Simulation of Communication Systems

By

Xiaoyuan Wu

Thesis submitted to the faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Approved:

Dr. William H. Tranter

Dr. Brian D. Woerner

Dr. Ira Jabcobs

October, 1998

Blacksburg, Virginia

Keywords: Simulations, Communication Systems

Simulation of Communication Systems

By

Xiaoyuan Wu

Committee Chair: Dr. William H. Tranter

Abstract

Digital communications and computers are having a tremendous impact on the world today. In order to meet the increasing demand for digital communication services, engineers must design systems in a timely and cost-effective manner. The number of technologies available for providing a given service is growing daily, covering transmission media, devices, and software. The resulting design, analysis, and optimization of performance can be very demanding and difficult. Over the past decades, a large body of computer-aided engineering techniques have been developed to facilitate the design process of complex technological systems. These techniques rely on models of devices and systems, both analytic and simulation, to guide the analysis and design throughout the life cycle of a system. Computer-aided design, analysis, and simulation of communication systems constitute a new and important part of this process.

This thesis studies different aspects of the simulation of communication systems by covering some basic ideas, approaches, and methodologies within the simulation context. Performance measurement of a digital communication is the main focus of this thesis. However, some popular visual indicators of signal quality, which are often generated in a simulation to provide a qualitative sense of the performance of a digital system, are also considered.

Another purpose of this thesis is to serve as a model for developing simulations or template of other systems. In other words, students learning to simulate a system can use the work presented here as a starting point.

Acknowledgement

First, I would like to express my most sincere thank to my advisor Dr. William Tranter, whose insight and invaluable advice has helped me to reach this important milestone of my life. He has been a mentor and friend in guiding me through my graduate career. It has been a great honor for me to work under the guidance of Dr. Tranter. I also would like to use this opportunity to thank Dr. Jacobs and Dr. Woerner to serve as my committee members. Without their expertise the completion of this thesis would not be possible. Their help and guidance are deeply appreciated.

I would also like to thank my parents and my sister for loving me and supporting me throughout my life. They have taught me the value of hard work and set great examples to me. Without them I would not be here today.

Last but not least, I would like to thank all my friends here at Virginia Tech. Without their help and support, I would not be able to complete this long journey.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

The complexity of communication systems and signal processing systems has grown considerably in recent years due to the advances in integrated circuit technology and in the demands placed on the systems themselves. This growth in complexity makes the traditional analytical methods of analysis more unsuitable for analyzing and designing modern communication systems. Therefore new and efficient design and analysis tools are needed. This demand can be met by using powerful computer-aided simulation-based techniques.

The simulation of communication systems is concerned with imitating some aspects of the behavior of communication systems without building actual hardware. The digital computer is used for this purpose. If each element of a physical communication system is represented by a mathematical model, the digital computer can serve as the "laboratory" for that system by connecting all these models together to simulate the actual system. This is referred to as a software model. The parameters of the model can be changed at will and the consequences can be observed without having to build hardware until the performance of the system is satisfactory.

1.1 History of Simulation-based Analysis and Design

The simulation of control systems using analog computers was used as a digital tool in the 1950s. The analog computer was a simulator of continuous systems that was composed of modular components that were interconnected via a patchboard to yield a block diagram configuration representing the system. The linear elements, such as integrators, summing junctions, and amplifiers were realized using operational amplifiers. Nonlinear modules such as limiters and arbitrary functions represented in piecewise linear form can be realized using devices and other nonlinear circuit elements. Any system whose behavior is described by a linear or nonlinear differential

equation with constant or with time varying coefficients can be reduced to a block diagram while in turn can be realized using components on an analog computer. By wiring the components according to the block diagram and exciting the model with appropriate signals one can simulate the dynamic behavior of a broad range of linear and nonlinear systems using the analog computer.

In 1960s, analog computer techniques started to be applied to digital simulation. The framework for digital simulation originated with block-oriented languages such as CSMP, MIDAS, and SCADS. These simulation languages emulated the behavior of analog computers on a component-by-component basis. These block-oriented simulation languages draw their motivation from the analog block diagram as a simple and convenient way of describing continuous systems. In mid 1960s, circuit analysis simulator such as ECAP and SPICE were also developed, which led to advances in numerical integration techniques and topological simplification of signal flowgraphs [2]. SPICE is still used extensively today.

Advances in discrete-time systems and digital signal processing have led to new approaches for digital simulation of systems. In the late 1960s and early 1970s, SYSTID, CSMP, CHAMP, LINK, and others were developed for aiding the analysis and design of satellite communication links [2].

In 1980s, interactive and menu driven simulators such as ICSSM, TOPSIM, and ICS were developed [2]. With these packages, simulations were performed on a mainframe or a super-minicomputer, and graphics terminals are used to provide a limited amount of interactive preprocessing as well as postprocessing.

Since mid 1980s, the simulation of communication systems has received more and more attentions as a separate subject within the electrical engineering community. In the past ten years, computer hardware and software technologies have undergone significant changes. Powerful workstations and personal computers have replaced mainframes as major computing tools. The current generation of simulation software packages such as BOSS, SPW, COSSAP, and others offer interactive, graphical, and user-friendly frameworks for simulation-based analysis and design of communication systems [2].

1.2 Overview of this thesis

This thesis deals with major aspects of the modeling and simulation of communication systems by studying two major simulation methods: the Monte Carlo method and the Quasianalytical method. It gives a detailed comparison between those two simulation methods. At the same time this thesis also explores another approach that differs from the conventional Monte Carlo method: Block Process.

Chapter 2 deals with many basic aspects of the simulation of communication systems, such as discrete-time representation of different signals as well as the lowpass representation of a bandpass signal.

Chapter 3 introduces the two major simulation methods, and an end-to-end Binary Phase Shift Keying (BPSK) communication system is used to demonstrate the effectiveness of the above methods.

Chapter 4 covers more sample systems to further demonstrate the strength and the weakness of two methods as well as how simulation of communication systems can be applied into analyzing and designing real-life communication systems.

Chapter 5, the concluding chapter, will summarize all the conclusions drawn from the previous studies and it is also hoped that this thesis will serve as a springboard for the reader to further explore the field of simulation of communication systems.

Another purpose of this thesis is to serve as a model for developing simulations or template of other systems. In other words, students learning to simulate a system can use the work presented here as a starting point.

Chapter 2

Representation of Signals in Simulation

2.1 Introduction

In a general sense, a system is defined as a combination and interconnection of several components to perform a desired task [1]. A signal is defined as the time history of some quantity, usually a voltage or current [7]. This thesis is concerned with the response of systems to random signals.

Most real life signals and systems that we wish to simulate are continuous. However, continuous-time signals and systems are represented by equivalent discrete-time signals and systems for computer simulation. In this chapter, we are to develop methods for obtaining such equivalent discrete representations.

2.2 Continuous and Discrete-Time Signal

A continuous signal is defined as a real or complex function of time x(t), where the independent variable, t, is continuous [2]. There are a few fundamentally important continuous signals, such as the unit step function u(t), the unit impulse function $\delta(t)$, the sinc function sinc(t), and complex exponential signals. Since these signals are well defined and very commonly used in the field of communications, this thesis will not spend any time defining and deriving these functions.

Discrete-time signal is only defined at discrete times $t = nT_s$, where n are integers. In most cases, such discrete-time signals result by sampling continuous signals at instants separated by the sampling interval T_s.

According to the sampling theorem, any continuous signal s(t) with a bandwidth of W can be uniquely represented by samples of s(t) taken at a rate of $f_s \geq 2W$ samples/sec. The minimum rate f_n

= 2W samples/sec is called the Nyquist rate. As a result, any real life continuous signal can be sampled at or above Nyquist rate. The resulting discrete-time signal can be used in simulation and this will not introduce any distortion into the system.

2.3 Representations of Bandpass Signals in Simulation

The fundamentals of simulation of digital communication system are the sampling theory and the complex envelope technique to represent signals and noise in a proper way [4]. In section 2.2, we covered the sampling theory briefly. In this section, we will cover the complex envelope technique.

Digital information-bearing signals are usually transmitted by some type of carrier modulation. When the bandwidth of such a signal is much smaller than the carrier frequency, this signal is defined as narrowband bandpass signals. Since we ultimately need to sample these functions, we wish to do so in the most efficient way possible. As mentioned previously, any continuous signal can be uniquely represented by the equivalent discrete model only if the sampling frequency is greater or equal to twice the highest frequency. In other words, any bandpass signal with bandwidth B and carrier frequency f_c has to be sampled at the minimum rate of $2*(f_c+B/2)$ according to the lowpass sampling theorem. However, any lowpass signal with bandwidth of W only needs to be sampled at a rate of 2W. It is known that the carrier modulated signals and bandpass system can, with minor restrictions, be analyzed and simulated as if they were low-pass. The technique used in the implementation of this idea is called complex envelope method [2].

Any carrier modulated signal $x(t)$ can be represented as

$$x(t) = r(t)\cos[2\pi f_c t + \varphi(t)] \tag{2.3.1}$$

which can be represented as

$$x(t) = \mathrm{Re}[r(t)e^{j\phi(t)}e^{j2\pi f_c t}] \tag{2.3.2}$$

where $r(t)$ is the amplitude modulation, $\varphi(t)$ is the phase modulation of the signal, and f_c is the carrier frequency. The signal

$$v(t) = r(t)e^{j\varphi(t)} \tag{2.3.3}$$

evidently contains all the information-related variations, and is of a low-pass nature. It is often called the complex low-pass equivalent or the complex envelope of the signal. If the narrowband condition defined earlier this section is satisfied one can show that bandpass filtering of $x(t)$ can be evaluated through an equivalent low-pass filtering operation with $v(t)$ as input.

In order to find the equivalent low-pass representation of a carrier modulated signal, one needs to begin with the modulated signal in the in-phase and quadrature form. Any bandpass signal $x(t)$ can be expressed as

$$x(t) = x_d(t)\cos 2\pi f_c t - x_q(t)\sin 2\pi f_c(t) \tag{2.3.4}$$

where $x_d(t)$ and $x_q(t)$ are the in-phase and quadrature components of the bandpass signal $x(t)$ respectively. It is known that if the bandwidth B of both the in-phase and quadrature components are such that B<< f_c, then the Hilbert transform of $x(t)$ is

$$\hat{x}(t) = x_d(t)\sin 2\pi f_c t + x_q(t)\cos 2\pi f_c t \tag{2.3.5}$$

The analytic signal of $x(t)$ is therefore

$$x_+(t) = x(t) + j\hat{x}(t) = [x_d(t) + jx_q(t)]e^{j2\pi f_c t} \tag{2.3.6}$$

The carrier modulated signal $x(t)$ is then

$$x(t) = \mathrm{Re}[x_+(t)] = \mathrm{Re}[\tilde{x}(t)e^{j2\pi f_c t}] \tag{2.3.7}$$

The equivalent low-pass representation or the complex envelope is then defined as

$$\tilde{x}(t) = x_+(t)e^{-j2\pi f_c t} = x_d(t) + jx_q(t) \tag{2.3.8}$$

2.3.1 Representation of the M-ary PSK Signals

An M-ary PSK signal can be represented as [3]

$$s(t) = A\cos[2\pi f_c t + \frac{2\pi}{M}(m-1) + \theta_i] \tag{2.3.9}$$

where $1 \le m \le M$, $0 \le t \le T$ and θ_i is the transmitter phase angle. Equation (2.3.9) can also be represented as

$$s(t) = A\mathrm{Re}\left\{ e^{j[\frac{2\pi}{M}(m-1)+\theta_i]} e^{j2\pi f_c t} \right\} \tag{2.3.10}$$

From equation (2.3.7), the complex envelope is given as [3],

$$\tilde{s}(t) = Ae^{j[\frac{2\pi}{M}(m-1)+\theta_i]} \tag{2.3.11}$$

Thus

$$x_d(t) = A\cos[\frac{2\pi}{M}(m-1) + \theta_i] \tag{2.3.12}$$

and

$$x_q(t) = A\sin[\frac{2\pi}{M}(m-1) + \theta_i] \tag{2.3.13}$$

We will use this signal form in Chapter 4 when the performance of M-ary PSK is studied.

2.3.2 Representation of the M-ary FSK Signals

An M-ary FSK signal can be represented as [3]

$$x(t) = A\cos[2\pi(f_c + I_m\Delta f)t + \theta_m] \tag{2.3.14}$$

or

$$x(t) = A\mathrm{Re}\left\{ e^{j2\pi I_m \Delta f t} e^{j\theta_m} e^{j2\pi f_c t} \right\} \tag{2.3.15}$$

where $1 \le m \le M$; $\Delta f = \frac{1}{T}$, which is the minimum frequency separation in order to guarantee the orthogonality between the signals; and $I_m = \frac{2m-1-M}{2}$. θ_m is the phase of each signal in the signal set.

Using equation (2.3.7), the complex envelope can be derived from equation (2.3.15) as

$$\tilde{x}(t) = Ae^{j(2\pi I_m \Delta f t + \theta_m)} \tag{2.3.16}$$

The complex envelope of $x(t)$ is then expressed as [3]

$$x_d(t) = A\cos(2\pi I_m \Delta f t + \theta_m) \tag{2.3.17}$$

and

$$x_q(t) = A\sin(2\pi I_m \Delta f t + \theta_m) \tag{2.3.18}$$

where $x_d(t)$ and $x_q(t)$ are the in-phase and quadrature components of the signal $x(t)$ respectively.

2.4 Filtering

Filtering is used in communication systems to select a desired signal and reject interference and noise. An ideal filter has a transfer function whose magnitude is flat within its "passband" and zero outside of this band of frequency; its midband gain is unity and its phase is a linear function of frequency.

In simulation, the filtering of bandpass signals is usually simplified to deal with the complex envelopes of signals and lowpass equivalents of bandpass filters. In MATLAB, classical filters can be easily defined with only a few parameters such as the passband, stop-band, and the tolerances allowed within these bands. These filters are described by polynomial with different coefficients. Then, the operation of filtering is realized using transposed direct form II network structure (details see Appendix C). Because the direct and quadrature components of a signal are orthogonal to each other, they need to be processed by the filter operation separately too.

Chapter 3

Simulation Techniques

The probability of symbol error is usually an important measure of the performance of a digital communication system. In the Monte Carlo simulation method the probability of symbol error is determined by designing a random experiment in which the communication system transmits symbols from an alphabet of size M total possible symbols, where M usually equals to 2^k. The measure of interest is the average number of errors in an arbitrarily long symbol sequence. That is, let N be the number of transmitted symbols, and n(N) be the number of observed errors. By the definition of relative frequency, which is an interpretation of probability, the probability $p = \lim_{N \to \infty} \frac{n(N)}{N}$ is the error probability associated with this system. If M = 2, the symbol error probability is usually referred to as the bit error probability. Generally, we use Bit-Error-Rate (BER) instead of bit error probability in simulation. Since each symbol has k bits for $M = 2^k$, each symbol error probability has an equivalent bit error probability. Therefore, we often use BER as a primary measurement of the basic reliability of a given system.

3.1 Two Major Methods of BER Measurement

Since performance evaluation is essential in the analysis and design of communication systems, the BER estimation is one of the primary goals for simulation of digital communication systems. The BER, which is defined as the fractional number of errors in a transmitted sequence, is certainly informative of the basic reliability of a system.

There are many simulation-based approaches for estimating the BER, and the Monte Carlo simulation approach discussed above is only one approach. Other approaches are importance

sampling, tail extrapolation and quasianalytical simulation. Here, we only look at two of those methods in detail. They are the Monte Carlo technique and the Quasianalytical technique.

3.2 Monte Carlo Method

A Monte Carlo simulation run can be viewed as a statistical experiment that is the software counterpart of an experiment performed on a real communication system. Therefore, a Monte Carlo simulation for a digital communication system can be developed from the mathematical model of the system. The system can be expressed as a block diagram in which each functional block is a signal processing operation realized as a DSP algorithm. We then pass information bits through the system. That is, we count the number of "successes" (errors, in this context) and divide by the total number of trials. The result is the relative frequency estimate of the error probability. This method requires no assumptions about the input processes or the system. In fact, in the implementation, the system is bypassed except for the output of the digital source and the decision device's version of the source. Since the source output is known, comparing the two sequences at some relative delay provides the empirical basis for an error rate. The block diagram of a simple communication system is illustrated in Fig. 3.1.

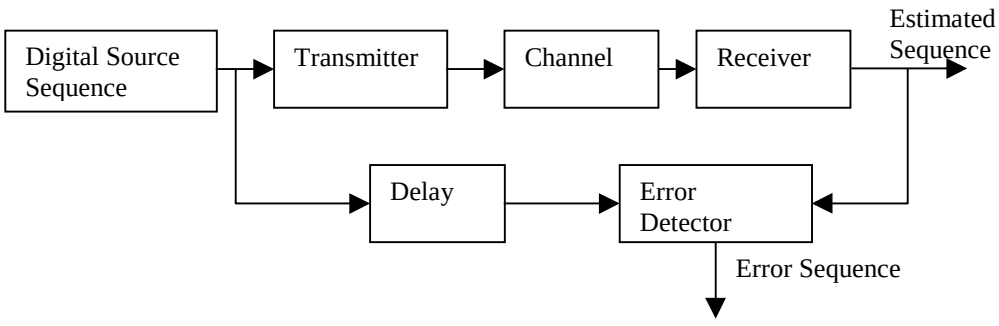


Fig. 3.1 Monte Carlo Estimation System Block Diagram

In order to implement the simulation of the system shown in Fig 3.1 it is necessary to know the delay of the system. This knowledge is, in effect, equivalent to symbol synchronization. For a carrier-modulated system, proper demodulation also requires carrier synchronization, namely, the "lining up" of the local oscillator phase with that of the incoming carrier. Carrier and symbol synchronization are essential functions in any digital system, and some form of synchronization is always implied. However, for purposes of discussing the Monte Carlo method or any other type of estimation techniques, the specific implementation of synchronization within the simulation is not central to the issue.

3.2.1 Quality of an estimator

Since a Monte Carlo simulation can be viewed as a statistical experiment, the value of any parameter obtained by such simulations is only an estimation of the true value. Therefore it is necessary for us to measure the quality of a Monte Carlo simulation. That is, the "closeness" of the estimate must be measured in a probabilistic sense. In this paper, three measures of quality of a Monte Carlo simulator are of interest: the bias, the variance, and the confidence interval.

3.2.1.1 **Bias of a Monte Carlo Simulator**

For almost all cases of interest, the simulator can be expressed as a weighted time-average:

$$\hat{Q} = \frac{1}{N}\sum_{i=1}^{N} w_i Y_i \tag{3.2.1}$$

where $\hat{Q}$ is the estimated value of a parameter with a true value of Q. Since we normally expect the estimation process to improve as N increases, an essential attribute of an estimator is that it converges to the true value as $N \to \infty$. Such an estimator is termed unbiased. Thus for an unbiased estimate $\hat{Q} \to Q$ as $N \to \infty$ for almost every sample function. Since $\hat{Q}$ is a random variable it has an associated distribution and pdf, say $f_{\hat{Q}}(q;N)$. As we have discussed, for an unbiased estimate

$$E(\hat{Q}) = \int_{-\infty}^{\infty} q f_{\hat{Q}}(q;N)dq = Q \tag{3.2.2}$$

It is important that a BER estimator is unbiased, since for an unbiased estimator the simulation result will, on the average, be correct.

As stated previously, for a Monte Carlo estimator in which N is the number of transmitted symbols, and N_e is the number of observed errors, the probability $\hat{p}_e = \frac{N_e}{N}$ is the estimated error probability associated with the system. As a result, we have $E(\hat{p}_e) = E(\frac{N_e}{N}) = \frac{E(N_e)}{N}$. Since $E(N_e) = Np_e$, it is clear that $E(\hat{p}_e) = \frac{Np_e}{N} = p_e$. Hence, the Monte Carlo estimator is unbiased [2].

Note that the above discussion does not require that the error events are independent. We therefore have that the Monte Carlo estimator $\hat{p}_e = \frac{N_e}{N}$ is, in general, an unbiased estimator. In the next section, we will derive an expression for the variance of the estimator. The result obtained will require independence of the error events. Because of bandlimiting and other important system attributes, we will see that this constraint is not often met in practice.

3.2.1.2 Variance of a Monte Carlo Estimator

The variance of an estimator,

$$\sigma^2(\hat{Q}) = E(\hat{Q}^2) - E^2(\hat{Q}) = \int_{-\infty}^{\infty} q^2 f_{\hat{Q}}(q;N)dq - E^2(\hat{Q}) \tag{3.2.3}$$

is a measure of the dispersion about the expected value. The smaller the variance, the better the estimator is, for a given N. Generally, $\sigma^2(\hat{Q}) \to 0$ only as $N \to \infty$, and we are faced with a tradeoff between estimator variance and sample size N, the latter being directly proportional to run-time in the simulation context.

It is convenient to use the moment generating function to determine the variance of a Monte Carlo simulator. The moment generating function is defined such that

$$\Gamma(z) = \sum_{k=-\infty}^{\infty} p_k z^k \tag{3.2.4}$$

Substituting $p_k = p\{N = k\} = \frac{N!}{k!(N-k)!} p_e{}^k (1-p_e)^{N-k}$, yields

$$\Gamma(z) = \sum_{k=0}^{N} \binom{N}{k} p_e{}^k (1-p_e)^{N-k} z^k = \sum_{k=0}^{N} \binom{N}{k} (p_e z)^k (1-p_e)^{N-k} \tag{3.2.5}$$

Using the binomial theorem which states that

$$(a+b)^m = \sum_{k=0}^{m} \binom{m}{k} a^k b^{m-k} \tag{3.2.6}$$

allows, the moment generating function $\Gamma(z)$, to be expressed as

$$\Gamma(z) = [p_e z + (1-p_e)]^N \tag{3.2.7}$$

From (3.2.4) it follows that

$$\Gamma'(z) = \sum_{k=-\infty}^{\infty} k p_k z^{k-1} \tag{3.2.8}$$

and

$$\Gamma''(z) = \sum_{k=-\infty}^{\infty} k(k-1) p_k z^{k-2} \tag{3.2.9}$$

Letting z=1, we get

$$\Gamma'(1) = \sum_{-\infty}^{\infty} k p_k = E\{k\} \tag{3.2.10}$$

and

$$\Gamma''(1) = \sum_{k=-\infty}^{\infty} k^2 p_k - \sum_{k=-\infty}^{\infty} k p_k \tag{2.3.11}$$

Thus

$$E\{k^2\} = \Gamma'(1) + \Gamma''(1) \tag{3.2.12}$$

We now use these results to obtain $E\{k\}$ and $E\{k^2\}$ [8],

$$\begin{aligned} \sigma_k{}^2 = \sigma_{N_e}{}^2 &= E\{k^2\} - (E\{k\})^2 \\ &= Np_e - Np_e{}^2 \\ &= Np_e(1 - p_e) \end{aligned} \tag{3.2.13}$$

3.2.1.3 Confidence Interval

The confidence interval is the most important measure of the quality of an estimator because it quantifies the measure of spread with an associated probability. Let $h_1(\hat{Q})$, $h_2(\hat{Q})$ be two functions of the estimator, such that, the interval (h_1, h_2) brackets the true value Q. The difference $h_1 - h_2$ is the width of the confidence interval. The probability associated with the condition $h_2 \le Q \le h_1$ is called the confidence level and is usually denoted $1 - \alpha$. Thus, a confidence interval and confidence level are defined through [2]

$$P \quad [h_2(\hat{Q}) \le Q \le h_1(\hat{Q}) = 1 - \alpha \tag{3.2.14}$$

Typical values for α are 0.05 and 0.01, corresponding to "95% confidence level" and "99% confidence level," respectively.

However equation (3.2.14) only defines confidence interval with certain confidence level in theory. Since the true value Q is unknown in almost all cases, equation (3.2.14) has little practical meaning. In order to evaluate the confidence interval, some kind of approximation is needed. The

most common methods to evaluate the confidence interval are Poisson approximation and Gaussian approximation.

Let N be the total number of symbols processed in a Monte Carlo simulation and N_e be the total number of errors observed. The estimated error probability $\hat{p}_e$ is therefore $\frac{N_e}{N}$. For any give N, N_e has a binomial distribution. Hence one can obtain a closed form for the confidence interval in terms of the cumulative beta distribution [5]. However, it is difficult to iteratively evaluate the binomial distribution when N is large. Therefore, we use two alternative approaches: Poisson approximation and Gaussian approximation.

It is well known that if there is a positive constant λ such that $\lim_{N\to\infty} N\,\hat{p}_e = \lambda$, the binomial distribution can be approximated by the Poisson distribution. Under the limiting condition the cumulative binomial distribution can be replaced by the cumulative Poisson distribution

$$\sum_{k=0}^{N_e} e^{-\lambda_1}\frac{{\lambda_1}^k}{k!} = \frac{\alpha}{2} \tag{3.2.15}$$

$$\sum_{k=N_e}^{N} e^{-\lambda_2}\frac{{\lambda_2}^k}{k!} = \frac{\alpha}{2} \tag{3.2.16}$$

As the above two equations to be solved for λ_1 and λ_2, we can construct the confidence interval $\left(\frac{\lambda_1}{N}, \frac{\lambda_2}{N}\right)$ with confidence level $1-\alpha$. A table of upper and lower values of λ for different confidence level for $0 \le N_e \le 50$ using the Poisson approximation can be found on page 499 [2].

The Poisson approximation gives us a usefully large range of values. For larger N_e such as $N_e > 10$, there exists another approximation, the Gaussian approximation, which is simpler to use. It is know that as $N \to \infty$, the estimated error probability $\hat{p}_e$ tends to have a normal distribution

with mean p_e and variance $\frac{p_e(1-p_e)}{N}$. Hence we can construct a confidence interval in the form [5]

$$\begin{aligned}\Pr ob\{&\frac{N}{N+d_\alpha{}^2}[\hat{p}_e+\frac{d_\alpha{}^2}{2N}-d_\alpha(\frac{\hat{p}_e(1-\hat{p}_e)}{N}+\frac{d_\alpha{}^2}{2N})^{1/2}]\\ &\le p_e \le \frac{N}{N+d_\alpha{}^2}[\hat{p}_e+\frac{d_\alpha{}^2}{2N}+d_\alpha(\frac{\hat{p}_e(1-\hat{p}_e)}{N}+\frac{d_\alpha{}^2}{2N})^{1/2}]\}\\ &=1-\alpha\end{aligned} \tag{3.2.17}$$

where p_e is the true value of the BER, and d_α is chosen so that

$$\frac{1}{(2\pi)^{1/2}}\int_{-d_\alpha}^{d_\alpha} e^{-t^2/2}dt = 1-\alpha \tag{3.2.18}$$

Let $\hat{p}_e = 10^{-v}$ and $N = N_e 10^v$, and under the assumptions that $\frac{N}{N+d_\alpha{}^2} \cong 1$ and $\hat{p}_e(1-\hat{p}_e) \cong \hat{p}_e$, which are usually more than amply satisfied, equation (3.2.23) can be expressed as

$$\Pr ob[y_- \le p \le y_+] = 1-\alpha \tag{3.2.19}$$

where the confidence interval (y_-, y_+) is given by

$$y_\pm = 10^{-v}\{1+\frac{d_\alpha{}^2}{2N_e}[1+(\frac{4N_e}{d_\alpha{}^2}+1)^{1/2}]\} \tag{3.2.20}$$

This interval is plotted in Figure 3.2 for confidence level of 90%, 95%, and 99%.

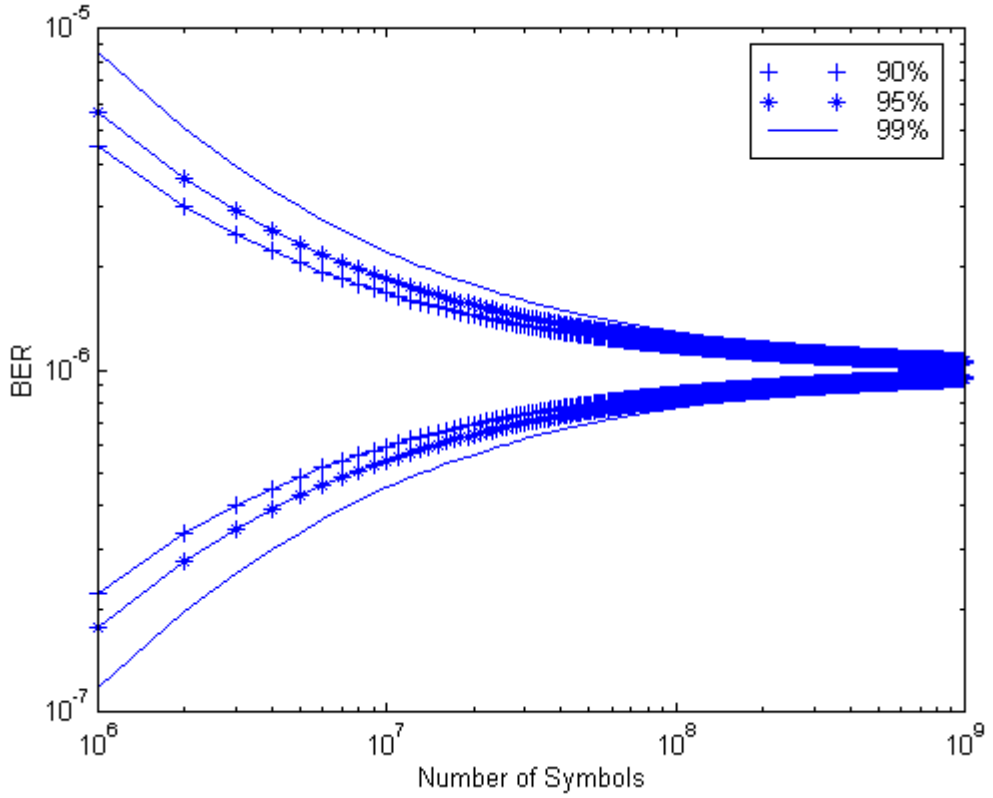


Fig. 3.2 Confidence Interval on BER when estimated value is 10^{-6} based on normal approximation [6]

3.2.2 Example: a BPSK System Simulation

In the previous sections, we introduced the Monte Carlo simulation method to evaluate the performance of communication systems. In this section, an end-to-end BPSK communication system is used as an example to further demonstrate the Monte Carlo simulation method and its properties.

The objective of this example is to evaluate the performance of a simple BPSK communication system with the presence of Additive White Gaussian Noise (AWGN), taking into account the inter-symbol interference (ISI) caused by the transmitting filter. A BPSK communication system with AWGN environment is a very easy system and consequently, it is relative easy to evaluate with an analytical solution. However, the non-linear effect of the transmitter filter is not as easy to follow

with the analytical solution. A Monte Carlo approach can simulate both AWGN environment and the ISI caused by the transmitting filter very well and thus provide us the accurate performance evaluation of such a system. Moreover, a Monte Carlo simulation also can provide us waveform level simulation products, which will help us to gain more insight of the system.

3.2.2.1 Signal Representation

As mentioned in chapter 2, it is more efficient to represent a bandpass signal with its equivalent complex envelope representation in a simulation. A bandpass BPSK signal can be represented mathematically as following

$$s(t) = A\sin[2\pi f_c t + (m-1)\pi + \theta_i] \tag{3.2.21}$$

where $m = 1, 2$ and θ_i is the transmitter phase angle.

Using the results derived in equation (2.3.12) and (2.3.13), the direct and quadrature components can be expressed as

$$s_D(t) = A\cos[(m-1)\pi + \theta_i] \tag{3.2.22}$$

and

$$s_Q(t) = A\sin[(m-1)\pi + \theta_i] \tag{3.2.23}$$

Therefore, the lowpass representation of a BPSK signal is

$$\begin{aligned} s_l(t) &= s_D(t) + js_Q(t) \\ &= A\cos[(m-1)\pi + \theta_i] + jA\sin[(m-1)\pi + \theta_i] \end{aligned} \tag{3.2.24}$$

in this example, A is set equal to 1 for simplicity.

3.2.2.2 The System Model

The block diagram of a BPSK communication system that we are studying in this example is shown in figure 3.3

The data signal is assumed to be uniformly distributed so that the probability for a binary 0 or a binary 1 is the same. The "*rand*" function in Matlab is used to generate such data signals. In this

simulation, -1 is binary 0 and 1 is binary 1 in order to get rid of the dc component of the signal. In Monte Carlo simulation, the whole simulation is conducted in the bit by bit fashion. That is, each data bit is generated and passed through the system to check if an error has been made before the second data bit is generated.

After the data bit is generated, it is modulated using the complex envelope representation that was derived in the previous section. In this simulation, the BPSK signal is assumed to have a random

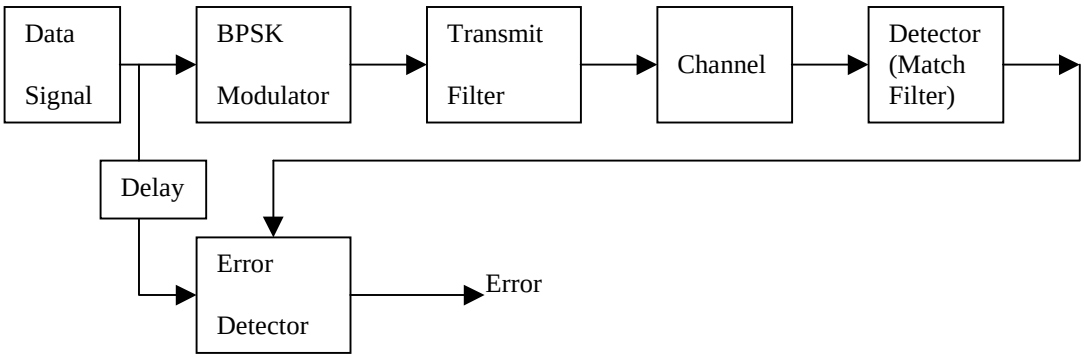


Fig. 3.3 BPSK system block diagram

phase angle uniformly distributed between 0 and 2π. It is our intention to show that the phase angle of the signal does not affect the performance of the system.

After the data signal is generated, it is sampled and then passed through a transmit filter. In this simulation, a sampling rate of 10 samples per bit is used for convenience. A transmitter filter is used here to limit the spectrum of the transmitted signal. In Matlab, there is a built-in *filter* command for filtering operation. However, this filtering operation is a parallel operation. That is, all data needs to be used as input of the *filter* command to get the right output of the filter. In Monte Carlo simulation, the simulator is operating in a sample-by-sample mode. Therefore, we have to either write a sequential filtering function ourselves or find a way to save the final state information of the filter for each sample and use it as the initial state for the next sample. Fortunately, both tasks are not very hard to achieve. Since saving each state of the filter for every sample is not very efficient, we will

use a simple sequential filter routine to accomplish such tasks (Details of the sequential filter routine is given in Appendix C). The transmitting filter used in this simulation is a Butterworth filter with the bandwidth and the order defined by the user. Since a Butterworth filter is not an ISI-free filter, the inter-symbol interference is introduced into the system by the transmit filter. It is also our intention to study the effect of the inter-symbol interference on the system's performance.

The output of the transmitter filter is then passed through an AWGN channel because it is the most commonly used model for noise in communication systems. Since E_b/N_o values are input parameters of the simulator and the energy per bit is assumed to be 1 in this example, we can calculate the noise power from those two parameters. The variance of the noise can be easily calculated using the following equation

$$\sigma^2 = \frac{E_b k}{E_b / N_0} \tag{3.2.25}$$

where k is the sampling rate.

After the signal has been passed through the channel, it needs to be received and detected for error. It is our intention to study the effects of the inter-symbol interference caused by the transmitting filter. A filter that is only matched with the noise portion seems to be a natural choice because it minimizes the effect of the noise. It can also be proven that in baseband, such a filter function is nothing more than an integrate-and-dump operation. Therefore, in this simulation the signal detection operation can be further simplified. At the receiving end, we only need to accumulate 10 samples, sum them up and make a hard decision by comparing the sum of the samples with the threshold value. The actual implementation of such task is very easy. However, there are some issues that need to be addressed here. First, we need to make sure that the 10 samples that are integrated are the samples from the same bit. This is, in a sense, a symbol synchronization problem in a simulation. Since each bit is sampled at the same rate, this task can be easily accomplished by lining up the first bit. In order to line up the first bit, we must measure the system delay. The easiest way to measure the delay is to test different value of delay. The value that yields the lowest BER is

the value of the system delay. One can also observe the delay from the eye-diagram. The filtered signal will not cross over at the same place as the signal before being filtered. The difference between the two diagram is also the system delay. Secondly, since we intend to show that the transmitter phase angle has no effect on system performance, the signal will have an arbitrary phase angle. The received signal constellation diagram is shown below

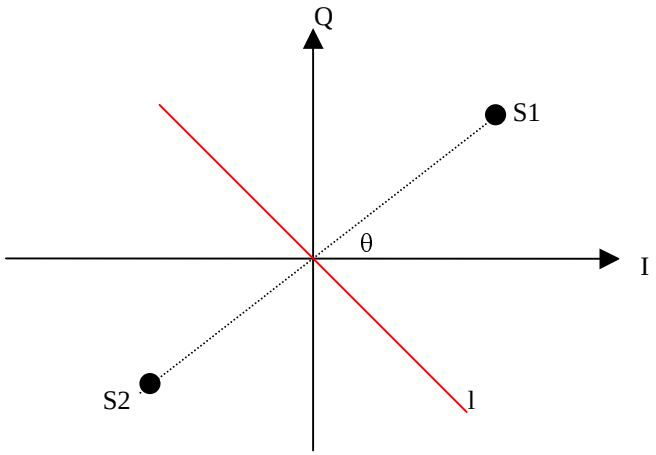


Fig. 3.4 Signal constellation diagram for BPSK signal with transmitting angle θ

It is obvious that the threshold for detecting signal s_1 and s_2 is *l*. It is also intuitive that if we rotate s_1 and s_2 θ degrees clockwise, the quadrature axis becomes the threshold and the error probability will still remain the same. In this simulation, the second detection scheme is used.

After the detection and decision have been made, the received bit can be compared with the original transmitted bit to reach an error statistic. The simulator keeps count the resulting error statistic until enough errors have been accumulated to estimate the BER with certain confidence level.

3.2.2.3 **Simulation Results**

Figure 3.5 shows the BER estimation for this BPSK system. It is shown in the graph that the simulated results are poorer than the theoretical value with the sole presence of the AWGN. This result clearly demonstrates the additional degradation on the system performance caused by the ISI

introduced by the transmit filter. If the bandwidth of the transmit filter is increased so that the filter causes little or no ISI, the simulated BER curve then should agree with the theoretical value. Since varying the bandwidth is a parametric study of the system, this graph will be generated using Quasianalytical technique and will be shown in the latter section of this paper.

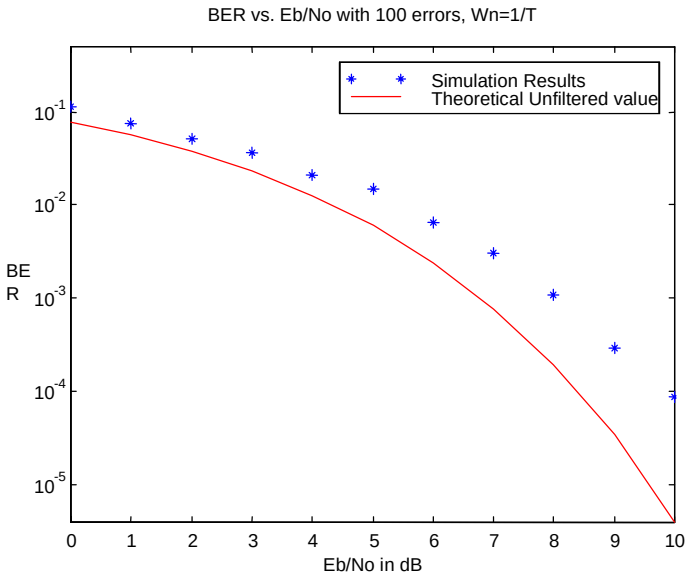


Fig. 3.5 Simulated BER performance curve with ISI

As analyzing and designing tool, Monte Carlo method can also generate the waveforms level simulation products such as eye-diagram and signal constellation diagram at different points of the system to help user to gain more insight of the behavior of the system. In this example, the waveforms are monitored at four different places, the signal before and after the transmitting filter and the signal before and after the receiver.

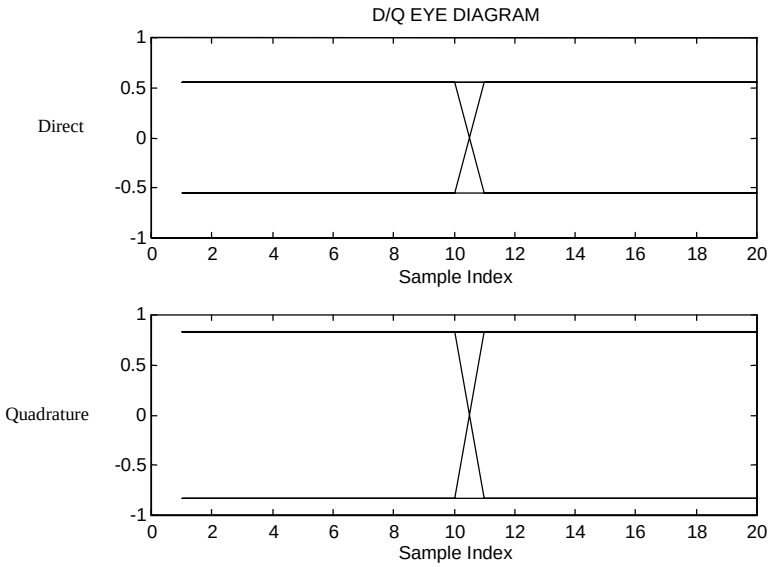


Fig. 3.6 Eye-diagram of the input of the transmitting signal

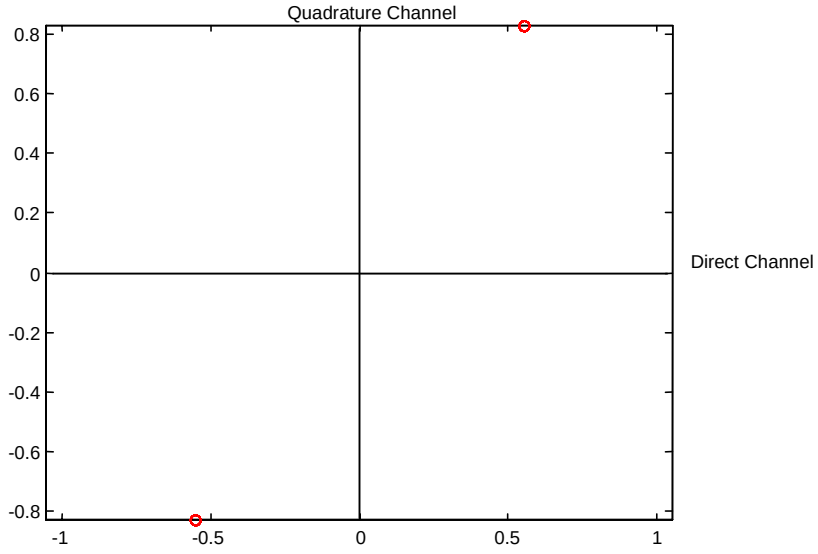


Fig. 3.7 Signal constellation of the input of the transmitting signal

Both figure 3.6 and figure 3.7 show the signal after the modulation and before filtering. The straight transition between the states in eye-diagram shows clearly that no distortion is present at this point.

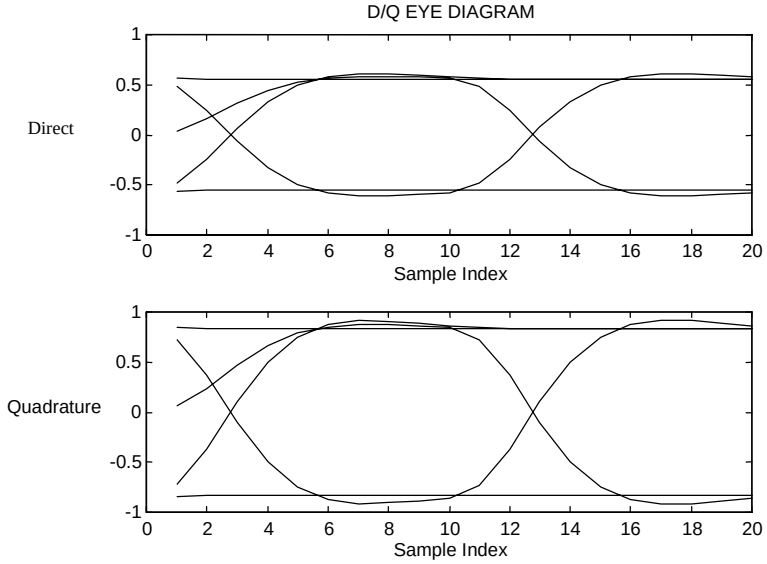


Fig. 3.8 Eye-diagram of the output of the transmitting filter

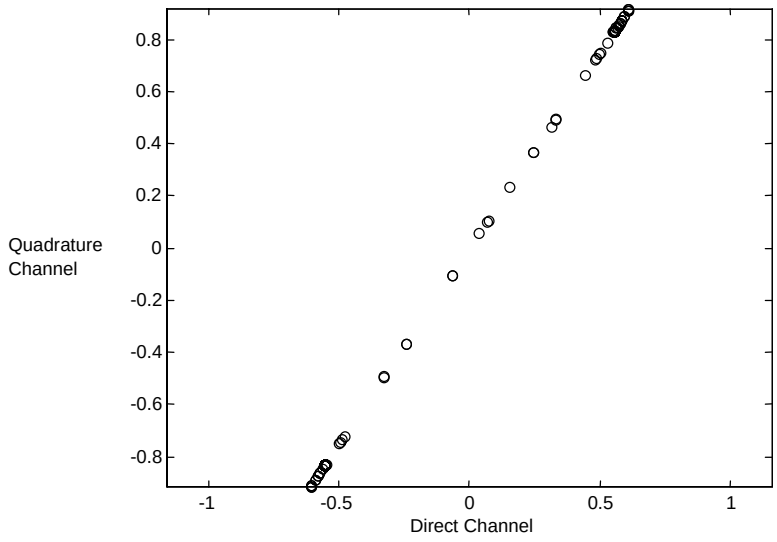


Fig. 3.9 Signal constellation of the output of the transmitting filter

The eye-diagram and the signal constellation diagram of the output of the transmitting filter are shown in Fig. 3.8 and Fig. 3.9. The effect of the inter-symbol interference (ISI) of the transmitting filter is clearly shown in these two diagrams. The eye-diagram does not have the straight transition between states any more. At the same time, it is worth noticing that there exists a transient state from the relaxed state to the first state at the first sample point, which is not present in the eye-diagram of the unfiltered signals. The signal constellation diagram also shows the signal points are scattered between the two original signal points.

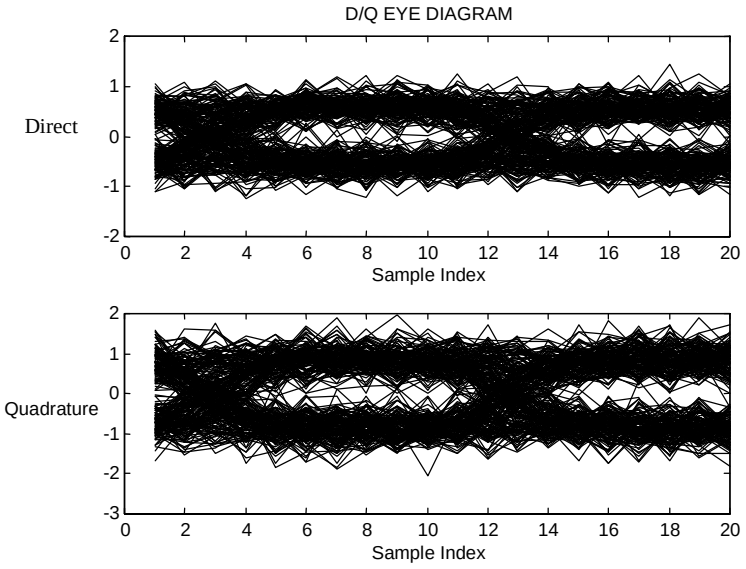


Fig. 3.10 Eye-diagram of the input of the receiver (E_b/N_o=15dB)

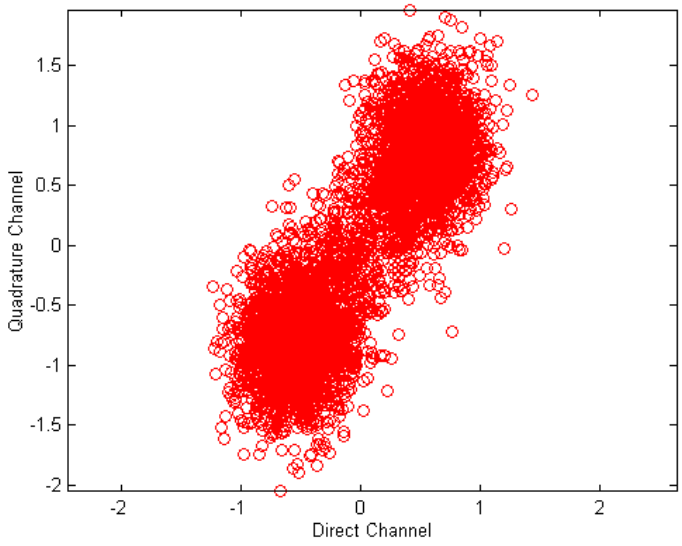


Fig 3.11 Signal Constellation of the input of the receiver

Figure 3.10 and 3.11 show the signal constellation and the eye-diagram of the signal after the noise has been added. In the eye-diagram, the signal states are close to each other due to the presence of the noise (closed eye). In the signal constellation diagram, the signal points are scattering around the original signal points. The histogram diagram shown in Fig. 3.12 illustrates how the signal points are distributed on the direct and quadrature axis. However, Fig. 3.12 is difficult to interpret due to the presence of the arbitrary transmitter phase angle θ. In order to have a conventional histogram that is easier to understand, we are going to rotate all signal points $-\theta$ degrees. The structure used to rotate the signal point is well known to us and is shown in Fig. 3.13. Set ϕ equal to $-\theta$, and the resulting $Y_d(t)$ and $Y_q(t)$ are the desired new direct and quadrature components of the signal. The resulting histogram is shown in Fig. 3.14. After the rotation, the quadrature axis becomes the threshold for detecting the two signal points s_1 and s_2. Therefore, all the error statistics will only be shown on the direct components of the signal. This is clearly illustrated in Fig. 3.14. The direct component has two sets of Gaussian distributed signal points with mean of 1 and –1.

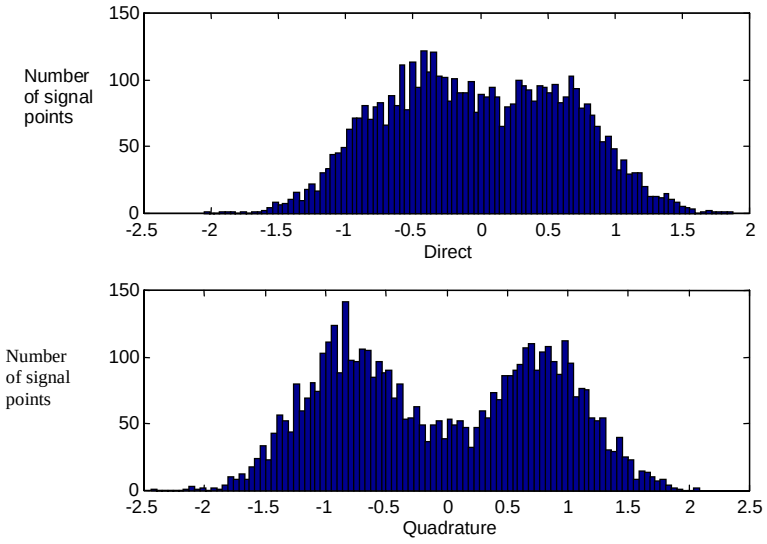


Fig. 3.12 Histogram of the signal points in the signal space

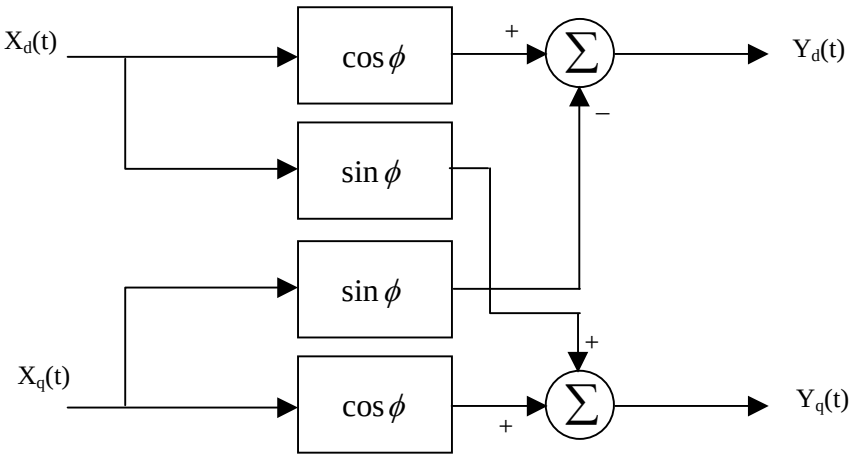


Fig. 3.13 Complex phase shift network

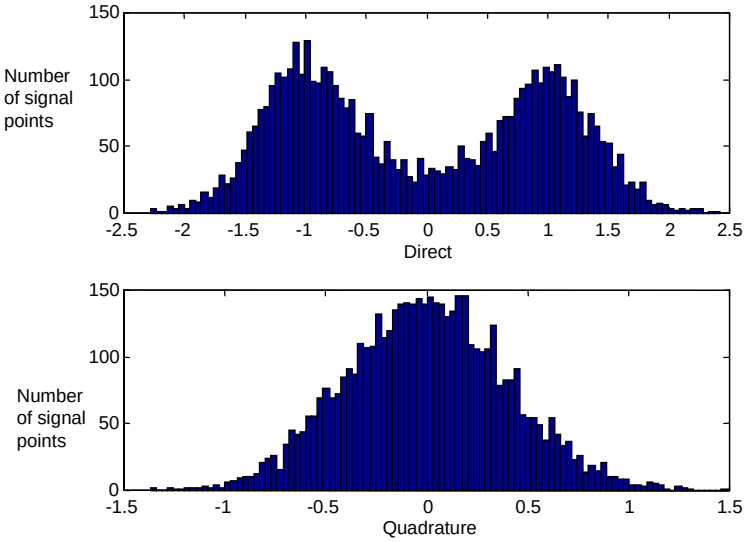


Fig. 3.14 Histogram of the signal points in the signal space

After the signal has been received, a hard decision is made on all the detected signals. The decision scheme used in this simulation was illustrated in Fig. 3.4. The signal constellation of the received signal is shown in Fig. 3.15.

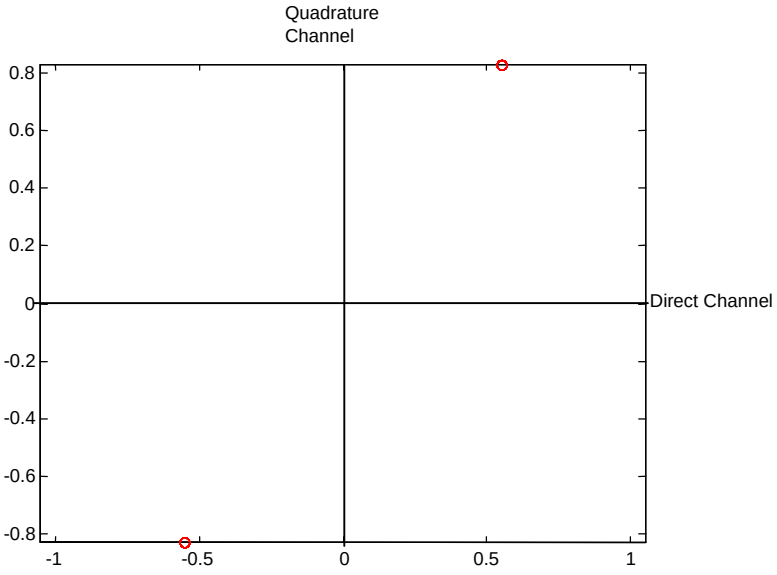


Fig. 3.15 The signal constellation of the received signal

The received signal has the same constellation diagram as the original signal, which means that the detection and hard decision schemes work very well. From this example, it is very clear that all the waveforms generated by the simulator can not only help us understanding the behavior of the system under different conditions, but also help us checking the correctness of the simulator itself. Therefore, these simulation products are very useful and important. This is a significant advantage of the Monte Carlo method.

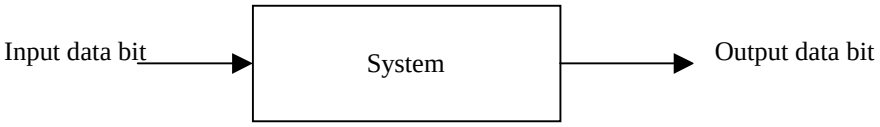
3.2.3 Block Analysis Method

As was shown in the previous section, a Monte Carlo simulation is run on a sample-by-sample basis. This structure is shown in Fig. 3.16(a). Each sample to be produced through the simulation is sampled, filtered, and detected sequentially. The process is straightforward and easy to program. Sequential time domain processing is efficient for system blocks that do not contain memory. Where memory is present in a system block the sample value at the output of a block is a function of past input samples as well as the present input sample. System blocks that are linear and not memoryless are typically expressed using a recursive relationship. Computer processing of these recursion relationships is time consuming and lead to inefficient simulations. Unfortunately, practical communication systems use one or more filters in their implementation and filters exhibit memory. Another approach is needed to increase simulation efficiency.

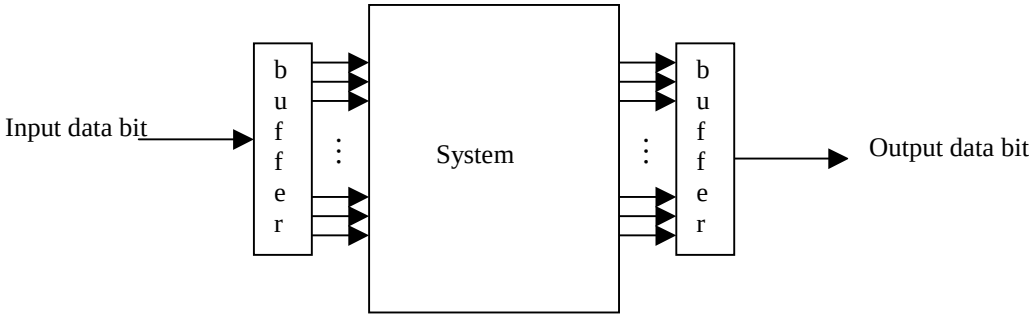
Computing the output of a linear filter, given the input and unit impulse response of the filter, is a convolution. We know that time domain convolution is an inefficient process on the computer and that implementing the process in the frequency domain is much more efficient. The implementation involves Fourier transforming the input signal, multiplying the resulting transform by the Fourier transform of the unit impulse response, and inverse transforming the result. Using the Fast Fourier Transform (FFT) to implement the transform operations results in a very efficient implementation, especially if radix-2 FFTs can be used. Since a block of samples must be Fourier transformed in order to implement the filter in the frequency domain, sample-by-sample processing is not

appropriate. The transform, or frequency domain, approach to simulation of system blocks having memory is referred to as Block Processing. The basic structure of this approach is shown in Fig. 3.16(b). The MATLAB routine "*filter*" is well suited to implement the approach discussed above.

Although the Block Analysis method follows the basic approach of the Monte Carlo method, the improvement on the simulation efficiency achieved by this method gives us some capabilities, such as parametric study, otherwise can only be achieved by the Quasianalytical technique. However, at the same time, the Block Analysis method still keeps the same waveform level simulation capabilities of the conventional Monte Carlo method.



(a) The conventional Monte Carlo method



(b) The Block Analysis method

Fig. 3.16 Block diagram of the conventional Monte Carlo method and the Block analysis method

3.2.3.1 The Performance Improvement of Block Analysis Method

In this section, we are going to study the performance improvement achieved by the Block Analysis method compared with the conventional Monte Carlo method. To do this, we are still going to simulate the sample BPSK system used in the previous sections. We are going to increase the block size from 1 to N, where N is an arbitrarily big number. The total simulation run-time is going to be monitored and plotted. Obviously, the block size of 1 is the same as the conventional Monte Carlo method. Thus, we can see how the simulation run-time varies with the different block size.

Before the simulation can be conducted, some preliminary study needs to be done to ensure the correctness of this simulation. First of all, because the system is not memory-less, it is necessary for us to save the system's state at the end of each block and use it as the initial state for the next block so that the Inter-Symbol Interference is not neglected. Fortunately, the "*filter*" command in Matlab has this function. It accepts the initial condition as a parameter, and outputs the final condition as another parameter. Therefore, this operation can be easily achieved. Secondly, due to the Fourier transform operation performed by the *filter* command, the block size of power of 2 is preferred to reach the optimal performance results.

Since the basic approach is the same for the Block Analysis method and the Monte Carlo method, we can modify the original Monte Carlo simulation code to construct the Block Analysis simulator. Figure 3.17 shows the performance improvement achieved by the Block Analysis method. When block size equals to 1, it is the same as the conventional Monte Carlo simulator. As the block size increased, the simulation run-time decreased very quickly. It is clearly shown in the graph that the computer run-time is reduced by a factor of 10 as the block size increases.

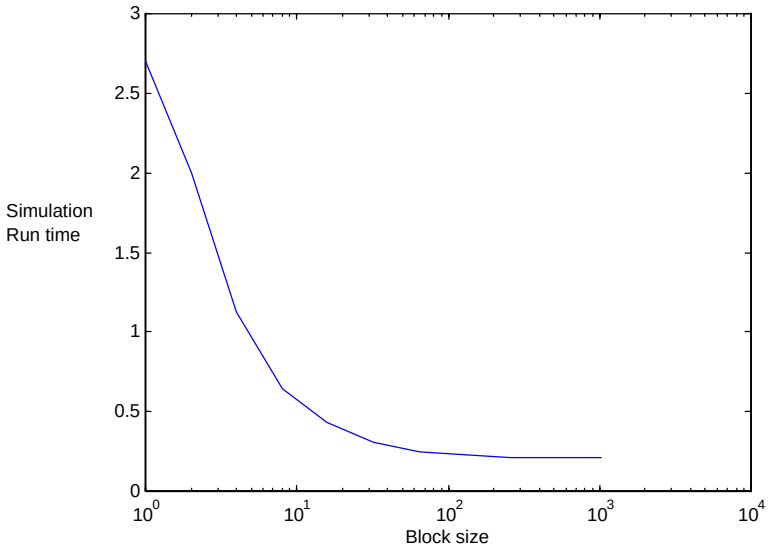


Fig. 3.17 Simulation Run-time as a function of block size

Since the only place that the Block Analysis method is different from the Monte Carlo method is the way it processes the data, the reliability parameters such as bias, variance and the confidence interval are essentially the same as the Monte Carlo method. Here, we will not cover them in detail again. The bottom line is that the Block Analysis method gives us an unbiased estimation of the BER.

3.2.4 Discussion

In the simulation context, Monte Carlo simulator represents the "true" situation for systems. It is very simple to implement and is independent of the specifics of the system. Therefore, it is one of the most fundamental simulation methods and should be included in any simulation package. The capability to generate waveform level simulation results in various position of the system also makes the Monte Carlo simulation essential as an analysis and design tool.

However, since the Monte Carlo method is just an estimation of the actual BER performance of the system, it can not be used blindly without any mention of confidence level. For without confidence level, any BER simulation done by Monte Carlo method has no practical meaning at all.

Although the Monte Carlo method is simple to implement and generally is the closest experiment next to actually building the actual system, the computer run-time usually is very long and even impractical. Therefore, for low BER systems, compromises are often necessary to get a relatively good estimation in reasonable time.

The Block Analysis method uses the matrix capabilities of Matlab to effectively reduce the simulation run-time with big block size. This performance improvement is very significant, especially for low BER system simulations. At the same time, the Block Analysis method also makes it possible to bring parametric study into waveform level simulation, which is also very helpful to gain more insight of the behavior of different communication systems.

3.3 Quasianalytical Estimation

Quasianalytical estimation technique, often known as semianalytical method, is the second simulation method studied in this paper. Basically, this technique combines both simulation and analysis. The simulation is used to generate a noiseless waveform at the receiver. Given this waveform and assuming that the noise is additive and has a known pdf, one can then calculate the probability of error with a formula [2]. This is the analysis portion. In the simulation part, we let the computer simulate the effect of the system distortion in the absence of noise, and then superimpose the noise on the noiseless waveform in the analytical part. The simulation part can be further clarified by Fig. 3.18. Fig. 3.18(a) shows a hypothetical transmitted bit stream, while Fig. 3.18(b) shows the corresponding noiseless received waveform at the input of the decision device. The value of this waveform at the kth sampling instant is denoted v_k. When the noise is superimposed, the resulting error probability is simply

$$Pe_k = \begin{cases} \Pr ob[noise > v_k] = \int_{v_k}^{\infty} f_n dn, & v_k < 0 \\ \Pr ob[noise < v_k] = \int_{-\infty}^{v_k} f_n dn, & v_k > 0 \end{cases} \tag{3.3.1}$$

where f_n is the pdf of the noise, which is assumed to be zero mean, and it is also implied that the threshold for detection is zero. Graphically, the distortion of $v_k + n$ is simply the distortion of n shifted by v_k. The probability of error is just the shaded area in Fig. 3.18(b) under the tail. The total probability of error can then be expressed as

$$Pe = \frac{1}{N} \sum_{k=1}^{N} Pe_k \tag{3.3.2}$$

where N is the total number of the possible states of v_k.

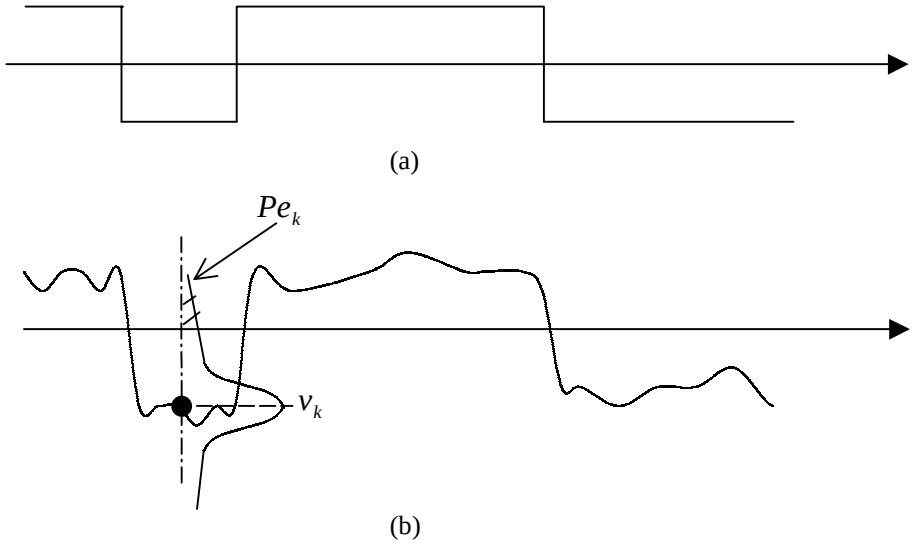


Fig. 3.18 Illustration of quasianalytical method.
(a) Transmitted bit stream. (b) Received waveform [6]

3.3.1 Restrictions and considerations for Quasianalytical technique

The biggest advantage gained by using Quasianalytical method is time saving. This is achieved because we do not have to wait for error to occur as in the case of Monte Carlo simulation. Instead, we simply calculate an integral, which in principle takes the same amount of time to evaluate regardless of the BER. However, the Quasianalytical technique cannot be applied to any arbitrary system. There are three restrictions associated with the Quasianalytical technique:

1. The system has to be linear from the point that the noise is introduced.
2. The memory of the system needs to be predetermined and should be relatively short. That is, if the amplitude of the received noiseless signal at any instance is a function of only relatively few preceding symbols, then a comparatively short sequence can produce all of the possible states of the received signal v_k.
3. The probability density function of the noise has to be a known function specified by the user in order to map all the possible states into an error probability. Thus, the overall bit error rate can be calculated.

Any system that satisfies the above three conditions can be simulated using the Quasianalytical technique.

There is a very simple way to determine the memory of the system. First, set the system at relax state, and then pass a step function at time t and monitor the output waveform. Since the system has memory, the output will have a transient state and settle at time t_1. The system then is determined to have a memory of $(t-t_1)/T_b$ bits, where T_b is the period of each bit. A small simulation program can easily accomplish this task. First a stream of zeroes can be sent through the system to make sure the system is completely relaxed and at time t, a stream of ones is sent. By monitoring the transient state of the response of the system, the memory of the system can be determined. This operation can be further clarified by the following example. Assume that a simple second order Chevychev II filter is

used in the system and our task is to determine the memory of this filter. By using the method mentioned above, the transient response of the filter can be obtained and is shown in Fig.3.19.

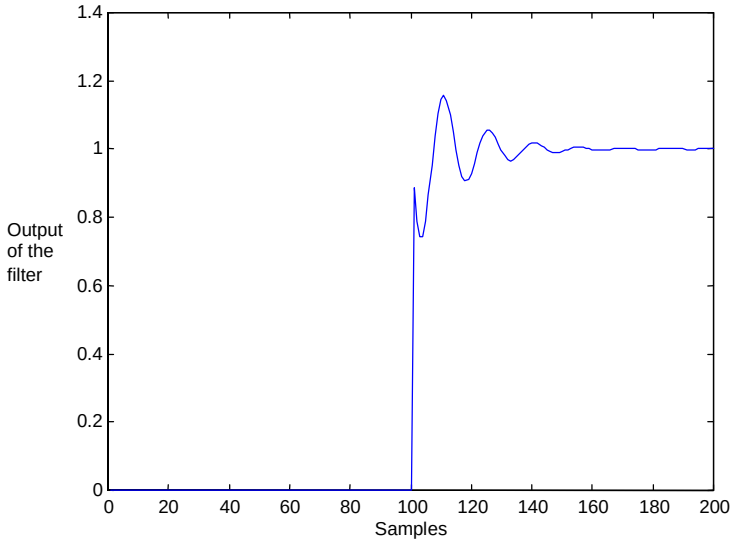


Fig. 3.19 The response of a second order Chevychev II filter

In this example, each information bit is sampled ten times and at the eleventh bit, a stream of ones is sent. The graph shows that the system's response settles to a steady state at about the 150^{th} sample, which means that the system's memory is about 50 samples or 5 bits.

After the memory of the system is decided, a maximum length PN sequence with length $N = Q^M$ will be generated, where Q is the size of the signal alphabet and M is system's memory in bits. Since the maximal PN sequence generator only generates $N-1$ sequences with the exception of the all zero case, the all zero case can be added on easily. All N sequences can then be sent through the system to produce all the possible states for the received signal v_k. This operation can be clearly demonstrated in Fig. 3.20.

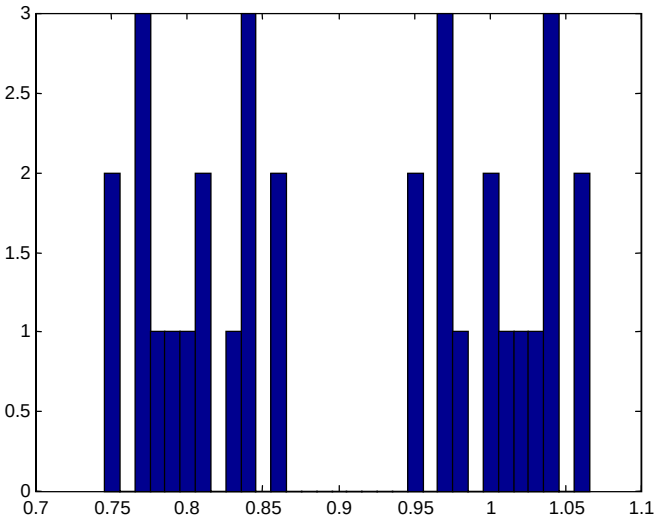


Fig. 3.20 All the possible states for the received signal

Using the same Chevychev II filter used above, since the memory of the filter is determined to be 5 bits, the total number of all the possible states of the received signal v_k is $N = 2^5 = 32$. Fig. 3.20 clearly shows that there are 32 total states for the received signal, which are dispersed along the axis.

Now all the possible states of the received signal have been determined. The last step is to map all these states into different error probability by using the pdf of the noise, and calculate the final error probability of the system. As we know that if a white noise is passed through a filter with transfer function $H(f)$, the average power at the output is

$$P_{n_0} = N_0 \int_0^\infty |H(f)|^2 df \tag{3.3.3}$$

where $\frac{1}{2}N_0$ is the two-sided power spectral density of the input. If the filter were ideal with bandwith B_N and midband gain H_0, the noise power at the output would be

$$P_{n_0} = N_0 B_N {H_0}^2 \tag{3.3.4}$$

Here B_N is referred to as the noise-equivalent bandwidth of $H(f)$. Obviously, in order to calculate the system's error probability, we need first to obtain the noise power at the receiver. As a result, the noise-equivalent bandwidth, B_N, needs to be calculated.

Once again, a small simulation program can easily accomplish the task of determining the noise-equivalent bandwidth B_N. From the equation (3.3.3) and (3.3.4), we can easily derive that

$$B_N = \frac{1}{H_0^{\,2}} \int_0^{\infty} |H(f)|^2 \, df \tag{3.3.5}$$

In the discrete form, equation (3.3.5) can be changed to

$$B_N = \frac{1}{H^2(1)} \frac{f_s}{2} \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 \, d\omega \tag{3.3.6}$$

where $H(1)$ is the dc response of the digital filter and f_s is the sampling frequency. By using Parseval's theorem and z-transform theory, equation (3.3.6) yields

$$B_N = \frac{f_s}{2} \frac{\sum_{n=-\infty}^{\infty} h^2(n)}{\left[\sum_{n=-\infty}^{\infty} h(n)\right]^2} \tag{3.3.7}$$

where $h(n)$ is the unit impulse response of the digital filter. Theoretically, equation (3.3.7) requires infinite amount of computing time to evaluate since the summation is from $-\infty$ to ∞. Therefore, an approximation is needed. That is, after the impulse response has declined a certain number of dB from the original value, we can approximate the response from then on is zero. Thus, the noise-equivalent bandwidth can be evaluated by sending a unit impulse function through the system to obtain the unit impulse response of the system. We can then using equation (3.3.7) to calculate the noise-equivalent bandwidth.

After the noise-equivalent bandwidth is found, we can then calculate the noise power and the noise standard deviation at the receiver. Thus the system error probability can be evaluated. For

example, with the system we used previously in an AWGN environment, the error probability can be easily evaluated using the Q-function.

$$P_e = \frac{1}{32}\sum_{i=1}^{32} Q\left(\frac{v_{k_i}}{\sigma_N}\right) \tag{3.3.8}$$

3.3.2 **Case Study**

Once again, we are going to use the simple BPSK system used in the previous section to demonstrate Quasianalytical method.

From the block diagram in figure 3.3, it is obvious that this BPSK is not a memoryless system due to the presence of the transmitting filter. Therefore, in order to get the right noiseless output waveform at the output, it is necessary to determine the memory of the system. Once we determined the memory of the system in term of bits, we can then pass all possible input sequence through the system to generate the proper noiseless output waveforms. Since an overestimation of the memory of the system will not affect the result of the simulation while an underestimate will cause an inaccurate estimation, estimating the system memory conservatively will guarantee the correctness of the simulation. However, if the system memory is estimated too conservatively, the maximal length PN sequence is going to become very long, thus increasing the simulation run-time. Therefore, it is also a trade off that try to estimate the system's memory correctly and at the same time maintain the simulation run-time to a minimum. The following graphs show the estimation of the memory of this BPSK system.

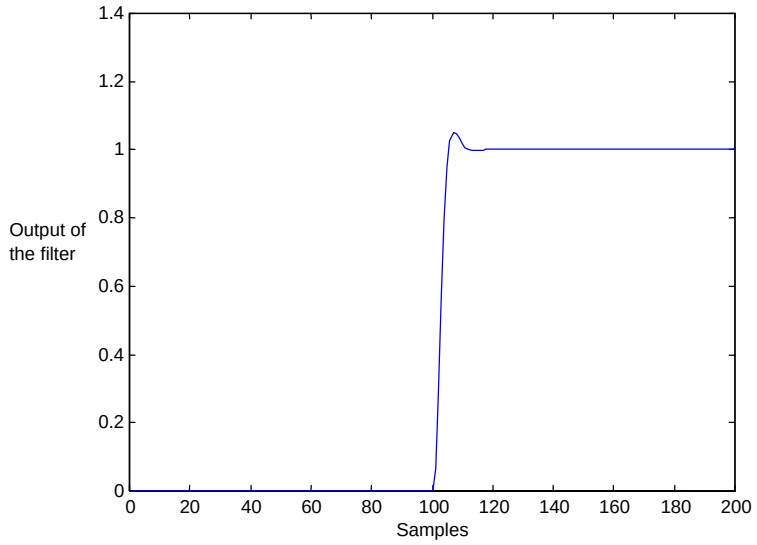


Fig. 3.21 ISI estimation of the transmitting filter

Since a step function is introduced at the 100th sample, the transient state lasts about 30 samples in this graph. Therefore, the ISI of the filter is about 3 bits. To ensure the correctness of the estimation, two more bits were added. This estimation will be used in the latter simulation.

After the memory of the system is determined, a maximum length PN sequence generator can be used to generate all possible input sequences. Since maximum length PN sequence generation are covered extensively in the literature, this thesis will not consider it in detail.

The above procedures in a sense provide us information about the reliability of the Quasianalytical estimator. Since the memory of the system is predetermined before any actual simulation was run, we could say that the estimator has no bias and zero variance. However if the estimation of the system memory is not precisely correct, the estimator then will be biased and it is not possible to quantify this bias in a general way.

After the generation of the noiseless output waveforms, the BER calculation is generally straightforward since the noise is additive and has a known pdf. In this case, the noise is AWGN, so the final calculation is just a simple Q-function evaluation. The BER curve then can be generated using the result from the calculation.

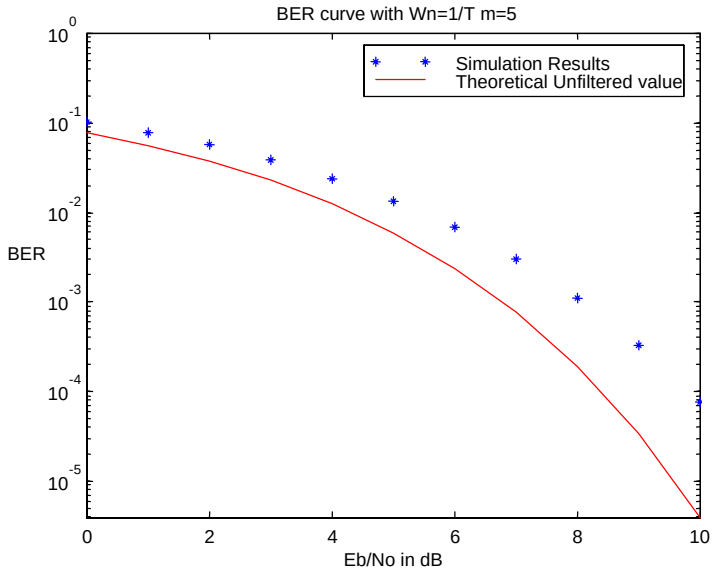


Fig. 3.22 BER curve simulated using Quasianalytical Technique

From the above graph, we can see that the result obtained using Quasianalytical technique agrees with the result from the Monte Carlo simulation.

To study more about the effect of the ISI on the system's performance, the bandwidth of the transmitting filter is increased and a new BER is plotted to show how the system's performance changes accordingly.

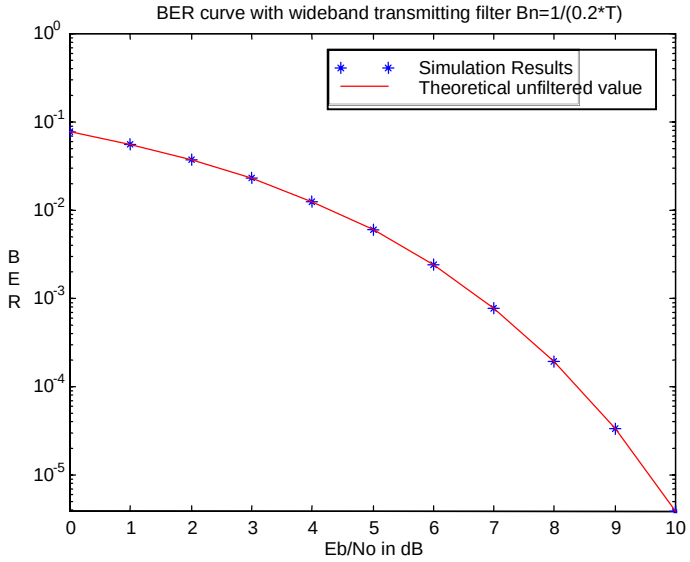


Fig. 3.23 BER curve with wideband transmitting filter

As it is shown in figure 3.23, when the bandwidth of the transmitting filter is increased so that there is little or no ISI, the system performance agrees with the theoretical unfiltered value very closely.

3.3.3 Discussion of the Quasianalytical simulation

There are many reasons that the Quasianalytical technique should be implemented. First and most important is its time saving property. In a linear channel, it can provide correct answers extremely fast. This may be very beneficial for simulations of low BER systems. It is also very useful for parametric studies that otherwise would be prohibited due to the extremely long computation time. Secondly, it can serve as a "sanity check" for the Monte Carlo method. As was mentioned in the previous section, the Monte Carlo method itself is not perfect. For example, random number generators are not ideal most of the time and are hard to check for long sequences. Therefore, a "sanity check" is necessary for Monte Carlo simulation results, and Quasianalytical method serves us well in this aspect.

However, Quasianalytical method is very problem specific and generally cannot be set up in advance to solve general problems [2]. On the other hand, Quasianalytical method is also unable to generate waveform level simulation product with the presence of noise, which sometimes might prohibit us to gain more insight of the behavior of the system.

Quasianalytical technique has lots of advantages and when used with combination with Monte Carlo method, they can provide very useful simulation results for analyzing and designing purposes. Therefore, there is no doubt that Quasianalytical technique should be implemented whenever is possible in simulation.

3.4 Conclusions

There are many choices in BER estimation techniques. The ones presented in this paper are just some of the simulation-based approaches for estimating the BER. Not all of these techniques apply

irrespective of the modulation/coding scheme and associated receiver structure. At the same time, all these techniques are not isolated from each other. Most times when combined with one another, the best simulation results can be reached. Therefore, one must consider which technique(s) should be implemented.

A. Monte Carlo method. Monte Carlo runs can be viewed as the "truest" situation for nonlinear system next to building the actual system. For this reason, each simulation package should include Monte Carlo capability, not only for BER estimation, but also calibrating and monitoring the fundamental behaviors of the system. For low BER systems, the simulation run-time could be unreasonably long and some sort of the alternative approach such as the Block Analysis method must be used. The implementation of the Monte Carl method is very simple and is independent of the specifics of the system. Therefore, it is possible for us to set up the basic structure in advance for general purposes and make minor modifications to accommodate each individual system.

B. The Quasianalytical technique. The biggest advantage of the Quasianalytical technique is time saving. In a linear channel situation, this technique can provide us the correct answer extremely fast. The speed of the Quasianalytical technique can also be used to perform many parametric studies which would be impossible otherwise. It also can serve as "sanity check" for Monte Carlo method due to the imperfection of the Monte Carlo method itself. However, the "analytical" part, which relates the simulated waveforms to the error rate, depends on the actual simulation method. Therefore, the implementation of the Quasianalytical technique is very much problem specific and generally cannot be set up in advance.

Chapter 4

Case Study

4.1 **Introduction**

In chapter 3, we studied some basic simulation approaches and methodologies. In this chapter, we present one more case study to further illustrate different ideas and techniques used in developing simulations.

We used a simple simulation of a BPSK communication system in the last chapter as an example. In this chapter, we are going to expand this example even further and construct a simulator for a general M-ary PSK system. The objective of this case study is to study and evaluate the error probability performance (BER) of the system in an AWGN environment, taking into account the ISI effect at the transmitter. At the same time, visual indicators of the system, such as eye diagram and signal constellation diagram, are also of strong interest to us in order to study the difference and similarity of the different level of PSK.

4.2 **Methods of Performance Evaluation Using Simulation**

Due to the combination of the arbitrary level of the signal alphabet and the ISI effect, a pure analytical solution to such a problem is somewhat tedious and involves a great deal of repetitive work. Moreover, such a solution does not provide easy access of the desired parametric study and evaluation. Simulation is therefore a more desirable option for such a task. However, a Monte Carlo simulation approach is very time consuming, especially when $\frac{E_b}{N_0}$ is large. On the other hand, a Quasianalytical approach has its limitation too. First of all, as the signal alphabet becomes large (8-ary, 16-ary and etc.), the length of the required sequence increases rapidly. Consequently this approach loses its time saving advantage quickly as M increases. Secondly, since the Quasianalytical

technique is very problem specific, modifications have to be made for each M. It is impossible to have a general simulator that uses M as an input parameter as in the case of the Monte Carlo approaches.

In order to reach our objective effectively, we need to apply a combination of different methodological approaches and simulation techniques. As mentioned above, the visual indicators such as eye-diagram and signal constellation diagram are not only important but also necessary for understanding and analyzing the system in the simulation. Therefore, a Monte Carlo simulator should be included to generate such diagrams. For the other facet of the performance, the BER estimation, there is no simple solution. Each case has to be dealt with differently. Although a Monte Carlo simulator is time consuming, it can provide us a general simulator for different levels of signal alphabet. On the other hand, the Quasianalytical technique is still very efficient for low value of M. Since all the modulation studied in this case belongs to the same family of modulation techniques, a low signal level system such as a QPSK system often behaves similarly as a system with higher signal level system such as 8-ary or 16-ary PSK system. Therefore, even though the Quasianalytical technique loses its advantage when M is large, it still can provide us great insight of the whole family of modulation techniques by simulating the low signal level system such as a QPSK system effectively. Hence, different simulation techniques need to be carefully chosen according to different problems.

4.3 Simulation of the M-ary PSK Signal

As mentioned before, both simulation approaches have their own advantages. Therefore, this simulator will have two parts: the Monte Carlo simulator and the Quasianalytical simulator. Thus, users can choose different simulator accordingly.

The Monte Carlo simulation procedure is as follows.

1. Generate a symbol s in the range of [1, M].
2. Generate random transmitter phase θ_i.

3. Construct the complex envelope of the signal by using equation (2.3.12) and (2.3.13)

$$x_d(t) = A\cos[\frac{2\pi}{M}(m-1) + \theta_i]$$
$$x_q(t) = A\sin[\frac{2\pi}{M}(m-1) + \theta_i]$$

where $x_d(t)$ and $x_q(t)$ are in-phase and quadrature components of the ith symbol. A is chosen to be 1 for convenience.

4. Sample the in-phase and quadrature components, $x_d(t)$ and $x_q(t)$ respectively, to generate the sampled signal $x_S(t)$.

5. Filter the in-phase and quadrature components of the sampled signal $x_S(t)$ to generate the transmitted signal $x_{tr}(t)$.

6. Generate noise according to the value of $\frac{E_b}{N_0}$, i.e. signal-to-noise ratio.

$$\frac{E_b}{N_0} = \frac{0.5 * A^2 T_b K}{\sigma^2 T_b} = \frac{K}{2 * \sigma^2} \tag{4.3.1}$$

where K is the sampling rate. Therefore, the standard deviation of the noise is

$$\sigma = \sqrt{\frac{1}{2 * \frac{E_b}{N_0}}} \tag{4.3.2}$$

The variation of $\frac{E_b}{N_0}$ is achieved by changing the noise variance and keeping the energy per bit, E_b, constant.

7. Construct the received signal by summing up the transmitted signal and the noise signal.

$$r_d(t) = x_{tr\,d}(t) + n_d(t) \tag{4.3.3}$$

$$r_q(t) = x_{tr\,q}(t) + n_q(t) \tag{4.3.4}$$

where $r_d(t)$ and $r_q(t)$ are the in-phase and quadrature components of the received signal $s(t)$.

8. Calculate the output of the matched filter.

$$output = \sum_{k=1}^{K} s(t) \tag{4.3.5}$$

where K is the sampling rate, which is 10 in this simulation.

9. Determine the received symbol in terms of maximum likelihood estimation.

$$s\_est = k$$

where

$$z_k = \max(z_i) \quad i = 1, 2, \ldots, M$$

10. Update the symbol error statistics.

The procedure for the Quasianalytical approach is as following.

1. Estimate the memory of the system.
2. Generate the maximum length sequence. If the system's memory is Q symbols, for M-ary PSK modulation, the length of the required sequence is $N = M^Q$.
3. Filter all the sequences to generate all possible transmitted sequences.
4. Calculate the output of the matched filter to obtain the noiseless result of the system.
5. Calculate the noise variance for different values of $\frac{E_b}{N_0}$.
6. Calculate the symbol error probability.

4.4 Simulation Results

Different simulation techniques are chosen for different value of M. When M=4 (QPSK system), the length of the required sequence is $M^Q = 4^5 = 1024$, assuming system's memory is five symbols.

This value is still reasonable small. Therefore, the Quasianalytical approach should be chosen to generate the symbol error probability curve. However, when $M \geq 8$, the length of the required sequence becomes very large. As a result, the computer run-time of a Quasianalytical simulator will increase dramatically too. Therefore, the Monte Carlo method should be used because it can provide us a general solution for all values of M.

4.4.1 QPSK System Simulation Results

The Quasianalytical technique is chosen to generate the symbol error probability for QPSK systems. The result is plotted in Figure 4.1.

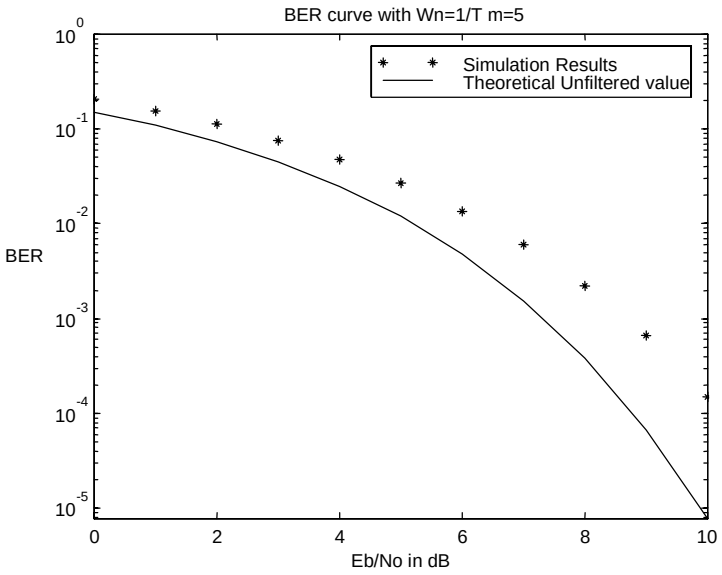


Fig. 4.1 Symbol Error Probability curve of QPSK system

The solid line is the theoretical value in an AWGN environment without the ISI effect. The difference between the simulated result and the theoretical unfiltered value clearly demonstrates the ISI effect on the system performance. To further demonstrate the ISI effect, the bandwidth of the transmitter filter is increased and another simulation is run to study how the performance of the system will change.

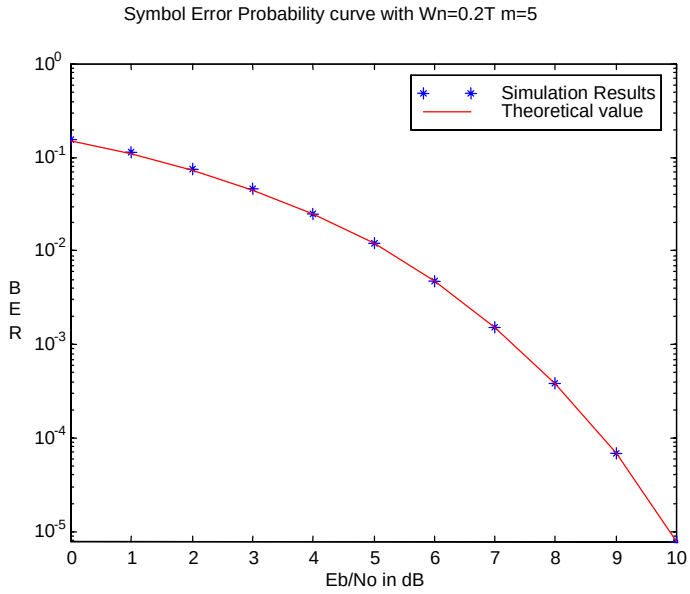


Fig. 4.2 Symbol Error Probability curve of QPSK system

It is clearly shown in Figure 4.2, when the bandwidth of the transmitter filter is increased so that the ISI effect can be neglected, the simulated result agrees with the theoretical value.

Besides the symbol error probability curve, other visual analysis tool such as eye diagrams and the signal constellation diagrams are also our interests. Therefore, the Monte Carlo simulator is used here to generate the desired diagrams.

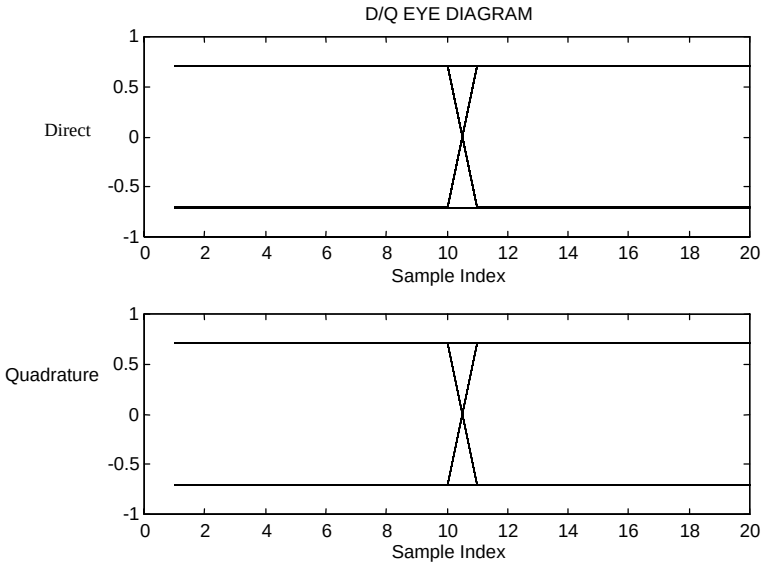


Fig. 4.3 Eye-diagram of the input of the transmitter filter

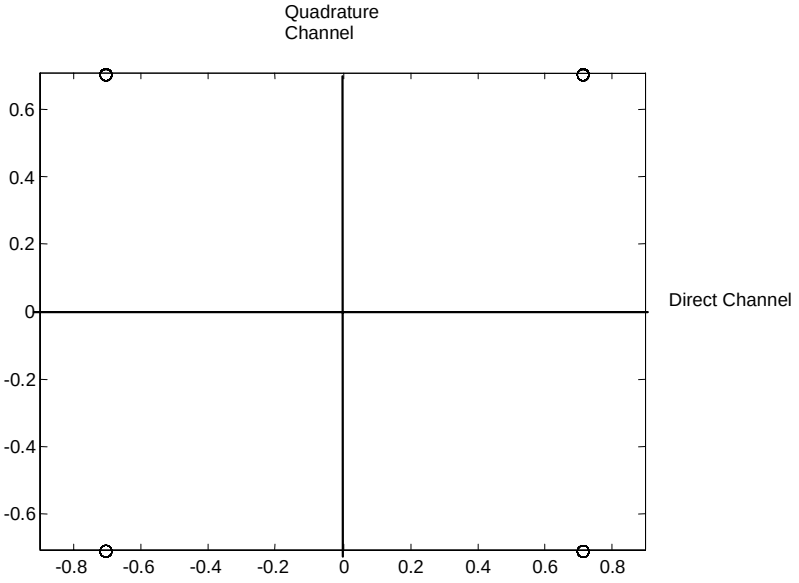


Fig. 4.4 Signal Constellation Diagram of the input of the transmitter filter

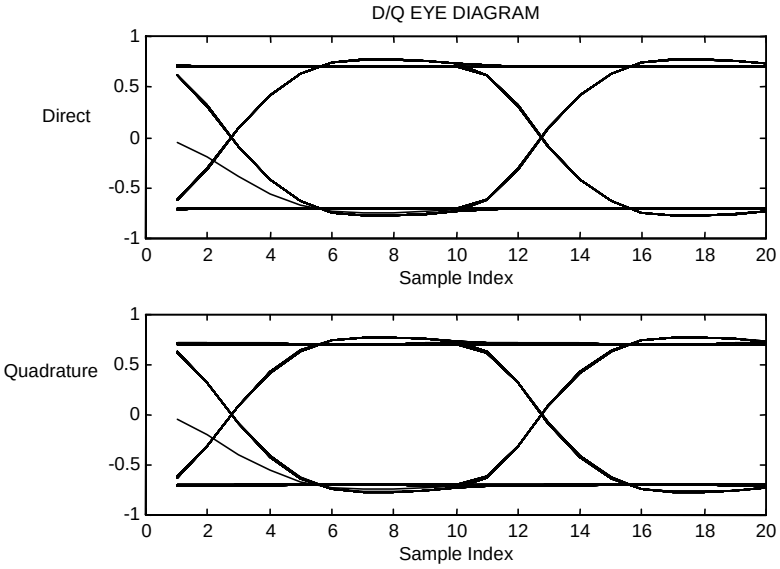


Fig. 4.5 Eye-diagram of the output of the transmitter filter

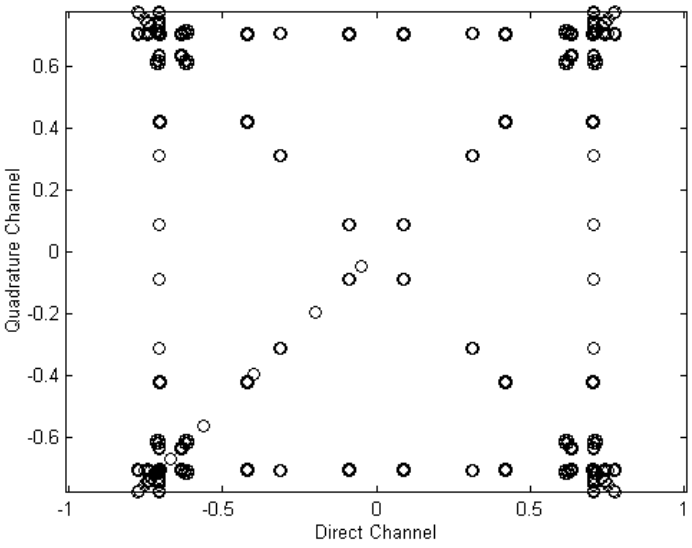


Fig. 4.6 Signal Constellation of the output of the transmitter filter

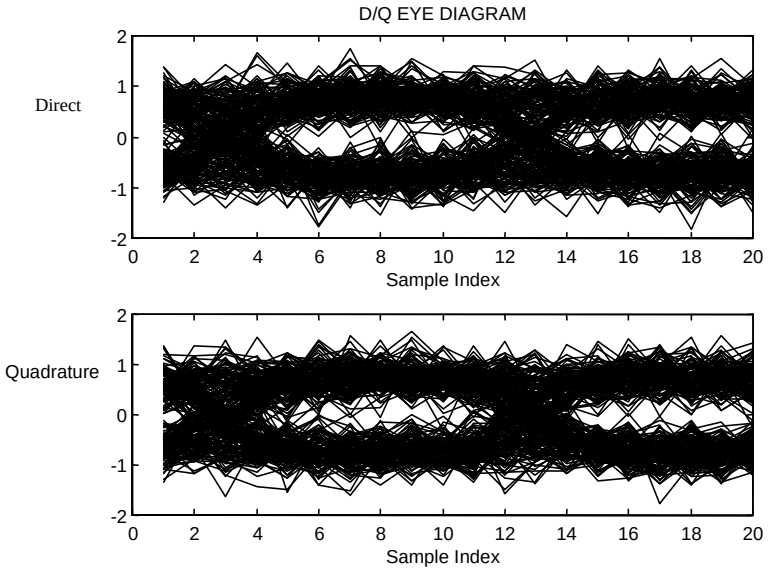


Fig. 4.7 Eye diagram of the input of the receiver (E_b/N_o=15 dB)

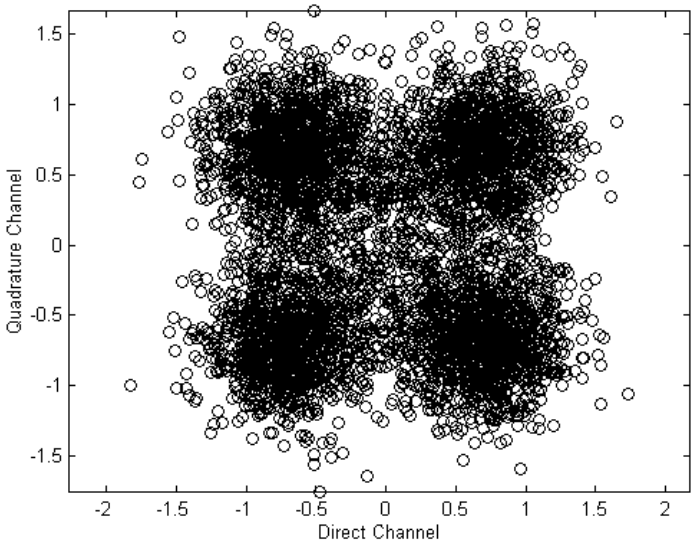


Fig. 4.8 Signal Constellation of the input of the receiver (E_b/N_o=15 dB)

The above diagrams clearly demonstrate the ISI effect of the filter and the effect of the noise at the different nodes of the system. The diagrams also show visually how the channel behaves in the time domain (eye diagram) and in the signal space (signal constellation diagram). Fig. 4.3 and Fig. 4.4 are the eye-diagram and the signal constellation diagram of the original transmitted signal respectively. It is clearly shown in Fig. 4.3 that the transition between the two states are straight transitions, which means that no distortion has been introduced at this point. Fig. 4.5 and Fig. 4.6 are the eye-diagram and signal constellation diagram of the signal after filtering. The ISI introduced by filtering causes the signal being dispersed in the signal space which is shown in Fig. 4.6. Hence, the state transition is no longer straight. It is also worth noticing that at the beginning of the time, there is a transition state coming out of zero state, which is not shown in Fig. 4.3. This is because that the filter is completely relaxed before any data is introduced. Fig. 4.7 and Fig. 4.8 are the eye-diagram and signal constellation diagram of the signal after it has been passed through an AWGN channel. In those two figures, the effect of the noise is clearly shown. In Fig. 4.7, it is shown that the "eye" has closed somewhat due to the presence of the noise. If E_b/N_o decreases more, the "eye" will eventually close completely. The similar effect is also shown in Fig. 4.8. The signal points are scattered all over the place instead of being dispersed in certain trajectories in Fig. 4.6. Similarly, when E_b/N_o decreases, the signal points will be scattered into even wider area. Combining all six figures, we can obtain a clear picture how the signal behaves at the various points of the system.

4.5 Conclusions

In this case study we had two objectives; the determination of the symbol error probability and the study of system behavior in an AWGN environment taking into account the ISI introduced by the transmitter filter. For different level of signal alphabet, different technique should be chosen to optimize the efficiency of the simulation process.

For $M = 4$, the length of the maximum length PN sequence required for the Quasianalytical technique is short. Therefore, we should choose the Quasianalytical technique to take full advantage

of its speed to generate the symbol error probability curve. For $M \geq 8$, the length of the maximum length PN sequence becomes long which prevents the Quasianalytical simulator from a rapid computation. On the other hand, the Monte Carlo method fits these situations perfectly. By using the Monte Carlo approach, it is possible for us to develop a single simulator that takes M as an input and generates all the desired diagrams. This is not possible for a Quasianalytical simulator because a separate maximum length PN sequence generator needs to be constructed for each value of M.

An important lesson given by this case study is that each new problem should be examined carefully for efficient means of solution. Any simulation approaches should not be accepted blindly, especially if they lead to large computation time.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We studied some basic simulation techniques in this paper. These techniques are primarily used for performance evaluation of different communication systems. They also can be used for configuring and executing waveform level simulations. All those capabilities provide us wonderful tools for analysis of proposed designs.

The Monte Carlo technique is very simple to understand and implement. It can be viewed as the "truest" situation next to building the actual system. It is also independent from the specific problem; therefore, it is possible for us to develop a general model and exchange different modules for the specific problem. However, despite of all the advantages that the Monte Carlo technique has, it is very inefficient. It has to wait for errors to happen, and thus it leads to large computational burdens in most cases.

The Quasianalytical technique's most overriding appeal is its speed. It enables us to simulate complex systems in a relatively short amount of time, and it also makes parametric study possible. Therefore, the Quasianalytical technique should be used whenever is possible. However, the Quasianalytical technique can not be used on the systems and under all the situations. There are constraints for the Quasianalytical technique such as the distribution of the noise has to be known and the noise has to be additive. At the same time, in order to take the full advantage of the time saving property of the Quasianalytical technique, the system memory has to be relatively short and the signal alphabet has to be moderately small. The Quasianalytical technique is also very problem specific so it is generally impossible to be set up in advance.

All the simulation techniques should not be isolated from each other because in most cases, a combination of the different techniques provides the best solution for the problem. Therefore, one must treat each new problem with a fresh eye for the most efficient means of solution. No technique should be the "universal" or "standard" technique.

5.2 **Future Work**

Simulation has been widely used in the communication industry as an analysis and design tool for the past decade. However, it is also clear that each organization has developed its own simulation package and there is no industry-wide standard, such as the SPICE package used for circuit simulation. In the past few years, MATLAB has been adapted widely to develop simulation for various communication systems. In times to come, more complete simulation packages, providing color graphics input-output, symbolic debugging, with a variety of data display capabilities ranging from tables to charts to graphs available singly or in combinations, will be available.

In addition to the development of software simulation package, efforts are also underway towards the development of firmware simulation systems using the hybrid hardware/software computing approach. The application of fast digital signal processors to moderate data rates will bring us to the stage where the software simulation models of receivers and transmitters will be used as is in a firmware implementation of the actual systems.

The challenge we are facing today is to use computer-aided design tools to get on with the revolution that is upon us all in information transmission and processing.

Appendix A.

Basic Structure of a Simulation

It is very helpful to construct a simulation in a manner that is flexible to use and easy to maintain. In most simulations, the input and output is usually a very large portion of the whole program. Therefore, if we separate the input and output from the main program body, the simulation program will be much easier to maintain and understand. For this reason, the simulation should be divided into three parts: preprocessor, simulator, and postprocessor.

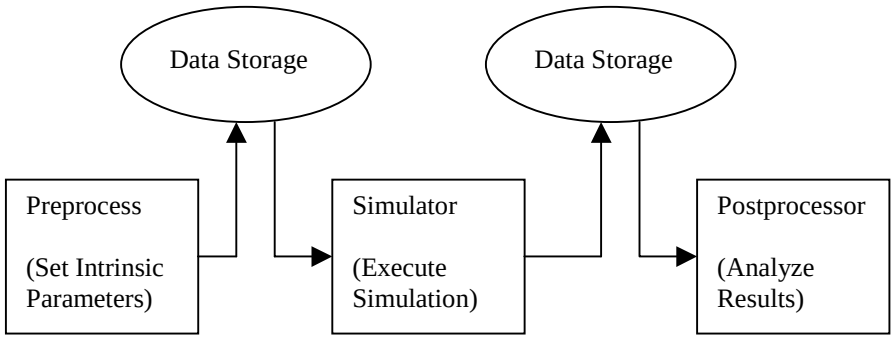


Fig. A.1 Basic Simulation Architecture

Preprocessor is the initialization part of the program. It reads all the user defined intrinsic system parameters and stores the data in data storage which can either be in the computer's memory or in a file. This is the stage in which users define the system. Simulator is the main body of the simulation. It reads all the data from the data storage and executes the simulation. At the end, simulator passes all the end results into the data storage waiting for postprocessor to analyze them. Postprocessor is the output stage of the program. It reads all the data, such as error statistic, generated by simulator and outputs them visually, such as BER curves and signal constellations. Postprocessor can also produce some kind of user interface that allows users to choose the desired diagrams. The following figures are the preprocessor and postprocessor used for the BPSK simulation example.

```
EDU» initial
Input "B" for BER calculation or "W" to generate all other diagrams ==> w
The period of each information bit (0.001) ==> 0.001
The number of samples per bit (10) ==> 10
The phase angle of Transmitter (pi/8) ==> pi/4
The order of the filter (2) ==> 2
The total number of signal needs to be generated (500) ==> 500
The desired Eb/No value (in dB) to generate all the waveforms (15dB) ==> 15
Choose one or more points of the system where you would like to see eye diagrams and
signal constellation diagrams
(i.e. MFD to choose all)
M ==> at modulator output (without filtering)
F ==> at transmitter filter output
D ==> at detector (integrate and dump) input
mfd
```

Fig. A.2 Preprocessor for the BPSK system

The first input of this preprocessor is the choice between the BER curve and other diagrams. Since we don't need as many runs to generate eye diagram and signal constellations, this input parameter allows us to generate eye diagrams and signal constellations much quicker. The following parameters are users defined intrinsic parameters. The last input of this preprocessor is where user would like to see the eye diagram and signal constellations. This allows user to insert probe at different node of the system to monitor the behavior of the system at different stages.

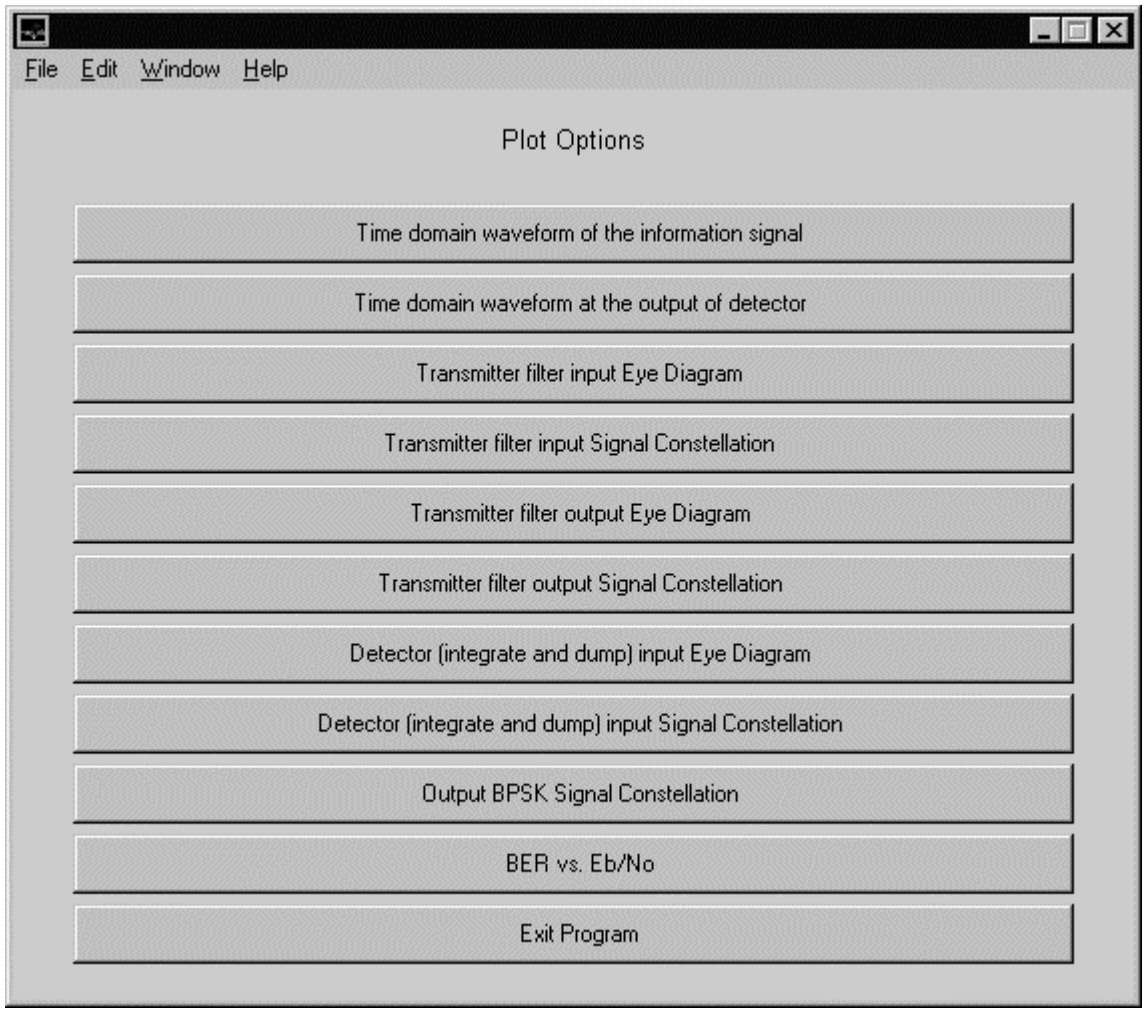


Fig. A.3 Postprocessor of BPSK simulation

Figure A.3 is the user interface created by the postprocessor. User is able to choose different output very easily.

Another advantage to separate the preprocessor, simulator, and postprocessor is flexibility. In most cases, preprocessor and postprocessor are very similar for all the different communication systems. Therefore, only little modifications are needed for different simulation. This can be very convenient. At the same time, by using this architecture, we are able to take full advantage of different software packages. For example, Matlab is very powerful for graphic output and is slow at executing speed. Therefore, we can use Matlab to construct preprocessor and postprocessor to take full advantage of its graphic output capability while using C or other computer languages to construct simulator to increase the speed of program, and thus achieve the optimal simulation performance.

Appendix B.

Random Data Generator

Any simulation involving one or more stochastic process requires the generation of random numbers. In the previous examples, we used the built-in *rand* function in Matlab to achieve such a task. However, the distribution of the generated data sequence is unverified and in most cases, it is very difficult to verify the distribution of the final data sequence. Therefore, it will be helpful if we can construct our own random data generator with know period and known distribution.

The most important and basic data generator is the uniform distributed data generator. The method of linear congruence is the most popular method of generating random data with uniform distribution. The random sequence is obtain by the recursion relationship

$$x_{N+1} = (Ax_N + c) \bmod m \qquad \text{(B.1)}$$

where m is the modulus, A is the multiplier, c is the increment, and x_0 is the initial value. The question now is how to choose $m, A, c,$ and x_0. Clearly if x_N is an integer, there are only m different values which x_N can take on. Thus, the best we can do is to make the sequence periodic with period m. Therefore, the parameter m should be chosen according to two considerations:

1. We wish a long period (m larger)
2. We wish to generate the values x_N quickly.

As a result, a common choice is to let $m = 2^E$ where we assume an E bit computer. The increment, c, is often set equal to zero. With $c = 0$ the random number can be generated slightly faster. However, maximum length cannot be achieved with $c = 0$.

The most general theorem to generate a maximum length data sequence is

1. c and m are relative prime (no common prime factors).
2. $A-1$ is a multiple of every prime p which divides m.

3. $A-1$ is a multiple of 4 if 4 divides m.

Proof: Knuth [7]

A nice consequence from this theorem:

1. c is odd.
2. $A = 4k+1$ where k is integer.
3. $m = 2^n$.

The proof is simple:

1. The only prime factor for m is 2, and if c is odd, it is obvious that m and c don't share any common prime factor. Therefore, they are relatively prime.
2. $A-1 = 4k$, which is clearly a multiple of 2.
3. $A-1 = 4k$, which is also a multiple of 4.

If we divide all x_N by m, we have a uniform random generator between 0 and 1.

Now we can generate the random variable with general pdf using the uniform random generator. If we want to generate a random variable with a pdf of

$$f_X(x) = \begin{cases} c_i & x_{i-1} \le x \le x_i \quad i = 1,2,\ldots,n \\ 0 & otherwise \end{cases} \tag{B.2}$$

P_i and F_i are defined as

$$P_i = \int_{x_{i-1}}^{x_i} f_X(x)dx, \quad i = 1,2,\ldots,n \tag{B.3}$$

$$F_i = \sum_{j=1}^{i} P_j = \sum_{j=1}^{i} \int_{j-1}^{j} f_X(x)dx \tag{B.4}$$

The procedure to generate such a random variable is as following:

1. Generate a uniformly distributed random variable U from U(0,1)

2. Find i from

$$F_{i-1} = \sum_{j=1}^{i-1} P_j < U \leq \sum_{j=1}^{i} P_j = F_i \qquad \text{(B.5)}$$

3. $X \leftarrow x_{i-1} + \frac{1}{c_i}(U - F_{i-1})$

4. Deliver X

Appendix C.

Sequential Filter Function in Matlab

In the previous chapters, we used filter function repeatedly. However, the *filter* command in Matlab is a block operation (details see Matlab manual). When we need to run a simulation in a sample by sample fashion, some other type of filtering operation needs to be used. Therefore, we constructed sequential filter function to perform such tasks.

It is known to us that any filter with a transfer function as

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-1} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}} \qquad \text{(C.1)}$$

can be expressed with the following diagram using Transposed Direct Form II Network Structure.

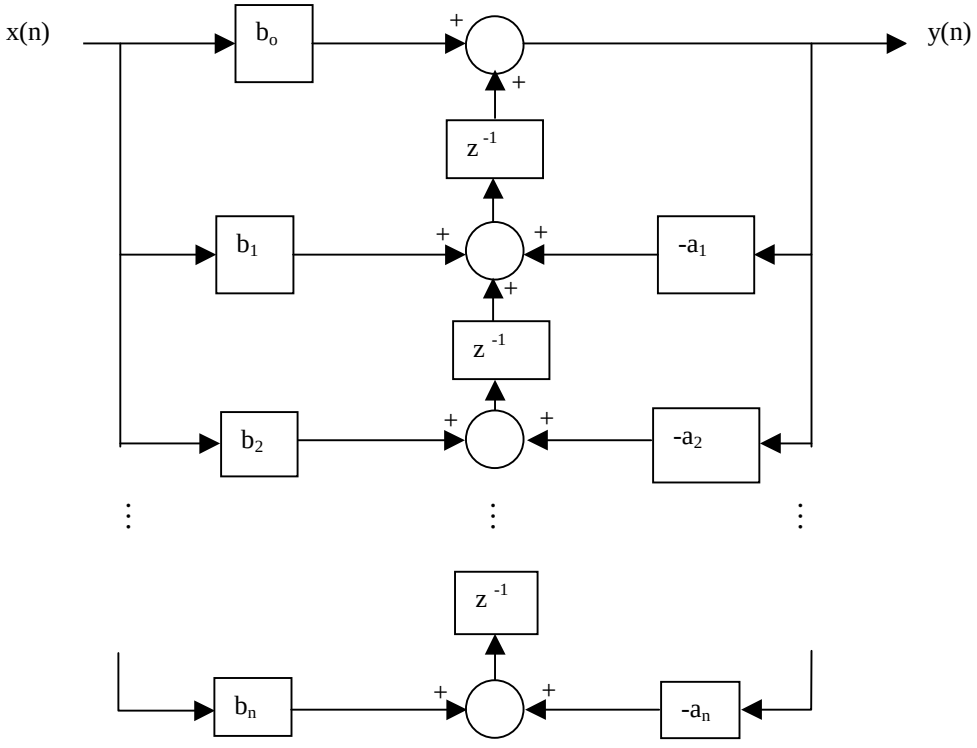


Fig. C.1 n-th order Transposed Direct Form II Network

According to the z-transform property, each z^{-1} block in the above diagram is equivalent to an time delay of one sample in time domain. Therefore, the above diagram can be simplified as a simple shift register with $n+1$ states. That's how we are going to construct our sequential filter function. The following is the Matlab code for this function. It is very simple and self-explaining.

```
function [out,sreg]=filter_seq(A,B,n,sreg1,in)

% Filter that filters data sequentially
%
% A, B & sreg1 are 1X(n+1) matrixs
% Copyright Dr. William H. Tranter

global out;
global sreg;

out=B(1)*in+sreg1(1,1);    % Calculate the output of the filter
sreg1=sreg1+B*in-A*out;    % Calculate the new states of the shift
register
sreg=[sreg1(2:n+1),0];     % Shift the new states to prepare for next
sample
```

Reference

1. Rodger E. Ziemer, William H. Tranter, and D. Ronald Fannin, Signal & Systems: Continuous and Discrete, Prentice Hall

2. Michel C. Jeruchim, Philip Balaban, and K. Sam Shanmugan, Simulation of Communication Systems, Plenum Press, New York and London

3. John G. Proakis, Digital Communications, McGraw-Hill Inc. 1995

4. Leon W. Couch II, Digital and Analog Communication Systems, 4th Edition, Macmillan Publishing Company, 1993.

5. A. M. Gardner, Common pitfalls in the application of stationary process theory to time-sampled and modulated signals, IEEE Trans. Commun. **COM-35**(5), 529-534 (1987).

6. M.C. Jeruchim, Techniques for estimating the bit error rate in the simulation of digital communication systems, IEEE J. Selected Areas Commun. **SAC-2**(1), 153-170 (1984)

7. R. E. Ziemer, and W. H. Tranter, Systems, Modulation, and Noise, Houghton Mifflin Company, 1976

8. W. H. Tranter, Class Notes for Simulation of Communication Systems