8-1-1991

Efficient Importance sampling Simulations for Digital Communication Systems

Jyun-Cheng Chen
Purdue University

EFFICIENT IMPORTANCE SAMPLING SIMULATIONS FOR DIGITAL COMMUNICATION SYSTEMS

A Thesis
Submitted to the Faculty

of

Purdue University

by

Jyun-cheng Chen

In Partial Fulfillment of the
Requirements for the Degree

of

Doctor of Philosophy

August 1991

To my parents, my wife Wen-chi and my son Eric.

ACKNOWLEDGMENTS

I would like to thank Prof. John S. Sadowsky, my academic advisor, who has been very supportive throughout my graduate study at Purdue. He is the major force behind the development of my interest in and understanding of digital communications. He is also the nicest tough advisor one can get.

My gratitude also goes to Professors Saul Gelfand, Tony Hsiao, Edward Delp, Michael Fitz, Philip Protter and Bruce Schmeiser at Purdue, and Professor Kung Yao at the UCLA for their guidance, encouragement and assistance. I am also grateful to Carey Parker who proofread several chapters of this thesis. Many of the computer simulation jobs cannot be completed in time without the help of George Goble of the Purdue Engineering Computing Networks.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

APPENDICES

Figure

ABSTRACT

Chen, Jyun-cheng. Ph.D., Purdue University, August 1991. Efficient Importance Sampling Simulations for Digital Communication Systems. Major Professor: John S. Sadowsky.

Importance sampling is a modified Monte Carlo simulation technique which can dramatically reduce the computational cost of the Monte Carlo method. A complete development is presented for its use in the estimation of bit error rates P_b for digital communication systems with small Gaussian noise inputs. Emphasis is on the optimal mean-translation Gaussian simulation density function design and the event simulation method as applied to systems which employ quasi-regular trellis codes. These codes include the convolutional codes and many TCM (Ungerboeck) codes. Euclidean distance information of a code is utilized to facilitate the simulation. Also, the conditional importance sampling technique is presented which can handle many non-Gaussian system inputs. Theories as well as numerical examples are given. In particular, we study the simulations of an uncoded MSK and a trellis-coded 8-PSK transmissions over a general bandlimited nonlinear satellite channel model. Our algorithms are shown to be very efficient at low P_b compared to the ordinary Monte Carlo method. Many techniques we have developed are applicable to other system simulations as building blocks for their particular system configurations and channels.

1. INTRODUCTION

1.1 Problems and Goals

Consider the conceptual block diagram Figure 1.1 of a communication system. The transmitted signal and the corrupting noise are input to the system which then decides what signal has been sent. (There may be more than one noise process, and these noise processes may enter the system at different points.) In digital communications, the transmitted signal represents an information bit stream. The system will make incorrect decisions on some bits along the way because of the random noise. We are most interested in the average probability of a transmitted bit being received in error, P_b, or the bit error rate (BER). This is a universal performance criterion for digital communication systems. P_b depends on the statistics of the input noise process(es) which are assumed known to us.

For simple system models, analytical methods can be used to derive an exact expression for P_b. Complex systems, especially nonlinear, however, are often mathematically intractable. For example, analyzing a digital satellite link with bandlimiting filters and nonlinear amplifiers is tedious, if not impossible. Many coding and

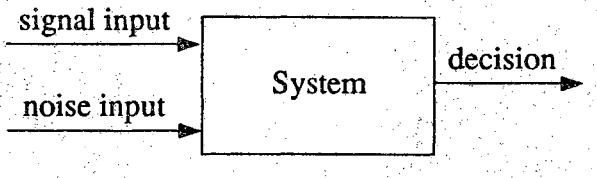


Figure 1.1 A conceptual block diagram of communication systems.

modulation schemes are also difficult to analyze even under simplified channel model assumptions.

The Monte Carlo simulation [5, 24, 45] is a popular alternative. The key advantage of the Monte Carlo method is that it can incorporate many system degradation factors simultaneously. But this generality comes with a price. Being an empirical estimator, the required computations increase with $1/P_b$. Consequently, Monte Carlo simulation of $P_b < 10^{-5}$ is generally considered infeasible because of excessive computer run time.

Some variance reduction schemes of the Monte Carlo method have been proposed to speed up the simulation, such as, extreme value theory (e.g., [69]), tail extrapolation (e.g., [68]), semi-analytical (e.g., [41]) and importance sampling [13, 25, 28, 32, 36, 37, 39, 40, 46, 48, 49, 53, 70]. The importance sampling (IS) method is more attractive than the others because IS retains the nice properties of Monte Carlo method (unbiasedness, simulating unapproximated complex system models) and its efficiency gain (over the Monte Carlo method) is more promising. In addition to estimating P_b, importance sampling has found applications in many other areas of communications, e.g., estimations of the false alarm probability in radar systems [33] and the time delay/blocking probability for digital packet/circuit switching networks [42, 48].

Shanmugan and Balaban's paper in 1980 [53] was the first one to use importance sampling to estimate P_b for communication systems with small Gaussian noise inputs. They demonstrated significant lowering of computations, not increasing with $1/P_b$, in simulating uncoded systems operating on linear memoryless channels. Since then, most work in the literature has concentrated on extending their concept to the more practical channel models, in particular channels with memory, and has achieved various degrees of success. However, optimization has been performed mostly for linear channels. For nonlinear channels, efficiency gain falls somewhat short of the theoretical limit. Also, the majority of the previous work has been for uncoded systems and Gaussian noise inputs alone.

This thesis intends to give a comprehensive development of importance sampling techniques for simulations of digital communication systems. Our primary interest is

in the trellis-coded systems. (Uncoded systems will be shown to be a special case.) The channel models will include Gaussian noise, nonlinearity and intersymbol interference (ISI). Non-Gaussian system disturbances such as synchronization error and/or phase jitter, cross-channel and/or co-channel interference and fading may also be considered. Therefore, the techniques presented in this thesis may be applied to many practical channels including telephone channels, terrestrial microwave links and satellite channels. In particular, we will use a satellite channel model to illustrate many of these techniques. Our purpose is to demonstrate that efficient importance sampling simulations do exist for more practical and complex systems. We will not be able, however, to present a "universal" importance sampling algorithm that works for many systems. Instead, the fundamental principles we have developed can be used as "building blocks" for construction of various special purpose algorithms. In fact, it will become clear later that efficient importance sampling algorithms use extensive side information of the systems they are simulating. Therefore, these importance sampling schemes are strongly system-dependent. In general, there is a tradeoff between the complexity and efficiency of the simulation algorithm.

We now specify our problems and goals more precisely. Trellis codes have been widely used in digital communication systems due to their efficiency in utilizing the power and bandwidth resources. This class of codes includes the well-known convolutional codes and the Trellis-Coded Modulation (TCM) codes. They can be fully described by their trellis diagrams, and the Viterbi decoder is the maximum likelihood (ML) decoder for memoryless channels. Let us first look at how conventional simulation is performed for trellis-coded systems without importance sampling.

Examples of ordinary Monte Carlo simulation of trellis-coded systems can be found in Heller and Jacobs [26]. The simulation is virtually a computer duplication of the actual system operation. A continuous stream of data samples, noise-corrupted and system-transformed, are input to the decoder. P_b is estimated by the relative frequency of erroneously decoded bits. We call this the "stream simulation" method. There are two problems associated with this conventional simulation. Firstly, the

occurrences of bit errors under normal operating conditions are rare events which may further be corrected by the decoder. Consequently, a large sample size and long computer run time are required to estimate a small P_b. Secondly, because of the memory inherent from channel encoding, stream simulation generates a sequence of correlated Viterbi decoder branching decisions even if the channel is memoryless. As we will see, this makes the design of a good importance sampling scheme difficult. Also, it is not an easy task to estimate the variance of the estimator (which is used to assess the quality of an estimate) if the decisions are correlated.

Importance sampling provides a solution to the first problem. The basic idea of importance sampling is rather simple. Since a Monte Carlo estimator for P_b is basically an error counter, efficiency may be enhanced by artificially increasing the number of errors in a controlled way. This can be accomplished by biasing the probability density function (pdf) of the input random variables, i.e., we use a "simulation density" which is different from the true "model density." Each error count then is "unbiased" or weighted (to offset its significance) in such a way that yields an unbiased estimator. For example, if the relative frequency of an error event is increased by a factor of 10, then the average (or expected) weighting factor should be 1/10.

The main issue in importance sampling is the selection of simulation density function which is a key factor in determining the simulation algorithm's efficiency. (The efficiency of a simulation algorithm is measured by the estimator's variance or ultimately the CPU time the algorithm takes to achieve a certain degree of accuracy.) Until today, most work in the design of importance sampling for the estimation of P_b for systems with Gaussian noise inputs has centered around two distinct approaches: 1) variance scaling, i.e., increasing the noise power, and 2) mean translation of the probability distributions of the input noise samples.

The mean-translation biasing technique, which is the one we promote here, was first presented by Lu and Yao [36] for uncoded systems and linear channels. The mean of each input Gaussian noise sample is shifted (translated) to a "simulation mean" which can be found by an optimization procedure. We will see in Chapter 4

that this biasing technique is more efficient than the variance-scaling biasing. In addition, instead of the stream simulation method commonly being used for uncoded systems, Lu and Yao also employ what we will call the "event simulation" method. Each simulation run is directed to decide whether or not the event "test bit error" has occurred. The simulation density function can then be optimized only within the scope of each simulation run and the decisions in successive runs are independent. By their approach, they achieve efficiency gains that are previously considered not possible for systems with memory [13, 25]. Event simulation also greatly eases the evaluation of estimator variance. We will use a simple example in the next section to illustrate the basic idea of event simulation.

Event simulation with mean-translation biasing for convolutional codes was developed by Sadowsky [46]. The bridge allowing this crossover from uncoded to trellis-coded systems as well as a solution to the aforementioned second problem is the event simulation method for Viterbi decoders introduced in that paper. This method effectively decomposes the Viterbi decoder simulation into simulating independent error events. An optimal mean-translation simulation density then is designed for each specific error event pattern and the decoder also makes independent decisions — whether or not the "specific error event" has occurred, in consecutive runs. An additional advantage of event simulation for coded systems is that the code distance information can be utilized to facilitate the simulation. The channel model considered in [46] is linear memoryless.

Lu and Yao's mean-translation biasing can be optimized only for linear and linearized moderately nonlinear channels. A different approach is taken by Sadowsky and Bucklew [49] who use an asymptotical efficiency criterion and a large deviations theory argument to arrive at the asymptotically optimal mean translation for essentially arbitrary nonlinear systems with small Gaussian noise inputs. For the special case of linear channels, their result agrees with Lu and Yao's. Their work is in fact very general which is not restricted to the BER estimation problem, and it does not require the noise process to be Gaussian. We will present in Chapter 4 the theoretical

aspects of importance sampling which results in the same optimal Gaussian simulation density design. Our arguments are not from the large deviations theory point of view but the ideas are similar.

The mean-translation biasing, event simulation method along with the "conditional importance sampling" (briefly touched in [46] and [49]) constitute the fundamental principles of our research. Conditional importance sampling is a very powerful technique which provides us a convenient tool to handle many non-Gaussian system inputs. In this thesis, we will give a systematic presentation of these three basic techniques. We will also demonstrate via examples how to apply them to construct procedures for simulations of uncoded and trellis-coded systems operating on linear and nonlinear channels. A satellite channel model will be our example of nonlinear channel where nonlinearity is present due to the nonlinear traveling wave tube amplifier (TWTA). Finding the optimal Gaussian mean-translation simulation density for nonlinear systems is considerably more complicated than for linear systems. We will show how to simulate linear convolutional codes and nonlinear quasi-regular TCM codes using the event simulation method and conditional importance sampling. Nonlinear codes present additional complexity in both analysis and simulation because we cannot assume a particular codeword is transmitted, and the error probabilities must be averaged over all possible codewords.

This thesis is organized as follows. Chapter 2 introduces the basics of importance sampling under the general framework of estimating any decision error probability. We will also review other works in applying importance sampling to the estimation of BER in digital communications and comment on their advantages/disadvantages. Chapter 3 is an overview of some background of digital communication systems needed in reading this thesis. Chapter 4 is a presentation of the theoretical aspects of importance sampling which leads to the optimal Gaussian density design for general nonlinear channels. The treatment will be different from that of Sadowsky and Bucklew [49] but the ideas are similar. New results will be presented which can identify precisely

what is meant by "moderately nonlinear" channel. The conditional importance sampling technique will be also introduced. Event simulation methods for uncoded and trellis-coded systems will be discussed in Chapter 5 along with two linear, memoryless channel simulation examples. Chapter 6 uses both uncoded and trellis-coded system examples to demonstrate the implementation of mean-translation biasing, event simulation and conditional importance sampling for simulations of nonlinear channels. Numerical data will be given in Chapter 5 and 6 to show the power of our algorithms. The final chapter contains the concluding remarks.

Hopefully, this thesis can serve as an up-to-date summary of the mean-translation biasing with event simulation and conditional importance sampling approach in importance sampling research. It is possible that even more powerful simulation schemes will be found in the future. We hope that this thesis is a starting ground to that road for interested readers. Part of this thesis has been published in [9, 10].

1.2 Event Simulations vs. Stream Simulations

We devote this section to a comparison between the event simulation and the stream simulation. The event simulation method is philosophically different from the conventional stream simulation method and is vital to our work. Its importance warrants a simple example in this introductory chapter to illustrate the difference between these two methods. A more general description of event simulation and the event simulation method for trellis codes will be presented in Chapter 4. Consider an uncoded binary baseband transmission over a linear channel with memory as shown in Figure 1.2. Since this is a linear system, one sample per signaling interval T is sufficient for both signal and noise. Let X_k and Y_k denote the signal and noise samples respectively during time interval $[(k-1)T, kT], k \geq 0$. We call X_k ($X_k = \pm 1$) a "test bit" because it represents a transmitted information bit "1" or "0". The memory length of the system is assumed to be T. The block labeled "D" is the unit delay operator and λ is the linear system coefficient determined by the system finite impulse response (FIR) function. The test statistic R_k at the demodulator output can be

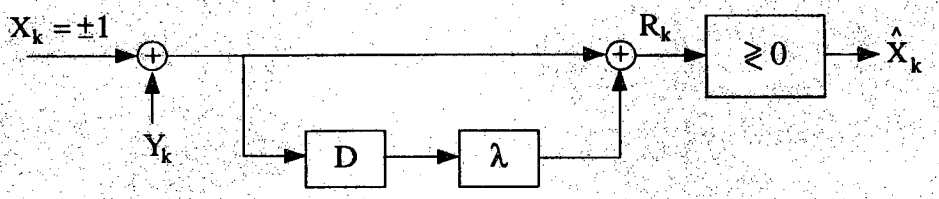


Figure 1.2 A simple system to compare stream and event simulations.

expressed as $R_k = X_k + Y_k + \lambda(X_{k-1} + Y_{k-1}) = \xi(X_{k-1}, X_k, Y_{k-1}, Y_k)$. The decision device decides on $\hat{X}_k = 1$ if $R_k \geq 0$ and $\hat{X}_k = -1$ if $R_k < 0$. Note that the output decision during the time interval $[(k-1)T, kT]$ depends not only on the current signal input X_k but also the previous input X_{k-1} which is known as the intersymbol interference.

The conventional stream simulation method is straightforward. It generates a sequence of random test bits and noise samples — $\{X_k\}_{k=0}^{L}$ and $\{Y_k\}_{k=0}^{L}$, computes $\{R_k\}_{k=1}^{L}$, and counts the number of decisions $\hat{X}_k \neq X_k$. This procedure is analogous to the actual system operation where we have a continuous stream of input bits and output bit estimates. If n out of L test bits are in error, P_b is estimated as n/L. In mathematical form, the empirical estimator for P_b is

$$\hat{P}_b^{(st)} = \frac{1}{L} \sum_{k=1}^{L} 1_{[0,\infty)}(-X_k R_k), \tag{1.1}$$

where the superscript (st) denotes stream simulation and $1(\cdot)$ is the indicator function defined as

$$1_A(Y) = \begin{cases} 1, & Y \in A; \\ 0, & \text{otherwise.} \end{cases} \tag{1.2}$$

Note that $\mathrm{E}[\hat{P}_b^{(st)}] = P(\hat{X}_k \neq X_k) = P_b$ for all k.

In the event simulation, instead of generating L test bits and noise samples and sequentially testing each one of them, we fix one test bit but test it L times. For example, suppose that we fix $X_1 = -1$ and define a random vector $\mathbf{Z} = (X_0, -1, Y_0, Y_1)$,

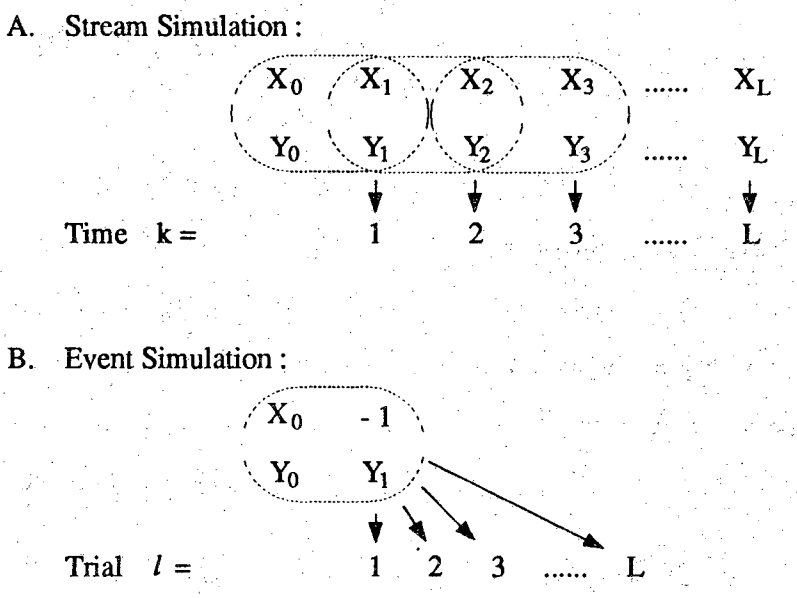


Figure 1.3 Event simulation vs. stream simulation.

then $\hat{X}_1 = 1$ if and only if $\xi(\mathbf{Z}) = -1 + Y_1 + \lambda(X_0 + Y_0) \geq 0$. The event simulation, no longer an emulation of the system operation, is purely a statistical estimator of the unknown quantity $P(\xi(\mathbf{Z}) \geq 0) = \mathrm{E}[1_{[0,\infty)}(\xi(\mathbf{Z}))]$. It generates L sets of random samples $\mathbf{Z}^{(\ell)}, \ell = 1, \ldots, L$, computes $\xi(\mathbf{Z}^{(\ell)})$, and counts the number of error events $\{\xi(\mathbf{Z}^{(\ell)}) \geq 0\}$. If n out of L trials result in the desired event, P_b is estimated as n/L. The estimator can be expressed as

$$\hat{P}_b^{(ev)} = \frac{1}{L} \sum_{\ell=1}^{L} 1_{[0,\infty)}(\xi(\mathbf{Z}^{(\ell)})), \tag{1.3}$$

where we have intentionally used the intermediate variable l instead of k to indicate that it is no longer a time index. The superscript (ev) denotes event simulation. Figure 1.3 is a graphical illustration of the two simulation methods where each dashed circle indicates the set of random variables that each decision is based on. Notice the overlapping of random variables in successive runs of stream simulation.

Note that $\mathrm{E}[\hat{P}_b^{(ev)}] = P(\xi(\mathbf{Z}) \geq 0) = P(\hat{X}_1 \neq X_1) = P_b$. We also notice that in event simulation each decision requires the generation of more samples than stream

simulation (3 samples versus 2 samples). As the system memory increases, the ratio increases linearly.

This example clearly demonstrates the fundamental difference between the two methods. Stream simulation can be viewed as an operation on the time axis while event simulation operates entirely on some probability space. Obviously, in simulating systems with memory, event simulation has the advantage that the terms in (1.3), $1_{[0,\infty)}(\xi(\mathbf{Z}^{(\ell)})), \ell = 1, \ldots, L$, are independent because they use different sets of random variables whereas in (1.1) R_k and thus $1_{[0,\infty)}(-X_k R_k), k = 1, \ldots, L$, are correlated. The estimator variance of $\hat{P}_b^{(ev)}$ can be easily estimated by a sample variance estimator while it is not so for $\hat{P}_b^{(st)}$. In addition, although event simulation is computationally more costly for each run, its overall cost can be much lower than the stream simulation when an efficient importance sampling scheme is applied. Lu and Yao [36] and Sadowsky [46] have clearly demonstrated this in their papers. This can be explained as follows.

In a heuristic sense, the simulation density is simply easier to optimize for each independent run in event simulation. In stream simulation, one particular noise sample's impact on consecutive decisions varies with time, hence its optimal biasing will be different for different test bits. This can create a biasing conflict. For example, suppose that $X_1 = -1$ and $X_2 = 1$. With mean translation, the mean of Y_1 should be positive in order to cause an error on the $k = 1$ decision, while it should be negative for the $k = 2$ decision. Similarly, with variance scaling, the variance of Y_1 should be increased for the $k = 1$ decision but be decreased for the $k = 2$ decision. Therefore compromise has to be made. As the system memory increases, the compromise becomes greater. For event simulation, no sacrifice on optimality is necessary because successive decisions depend on independent sets of random variables. In the above simple example, optimal biasing on Y_0 and Y_1 can be tailored for each of the only two possible signal combinations $(X_0, X_1) = (-1, -1)$ and $(X_0, X_1) = (1, -1)$. Thus, in each run of the event simulation, an optimal biasing is used after choosing X_0.

In summary, the key of event simulation is that we fix the transmitted signal and focus on estimating the probability of the event that a decision error occurs. The advantages of this simulation method are: 1) it is easier to evaluate the estimator variance; and 2) it allows signal-dependent importance sampling biasing which yields higher efficiency gain because no biasing conflict would occur. It will be also demonstrated later that event simulation permits "signal biasing" (biasing of the distribution of X_0 in the above example) which enables us to further concentrate our computational effort on the "important error events" and thus increases the importance sampling efficiency. More will be said about event simulation and signal biasing in Chapter 4 and the examples in Chapters 5 and 6. In contrast, this signal biasing is not permissible with the stream simulation, again, because of biasing conflicts caused by correlated decision statistics.

2. FUNDAMENTALS OF IMPORTANCE SAMPLING

This chapter introduces the basic notions of importance sampling and reviews previous work in its application to the BER estimation problem. We adopt the notational convention that upper case letters represent random quantities and lower case letters represent deterministic quantities or dummy variables. Bold-faced letters indicate vectors or sequences.

2.1 The Monte Carlo Method

Consider the block diagram of Figure 2.1 for a general mean estimation problem. Suppose that $\mathbf{Y}$ is a random vector with joint pdf (model density) $f_{\mathbf{Y}}(\mathbf{y})$, which is known to us, and we want to estimate the unknown quantity $\alpha = \mathrm{E}[g(\mathbf{Y})]$, i.e., the population mean of $g(\mathbf{Y})$, where $g(\cdot)$ is any (possibly nonlinear) function such that $g(\mathbf{Y}) \geq 0$. In some literature, the density function $f_{\mathbf{Y}}(\cdot)$ is called the "input density" in contrast to the "output density" $f_{g(\mathbf{Y})}(\cdot)$. The transfer function $g(\cdot)$ is often complicated enough to prevent us from directly computing statistics of the output $g(\mathbf{Y})$. (We won't need simulation if we can do this.) We will be working on input density $f_{\mathbf{Y}}(\cdot)$ exclusively.

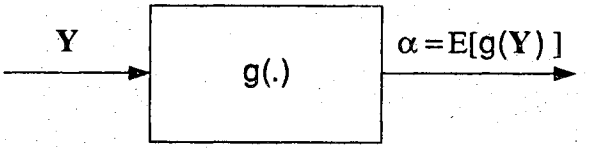


Figure 2.1 A general block diagram for mean estimation.

Of particular interest to us is the special case that $\alpha = P_e$ is a decision error probability. In this case, the components of $\mathbf{Y}$ are sampled values of the input noise process(es) that contribute to the decision, and $g(\cdot) = 1_E(\cdot)$ where E is the "error set," i.e., the set of noise inputs that result in a decision error. $1_E(\cdot)$ is the indicator function as defined by (1.2). Note that $P_e = \mathrm{E}[1_E(\mathbf{Y})] = P(\mathbf{Y} \in E)$. We will concentrate on this decision error probability. However, the following arguments can be easily generalized by replacing $1_E(\cdot)$ with $g(\cdot)$.

Expressing the expectation $P_e = \mathrm{E}[1_E(\mathbf{Y})]$ as an integral, we have

$$P_e = \int 1_E(\mathbf{y}) f_{\mathbf{Y}}(\mathbf{y})\, d\mathbf{y}, \tag{2.1}$$

where the integration (or summation if $f_{\mathbf{Y}}(\cdot)$ is a probability mass function) is multidimensional. To evaluate the integral (2.1), the Monte Carlo method uses a sample mean estimator which generates L independent identically distributed (i.i.d.) samples, $\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \ldots, \mathbf{Y}^{(L)}$, from the density function $f_{\mathbf{Y}}(\mathbf{y})$, then computes the sample mean of $1_E(\mathbf{Y})$. In mathematical form, the estimator (a random variable) of P_e is

$$\hat{P}_e = \frac{1}{L} \sum_{\ell=1}^{L} 1_E(\mathbf{Y}^{(\ell)}), \tag{2.2}$$

where $1_E(\mathbf{Y}^{(\ell)})$ acts like an error detector and (2.2) is simply an error counter. As $L \to \infty$, the estimate converges to the true value P_e. Note that the Monte Carlo method can be also used to obtain an empirical histogram of the random variable (r.v.) $1_E(\mathbf{Y})$ (or $g(\mathbf{Y})$ in general) which may be useful in some applications.

Since each $\mathbf{Y}^{(\ell)}$ is sampled from $f_{\mathbf{Y}}(\mathbf{y})$, we have $\mathrm{E}[\hat{P}_e] = \mathrm{E}[1_E(\mathbf{Y})] = P_e$. Hence, the Monte Carlo estimator is unbiased. The variance of $\hat{P}_e$ is

$$\mathrm{var}[\hat{P}_e] = \frac{1}{L}\mathrm{var}[1_E(\mathbf{Y})] = \frac{P_e}{L}(1 - P_e). \tag{2.3}$$

The required number of independent simulation runs L can thus be expressed as a function of the "standard error," ϵ, and P_e,

$$\epsilon \equiv \frac{\sqrt{\mathrm{var}[\hat{P}_e]}}{P_e} \quad \Rightarrow \quad L \approx \frac{1}{\epsilon^2 P_e}, \tag{2.4}$$

where the approximation holds if P_e is small. Note that L increases with $1/P_e$. For a decision error probability of 10^{-6} and $\epsilon = 10\%$, this would require 10^8 simulation runs — an expensive task even for a powerful computer.

The standard error, ϵ, defined as the ratio between the standard deviation of the mean estimator and the true population mean, is a measure of the quality of an estimator. Note that the sample mean estimator (2.2) is binomially distributed ($1_E(\mathbf{Y})$ is a Bernoulli random variable with mean P_e) which converges to a Gaussian random variable as $L \to \infty$. Often, it is desired that the bias of the estimator satisfies the following equalities

$$0.95 = P(|\hat{P}_e - P_e| < \delta) = P\left(\frac{-\delta}{\sigma_{\hat{P}_e}} < \frac{\hat{P}_e - P_e}{\sigma_{\hat{P}_e}} < \frac{\delta}{\sigma_{\hat{P}_e}}\right), \tag{2.5}$$

where $\sigma_{\hat{P}_e}$ is the standard deviation of the sample mean estimator $\hat{P}_e$. Approximating $(\hat{P}_e - P_e)/\sigma_{\hat{P}_e}$ with a standard Gaussian r.v., we have $\hat{P}_e \in [P_e - \delta, P_e + \delta]$, where $\delta = (1.96\epsilon)P_e$. For example, an $\epsilon = 10\%$ is equivalent to $\delta = 0.2P_e$. When L is not very large, the Chebyshev inequality $P(|\hat{P}_e - P_e| > \delta) \leq \sigma^2_{\hat{P}_e}/\delta^2$ can be used.

Equations (2.4) and (2.5) can only give us a qualitative description of the relationship between L, ϵ and how close the estimate is to the true value. In simulation, both P_e and $\sigma_{\hat{P}_e}$ are unknown to us and need to be empirically estimated. That is, as the simulation progresses, we have no idea what the exact value of ϵ and consequently how good the estimate is. In this case, the "confidence interval" is commonly used to assess the quality of the estimate in Monte Carlo simulation. A confidence interval is a random interval $[r_1, r_2], r_1, r_2 \in \mathbf{R}$, such that with certain probability (usually 0.95) this interval would cover the true value. (This is different from the last paragraph where the estimate falls in a fixed interval.) As L increases, the confidence interval becomes narrower. For simple systems such as binary signalings the relationship between the confidence interval and the required number of runs L in Monte Carlo simulation can be found analytically [31].

For importance sampling, an exact quantitative relationship between L and the confidence interval is difficult to establish. Instead, we use the "empirical precision,"

an estimate of ϵ which is readily available in simulation, $\varepsilon = \hat{\sigma}_{\hat{P}_e}/\hat{P}_e$ where $\hat{\sigma}_{\hat{P}_e}$ is an estimate of $\sigma_{\hat{P}_e}$, as an indicator of the quality of the estimate. There is a loose relationship between ε and the confidence interval which can be easily obtained. Suppose that we can estimate $\sigma_{\hat{P}_e}$ closely. Replacing $\sigma_{\hat{P}_e}$ with $\hat{\sigma}_{\hat{P}_e}$ in (2.5), we have $P_e \in [\hat{P}_e - \delta, \hat{P}_e + \delta]$, where $\delta = (1.96\varepsilon)\hat{P}_e$. Therefore, an empirical precision $\varepsilon = 10\%$ is equivalent to saying that, with probability 0.95, $P_e \in [0.8\hat{P}_e, 1.2\hat{P}_e]$ when L is large, or $P_e \in [0.5\hat{P}_e, 1.5\hat{P}_e]$ with Chebyshev inequality when L is only moderate.

2.2 The Importance Sampling Technique

Importance sampling is motivated by a modification to (2.1). Instead of the true model density function $f_{\mathbf{Y}}(\mathbf{y})$, we introduce a simulation density $f^*_{\mathbf{Y}}(\mathbf{y})$, provided that the absolute continuity is satisfied, i.e., $f^*_{\mathbf{Y}}(\mathbf{y}) > 0$ whenever $1_E(\mathbf{y})f_{\mathbf{Y}}(\mathbf{y}) > 0$ [21]. Let us rewrite (2.1) as

$$\begin{aligned} P_e &= \int 1_E(\mathbf{y}) \frac{f_{\mathbf{Y}}(\mathbf{y})}{f^*_{\mathbf{Y}}(\mathbf{y})} f^*_{\mathbf{Y}}(\mathbf{y})\, d\mathbf{y} \\ &= \mathrm{E}^*[1_E(\mathbf{Y})w(\mathbf{Y})], \end{aligned} \tag{2.6}$$

where E^* denotes expectation with respect to the distribution $f^*_{\mathbf{Y}}(\cdot)$ and the *a posteriori* likelihood ratio

$$w(\mathbf{y}) = \frac{f_{\mathbf{Y}}(\mathbf{y})}{f^*_{\mathbf{Y}}(\mathbf{y})}$$

is called the importance sampling weight.

To evaluate the integration in (2.6) we again appeal to the sample mean estimator. That is, the importance sampling simulation generates i.i.d. random samples from the biased simulation distribution $f^*_{\mathbf{Y}}(\mathbf{y})$ and empirically estimates the mean of $1_E(\mathbf{Y})w(\mathbf{Y})$. The IS estimator of P_e can thus be written as

$$\hat{P}^*_e = \frac{1}{L} \sum_{\ell=1}^{L} 1_E(\mathbf{Y}^{(\ell)})w(\mathbf{Y}^{(\ell)}) \tag{2.7}$$

where and hereafter the superscript * is used to indicate operations related to importance sampling density $f^*(\cdot)$. Note that in ordinary Monte Carlo simulation, the

density function is not biased, i.e., $f^*_{\mathbf{Y}}(\mathbf{y}) = f_{\mathbf{Y}}(\mathbf{y})$, in such case $w(\mathbf{Y}^{(\ell)}) = 1$ for all $\ell = 1, 2, \ldots, L$.

Let us examine the properties of importance sampling estimator (2.7). The simulation data $\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \ldots, \mathbf{Y}^{(L)}$ are i.i.d., therefore we have

$$\begin{aligned} \mathrm{E}^*[\hat{P}^*_e] &= \mathrm{E}^*[1_E(\mathbf{Y})w(\mathbf{Y})] \\ &= \int 1_E(\mathbf{y}) \frac{f_{\mathbf{Y}}(\mathbf{y})}{f^*_{\mathbf{Y}}(\mathbf{y})} f^*_{\mathbf{Y}}(\mathbf{y})\, d\mathbf{y} \\ &= P_e. \end{aligned}$$

Hence the importance sampling estimator is also unbiased. The estimator variance is

$$\begin{aligned} \mathrm{var}^*[\hat{P}^*_e] &= \frac{1}{L}\mathrm{var}^*[1_E(\mathbf{Y})w(\mathbf{Y})] \\ &= \frac{1}{L}\left\{\mathrm{E}^*[1_E(\mathbf{Y})w^2(\mathbf{Y})] - P_e^2\right\} \\ &= \frac{1}{L}\left\{\int 1_E(\mathbf{y})w(\mathbf{y})f_{\mathbf{Y}}(\mathbf{y})\, d\mathbf{y} - P_e^2\right\}. \end{aligned} \tag{2.8}$$

Comparing (2.8) with the variance of the Monte Carlo estimator (2.3) which is rewritten as

$$\mathrm{var}[\hat{P}_e] = \frac{1}{L}\left\{\int 1_E(\mathbf{y})f_{\mathbf{Y}}(\mathbf{y})\, d\mathbf{y} - P_e^2\right\},$$

the IS estimator has the extra term $w(\mathbf{y})$ in the integrand. Apparently, if the simulation density $f^*_{\mathbf{Y}}(\mathbf{y})$ is such that $f^*_{\mathbf{Y}}(\mathbf{y}) > f_{\mathbf{Y}}(\mathbf{y})$, i.e., $w(\mathbf{y}) < 1$, for all $\mathbf{y}$ in the error region E, then $\mathrm{var}^*[\hat{P}^*_e] < \mathrm{var}[\hat{P}_e]$. Equivalently, for an equal variance, the importance sampling estimator would require a smaller value of L, i.e., fewer simulation runs. The efficiency of importance sampling can be defined as

$$\text{Efficiency } \eta = \frac{L_{\mathrm{MC}}}{L_{\mathrm{IS}}}, \quad \text{given } \mathrm{var}^*[\hat{P}^*_e] = \mathrm{var}[\hat{P}_e],$$

where L_{MC} and L_{IS} denote numbers of independent simulation runs for ordinary Monte Carlo simulation and importance sampling respectively. Note that this definition of efficiency implies that the computational costs are the same for each run in

Monte Carlo simulation and importance sampling. This, however, is not so. With stream simulation method, the importance sampling algorithm has an overhead of computing importance sampling weights and unbiasing data. With event simulation method, as mentioned in Chapter 1, we need to generate more samples and the overhead is even greater. Both methods also have the extra cost of computing optimal biasings if a time varying (signal dependent) scheme is used. Nevertheless, importance sampling algorithms with stream simulation setting often still use η as the measure of efficiency in the literature because of the small overhead cost. However, since we will use event simulation, the "computer CPU time efficiency," i.e., the ratio of CPU times required for Monte Carlo simulation and importance sampling for the same variance, is a more appropriate and straightforward measurement of efficiency.

The selection of a simulation density is an important factor in determining the simulation algorithm's efficiency. A qualitative characterization of "good" simulation densities can be established as follows. Returning to (2.8), for a required standard error ϵ, we have

$$L = \left\lceil \frac{\text{var}^*[1_E(\mathbf{Y})w(\mathbf{Y})]}{\text{var}[\hat{P}_e]} \right\rceil = \left\lceil \frac{\text{var}^*[1_E(\mathbf{Y})w(\mathbf{Y})]}{\epsilon^2 P_e^2} \right\rceil \tag{2.9}$$

where $\lceil x \rceil$ denotes the least integer $\geq x$ and ϵ is the standard error defined in the last section. Therefore, we would like to select a simulation density $f^*_{\mathbf{Y}}(\mathbf{y})$ to minimize L or equivalently to minimize $\text{var}^*[1_E(\mathbf{Y})w(\mathbf{Y})]$. The unconstrained optimal simulation density is well known [24]. Considering (2.8) again, note that we always have $\text{var}^*[\hat{P}^*_e] \geq 0$, and equality holds, by Jensen's inequality lemma, if and only if

$$1_E(\mathbf{y})\frac{f_{\mathbf{Y}}(\mathbf{y})}{f^*_{\mathbf{Y}}(\mathbf{y})} = \text{ constant w.p.1 with respect to } f^*_{\mathbf{Y}}(\mathbf{y}),$$

where "w.p.1" stands for "with probability one." Therefore $f^*_{\mathbf{Y}}(\mathbf{y}) \propto 1_E(\mathbf{y})f_{\mathbf{Y}}(\mathbf{y})$. After normalization, the optimal importance sampling density function is found to be

$$f^*_{\mathbf{Y}}(\mathbf{y})_{opt.} = \frac{1_E(\mathbf{y})f_{\mathbf{Y}}(\mathbf{y})}{P_e}. \tag{2.10}$$

Substituting (2.10) into (2.8), we have $\text{var}^*[1_E(\mathbf{Y})w(\mathbf{Y})] = \text{var}^*[\hat{P}_e^*] = 0$. That is, we need just one sample ($L = 1$) from this optimal distribution to estimate P_e. However, this solution is impractical. The denominator in (2.10), P_e, is what we want to estimate in the first place. Therefore, finding the optimal simulation density is as difficult as finding P_e. We cannot compute the importance sampling weight either, which is again P_e w.p.1.

The optimal simulation density function (2.10) does give us some hints as to what a good simulation distribution should be, namely,

1. $f_{\mathbf{Y}}^*(\mathbf{y})_{opt.}$ puts all its probability mass in the error region E because of the indicator function.

2. In the error region, $f_{\mathbf{Y}}^*(\mathbf{y})_{opt.}$ is proportional to $f_{\mathbf{Y}}(\mathbf{y})$.

3. $f_{\mathbf{Y}}^*(\mathbf{y})_{opt.}$ need not be memoryless or stationary even if the channel is.

Thus, the "important samples" are those $\mathbf{y}$ in the "important regions" — those regions where $1_E(\mathbf{y})f(\mathbf{y})$ are relatively large. An efficient simulation density design therefore should first identify the important error regions which we will see are signal-dependent. (Although signal inputs are not shown in Figure 2.1, they are present, and the error set in general strongly depends on the signal inputs.) Then we bias the model density function in such a way that more samples are taken from these regions. Later we will see that our nonstationary mean-translation biasing used in conjunction with the event simulation method is precisely one such efficient scheme.

If we take a closer look at the real computational cost of a simulation algorithm, which can be measured by the computer run time, it is

$$\text{total cost} = L \times \text{ per sample cost.}$$

The optimal simulation density $f_{\mathbf{Y}}^*(\mathbf{y})_{opt.}$ has $L = 1$ but very high (possibly infinity) per sample cost. It is difficult to express the total cost as a function of the simulation density (or its parameters) and then perform minimization. Instead, in practice, we often control the per sample cost by restricting our candidate simulation densities

to a "candidate family" then minimize $\mathrm{var}^*[1_E(\mathbf{Y})w(\mathbf{Y})]$ only within this family. In other words, we can only emulate the proportionality $f^*_{\mathbf{Y}}(\mathbf{y}) \propto 1_E(\mathbf{y})f_{\mathbf{Y}}(\mathbf{y})$ to the best of this family. The candidate family should contain simulation densities $f^*_{\mathbf{Y}}(\cdot)$ which allow us to generate samples and compute the importance sampling weights easily. For example, if the model density $f_{\mathbf{Y}}(\cdot)$ is n-dimensional Gaussian, then the choice of a candidate family of Gaussian distributions would seem natural. If the model density $f_{\mathbf{Y}}(\mathbf{y}) = \prod_k^n f_k(y_k)$, i.e., the components of $\mathbf{Y}$ are independent, then a candidate family of product form densities is preferred. A candidate family of arbitrary densities may result in a lower $\mathrm{var}^*[1_E(\mathbf{Y})w(\mathbf{Y})]$ but not be computationally efficient.

Finally, as mentioned earlier, the estimator variance needs to be estimated empirically. A logical choice is the sample variance estimator. That is, $\mathrm{var}^*[1_E(\mathbf{Y})w(\mathbf{Y})]$ is estimated by

$$\hat{S}^2 = \frac{1}{L}\left\{\sum_{\ell=1}^{L}\left[1_E(\mathbf{Y}^{(\ell)})w(\mathbf{Y}^{(\ell)}) - \hat{P}_e^*\right]^2\right\}. \tag{2.11}$$

(An unbiased estimator actually has the multiplicity $1/(L-1)$ instead of $1/L$ but the difference is negligible for large L.) And $\mathrm{var}^*[\hat{P}_e^*]$ is estimated by $\hat{S}^2/L$. The simulation is terminated when the empirical precision $\varepsilon = \sqrt{\hat{S}^2/L}/\hat{P}_e^*$, which is checked periodically during the course of simulation, falls below a preset empirical precision. Usually we set $\varepsilon = 10\%$ for a rough 95% confidence interval of $[0.8\hat{P}_e^*, 1.2\hat{P}_e^*]$ as explained in the last section.

2.3 Previous Works and Comments

A. Uncoded systems:

Importance sampling idea first appeared in 1949 [24], and has been used in diverse fields, such as, physics, operations research and statistics. Its applications to the estimation of BER for uncoded digital communication systems operating on ideal linear memoryless channels have achieved significant computational gains. Efficiency η on the order of 10^4 is reported in Shanmugan and Balaban [53] in simulating a

BER of 10^{-6}. Their simulation density is a uniform (stationary) variance scaling for every Gaussian input noise sample. The conventional stream simulation is used. We will call importance sampling schemes employing the uniform variance scaling and the stream simulation "conventional importance sampling" or CIS.

For nonideal channels, their gains are moderate, for example, on the order of 10^1 for a BER of 10^{-4} with the presence of a limiter (nonlinearity) and a simple receive filter (ISI). The deficiency results from 1) the difficulty in determining the optimal variance scaling for nonlinear channel and 2) the so-called "dimensionality effect." Dimensionality effect in the importance sampling literature refers to the effect that efficiency gain drops rapidly as the "dimension" of the system increases. The dimension of a system is roughly defined as the number M of noisy samples that affect one output decision. It is easy to see the cause of dimensionality effect. Recall from the last section that importance sampling efficiency is achieved by concentrating the probability mass of the simulation density $f^*_{\mathbf{Y}}(\mathbf{y})$ on the important regions where $1_E(\mathbf{y})f_{\mathbf{Y}}(\mathbf{y})$ are large. CIS achieves this goal to some degree by simply pushing the probability mass out from the origin of the n-dimensional sample space uniformly in all directions. However, as the dimension of the system increases, there will be more and more wrong directions, and hence, less and less of the probability mass would really end up in the important regions. We will explain graphically this dimensionality effect in Chapter 4 when we compare the variance-scaling biasing and mean-translation biasing.

There are two obvious ways to reduce the dimensionality effect: 1) reduce the number of samples per symbol and 2) compute the IS weight using less than M samples. The first method introduces modeling error while the latter, as noted by Shanmugan and Balaban [53], results in a biased estimator. Both the modeling error and bias are hard to quantify and predict. Thus, these two solutions are not attractive. Shanmugan and Balaban also note that if we compute the weight using more than M samples (sometimes the exact memory length of the system is unknown), the estimator remains unbiased but the estimator variance is increased simply because more terms

are included. Also, if the memory truncation occurs at where the impulse response function of the bandlimiting filter has significant value, the variance increase tends to be large. We will see that these problems can be eliminated by using event simulation where the importance sampling weight is computed using exactly M samples, no less and no more.

Although gains are moderate for nonideal channels, Shanmugan and Balaban's CIS algorithm is simple and easy to implement. Also, a saving factor of even 2 can be significant if the Monte Carlo method requires hundreds of hours to run. Therefore, CIS was quickly adopted in commercial simulation softwares like TOPSIM III [38].

In an effort to reduce the dimensionality effect, Davis [13] proposes a nonstationary variance-scaling scheme. Only $M' < M$ samples are biased ($M' = 1$ is suggested for bi-phased transmission, i.e., no biasing for the ISI components) since these samples are the most important ones in decisions making. The result is an unbiased estimator with no dimensionality effect but gains are low ($\sim 10^1$) because now some important error regions are under-sampled. They also propose to approximate the nonlinear system with a linearized model in finding the optimal variance scaling. This linearization is never very desirable. As the system operating point moves toward the nonlinear region, the modeling error increases which, more often than not, we will not be able to quantify.

Another notable modification to the CIS is done by Jeruchim et al. [25, 32] in which they propose the EIS (efficient importance sampling) technique. EIS still employs stream simulation and variance scaling. They force the dimension to be just unity (thus no dimensionality effect) by computing the IS weight as if the system is linear which admits a reduction to one dimension. A satellite channel with both uplink and downlink noises is studied. The CPU time efficiency gains are in the range of 10^1–10^2 for channels with nonlinearity which are quite good for a system this complex. However, their approach has some critical assumptions and approximations which make the algorithm's usefulness in other situations (e.g., different filters, severe nonlinearity) unclear:

1. Regression method is used to learn the system impulse response functions for both noise processes. This effectively linearizes the channel for uplink noise which passes through nonlinearity.

2. The ISI-contributed portion of the decision voltage is assumed to be uniformly distributed.

3. The noise statistic at the decision time is assumed to be Gaussian.

The second and the last assumptions can be of concern for some systems. Also, computing the IS weight at the channel output while the biasing is done at the input results in a biased estimator [52]. Another drawback of EIS is that the required number of simulation runs L is preset by analytical calculation based on the above assumptions/approximations. Thus, at the end of simulation, we really have no idea of the exact bias and variance of the estimator. The quality of an estimate has to be confirmed by other estimators (e.g., error bounds, Monte Carlo or semi-analytical).

Lu and Yao [36] take a major departure from all previous work and achieves significant gains without dimensionality effect or bias. Their algorithm will be discussed in detail in Chapter 5. Two key new concepts, as mentioned in Chapter 1, are mean-translation biasing and event simulation. The noise vector $\mathbf{Y}^{(\ell)}$ is biased by shifting (or translating) its mean vector. The result, for example, is 54 times more efficient than CIS for a BER of 10^{-6} and $M = 3$. The reason why there is no dimensionality effect is also clear. The optimal mean translation is the point where $1_E(\mathbf{y})f_{\mathbf{Y}}(\mathbf{y})$ is maximized. This is the "most important point" in the error set, hence, optimized mean translation will place much more of its probability mass in the important region than CIS. Dimension is not a factor in this approach so long as we can locate that maximum point on the n-dimensional space. Their success proves that low computational gains for systems with memory is not a natural limitation of importance sampling as once suggested. It also becomes clear that nonlinearity rather than memory is really more of a challenge to importance sampling.

Lu and Yao's optimization method in finding the simulation mean works only for linear systems. Sadowsky and Bucklew [49] use a different approach to arrive at the same result but their method also applies to nonlinear systems. In short, they show that within the closure of the error set, there exist some points called "minimum rate points" such that the rate at which the decision error probability decays with the increased signal-to-noise ratio (SNR) is dominated by these points. When the model density is exponentially twisted to these minimum rate points, asymptotical efficiency is achieved. For a Gaussian random vector $\mathbf{Y}$, the exponentially twisted function reduces to simply mean translation of the model density function. And the simulation density is some convex combination of the mean-translated (to the minimum rate points) model density functions. When there is only one minimum rate point, called the "dominating point," we just shift the mean to this point. That is, if $\mathbf{Y}^{(\ell)} \sim N(\mathbf{0}, \mathbf{C})$, then the simulation distribution is $N(\mathbf{v}, \mathbf{C})$, where $\mathbf{C}$ is the unscaled covariance matrix and $\mathbf{v}$ is the dominating point. We will in general assume in our study that the dominating point exists. The dominating point $\mathbf{v}$ depends on the signal as well as other system parameters. Hence, the task of IS design is to find the dominating point. We direct readers to their paper for a complete treatment. Our work is closely related to theirs and the common fundamental ideas will be discussed in Chapter 4.

B. Coded Systems:

Dimensions in coded systems are greater than in uncoded systems. It is a combination of the channel memory (due to filtering and/or other memory devices) and the memory inherent from the code. This is even more true for trellis codes for which the code memory is theoretically infinite. Herro and Nowark's paper [28] is apparently the first to study importance sampling applied to Viterbi decoding. The CIS method is used in their study. Not surprisingly, the efficiency diminishes as the "effective decoder memory," which is proportional to the code's constraint length, increases. Also, as noted by Shanmugan and Balaban, memory truncation while computing the

importance sampling weight generates biased results. They conclude that importance sampling is useful only for short constraint length codes and small BER's. The effective decoder memory for a particular code must be chosen accurately to account for all noise samples that have significant impact on the decoder's decision. Also, the optimal uniform variance scaling has to be found by trials. Over-biasing may result in a variance even greater than the Monte Carlo method.

In parallel to Lu and Yao's approach for uncoded systems, Sadowsky [46] develops the event simulation method for Viterbi decoders and he uses a union bound argument to design the mean-translation simulation density function. This biasing scheme coincides with the optimal mean-translation biasing obtained by the method in Sadowsky and Bucklew [49] for the special case of AWGN channel. The gains are extremely high. We will explain his event simulation method in Chapter 5 and use two trellis code examples to illustrate the principles in dealing with coded system simulations. A more complicated situation, coded systems operating on nonlinear channels with memory, will be covered in Chapter 6. The event simulation method for Viterbi decoders is what makes efficient importance sampling simulations for trellis-coded systems possible.

Although we will not consider the simulation for the other major branch of error-correcting codes, the block codes, [46] has given an example of the Hamming (7,4) code employing importance sampling. It is conceived that importance sampling can, at least in principle, apply to other more complex block codes such as the BCH and RS codes and practical channel models. We leave this for future research.

3. TCM CODES AND SIMULATION CHANNELS

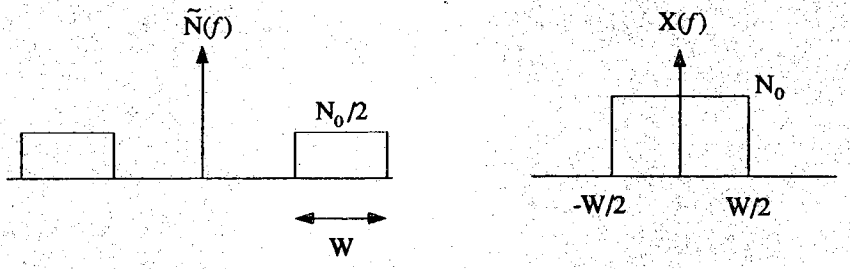
3.1 The Additive White Gaussian Noise in Simulations

Communication systems analysis often assumes that the system input noise process in Figure 1.1 is an additive stationary zero-mean white Gaussian noise process. This noise process is quite faithful in modeling the radio atmosphere noise (rain-induced noise, earth background noise, etc.) and the transmitter/receiver equipment thermal noise. Also, this assumption is easy to work with, especially in the signal space analysis of digital communication systems [43, 64, 72].

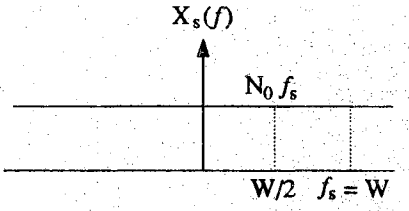
An AWGN process $N(t)$ has a two-sided power spectral density (p.s.d.) $N_0/2$ over the entire frequency spectrum and any finite collection of noise samples has a joint Gaussian distribution. This noise model leads to the concept of the matched filter receiver if the channel is linear. If the channel is also bandlimited, which is the case in practical systems, Nyquist's ISI-free criterion for filter design applies. In this thesis, when we say ideal AWGN channel we will mean an infinite bandwidth channel with a corrupting AWGN process.

Digital computers can handle only discrete-time events, thus sampling of a continuous time process is necessary in simulation. Note that we can not sample the infinite bandwidth AWGN directly, the samples would have infinite variance. Also, computer simulation of a narrowband (comparing to the carrier frequency) bandpass signal/system often uses its complex baseband equivalent to reduce the number of samples required by the sampling theorem. The discrete-time baseband equivalent of the AWGN is obtained as follows.

First we approximate the infinite bandwidth noise process $N(t)$ with a bandlimited noise $\tilde{N}(t)$ with p.s.d. $N_0/2$ and a bandwidth W much wider than the system



(a) Bandlimited AWGN. (b) Lowpass equivalent of (a).



(c) Sampled process of (b).

Figure 3.1 Power spectra of bandlimited, lowpass and sampled AWGN.

bandwidth to reduce modeling error. See Figure 3.1(a). The noise process $\tilde{N}(t)$ can be expressed as [43]

$$\tilde{N}(t) = X(t)\cos\omega_c t - Y(t)\sin\omega_c t,$$

where ω_c is the carrier angular frequency and $X(t)$ and $Y(t)$ are lowpass zero mean Gaussian noise processes with two-sided p.s.d. N_0 and bandwidth $W/2$ as shown in Figure 3.1(b). The complex signal $Z(t) = X(t) + jY(t)$ is the baseband equivalent and contains all information of $\tilde{N}(t)$. The autocorrelation functions for $X(t)$, $Y(t)$, and $Z(t)$ are

$$\phi_{XX}(\tau) = \phi_{YY}(\tau) = \phi_{ZZ}(\tau) = WN_0\text{sinc}(W\tau)$$

where $\text{sinc}(x) = \sin(\pi x)/\pi x$, and $\phi_{XY}(\tau) = 0$ for all τ. That is, the in-phase (I) and quadrature (Q) components of $\tilde{N}(t)$ are uncorrelated (thus independent) Gaussian processes.

Now suppose that we sample N_s times during an interval of T seconds. Let $f_s = 1/\triangle = N_s/T$ denote the sampling rate, where $\triangle$ is the time between samples. The sampled discrete-time lowpass noise process $X_s(t)$ (and $Y_s(t)$) has the p.s.d. of Figure 3.1(c) with no aliasing if we have $W = f_s$, provided that W is much greater than the system bandwidth. The autocorrelation functions of $X_s(t)$ and $Y_s(t)$ are

$$\phi_{X_s X_s}(\tau) = \phi_{Y_s Y_s}(\tau) = N_0 f_s \delta(\tau)$$

where $\delta(\cdot)$ is the unit impulse function. Therefore, samples of $X_s(t)$ and $Y_s(t)$ are i.i.d. Gaussian r.v.'s with the same variance $N_0 f_s = N_0 N_s R$ where $R = 1/T$. We will set T equal to the transmitted channel symbol duration and R is called the data baud rate.

For the ideal AWGN channel and many other linear channel models, the noise statistics at the demodulator output can be found analytically and we don't have to sample the input noise process. When the channel is nonlinear, many samples per channel symbol are required to simulate the nonlinearity. The satellite channel is one such case where the transmitter and transponder high power amplifiers are often driven at near saturation to achieve power efficiency and are the major sources of nonlinearity. Depending on the modulation scheme and the severity of nonlinearity, 8–64 samples (or more) per symbol are usually used. Nonlinearity tends to spread the signal spectrum [22], thus the more severe the nonlinearity, the more samples per symbol will be needed.

The next section will introduce the trellis codes and analyze their error performance on the ideal AWGN channel. Section 3.3 describes a satellite channel which is the channel model we will pay much attention to. A satellite channel is often too difficult to analyze because of the nonlinearity and its many distortion sources. Analytical tools do exist for the analysis of nonlinear systems with memory (e.g., [50]), but there are problems associated with these methods. Performance evaluation of satellite systems via computer simulation has remained the dominant means since the beginning of the satellite era.

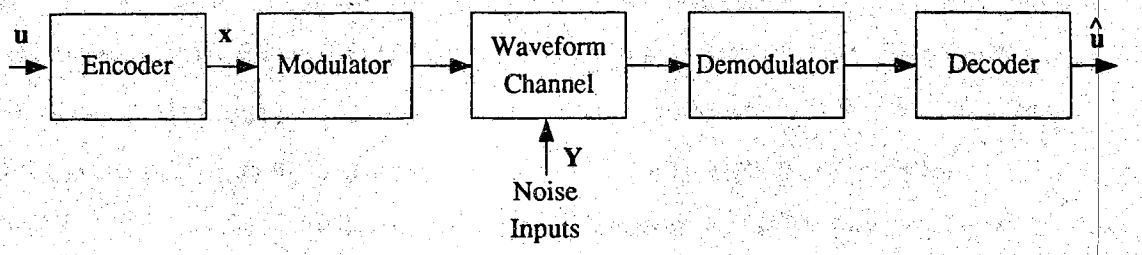


Figure 3.2 Block diagram of a digital communication system.

3.2 The Trellis-Coded Modulation

3.2.1 TCM Codes

Consider the block diagram of a typical digital communication system shown in Figure 3.2. Traditionally, the channel encoder and modulator are two independent functional units — the encoder transforms and adds bit redundancy to the data stream for later error correction/detection while the modulator chooses transmitted signals which can be easily distinguished by the receiver after transmission over the waveform channel. They more or less are designed separately as long as the combination achieves BER, power and bandwidth specifications. With an M-ary modulation, a code rate R ($R < 1$) block code or convolutional code would result in a factor of $1/R$ bandwidth expansion compared to the uncoded system at the same effective data rate. Note that for many modulation schemes (e.g., binary signalings, QPSK, OQPSK, MSK), the Hamming distance between any two discrete codewords (number of digits in which they differ) is a linear function of the Euclidean distance between their respective transmitted signals in the signal space. Therefore, the criterion for designing channel encoders has been to maximize the minimum Hamming distance d_{min} between discrete codewords.

To have the error-correcting capability of channel coding while maintaining the same spectral efficiency as the uncoded transmission, the signal set S (S = $\{p_i(t), 1 \leq i \leq M; p_i(t) = 0$, for $t > T$ and $t < 0\}$ = the set of all possible transmitted signals) must be expanded to accommodate the added bit redundancy. However, now the signal space becomes more crowded and there are more neighboring signal points. The probability of mistaking one signal with the others is increased. The coding gain can be offset by this effect. For example, an uncoded QPSK system can be replaced by a code rate 2/3, 64-state convolutional encoder, which has $d_{min} = 7$, followed by an 8-PSK modulator. However, after going through all the trouble of encoding and decoding, the coded system performs only as good as the uncoded QPSK [58]. One possibility to overcome this adversity is to let the signal mapping function be dependent on the channel encoding. This is the motivation behind the development of TCM [58].

TCM was introduced by Ungerboeck [57, 60]. It is one of the so-called combined modulation and coding techniques [2, 54]. One common concept of these techniques is to design the encoder and modulator jointly to directly maximize the Euclidean distance between transmitted signals. The encoder and modulator are viewed as one single block which maps information bits directly to the transmitted channel signals. The codes thus designed are also called "modulation codes" because coding can then be thought of being embedded in the modulation process. This class of codes often can achieve coding gains with no or little sacrifice of bandwidth efficiency and data rate, but at the expense of system complexity. This concept takes us one step further toward Shannon's prediction of the existence of a both bandwidth and power efficient error-free coding scheme so long as the data rate is less than the channel capacity and the signal-to-noise ratio (SNR) is greater than -1.59 dB [20].

Figure 3.3(a) shows the block diagram of a TCM transmitter which consists of a linear binary convolutional encoder (bit-wise modulo-2 addition of any two codewords is another valid codeword) followed by a signal mapper. This structure was introduced by Forney et. al. [18]. During each signaling period $kT, k = 1, 2, \ldots$, a block of

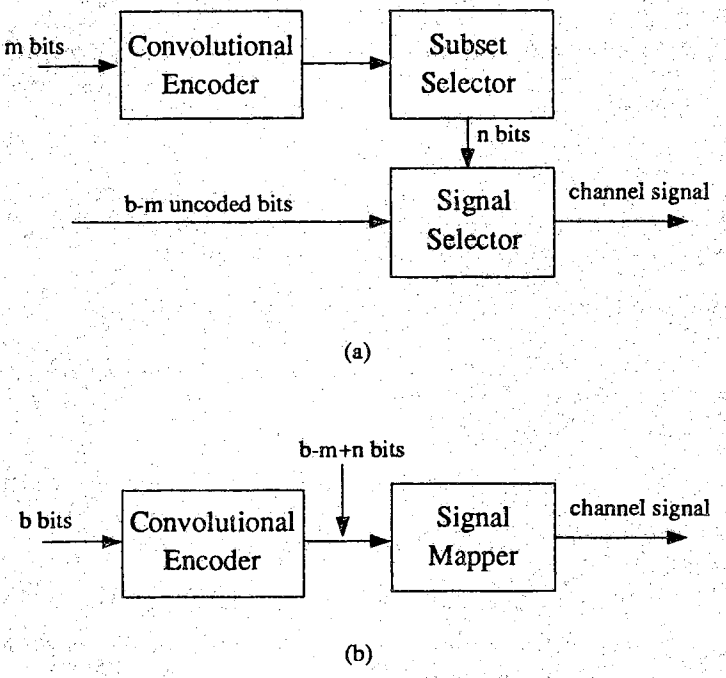


Figure 3.3 Block diagram of a TCM transmitter.

b information bits are to be transmitted, where T is called the symbol duration. Among them, m bits are shifted into an (n, m) convolutional encoder while the rest of $(b-m)$ bits are uncoded and sent directly to the block labeled signal selector. The n-bit outputs of the convolutional encoder, called the "subset selector," is to choose a set S′ of signals from the signal set S, where S′ $\subset$ S. The uncoded bits then select from this subset the actual transmitted channel signal. Thus the signal set S is divided into 2^n subsets, each having 2^{b-m} elements. The binary $(n+b-m)$-tuple of the n-bit encoder output plus the $(b-m)$ uncoded bits is called a "signal selector." For some TCM codes which do not have uncoded bits, each subset would contain only one signal and the subset selector is the signal selector. We will call the b-bit input data an "information symbol," the corresponding signal selector a "code symbol," and the transmitted signal a "channel symbol." A codeword is a sequence of code symbols. The terms "code symbol" and "signal selector" will be used interchangeably; so will "codeword" and "signal selector sequence."

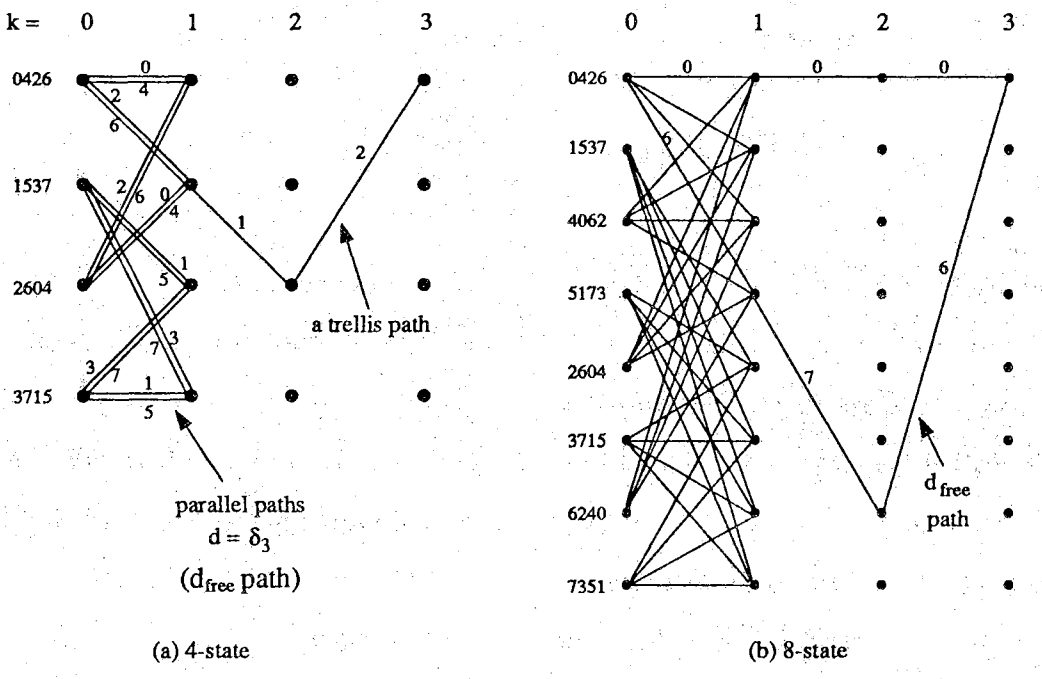


Figure 3.4 Examples of trellis diagrams for 8-PSK TCM codes.

The trellis diagram representation is a standard for trellis codes. A trellis diagram is a state transition diagram evolving in time and the transition branches are labeled with channel symbols or signal selectors. Two examples of TCM trellis diagrams are shown in Figure 3.4, both have the same input block length $b = 2$ and the 8-PSK signal set ($|S| = M = 8$). At any time instance kT (we will often refer to simply as "time index" or "stage" k), a trellis node represents one of the finite convolutional encoder states. An encoder state is defined by the values of ν past information bits stored in the shift registers of the convolutional encoder, where ν is called the constraint length. The total number of states is 2^{ν}. The incoming m information bits together with the current encoder state determine the transition from one state at time kT to some state at time $(k+1)T$ as well as the subset selector. There are 2^m possible state transitions originating from each node. Uncoded bits have no influence on state transitions, their presence is represented by letting the state transitions have parallel branches. The number of parallel branches for each state transition is 2^{b-m}, and they

are labeled by signals from the same signal subset. The one particular branch is chosen by the uncoded bits. The total number of branches leaving and entering each state is thus $2^m 2^{(b-m)} = 2^b$. Figure 3.4(a) has parallel branches ($m = 1$) while in (b) $m = b = 2$ and hence no parallel branches. A trellis path is a sequence of branches that progress in time. Given the initial encoder state, there is a 1-to-1 correspondence between the transmitted information sequence and the signal selector sequence. In addition, that signal selector sequence is equivalent to a trellis path or an encoder state sequence in the trellis diagram. A TCM code can be completely described by its trellis diagram.

If we view the uncoded bits, if there are any, as trivial connections in a larger convolutional encoder, a more general system block diagram is shown in Figure 3.3(b). Note that by doing this, the linearity of the discrete encoder is preserved — it is still a linear convolutional encoder, we just eliminate the uncoded bits. In terms of the trellis diagram, there will be more states but no parallel paths. Therefore, the major conceptual difference between TCM codes and ordinary convolutional codes is the signal mapping function (modulation). We assume that the readers have had basic exposure to the convolutional codes and their performance bounds computation. Standard treatment of convolutional codes can be found in, e.g., [12, 34, 64].

Conventionally, for a rate $R = m/n$ convolutional code, the selection of an M-ary signal set is independent of m and n. Some commonly used modulation schemes are BPSK, FSK, QPSK (and its variations, such as, OQPSK, DPSK, MSK) and QAM. The rules of signal assignment (mapping of signal selectors to channel symbols) are according to Gray coding when possible so that neighboring signals differ in the least number of bits to minimize bit errors. This type of conventional convolutional codes which have no special structure in the signal mapping function can be considered as a special case of TCM.

TCM employs an expanded M-ary signal set in the sense that $\mathrm{M} = 2^{b-m+n} > 2^b$, i.e., the size of the signal set is larger than that required by the uncoded transmission. Coding gain can be viewed as coming from the redundancy of the expanded signal set

rather than from the redundancy of transmitting additional channel symbol pulses. Thus bandwidth expansion is not required. One class of commonly used TCM codes has $b = m$ and $n = m + 1$, i.e., the signal sets are expanded by a factor of 2. The rules of signal assignments for TCM codes are called "mapping by set partitioning" whose objective is to maximize Euclidean distances between discrete codewords. The Euclidean distance between two discrete codewords is defined as the Euclidean distance between their corresponding channel symbols in the signal space. As mentioned earlier, this mapping function should depend on the encoder so that the loss of coding gain because of the expanded signal set can be recovered.

We will study the nonlinear satellite channel where constant envelope signalings are preferred, therefore let us take the 2-dimensional 8-PSK signal set for our example. This signal set has been also adopted in many practical modem designs [15, 19, 59]. Wei [66] and Part II of [58] outline the general rules of mapping by set partitioning for any Z_k type signal sets and the desired "free distance," d_{free}. A signal set is of type Z_k if it is a subset of the k-dimensional lattice with integer coordinates. The free distance d_{free}, defined as the minimum Euclidean distance between all possible codewords, is an important property of TCM codes. We will see in the next subsection that, for the AWGN channel, the free distance is a good indicator of error performance at high SNR.

Figure 3.5 shows the 8-PSK signal constellation and a signal mapping based on the rules of mapping by set partitioning. The one digit number next to a signal point is its "signal label," and the 3-bit symbol inside the parentheses is the signal selector. This particular mapping of signal selectors to signal labels is known as the "natural mapping" because the signal labels happen to be the octal representation of their signal selectors. The reasoning of mapping by set partitioning for this 8-PSK signal set is as follows. Notice that there are four possible Euclidean distances (ED) between signals. Normalizing the signal power such that all signal points lie on the unit circle in the signal space, we have $\delta_0^2 = 0.5858$, $\delta_1^2 = 2$, $\delta_2^2 = 3.4142$, and $\delta_3^2 = 4$. The minimum ED for signals in S is δ_0. A "careless" signal mapping might result

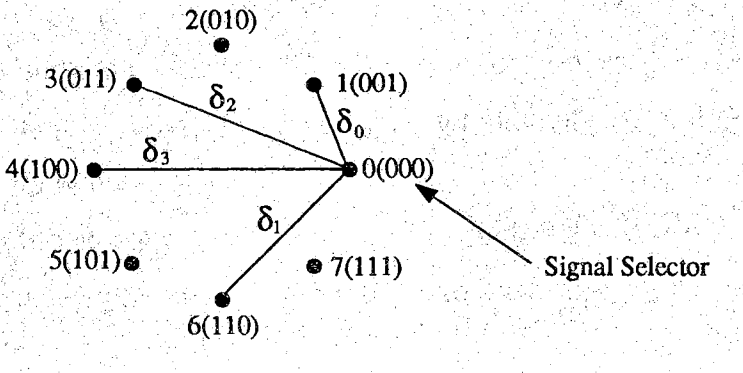


Figure 3.5 8-PSK signal constellation and the natural mapping.

in two valid paths having this minimum distance. To avoid this and to increase the distances between paths, we first partition S into 2^n subsets with increased intrasubset minimum Euclidean distance: The first partition results in 2 subsets, {0,2,4,6} and {1,3,5,7}, and the minimum ED within each subset is δ_1. One further partition results in 4 subsets, {0,4}, {2,6}, {1,5} and {3,7}, and the minimum intrasubset Euclidean distance is δ_3. The next partition, necessary only if $n = 3$, produces 8 subsets, each having only one single signal and the intrasubset Euclidean distance is defined to be infinity. The rules of mapping by set partitioning then are: [58]

1. Parallel branches are associated with signals with maximum distance $\delta_3 = 2$ between them, i.e., the signals in the subsets {0,4}, {1,5}, {2,6} or {3,7}.

2. Four branches leaving or entering one state are labeled with signals with at least distance $\delta_1 = \sqrt{2}$ between them, i.e., the signals in the subsets {0,2,4,6} or {1,3,5,7}.

3. All 8-PSK signals are used in the trellis diagram with equal frequency.

The two examples in Figure 3.4 are labeled by these rules and the result is the natural mapping in Figure 3.5.

Comparing Figure 3.4(a) with an uncoded QPSK system, they both have the spectral efficiency of 2 bits/sec/Hz. Starting from any state, the two paths which

diverge and later remerge that have the minimum distance, d_{free}, are the one-branch parallel paths. (This is the reason why they are assigned the set of signals with the greatest intrasubset distance δ_3.) That is, $d_{free} = \delta_3 = 2$. For the uncoded QPSK, $d_{free} = \sqrt{2}$. Therefore there is a 3 dB improvement for this code without bandwidth increase or data throughput decrease. For the Figure 3.4(b) code, two paths with the minimum distance are shown and $d^2_{free} = 2\delta_1^2 + \delta_0^2 = 4.5838$, i.e., a coding gain of 3.6 dB. For the 8-PSK signal set, coding gains of 3, 4 and 4.8 dB schemes over the uncoded QPSK transmission on the AWGN channel have been found with 4-state, 8-state, and 16-state convolutional encoders respectively. Theoretically, 7–8 dB coding gain can be achieved with TCM [57].

Consider the mapping of Figure 3.5. The Hamming distance (HD) between two discrete code symbols is no longer a linear function of their Euclidean distance. For example, $d_{\text{HD}}(000, 100) = 1 < d_{\text{HD}}(000, 110) = 2$, but $d_{\text{ED}}(000, 100) = 2 > d_{\text{ED}}(000, 110) = \sqrt{2}$. A signal mapping such that the Euclidean distance between two signals depends only on the Hamming distance of their signal selectors is called a regular mapping, otherwise non-regular. TCM codes in general have non-regular mappings and thus are non-regular codes. By definition, a regular code [6] is such that the distance between two codewords depends only on the Hamming distance of their input information bit sequences. Convolutional codes with regular mappings are regular because of the linearity (linear codes are trivially regular codes) of the encoders and regularity in signal mappings. Regular codes have a nice property that the number of codewords at distance d from one particular codeword is the same for all codewords and all d. Therefore in computing the BER performance we can assume a particular codeword is transmitted, in particular, the all 0 sequence. For non-regular codes, error performance analysis is more difficult. This is the subject of the next subsection.

At the receiving end, the sequence of transmitted signals is demodulated and then decoded by a maximum likelihood sequence decoder — the Viterbi decoder. We will use the unquantized soft decision Viterbi decoder to avoid loss of information and

generally 2 dB of SNR if a two-level hard-quantized decoder is used. Synchronizations of carrier phase and frequency, symbol and bit timings at the receiver is assumed throughout unless otherwise stated.

There are other more complex designs of signal sets and their mapping rules than the 2-dimensional 8-PSK with natural mapping described above. For example, rotational invariant TCM codes using 2-dimensional signal constellations are discussed in Wei [67]. Multidimensional TCM codes [7, 18, 66] can achieve higher coding gains and/or reduced system complexity by using multi-dimensional (> 2) signal sets. Recall that the expanded 2-dimensional 8-PSK signal set causes the loss of coding gain obtained from convolutional encoding. The signal set S has a minimum ED ($\delta_0^2 = 0.5858$) much smaller than the uncoded QPSK system ($d_{free}^2 = 4$) to start the set partitioning. In multi-dimensional TCM codes, the added bit redundancy or signal set expansion is evenly absorbed (shared) by all of the constituting 2-dimensional signal constellations [66]. We can thus have a simple per 2-dimensional constellation or increased intrasubset ED. For the moment, the 8-PSK signal set suffices to serve our purpose of simulation algorithm study. The simulation algorithms we will present are applicable to any TCM codes with quasi-regularity which is explained in the next section.

3.2.2 Union Bounds and the RC Algorithm

An analytical solution for P_b is difficult to obtain for even simple TCM codes. Instead, an error bound or simulation is usually used. An union bound on P_b for the case of the ideal AWGN channel will be discussed in this section while simulations for the AWGN channel and the satellite channel will be covered in later chapters.

Let $\mathbf{x} = (x_1, x_2, \cdots, x_\tau, \cdots)$ and $\mathbf{x}' = (x_1', x_2', \ldots, x_\tau', \ldots)$ be the transmitted codeword (signal selector sequence) and decoded codeword respectively. By notational convention, x_k is the signal selector produced by the state transition from time index $k-1$ to k. The decoded codeword $\mathbf{x}'$ is a "first event error" of length τ if the two trellis paths representing $\mathbf{x}$ and $\mathbf{x}'$ diverge at the same state at trellis time index

$k = 0$, remerge for the first time at some state after τ branches, and $x_k = x'_k, k > \tau$. We allow $\mathbf{x}' = \mathbf{x}$ ($\tau = 0$) to be a trivial error event. Note that it may happen that $x_k = x'_k$ for some $k, 1 < k < \tau$. That is, some signal selectors on the first event error path may be "accidently" correct. The squared Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$, denoted as $d^2(\mathbf{x}, \mathbf{x}')$, is

$$d^2(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^{\tau} d^2(x_k, x'_k).$$

We have dropped the subscript ED for Euclidean distance because this is the distance measure we will be mainly concerned with. On the ideal AWGN channel, the squared Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$ determines the likelihood of decoding $\mathbf{x}'$ while $\mathbf{x}$ is sent. Note that we consider only the ML Viterbi decoder which uses unquantized demodulator output and makes a decision at the end of the data. For the fixed-lag and/or quantized Viterbi decoder used in some practical applications, the relationship is slightly weaker.

Let $\mathbf{x}' = \mathbf{x} + \mathbf{e}$ where $\mathbf{e} = (e_1, e_2, \cdots)$ is the "signal selector error sequence" and e_k is obtained by bit-wise modulo-2 addition of x_k and x'_k, e.g., $e_k(x_k, x'_k) = e_k(010, 110) = 100$. If $e_k = 0, k > \ell(\mathbf{e})$, we call $\ell(\mathbf{e})$ the "length" of $\mathbf{e}$. Note that because of the linearity of the convolutional code, $\mathbf{e}$ is also a valid signal selector sequence (codeword), specifically, one which is a first event error of length $\ell(\mathbf{e})$ of the all 0 signal selector sequence. That is, the trellis path representing $\mathbf{e}$ diverges from the all zero path at time index $k = 0$ and merges back to the all zero path for the first time at $k = \ell(\mathbf{e})$. Let $P_d = Q(d/2\sigma)$, where σ is an SNR determined constant factor, d is the Euclidean distance between two codewords and $Q(\cdot)$ is the complementary Gaussian distribution function defined as

$$Q(\gamma) = \frac{1}{\sqrt{2\pi}} \int_{\gamma}^{\infty} e^{-t^2/2}\, dt.$$

Suppose that the transmitted codeword is of finite length, which is the case in practical applications. Let C be the set of all possible codewords and we assume throughout that all codewords are equally likely to be sent. The "first event error

probability" P_e can be expressed and upper bounded as follows. Additions between two codewords are bit-wise modulo-2 additions.

$$\begin{aligned} P_e &= \mathrm{E}[P_e(\mathbf{X})] \\ &= \mathrm{E}\left[\sum_{\mathbf{e}\neq\mathbf{0},\mathbf{e}\in\mathrm{C}} P(\mathbf{X}+\mathbf{e}|\mathbf{X})\right] \qquad (3.1) \\ &= \mathrm{E}\left[\sum_{d=d_{free}}^{\infty} A_d(\mathbf{X})P(\mathbf{X}+\mathbf{e}|\mathbf{X})\right] \qquad (3.2) \\ &\leq \mathrm{E}\left[\sum_d A_d(\mathbf{X})P_d\right] = \sum_d A_d P_d, \qquad (3.3) \end{aligned}$$

where $P_e(\mathbf{X}) = \mathrm{E}[P_e|\mathbf{X}]$ is the conditional first event error probability given that $\mathbf{X}$ is the transmitted codeword, $A_d(\mathbf{X})$ is the number (multiplicity) of codewords at distance d from $\mathbf{X}$, and $A_d = \mathrm{E}[A_d(\mathbf{X})]$ is the expected number of codewords at distance d from a codeword. Equation (3.1) results from the linearity of the convolutional code. Equation (3.2) follows (3.1) by grouping $\mathbf{e}$'s such that $d(\mathbf{X},\mathbf{X}+\mathbf{e}) = d$. The inequality (3.3) holds because the decoding error region $\{\mathbf{X}+\mathbf{e}$ is decoded$\}$ is a subset of the error region when $\mathbf{X}+\mathbf{e}$ and $\mathbf{X}$ are the only two admissible decoding options. Equation (3.3) is the "union bound" of P_e. Note that this bound is valid only if the channel is AWGN. The expectations in (3.1)–(3.3) are taken with respect to all codewords because for non-regular codes the distance structures are not the same for all codewords. That is, $A_d(\mathbf{x}_i) \neq A_d(\mathbf{x}_j), \mathbf{x}_i \neq \mathbf{x}_j$. For regular codes, by definition, we have $A_d(\mathbf{x}) = A_d$ for all $\mathbf{x}$, and thus $P_e(\mathbf{X}) = P_e$.

If the Viterbi decoder we used is maximum likelihood, P_e is the probability that, given the decoded node is on the correct path at some time index k, the next branching decision x_{k+1} will be incorrect. Or equivalently at this moment the decoder will eliminate the correct path in favor of another path with a larger metric. This is because the distance distribution of paths diverging from and merging into a state are identical. If the paths are forced to merge as is done in the practical fixed-lag (near maximum likelihood) decoder, (3.1) is an upper bound of P_e [12]. Usually the

truncation length is chosen to be 4–5 times of the code constraint length and the difference is negligible.

Equation (3.1) is a sum over all possible $\mathbf{x}$ and $\mathbf{e}$ which is difficult to compute analytically even for a simple code. (Suppose that the codeword length is 200 and the input block length $b = 2$. Then $|\mathrm{C}| = (2^2)^{200}$.) The error bound (3.3) is more computable. Its most significant term, $A_{d_{free}} P_{d_{free}}$, can be easily obtained by computer search and is often used as a criterion for code designs. However, TCM codes often have dense distance spectra — there may be many possible distances close to d_{free} with comparable or even much greater multiplicities than $A_{d_{free}}$. Therefore, a code design based on the free distance term alone is not necessarily a better code than another one with a greater d_{free}. This is especially true when at low SNR where $P_{d_{free}}$ is not much more significant than the values of P_d of its neighboring distances. Obviously there is a need for us to be able to evaluate more terms in (3.3).

The bit error probability P_b can be computed and upper bounded similarly. We have

$$\begin{aligned} P_b &= \mathrm{E}[P_b(\mathbf{X})] \\ &= \mathrm{E}\left[\frac{1}{b}\sum_{\mathbf{e}\in\mathrm{C}} \mathrm{N}_b(\mathbf{X}, \mathbf{X}+\mathbf{e}) P(\mathbf{X}+\mathbf{e}|\mathbf{X})\right] \qquad (3.4) \\ &= \frac{1}{b}\mathrm{E}\left[\sum_{d=d_{free}}^{\infty} B_d(\mathbf{X}) P(\mathbf{X}+\mathbf{e}|\mathbf{X})\right] \\ &\leq \frac{1}{b}\sum_{d} B_d P_d, \qquad (3.5) \end{aligned}$$

where $\mathrm{N}_b(\mathbf{X}, \mathbf{X}+\mathbf{e})$ is the number of bit errors if $\mathbf{X}+\mathbf{e}$ is decoded and $\mathbf{X}$ is transmitted ($\mathrm{N}_b(\mathbf{X}, \mathbf{X}) = 0$), $B_d(\mathbf{X})$ is the total number of bit errors for paths at distance d from $\mathbf{X}$, and $B_d = \mathrm{E}[B_d(\mathbf{X})]$ is the average number of bit errors for paths at distance d from the correct path. Note that by linearity of the convolutional code, we have $\mathrm{N}_b(\mathbf{X}, \mathbf{X}+\mathbf{e}) = n_b(\mathbf{e})$, i.e., the number of bit errors depends only on the error sequence $\mathbf{e}$. We also remark that (3.4) is in fact an upper bound of P_b [64]. The bound is tight and the distinction has been usually neglected in the literature.

A profile of A_d or B_d with respect to the distance d is called the distance spectrum of the code. Distance spectra provide us the necessary information to compute the union bounds (3.3) and (3.5). We will want to compare our simulation results against the error bounds as a way to confirm the accuracy of our estimates. Many algorithms have been proposed to obtain the distance spectrum [8]. A fast bi-directional stack algorithm is given by Rouanne and Costello in [44] which we will call the RC Algorithm. With some modification, their algorithm also provide valuable information which can facilitate the simulation. The RC Algorithm works for linear, regular and the so-called quasi-regular codes. For quasi-regular codes (linear and regular codes are special cases), the distance spectrum can be computed by assuming the all-zero sequence is transmitted. This is a very large class of codes which includes the ordinary convolutional codes and many of the practical TCM codes. Recall that for a linear convolutional code with regular mapping, the ED and HD are related by a linear function. In this case, the distance spectrum can be reduced to the commonly known "weight spectrum" after scaling.

Quasi-regularity of a code is an important characteristic which we will exploit in the simulation. Therefore, let us first examine the definition and properties of quasi-regular codes. Let s and s' be two states in the trellis diagram and e be the signal selector error of two signal selectors which are produced by two state transitions originating from s and s'. We define the "distance polynomial" $p_{s,s',e}(z)$ as

$$p_{s,s',e}(z) = \sum_x P(x|s) z^{d^2(x,x+e)}, \tag{3.6}$$

where $P(x|s)$ is the probability of producing the signal selector x given that the encoder is in state s. The polynomial $p_{s,s',e}(z)$ is defined only for the e for which there exist a branch that leaves s and a branch that leaves s' whose signal selector error is e. Also, in general, we have $p_{s,s',e}(z) = p_{s',s,e}(z)$. Examining the trellis diagrams in Figure 3.4, the eight possible distance polynomials are

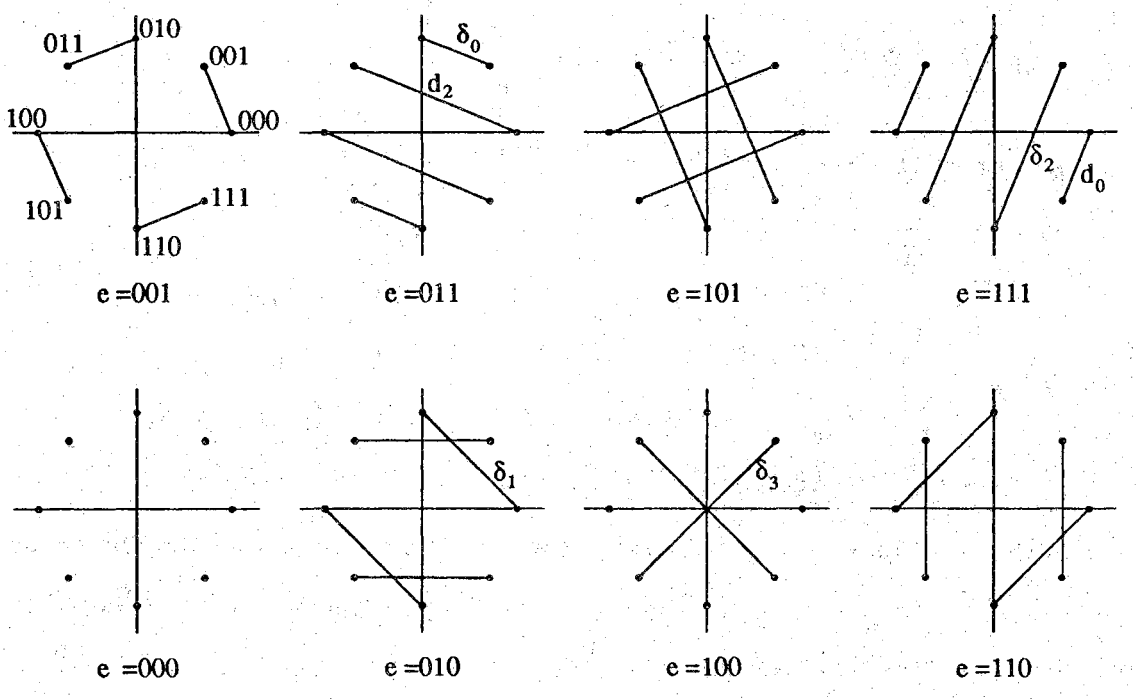


Figure 3.6 Signal selector error and distance for 8-PSK mapping.

$$
\begin{aligned}
p_{s,s',000}(z) &= 1, & p_{s,s',001}(z) &= z^{\delta_0^2}, \\
p_{s,s',010}(z) &= z^{\delta_1^2}, & p_{s,s',011}(z) &= \tfrac{1}{2}z^{\delta_0^2} + \tfrac{1}{2}z^{\delta_2^2}, \\
p_{s,s',100}(z) &= z^{\delta_3^2}, & p_{s,s',101}(z) &= z^{\delta_2^2}, \\
p_{s,s',110}(z) &= z^{\delta_1^2}, & p_{s,s',111}(z) &= \tfrac{1}{2}z^{\delta_0^2} + \tfrac{1}{2}z^{\delta_2^2}.
\end{aligned}
\tag{3.7}
$$

These polynomials can also be represented graphically as in Figure 3.6 where, for every e, all terms in the sum of (3.6) are shown with x and $x + e$ connected by lines. A trellis code is quasi-regular if and only if [44] (1) it consists of a linear binary encoder followed by a mapper and (2) the distance polynomials are independent of the pair of states, i.e., for any two pairs of states (s_1, s_1') and (s_2, s_2'), $p_{s_1,s_1',e}(z) = p_{s_2,s_2',e}(z)$. Regular codes can be considered as a special case of quasi-regular codes where the distance between two codewords depends only on e (the Hamming distance between them) and not on the signal selectors, hence, $p_{s,s',e}(z)$ is a monomial. In the above examples, $p_{s,s',011}(z)$ and $p_{s,s',111}(z)$ are not monomials. Therefore they are not regular codes, but they are quasi-regular because the polynomials do not depend on (s, s').

An example of regular code is a convolutional code with QPSK and Gray coding. In that case the distance polynomials are, independent of the number of states,

$$\begin{cases} p_{s,s',00}(z) = 1, \\ p_{s,s',01}(z) = z^{\delta_1^2}, \\ p_{s,s',10}(z) = z^{\delta_0^2}, \\ p_{s,s',11}(z) = z^{\delta_1^2}, \end{cases}$$

where $\delta_0 = \sqrt{2}$ and $\delta_1 = 2$ for a normalized signal power. Note that they are all monomials and do not depend on (s, s').

From the polynomials in (3.7) or from Figure 3.6, we note that for some signal selector errors there are more than one distance associated with them. That is, $d(x, x+e)$ is not unique which depends on x. In our examples, there are only two such signal selector errors, they are $e \in \{011, 111\} = \mathcal{E}$. For those e, all possible distances and their probabilities of occurrences are contained in the information provided by their distance polynomials. For example, in (3.7), the distance polynomial of $e = 011$ (or $e = 111$) indicates that $d(x, x+e) = \delta_0$ or δ_2 with equal probability. And Figure 3.6 tells us what exactly those pairs of x and $x + e$ are. The "worst-case distance" of e is defined as $d_w(e) = \min_x d(x, x + e)$. For a sequence $\mathbf{e}$, $d_w^2(\mathbf{e}) = \sum_k d_w^2(e_k)$. Here, we have $d_w(011) = d_w(111) = \delta_0$. Note that for $e \notin \mathcal{E}$, we have $d_w(e) = d(x, x + e)$ for all x.

Return to the distance spectrum and RC Algorithm. The original RC Algorithm does not keep track of the signal selector error sequences $\mathbf{e}$ as it progresses. For the purpose of simulation, we do like to have a list of $\mathbf{e}$'s which have small-to-moderate worst-case distances. Therefore we modify the RC Algorithm a little bit and the algorithm can be described as follows. First we assume that the all zero path is the correct path. In this case, the set of all signal selector error sequences (i.e., the set C) is simply the set of first event error paths of the all zero path on the trellis diagram. This set of first event error paths can be sequentially constructed by using a bi-directional (forward and backward) stack algorithm. The output of the algorithm is therefore a list of signal selector error sequences. The multiplicities $A_d(\mathbf{0})$ and $B_d(\mathbf{0})$ can be

obtained from this list. Note that, from Figure 3.6, $d(000,011) = d(000,111) = \delta_0$, i.e., the first event error paths of the all zero path always have the worst-case distances. Therefore, $A_d(\mathbf{0})$ (or $B_d(\mathbf{0})$) versus the distance d can be called the "worst-case distance spectrum." (This is slightly different from Rouanne and Costello's definition of worst-case distance spectrum, but more intuitively satisfactory.) The bi-directional stack algorithm is described formally as follows.

Modified RC Algorithm:

- Input: Convolutional encoder configuration, signal mapping function, and the desired maximum worst-case distance d_{max}.
- Output: A list of signal selector error sequences $\mathbf{e}$ such that $d_w(\mathbf{e}) < d_{max}$, their respective worst-case distances, lengths $\ell(\mathbf{e})$, and information weights $n_b(\mathbf{e})$.
- Method:
  1. Construct tables containing trellis diagram information.
  2. Forward (or backward alternatively) extension. For the successor (predecessor) state is not the zero state, extend path. Push new paths into the forward (backward) stack with updated terminal states, worst-case distances, information weights, lengths, and error sequences $\mathbf{e}$. Delete old path.
  3. For the newly created paths, check mergers with all paths in the opposite direction stack. If mergers happen, push the complete paths into the output stack. If $d > d_{max}$, stop.
  4. Choose the path with shortest distance and length in the stack. Go to 2.

The multiplicity $A_d(\mathbf{0})$, for $d < d_{max}$, is obtained by simply counting the number of $\mathbf{e}$ that have the same worst-case distance d. $B_d(\mathbf{0})$ is obtained similarly except each count is weighted by $n_b(\mathbf{e})$. We are, however, more interested in the distance

spectrum. For quasi-regular codes, the distance spectrum can be easily computed using the distance polynomials. For example, suppose that $A_{d_{free}}(\mathbf{0}) = 6$, i.e., there are 6 error sequences having the same worst-case distance d_{free}. For a fixed $\mathbf{e}$, let $n_{\mathcal{E}}(\mathbf{e})$ denote the number of appearances of $e_k, k = 1, \ldots, \ell(\mathbf{e})$, such that $e_k \in \mathcal{E}$. Assume that, for these 6 error sequences, there are two with $n_{\mathcal{E}}(\mathbf{e}) = 2$, three with $n_{\mathcal{E}}(\mathbf{e}) = 1$ and one with $n_{\mathcal{E}}(\mathbf{e}) = 0$. See the distance polynomials in (3.7). We find that for each appearance of $e_k \in \mathcal{E}$, there are 2 equally likely distances (δ_0 and δ_2) depending on the transmitted code symbol x_k. Therefore, for a fixed $\mathbf{e}$, the probability of having the worst-case distance is $(1/2)^{n_{\mathcal{E}}(\mathbf{e})}$. Hence, we have $A_{d_{free}} = 1 \cdot (1/2)^0 + 3 \cdot (1/2)^1 + 2 \cdot (1/2)^2 = 3$. Note that if we are considering $A_d, d > d_{free}$, the contribution to the multiplicity from error sequences with worst-case distances less than d must be also considered. Furthermore, we are fortunate that, for this 8-PSK with the natural mapping, the distance polynomials for all elements in $\mathcal{E}$ are identical. If this is not the case, we have to count the appearance of each element (of $\mathcal{E}$) in $\mathbf{e}$ separately rather than using the total number $n_{\mathcal{E}}(\mathbf{e})$.

This algorithm works for all quasi-regular codes which include the codes that can be analyzed by Zehavi and Wolf's [71] transfer function bounds. An even broader class of codes than quasi-regular is defined in Schlegel [51] for which the distance spectrum is computable. In there, the Euclidean distance between two codewords is expressed as a quadratic form. This expression makes possible the computation of distance spectrum for some codes even when linear ISI is present. The utilization of his method to our simulation problem is left for future work.

For small SNR's, the error bounds (3.3) and (3.5) are loose. Also, they do not hold if the channel model is not the ideal AWGN. In those cases, simulations are desired. To estimate P_e and P_b, we will be working with (3.1) and (3.4). Note that the two equations are in nature event-oriented in which the term $P(\mathbf{X} + \mathbf{e}|\mathbf{X})$, the probability of the event "decoding $\mathbf{X} + \mathbf{e}$ while $\mathbf{X}$ is transmitted," is what we will be estimating. One major advantage of this event simulation approach is that we can concentrate our computation on the terms that have significant contributions to the

values of P_e and P_b. We will use the distance information to identify these terms in Chapters 5 and 6. Details and justification of the event simulation method for trellis codes will be given in the next chapter. As mentioned earlier, this event simulation approach is conceptually different from the conventional stream simulation which is simply an imitation of the system operation.

3.3 A Satellite Channel Model

The satellite channel [16, 23, 55, 61] is the channel model we will be mainly concerned with besides the ideal AWGN channel. A simplified block diagram of a digital communication system operating on a bend-pipe (non-regenerative) type satellite channel is shown in Figure 3.7. A passband model is described below although the simulation will be conducted entirely on the baseband by using the complex baseband representation for all relevant waveforms and components.

The transmit IF (intermediate frequency) bandpass filter (BPF) $H_1(f)$ is mainly for pulse shaping as well as reducing the spectral energy spread. Its selection is an important part of the system design and is basically a tradeoff between the in-band noise power, ISI and cross-channel interference. A Nyquist, Butterworth or Chebyshev filter is often used. For the most commonly used Nyquist filter, the pulse shaping function is usually evenly shared by the transmit and receive filters and thus $H_1(f)$ is a square-root Nyquist filter. This filter is often compensated, resulting in a notch filter, because the Nyquist filter is ISI-free only if an impulse train is transmitted. The roll-off rate of the Nyquist filter needs to be carefully designed. Note that filtering generates signal envelope variation, therefore the selection of $H_1(f)$ also depends on the modulation scheme and the nonlinearity of the channel. In our simulation program, we will allow the discrete baseband equivalents of all filters to be any finite impulse response functions (FIR). The baseband equivalent of a bandpass filter can be easily computed from the transfer function $H_1(f)$ or the impulse response $h_1(t)$ [43]. Our program will specify all filters directly in the baseband.

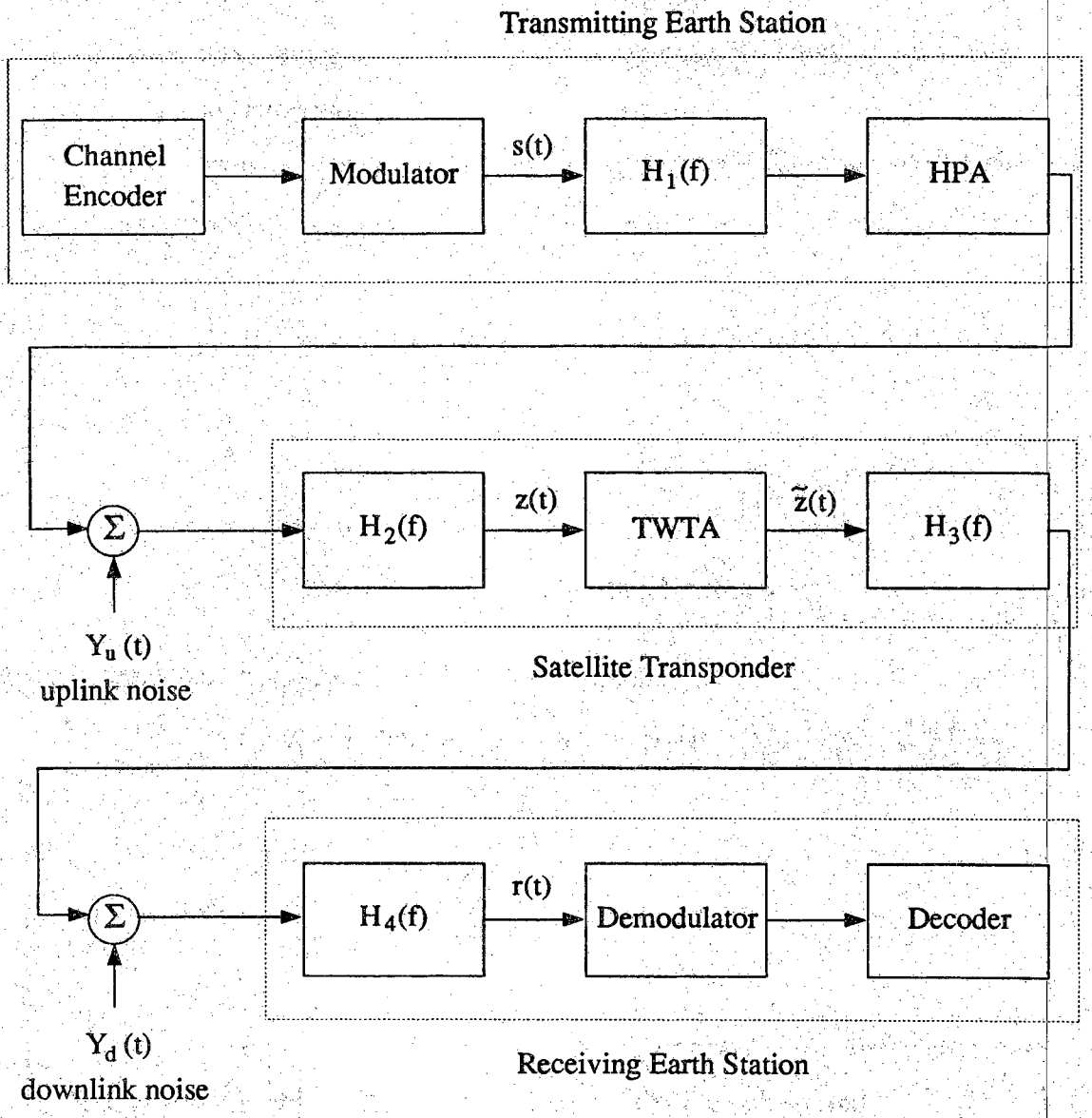


Figure 3.7 Simplified block diagram of a satellite communication system.

Filter operations tend to be one of the more CPU-intensive parts of the simulation. For this reason, many Monte Carlo simulations are performed in the frequency domain and use the FFT for filtering rather than the more time consuming convolution. The FFT is more efficient than convolution if the data length is large. For our importance sampling algorithm employing the event simulation method, the data length is usually small-to-moderate, and the computation required to find the optimal mean-translation simulation density turns out to be the dominant cost of computation. Therefore, we will simulate the system entirely in the time domain.

We have omitted the up-converter following $H_1(f)$ because it is transparent in our complex baseband simulation. The up-converter translates the carrier frequency from IF to RF (radio frequency), the latter is usually in the L band (1–2 GHz), C band (4–6 GHz) or Ku band (12–14 GHz) for commercial geostationary satellite systems such as the MSAT (Mobile Satellite) and INTELSAT (International Telecommunications Satellite) systems. The earth station high power amplifier (HPA) provides uplink transmitting power which can be as high as several thousand watts to overcome the great ($\sim$ 200 dB in Ku band) free space loss. A wideband BPF sometimes appears after HPA to reduce the cross-channel interference.

The uplink (UL) and downlink (DL) noises are assumed to be independent additive white Gaussian processes. As mentioned at the beginning of this chapter, they are very good models for the satellite channel noise. Our algorithm actually allows the two noise processes to be correlated but the computation in this case will be considerably more complex. For some applications where the fixed earth station is able to provide large uplink power such that, in spite of the small receive antenna size and antenna gain at the satellite, the uplink enjoys a much greater SNR than the downlink which has large earth receive antenna but limited satellite on-board power. In such cases, it is a common practice to ignore the uplink noise (SNR $= \infty$) and add downlink noise analytically. This greatly simplifies the analysis and simulation since only the uplink noise is passed through the satellite nonlinear amplifier. This practice of course is not valid [56] in many other applications such as the MSAT and VSAT (Very Small

Aperture Terminal) satellite systems where the transmit power is limited and the uplink SNR may be even less than the downlink SNR. Our IS algorithm is capable of handling any combination of uplink and downlink SNR's. If the system is linear, the overall system SNR, in terms of E_b/N_0, can be expressed as [16]

$$\left(\frac{E_b}{N_0}\right)_{total} = \frac{(E_b/N_0)_u (E_b/N_0)_d}{(E_b/N_0)_u + (E_b/N_0)_d}, \tag{3.8}$$

where the subscripts "u" and "d" denote uplink and downlink respectively. If the channel is nonlinear, the above equation is only an approximation.

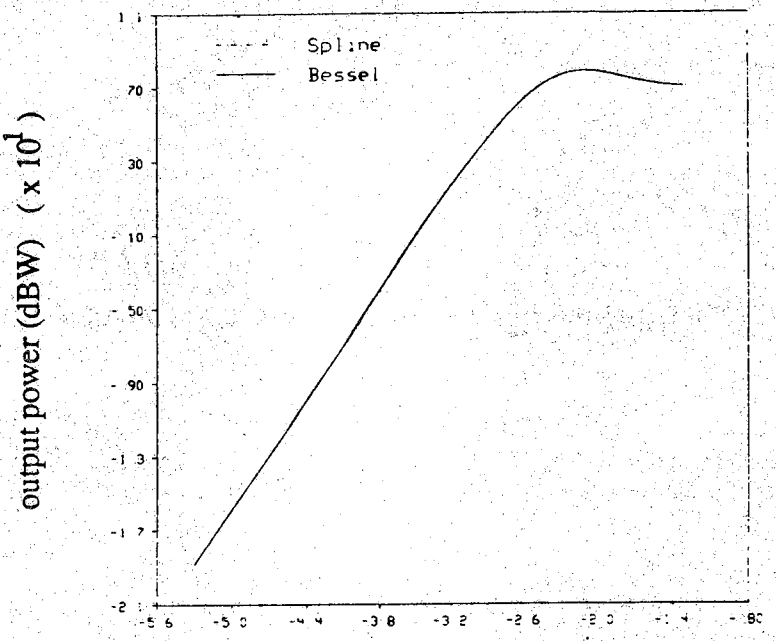
The satellite input filter $H_2(f)$ is in fact a cascade of filters, including the low noise amplifier (LNA), receive filter and demultiplexing filter. The transponder HPA, usually in the form of TWTA, will be described shortly. A frequency translation from uplink to downlink on the transponder is omitted in the diagram. The satellite output filter $H_3(f)$ is also the combination of several filters which are responsible for reducing the intermodulation and spectral spreading (and hence adjacent channel interference or ACI) caused by the TWTA. Note that, although only one channel is shown in Figure 3.7, there are usually many channels sharing one common transponder.

At the receiving end, the presence of ACI, co-channel interference (CCI) and multipath effect usually can be modeled as additive noise processes before the receive filter $H_4(f)$. For example, in [27], the total effect of the ACI and multipath is modeled as a signal-correlated Gaussian process. The earth receive filter $H_4(f)$ includes the front-end LNA and BPFs. A frequency down-converter before the demodulator is again omitted. The demodulator removes the carrier and produces one (complex) sample per symbol interval T. We will sample the demodulator output waveform at the midpoint of each symbol interval, but the program will allow search for the optimal sampling timing within the signaling interval. We will assume that the demodulator is synchronized with the transmitter such that the mid-band group delay is zero. However, since the algorithm will accept any FIR filters, static synchronizer phase error $\Delta\phi$ and/or frequency offset Δf could be easily incorporated by introducing a complex phase factor $\exp(j[\Delta\phi + 2\pi\Delta f t])$ to the receive filter impulse response

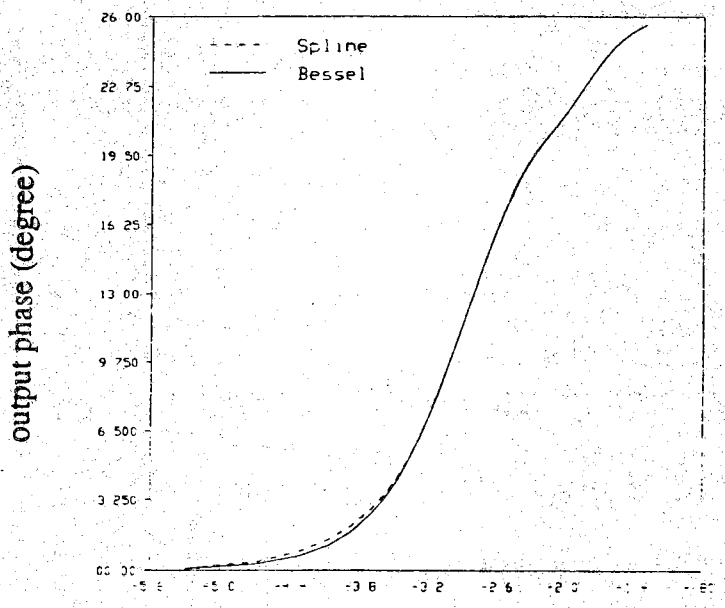
function. Likewise, a static timing error $\Delta\tau$ can be modeled by simply time-shifting the receive filter impulse response function. A perfect symbol timing will be also assumed.

The coding and modulation schemes we will cover include the uncoded BPSK, QPSK and MSK, convolutional codes with BPSK/QPSK and 8-PSK TCM codes. Time domain pulse shaping is permissible at the modulator although the rectangular pulse is the norm. Soft-decision ML Viterbi decoders, which accept the complex demodulator sampler output without quantization, will be used for convolutional and TCM codes. The branch metric function which compares a single demodulator output symbol r and a signal selector x is $Re[s(x)^*r]$ where $Re[\cdot]$ denotes "real part," $s(x)$ is the complex signal pulse amplitude selected by x, and the superscript * is the complex conjugate. This is the maximum likelihood branch metric for a 1 or 2-dimensional signaling operating on a linear memoryless channel.

The satellite HPA usually is the light-weight, small-sized traveling wave tube amplifier (TWTA). To achieve power efficiency, it is often driven at near saturation. This introduces the so-called AM/AM and AM/PM conversion effects, where AM and PM stand for amplitude modulation and phase modulation respectively. This amplitude and phase nonlinearity is the main reason which makes analysis difficult. Also, since there are usually many channels sharing one common transponder, TWTA is also known to cause intermodulation and spectral spreading and thus the ACI. An example of TWTA AM/AM and AM/PM characteristics is shown in Figure 3.8. This is for the Hughes 261H TWTA used in the INTELSAT VI and V. The curves labeled "Bessel" and "Spline" are curve fits to the measured data (not shown) using Bessel functions and third order polynomials spline approximation respectively. They closely agree to the measured data in all normal operating regions and are indistinguishable except in the low input power region of the AM/PM curve. A TWTA is usually specified by its input power backoff, IBO, or its corresponding output backoff. The IBO is the difference in dB between the actual average input power of the modulated signal and the carrier power input to the TWTA required to saturate it. For example,



(a) AM/AM conversion.



(b) AM/PM conversion.

Figure 3.8 Hughes 261H TWTA nonlinear characteristics.

in Figure 3.8, a 4 dB input backoff (IBO = 4 dB) from the saturation point (-21 dBmW) is equivalent to a 1 dB output backoff.

For the purpose of simulation, we need a complex baseband model for the TWTA. A quadrature model described in [3, 29] fits our purpose perfectly and the curves labeled "Bessel" in Figure 3.8 are the result of this model. For our IS algorithm, we prefer functional descriptions of the nonlinearity to be differentiable. We now investigate this quadrature model. The output signal of the TWTA responding to an input signal of the form

$$\begin{aligned} z(t) &= A(t)\cos(\omega_c t + \theta(t)) \\ &= z^I(t)\cos(\omega_c t) - z^Q(t)\sin(\omega_c t), \end{aligned}$$

where $z^I(t) = A(t)\cos\theta(t)$ and $z^Q(t) = A(t)\sin\theta(t)$, can be expressed as

$$\begin{aligned} \tilde{z}(t) &= U(A(t))\cos[\omega_c t + \theta(t) + G(A(t))] \\ &= R^I(A(t))\cos(\omega_c t + \theta(t)) - R^Q(A(t))\sin(\omega_c t + \theta(t)) \\ &= \tilde{z}^I(t)\cos(\omega_c t) - \tilde{z}^Q(t)\sin(\omega_c t). \end{aligned}$$

The instantaneous quadrature nonlinearity $R^I(A) = U(A)\cos(G(A))$ and $R^Q(A) = U(A)\sin(G(A))$ can be approximated by Bessel functions:

$$R^I(A) = c_1 A e^{-c_2 A^2} I_0(c_2 A^2) \tag{3.9}$$

$$R^Q(A) = s_1 A e^{-s_2 A^2} I_1(s_2 A^2), \tag{3.10}$$

where $I_0(\cdot)$ and $I_1(\cdot)$ are modified Bessel functions of the first kind of order 0 and 1 respectively and the constants are $c_1 = 1.61245$, $c_2 = 0.053557$, $s_1 = 1.71850$ and $s_2 = 0.242218$ for this particular TWTA.

The input-output relationship between the instantaneous I and Q channel envelopes can thus be obtained as follows.

$$\begin{aligned} \tilde{z}^I &= U(A)\cos(\theta + G(A)) \\ &= R^I\cos(\theta) - R^Q\sin(\theta) \\ &= c_1\eta_0(c_2 A^2)z^I - s_1\eta_1(s_2 A^2)z^Q, \end{aligned} \tag{3.11}$$

where $\eta_0(\alpha) = e^{-|\alpha|}I_0(\alpha)$ and $\eta_1(\alpha) = e^{-|\alpha|}I_1(\alpha)$ are exponentially scaled modified Bessel functions which are available as IMSL subroutines [30]. Similarly, it can be shown that

$$\tilde{z}^Q = c_1\eta_0(c_2A^2)z^Q + s_1\eta_1(s_2A^2)z^I \tag{3.12}$$

To find the optimal mean-translation simulation density, we will show that it is necessary to take derivatives of (3.11) and (3.12). An alternative to the Bessel approximation which allows quick computation of derivatives is the spline polynomial approximation. For example, we can approximate $R^I(A)$ and $R^Q(A)$ piecewisely with third order polynomials. See Figure 3.8. In order to locate each region easily, we divide the input amplitude A into 10 regions: [0,1), [1,2), ..., and [10,∞). Therefore the integer part of A is the region index. In each region, the approximations are

$$R^I(A) = C_3A^3 + C_2A^2 + C_1A + C_0 \tag{3.13}$$
$$R^Q(A) = S_3A^3 + S_2A^2 + S_1A + S_0. \tag{3.14}$$

Next we have to find the coefficients C_i and $S_i, i = 0,1,2,3$. Note that the derivatives of (3.13) and (3.14) are

$$R^I(A)' = 3C_3A^2 + 2C_2A + C_1$$
$$R^Q(A)' = 3S_3A^2 + 2S_2A + S_1.$$

Taking derivatives of (3.9) and (3.10), C_i and S_i for each region can thus be obtained by solving the following linear system equations. Note that we use the beginning and end points of each region for boundary conditions to ensure continuity. Hence there are 8 equations and 8 unknowns.

$$\begin{aligned}
C_3A^3 + C_2A^2 + C_1A + C_0 &= c_1A\eta_0(c_2A^2) \\
S_3A^3 + S_2A^2 + S_1A + S_0 &= s_1A\eta_1(s_2A^2) \\
3C_3A^2 + 2C_2A + C_1 &= c_1\eta_0(c_2A^2) + 2c_1c_2A^2[\eta_1(c_2A^2) - \eta_0(c_2A^2)] \\
3S_3A^2 + 2S_2A + S_1 &= s_1\eta_1(s_2A^2) + \\
&\quad 2s_1s_2A^2[\eta_0(s_2A^2) - (1 + \frac{1}{s_2A^2})\eta_1(s_2A^2)].
\end{aligned}$$

The derivatives of Bessel functions follow [1] $I_0'(x) = I_1(x)$ and $I_1'(x) = I_0(x) - \frac{1}{x}I_1(x)$. The last region [10,$\infty$) should be handled differently. We set $R^I(A) = R^I(10) = C_0$ and $R^Q(A) = R^Q(10) = S_0$ for $A \geq 10$. The curves labeled "Spline" in Figure 3.8 are the result of this model which agree closely to the Bessel approximation.

With the spline approximation, it is easy to show that the input and output I and Q channel envelopes are related by:

$$\begin{aligned} \tilde{z}^I &= R^I(A)\cos\theta - R^Q(A)\sin\theta \\ &= (C_3A^2 + C_2A + C_1 + \frac{C_0}{A})z^I - (S_3A^2 + S_2A + S_1 + \frac{S_0}{A})z^Q \end{aligned} \tag{3.15}$$

and

$$\tilde{z}^Q = (S_3A^2 + S_2A + S_1 + \frac{S_0}{A})z^I + (C_3A^2 + C_2A + C_1 + \frac{C_0}{A})z^Q \tag{3.16}$$

We have described a satellite channel model in detail. For system simulations, subroutines often are not only related by inputs and outputs but also some system parameters. This is especially true for importance sampling program. Therefore, it is crucial to have a good understanding of the system. One important observation of a bandlimited nonlinear channel such as the satellite channel is that the probability of deciding on one codeword while the other is sent depends not only on the Euclidean distance between them but also the ISI and nonlinearity. Therefore, the Euclidean distance spectrum which we use to obtain the union bound of P_b for the AWGN channel is no longer valid. Nonetheless, Euclidean distance information is still useful in designing the signal biasing which we will discuss in Chapter 6.

4. IMPORTANCE SAMPLING THEORY

4.1 Systems with Gaussian Noise Inputs

In this section, we will develop the optimal Gaussian simulation distribution for systems with Gaussian noise inputs. The reason for restricting our candidate family for the simulation density to be Gaussian is stated in Chapter 2. Increasing the size of the candidate family surely increases the possibility of a higher efficiency gain. But while doing this, we must make sure that the expansion still permits optimization as well as easy generation of samples and computation of the IS weights.

Consider the general block diagram of Figure 4.1 for a digital communication system whose single output decision is based on the noise input vector $\mathbf{Y}$ and the "signal" input vector $\mathbf{x}$. Let $\mathbf{Y}$ be an n-dimensional zero mean Gaussian random vector with covariance matrix $\sigma^2\mathbf{I}$. (There is no loss of generality by assuming a diagonal covariance matrix because any correlation structure can be realized by including a linear transformation in the system model. Therefore, colored noise can also be considered.) The m-dimensional vector $\mathbf{x}$ models any other inputs (such as the ISI and/or error event pattern which we will discuss in the following sections) that have

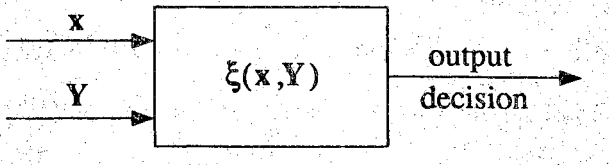


Figure 4.1 A general system block diagram with noise and signal inputs.

influence on the output decision. Let $\xi(\cdot,\cdot)$ be the "system response function" such that the output decision statistic D can be expressed as $D = \xi(\mathbf{x}, \mathbf{Y})$. The function $\xi(\cdot,\cdot)$, possibly nonlinear, may be fairly complex and need not be in closed form. Suppose that the decision is performed by a threshold test $D \gtrless 0$, and the decision error occurs when $D \geq 0$. This binary decision rule may seem inappropriate at first glance for some applications such as uncoded M-ary signalings or coded communications where more than two decision options are admissible. But we can always pick one particular decision error (one that would occur with high probability) and formulate it as a binary decision problem, i.e., we are only interested in one particular type of decision error. How this is done and why will be discussed in more detailed later.

We are interested in estimating the decision error probability $P_\gamma(\mathbf{Y} \in E(\mathbf{x}))$ (or the mean of $1_{E(\mathbf{x})}(\mathbf{Y})$ by the formulation of Chapter 2) where $\gamma = 1/\sigma^2$ is a normalized SNR parameter and $E(\mathbf{x}) = \{\mathbf{y} : \xi(\mathbf{x}, \mathbf{y}) \geq 0\}$ is called the "error set" of $\mathbf{x}$ which in general depends strongly on $\mathbf{x}$. However, at this level of the simulation algorithm, we will let $\mathbf{x}$ be a fixed parameter vector and thus $E(\mathbf{x})$ is a fixed subset of $\mathbf{R}^n$. Therefore, in this section we will usually suppress the dependency on $\mathbf{x}$ in the notation and denote the error set $E = E(\mathbf{x})$. We are primarily interested in the high SNR case and we will gain some important insight by considering asymptotic behavior as $\gamma \to \infty$ ($\sigma^2 \to 0$).

A special case of particular interest is $\xi(\mathbf{x}, \mathbf{y}) = a(\mathbf{x}) + \mathbf{b}(\mathbf{x})^t\mathbf{y}$, where $a(\cdot)$ is a real-valued function and $\mathbf{b} : \mathbf{R}^m \to \mathbf{R}^n$ is an n-dimensional vector function. This is an "affine" function of $\mathbf{y}$. However, we will refer to this as a "linear system" (even though strictly speaking it is not linear in $\mathbf{y}$) because the contribution from $\mathbf{y}$ to the decision voltage D is a linear function of $\mathbf{y}$. We only require that the system be affine in $\mathbf{y}$, any type of dependency for vector $\mathbf{x}$ is admissible. ($a(\mathbf{x})$ is treated as a constant and $\mathbf{b}(\mathbf{x})$ as a constant vector.) In this case, we have $E = E(\mathbf{x}) = \{\mathbf{y} : \mathbf{b}(\mathbf{x})^t\mathbf{y} \geq -a(\mathbf{x})\}$ which is a "half space" in $\mathbf{R}^n$.

We will assume throughout that the error set E has a nonempty interior and no isolated points, that is, $\bar{E} = \overline{E^o}$ where $\bar{E}$ and E^o denote the closure and interior of E respectively. We also assume $\mathbf{0} \notin \bar{E}$. ($\mathbf{0} \in \bar{E}$ means that the system is more likely

to make incorrect decisions than correct decisions for large γ.) Decision regions of practical interest always satisfy these simple regularity conditions. The boundary of E will be denoted $\partial E = \bar{E} \backslash E^o$. The notation $a_\gamma \sim b_\gamma$ means that $\lim_{\gamma \to \infty} a_\gamma / b_\gamma = 1$ and that this limit exists. Also, $Q(z) = \int_z^\infty (2\pi)^{-1/2} \exp(-\zeta^2/2)\, d\zeta$ is the standard Gaussian complementary distribution function.

There are several concepts to be presented in this section. For sake of clarity, we separate major arguments with numbered titles.

1. $P_\gamma(\mathbf{Y} \in E)$ vanishes exponentially fast as a function of γ.

Expressing the decision error probability as an integral, we have $P_\gamma(\mathbf{Y} \in E) = \int_E f_{\mathbf{Y}}(\mathbf{y})\, d\mathbf{y}$. Note that the integrand $f_{\mathbf{Y}}(\mathbf{y}) \propto \exp(-\gamma \|\mathbf{y}\|^2/2)$ vanishes exponentially fast as $\gamma \to \infty$. More specifically, $-\gamma^{-1} \log(f_{\mathbf{Y}}(\mathbf{y})) \sim \|\mathbf{y}\|^2/2$. We call $\|\mathbf{y}\|^2/2$ the "exponential rate of decrease" of $f_{\mathbf{Y}}(\mathbf{y})$ with respect to γ. By a classical asymptotic method due to Laplace, the rate of decrease of the integral $\int_E f_{\mathbf{Y}}(\mathbf{y})\, d\mathbf{y}$ is the minimum rate of decrease of the integrand over the range of integration E [14]. Under our regularity condition $\bar{E} = \overline{E^o}$, we may apply Laplace's method to conclude that the exponential rate of decrease of the decision error probability $P_\gamma(\mathbf{Y} \in E)$ is

$$-\lim_{\gamma \to \infty} \gamma^{-1} \log\left(P_\gamma(\mathbf{Y} \in E)\right) \quad = \quad \inf_{\mathbf{y} \in E} \frac{1}{2} \|\mathbf{y}\|^2. \tag{4.1}$$

We will call a point $\mathbf{v} \in \partial E$ such that $\|\mathbf{v}\|^2 = \inf_{\mathbf{y} \in E} \|\mathbf{y}\|^2$ a "minimum rate point." Therefore, a minimum rate point is a boundary point of E with the smallest L_2 norm, i.e., it is closest to the origin among all points in the error set E. Figure 4.2 shows an $n = 2$ example of an error set and a minimum rate point. Note that the minimum rate point may not be unique. If there is only one minimum rate point, we will call it the "dominating point." Since we have assumed that $\mathbf{0} \notin \bar{E}$, at least one minimum rate point always exists. Hence the right hand side of (4.1) can be replaced by $\|\mathbf{v}\|^2/2$. In other words, we have $P_\gamma(\mathbf{Y} \in E) = a_\gamma \exp(-(\|\mathbf{v}\|^2/2)\gamma)$ where the factor a_γ decays slower than the exponential rate, that is, $\lim_{\gamma \to \infty} \gamma^{-1} \log(a_\gamma) = 0$.

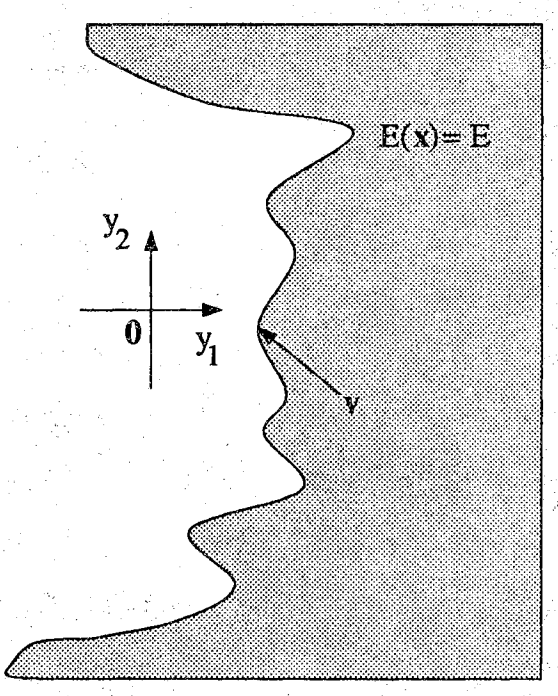


Figure 4.2 An $n = 2$ example of an error set and a minimum rate point.

Example 4.1:

Let us take the above special case of linear systems for example. The minimum rate point can be obtained by solving the following minimization problem:

$$\begin{cases} \min \ \|\mathbf{y}\|^2, \\ \text{subject to} \quad \xi(\mathbf{x},\mathbf{y}) = a(\mathbf{x}) + \mathbf{b}(\mathbf{x})^t\mathbf{y} = 0. \end{cases}$$

The minimization problem is to find $\mathbf{y}$ which are closest to the origin under the constraint that $\mathbf{y} \in \partial E$. Assume that $\xi : \mathbf{R}^m \times \mathbf{R}^n \to \mathbf{R}$ is continuous and differentiable with respect to $\mathbf{y}$. By continuity, we have $\partial E = \xi^{-1}(0) = \{\mathbf{y} : D = \xi(\mathbf{x},\mathbf{y}) = 0\}$, and thus the constraint. Using the Lagrange method [35], it is easy to show that the unique minimum rate point (dominating point) is $\mathbf{v} = \mathbf{v}(\mathbf{x}) = -(a(\mathbf{x})\mathbf{b}(\mathbf{x}))/\|\mathbf{b}(\mathbf{x})\|^2$. Also, the error set can be written as $E = \{\mathbf{y} : \mathbf{v}^t(\mathbf{y} - \mathbf{v}) \geq 0\}$ which is a half space tangent to the dominating point vector $\mathbf{v}$. It follows that $P_\gamma(\mathbf{Y} \in E) = P_\gamma(D \geq 0) = Q(\gamma^{-1/2}a(\mathbf{x})/\|\mathbf{b}(\mathbf{x})\|) = Q\left(\sqrt{\gamma\|\mathbf{v}\|^2}\right)$. ($D$ is a real Gaussian random variable with mean $a(\mathbf{x})$ and variance $\|\mathbf{b}(\mathbf{x})\|^2/\gamma$.) Using the well-known approximation

$Q(z) \sim (2\pi z^2)^{-1/2}\exp(-z^2/2)$ for large z, we have $P_\gamma(\mathbf{Y} \in E) = a_\gamma \exp(-(\|\mathbf{v}\|^2/2)\gamma)$ where $a_\gamma \sim (2\pi\|\mathbf{v}\|^2\gamma)^{-1/2}$. Therefore $P_\gamma(\mathbf{Y} \in E)$ indeed vanish with exponential rate $\|\mathbf{v}\|^2/2$. ¶

2. $L_\epsilon(\gamma)$, the required number of simulation runs for estimating $P_\gamma(\mathbf{Y} \in E)$, may increase exponentially fast as a function of γ.

Consider the simulation distribution to be Gaussian which is specified by its mean vector $\mathbf{v}_\gamma^*$ and its covariance matrix $\mathbf{C}_\gamma^*$. This is the most general formulation for any Gaussian simulation densities because both $\mathbf{v}_\gamma^*$ and $\mathbf{C}_\gamma^*$ can be functions of γ. Let $L_\epsilon(\gamma)$ be the number of simulation runs required to estimate $P_\gamma(\mathbf{Y} \in E)$ for a standard error ϵ. $L_\epsilon(\gamma)$ is found to be as (2.9) except now it is also a function of the SNR factor γ. Since per sample computational costs are the same for any Gaussian simulation distributions, we can use $L_\epsilon(\gamma)$ as the measure of computational cost. Using $\mathrm{var}[Z] = E[Z^2] - E[Z]^2$, we can rewrite (2.9) as

$$L_\epsilon(\gamma) \quad = \quad \left\lceil \epsilon^{-2} \left(\frac{\mathrm{E}_\gamma^* \left[(1_E(\mathbf{Y})w(\mathbf{Y}))^2 \right]}{P(\mathbf{Y} \in E)^2} - 1 \right) \right\rceil . \tag{4.2}$$

Proposition 1: Assume that the limit

$$\rho \quad = \quad - \lim_{\gamma \to \infty} \gamma^{-1} \log \left(\mathrm{E}_\gamma^* \left[(1_E(\mathbf{Y})w(\mathbf{Y}))^2 \right] \right) \tag{4.3}$$

exists. Then

$$\lim_{\gamma \to \infty} \gamma^{-1} \log (L_\epsilon(\gamma)) \quad = \quad \|\mathbf{v}\|^2 - \rho \tag{4.4}$$

is the "exponential rate of increase" of the number of simulation runs $L_\epsilon(\gamma)$. Moreover, $\rho \leq \|\mathbf{v}\|^2$, and hence, the exponential rate of increase in (4.4) is always ≥ 0.

Proof: Following the result of previous discussion, $P_\gamma(\mathbf{Y} \in E)^2$ vanishes with exponential rate $2 \times (\|\mathbf{v}\|^2/2) = \|\mathbf{v}\|^2$. Notice that for any $\gamma < \infty$, we should have $\mathrm{var}^*[1_E(\mathbf{Y})w(\mathbf{Y})] = \mathrm{E}_\gamma^*[(1_E(\mathbf{Y})w(\mathbf{Y}))^2] - P_\gamma(\mathbf{Y} \in E)^2 \geq 0$. Therefore, if $\rho > \|\mathbf{v}\|^2$

(ρ is the exponential rate of decrease of $\mathrm{E}^*_\gamma[(1_E(\mathbf{Y})w(\mathbf{Y}))^2]$) then $\mathrm{E}^*_\gamma[(1_E(\mathbf{Y})w(\mathbf{Y}))^2]$ would vanish faster than $P(\mathbf{Y} \in E)^2$ and this would violate $\mathrm{var}^*[1_E(\mathbf{Y})w(\mathbf{Y})] \geq 0$ for large γ. Therefore we must have $\rho \leq \|\mathrm{v}\|^2$.

If $\rho < \|\mathrm{v}\|^2$, then (4.4) follows by simply taking the logarithm of (4.2). If $\rho = \|\mathrm{v}\|^2$, there is a possibility of cancellation in the argument of the $\lceil\cdot\rceil$ function in (4.2) causing this argument to decrease possibly even faster than exponentially. (For example, in the case of the unconstrained minimum variance solution (2.10), the cancellation is $1 - 1 = 0$ because $\mathrm{E}^*_\gamma[(1_E(\mathbf{Y})w(\mathbf{Y}))^2] = P_\gamma(\mathbf{Y} \in E)^2$.) However, even if this happens we still have $L_\epsilon(\gamma) \geq 1$ (because of the $\lceil\cdot\rceil$ function). Therefore, the exponential rate of increase ($\sim \gamma^{-1}\log(L_\epsilon(\gamma))$) cannot be negative. ¶

Now, our goal is to specify $\mathbf{v}^*_\gamma$ and $\mathbf{C}^*_\gamma$ to minimize $L_\epsilon(\gamma)$. Or, asymptotically we should minimize the rate of growth of $L_\epsilon(\gamma)$. The term $\|\mathrm{v}\|^2$ in (4.4) is fixed because the minimum rate points do not depend on the SNR. Hence, we should maximize $\rho \leq \|\mathrm{v}\|^2$. We say that an importance sampling scheme is "exponentially efficient" if $\rho = \|\mathrm{v}\|^2$. (This is called the asymptotic efficiency in [49].) Clearly, any scheme for which $\rho < \|\mathrm{v}\|^2$ is ultimately very inefficient due to its exponentially increasing computational cost as indicated by (4.4).

3. For any "stable" variance-scaling scheme, $L_\epsilon(\gamma)$ does not increase exponentially fast if the simulation density is the optimized mean-translation biasing and the "forbidden set condition" is satisfied.

To maximize ρ with hope of achieving the exponential efficiency, we must characterize ρ in terms of $\mathbf{v}^*_\gamma$ and $\mathbf{C}^*_\gamma$. To do this, we first express the expectation $\mathrm{E}^*_\gamma[(1_E(\mathbf{Y})w(\mathbf{Y}))^2]$ as an integral

$$\mathrm{E}^*_\gamma\left[(1_E(\mathbf{Y})w(\mathbf{Y}))^2\right]$$

$$= \int_E w(\mathrm{y})^2 f^*_{\mathbf{Y}}(\mathrm{y})\,d\mathrm{y} \quad = \quad \int_E \frac{f_{\mathbf{Y}}(\mathrm{y})^2}{f^*_{\mathbf{Y}}(\mathrm{y})}\,d\mathrm{y}$$

$$= \left(\frac{\gamma}{2\pi}\right)^{n/2} \det\left(\gamma \mathbf{C}_\gamma^*\right)^{1/2}$$

$$\times \int_E \exp\left(-\gamma\|\mathbf{y}\|^2 + \frac{1}{2}(\mathbf{y}-\mathbf{v}_\gamma^*)^t \mathbf{C}_\gamma^{*-1}(\mathbf{y}-\mathbf{v}_\gamma^*)\right) d\mathbf{y}. \quad (4.5)$$

Assuming $\mathbf{v}_\gamma^* \sim \mathbf{v}_\infty^*$ and $\gamma\mathbf{C}_\gamma^* \sim \mathbf{C}_\infty^*$, we apply the Laplace's method and (4.3) to obtain

$$\rho = -\lim_{\gamma\to\infty} \gamma^{-1} \log\left(\det\left(\gamma\mathbf{C}_\gamma^*\right)^{1/2}\right)$$

$$+ \inf_{\mathbf{y}\in E}\left\{ \|\mathbf{y}\|^2 - \frac{1}{2}(\mathbf{y}-\mathbf{v}_\infty^*)^t \mathbf{C}_\infty^{*-1}(\mathbf{y}-\mathbf{v}_\infty^*)\right\}. \quad (4.6)$$

For later use (in Proposition 3) we note that (4.6) continues to hold even if some or all elements of $\gamma\mathbf{C}_\gamma^*$ tend to $+\infty$.

Proposition 2: Assume that there exists a dominating point $\mathbf{v} \in \partial E$. Consider any Gaussian importance sampling schemes such that $\mathbf{v}_\gamma^* \sim \mathbf{v}_\infty^* \in \mathbf{R}^n$ and $\gamma\mathbf{C}_\gamma^* \sim \mathbf{C}_\infty^*$ where $\mathbf{C}_\infty^*$ is a finite positive definite matrix, and define the "forbidden set" $\mathrm{F}(\mathbf{C}_\infty^*)$ as

$$\mathrm{F}(\mathbf{C}_\infty^*) = \left\{ \mathbf{y} : \|\mathbf{y}\|^2 - \frac{1}{2}(\mathbf{y}-\mathbf{v})^t \mathbf{C}_\infty^{*-1}(\mathbf{y}-\mathbf{v}) < \|\mathbf{v}\|^2 \right\}. \quad (4.7)$$

Then, $\rho = \|\mathbf{v}\|^2$ if and only if $\mathbf{v}_\infty^* = \mathbf{v}$ and $E \cap \mathrm{F}(\mathbf{C}_\infty^*) = \emptyset$.

Proof: Since $\mathbf{C}_\infty^*$ is assumed to be finite, we have $\lim_{\gamma\to\infty} \gamma^{-1}\log(\det(\gamma\mathbf{C}_\gamma^*)^{1/2}) = 0$ in (4.6). The infimum in the second term of (4.6) can be upper bounded by evaluating the argument at any point in $\bar{E}$. In particular, since $\mathbf{v} \in \partial E$, we have

$$\rho = \inf_{\mathbf{y}\in E}\left\{ \|\mathbf{y}\|^2 - \frac{1}{2}(\mathbf{y}-\mathbf{v}_\infty^*)^t \mathbf{C}_\infty^{*-1}(\mathbf{y}-\mathbf{v}_\infty^*)\right\}$$

$$\leq \|\mathbf{v}\|^2 - \frac{1}{2}(\mathbf{v}-\mathbf{v}_\infty^*)^t \mathbf{C}_\infty^{*-1}(\mathbf{v}-\mathbf{v}_\infty^*)$$

$$\leq \|\mathbf{v}\|^2.$$

We have assumed that $\mathbf{C}_\infty^*$ is positive definite, hence equality holds in the second inequality if and only if $\mathbf{v} = \mathbf{v}_\infty^*$. When $\mathbf{v} = \mathbf{v}_\infty^*$, the forbidden set condition $E \cap \mathrm{F}(\mathbf{C}_\infty^*) = \emptyset$ is necessary and sufficient for equality in the first inequality. ¶

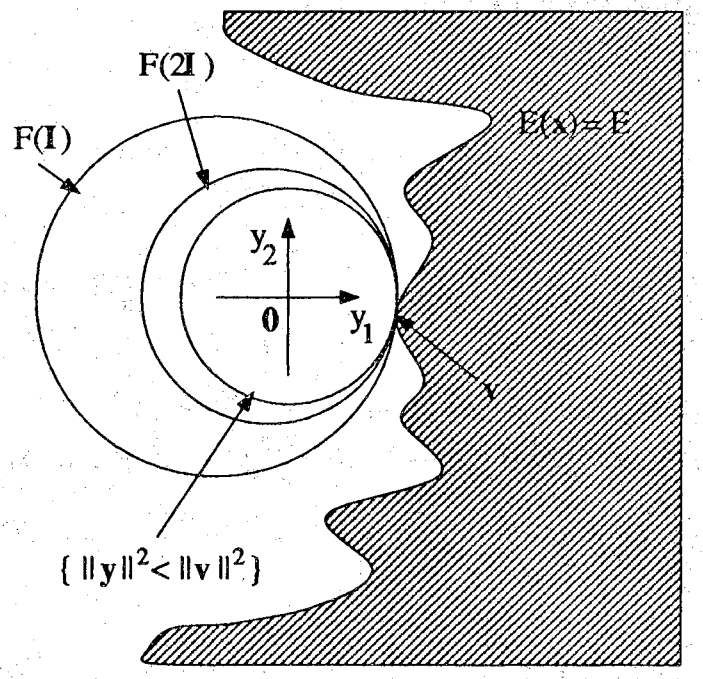


Figure 4.3 The forbidden set condition.

The assumption $\gamma \mathbf{C}_\gamma^* \sim \mathbf{C}_\infty^*$ in Proposition 2 considers the case in which the importance sampling covariance matrix is asymptotically stable relative to the model covariance matrix $\gamma^{-1}\mathbf{I}$. Notice that $\mathbf{v}$ is a boundary point of the forbidden set $\mathrm{F}(\mathbf{C}_\infty^*)$ (but $\mathbf{v} \notin \mathrm{F}(\mathbf{C}_\infty^*)$), and if $2\mathbf{I} - {\mathbf{C}_\infty^*}^{-1}$ is positive definite (which will usually be the case) this set is an ellipsoid. In the case of uniform variance scalings ($\mathbf{C}_\infty^* = c\mathbf{I}$), (4.7) reduces to, for $c > 1/2$,

$$\mathrm{F}(c\mathbf{I}) \quad = \quad \left\{ \mathbf{y} : \|\mathbf{y} + (2c-1)^{-1}\mathbf{v}\| \; < \; \frac{2c}{2c-1}\,\|\mathbf{v}\| \right\},$$

which is a sphere with center $-(2c-1)^{-1}\mathbf{v}$ and radius $2c\|\mathbf{v}\|/(2c-1)$. Figure 4.3 illustrates the situation for $c = 1$ and $c = 2$. In the case of linear systems, $\partial E = \{\mathbf{y} : \mathbf{v}^t\mathbf{y} = \|\mathbf{v}\|^2\}$ is a hyperplane that is tangent to the sphere $\mathrm{F}(\mathbf{C}_\infty^*)$. Assuming only that $2\mathbf{I} - {\mathbf{C}_\infty^*}^{-1}$ is nonnegative definite, it follows that $E \cap \mathrm{F}(\mathbf{C}_\infty^*) = \emptyset$ always holds for linear systems. So, in general the forbidden set condition $E \cap \mathrm{F}(\mathbf{C}_\infty^*) = \emptyset$ can be viewed as a restriction on the admissible "degree of nonlinearity."

Proposition 2 says, asymptotically, the mean vector of the Gaussian simulation density should be the dominating point provided that the forbidden set condition is satisfied. By (4.7), the forbidden set depends on the choice of the covariance matrix $\mathbf{C}_\infty^*$. It turns out that the optimal choice of $\mathbf{C}_\infty^*$ is difficult to determine for general systems. However, for linear systems a more precise result can be obtained which will provide us some useful insights. Consider the case in which we employ the asymptotically optimal (efficient) mean translation $\mathbf{v}_\gamma^* = \mathbf{v}$ and uniform variance scaling $\mathbf{C}_\gamma^* = c\gamma^{-1}\mathbf{I}$. (We consider here only the uniform variance scaling because it can be characterized by a parameter and it can be easily implemented. Non-uniform variance scalings will be discussed in the next Proposition.) We want to determine the optimal variance-scaling constant c. For $c > 1/2$, expression (4.5) reduces to

$$\begin{aligned} \mathrm{E}_\gamma^*\left[(1_E(\mathbf{Y})w(\mathbf{Y}))^2\right] &= c^{n/2}\left(\frac{c}{2c-1}\right)^{n/2}\exp\left(\frac{\|\mathbf{v}\|^2}{2c-1}\gamma\right)Q\left(\sqrt{\frac{4c}{2c-1}\|\mathbf{v}\|^2\gamma}\right) \\ &\sim (8\pi\|\mathbf{v}\|^2\gamma)^{-1/2}\,c^{n/2}\left(\frac{c}{2c-1}\right)^{(n-1)/2}\exp(-\|\mathbf{v}\|^2\gamma). \quad (4.8) \end{aligned}$$

When $c \leq 1/2$, we have $\mathrm{E}_\gamma^*[(1_E(\mathbf{Y})w(\mathbf{Y}))^2] = \infty$. Plugging (4.8) and $P_\gamma(\mathbf{Y}\in E)^2 \sim (2\pi\|\mathbf{v}\|^2\gamma)^{-1}\exp(-\|\mathbf{v}\|^2\gamma)$, which we have obtained in Example 4.1, into (4.2), we have

$$L_\epsilon(\gamma) \sim \sqrt{\frac{\pi}{2}}\,\|\mathbf{v}\|\,c^{n/2}\left(\frac{c}{2c-1}\right)^{(n-1)/2}\gamma^{1/2}. \quad (4.9)$$

Note that the exponential factors in the numerator and the denominator of (4.2) have cancelled. The approximations in (4.8) and (4.9) involve only Gaussian Q function approximations which are still quite accurate for only moderately large arguments.

Therefore, for exponentially efficient mean translations, $L_\epsilon(\gamma)$ grow like $\gamma^{1/2}$. Moreover, notice that the variance-scaling constant c impacts the computational cost only as a multiplicative factor $c^{n/2}(c/(2c-1))^{(n-1)/2}$. The optimal c can be obtained by minimizing this factor and the result is found to be

$$c_{opt} = \frac{2n-1}{2n} \approx 1. \quad (4.10)$$

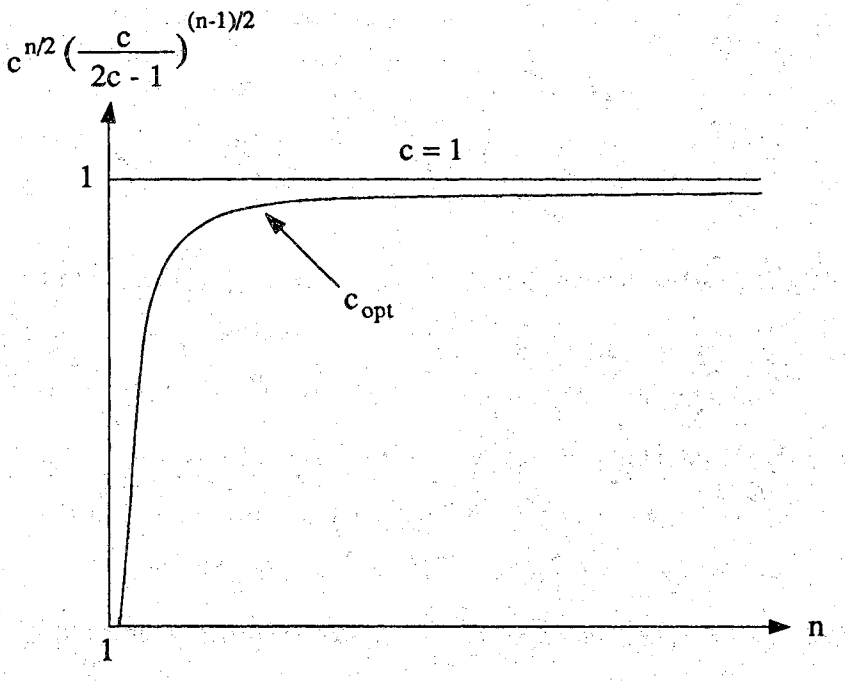


Figure 4.4 Dimensions in the optimal mean translation.

That is, no variance scaling ($c = 1$) is approximately asymptotically optimal, at least for linear systems. Moreover, notice that when $c = 1$, $\mathrm{E}^*_\gamma[(1_E(\mathbf{Y})w(\mathbf{Y}))^2]$ (and hence $L_\epsilon(\gamma)$) does not depend on the dimension n. In other words, for linear systems, exponentially efficient mean-translation biasing with no variance scaling (uniform or non-uniform) is free of the dimensionality effect. Figure 4.4 shows the multiplicative factor $c^{n/2}(c/(2c-1))^{(n-1)/2}$ as a function of n when $c = 1$ and $c = c_{opt}$. The saving on $L_\epsilon(\gamma)$ by employing a uniform variance scaling in addition to the optimal mean translation quickly diminishes as the dimension n goes up. Therefore, hereafter unless otherwise stated, when we consider an optimal mean-translation biasing, we will mean the asymptotically efficient mean translation without scaling the covariance matrix.

Another observation of (4.10) is that, if we plot c_{opt} against the dimension n, the optimal variance scaling factor increases from 1/2 ($n = 1$) to 1 ($n = \infty$). Recall that $\mathbf{C}^*_\gamma = c\gamma^{-1}\mathbf{I}$. A variance-scaling factor $c < 1$ actually means variance reduction rather than increase! Thus, in contrast to the variance-scaling biasing where the noise

variance is increased, our exponentially efficient mean-translation biasing requires a variance reduction. As an aside, some workers have been able to reduce many IS problems to 1-dimensional [13, 52], and thus eliminate the dimensionality effect. For $n = 1$, the asymptotical optimal variance-scaling factor for mean-translation biasings is $1/2$. This solution, however, is unstable because $\mathrm{E}^*_\gamma[(1_E(\mathbf{Y})w(\mathbf{Y}))^2] = \infty$ for $c = 1/2$. Therefore, as seen from Figure 4.4, although the potential saving on $L_\epsilon(\gamma)$ by variance-scaling the optimal mean-translation biasing is great for the $n = 1$ case, the chance of disaster is great too. The optimal variance-scaling factor c_{opt} must be chosen carefully ($c_{opt} = 1/2$ is an asymptotical solution). This is an open question. But in most cases, we would rather settle for $c = 1$.

The last two paragraphes present results for linear systems which we might hope to extrapolate to a "moderately nonlinear" system. Such arguments are common practice in the literature. However, this appeal to linear system behaviors neglects the forbidden set condition $E \cap \mathrm{F}(\mathbf{C}^*_\infty) = \emptyset$ which must be satisfied by a "moderately nonlinear" system. Notice that if $\mathbf{C}^{*\,-1}_\infty \approx \mathbf{0}$, then $\mathrm{F}(\mathbf{C}^*_\infty) \approx \{\mathbf{y} : \|\mathbf{y}\|^2 < \|\mathbf{v}\|^2\} =$ the sphere centered at the origin of radius $\|\mathbf{v}\|$. But by the definition of a minimum rate point, we always have $E \cap \{\mathbf{y} : \|\mathbf{y}\|^2 < \|\mathbf{v}\|^2\} = \emptyset$. Thus, variance scaling by increasing the noise variance weakens the forbidden set condition, and hence, provides a degree of robustness by admitting more severely nonlinear systems, as illustrated in Figure 4.3. (An optimal mean-translation biasing with forbidden set $\mathrm{F}(2\mathbf{I})$ is more robust than the one with $\mathrm{F}(\mathbf{I})$.) Of course, the cost of this robustness is an increase in computation. But in practice since it is normally not possible to verify the forbidden set condition, the robustness obtained by some degree of variance scaling may be justified in some cases.

In parallel to the above exponentially efficient mean translation, it is possible to achieve this exponential efficiency with just variance scaling. Proposition 3 below states the necessary and sufficient conditions of the optimal variance-scaling Gaussian density, which follows directly from (4.6), so we omit the straightforward proof.

Proposition 3: Assume that there is a dominating point $\mathbf{v}$ and consider any Gaussian importance sampling scheme such that $\mathbf{v}_\gamma^* \sim \mathbf{v}_\infty^* \neq \mathbf{v}$. Then $\rho = \|\mathbf{v}\|^2$ if and only if

$$\lim_{\gamma\to\infty} \gamma^{-1} \log\left(\det\left(\gamma \mathbf{C}_\gamma^*\right)^{1/2}\right) \quad = \quad 0 \tag{4.11}$$

and

$$\lim_{\gamma\to\infty} \frac{1}{2}(\mathbf{y}-\mathbf{v}_\infty^*)^t(\gamma\mathbf{C}_\gamma^*)^{-1}(\mathbf{y}-\mathbf{v}_\infty^*) \quad \leq \quad \|\mathbf{y}\|^2 - \|\mathbf{v}\|^2 \tag{4.12}$$

for all $\mathbf{y} \in E$ with equality for some $\mathbf{y} \in \bar{E}$. ¶

Notice that condition (4.12) for $\mathbf{y} = \mathbf{v} \in \bar{E}$ gives us the necessary condition

$$\lim_{\gamma\to\infty} (\mathbf{v}-\mathbf{v}_\infty^*)^t(\gamma\mathbf{C}_\gamma^*)^{-1}(\mathbf{v}-\mathbf{v}_\infty^*) \quad = \quad 0.$$

In other words, $\gamma\mathbf{C}_\gamma^*$ must tend to infinity at least in the dimension of $\mathbf{v} - \mathbf{v}_\infty^*$. Condition (4.11), on the other hand, says that the rate of growth must be less than exponentially fast.

Proposition 3 suggests a non-uniform variance scaling which scales the covariance matrix only in the dimension of $\mathbf{v} - \mathbf{v}_\infty^*$. Before we comment on this approach, let us first examine the CIS algorithm mentioned in Chapter 2 which is uniform variance scaling with unscaled mean vector. Therefore, we have $\mathbf{v}_\gamma^* = \mathbf{0}$ and $\mathbf{C}_\gamma^* = c_\gamma\gamma^{-1}\mathbf{I}$. In the case of CIS applied to a linear system, (4.5) reduces to, for $c_\gamma > 1/2$,

$$\begin{aligned} \mathrm{E}_\gamma^*\left[(1_E(\mathbf{Y})w(\mathbf{Y}))^2\right] \quad &= \quad c_\gamma^{n/2}\left(\frac{c_\gamma}{2c_\gamma-1}\right)^{n/2} Q\left(\sqrt{(2-c_\gamma^{-1})\|\mathbf{v}\|^2\gamma}\right) \\ &\sim \quad \frac{2^{-(n+1)/2}}{\sqrt{2\pi}\|\mathbf{v}\|}\left\{c_\gamma^{n/2}\exp\left(\frac{\|\mathbf{v}\|^2\gamma}{2c_\gamma}\right)\right\}\gamma^{-1/2}\exp\left(-\|\mathbf{v}\|^2\gamma\right). \end{aligned}$$

If $c_\gamma \leq 1/2$, as in the case of (4.8), we have $\mathrm{E}_\gamma^*[(1_E(\mathbf{Y})w(\mathbf{Y}))^2] = \infty$. The asymptotically optimal variance-scaling factor c_γ is found by minimizing the term in the bracket $\{\cdot\}$. The result is $c_{\gamma,opt} \sim \|\mathbf{v}\|^2\gamma/n$. Note that n can't be too large such that $c_\gamma \leq 1/2$. (This implies that the CIS scheme would blow up beyond some dimension n for large γ.) Plugging this $c_{\gamma,opt}$ back into the expression above, and then plugging into (4.2), we get

$$L_\epsilon(\gamma) \quad \sim \quad \sqrt{2\pi}\, 2^{-(n+1)/2}\, \|\mathbf{v}\|^{(n+1)}\, n^{-n/2}\, \exp(n/2)\, \gamma^{(n+1)/2}. \qquad (4.13)$$

Therefore, for an optimized CIS and a fixed n, $L_\epsilon(\gamma)$ grows in proportion to $\gamma^{(n+1)/2}$. This is substantially worse than the $\gamma^{1/2}$ (for all n) behavior obtained in (4.9) for the optimized mean translation. Moreover, the CIS computational growth does depend on the dimension n in an undesirable way. A more precise description of the dimensionality effect like in Figure 4.4 is difficult to obtain. However, note that (4.13) is approximately the $L_\epsilon(\gamma)$ for a large γ. In that case, the growth of the last term, $\gamma^{(n+1)/2}$, relative to n far outweights the others. Thus, it is clear that the CIS is very inefficient for a large n and SNR. This will be confirmed by an example in the next section.

The technique proposed by Davis [13] in an effort to reduce the dimensionality effect of the CIS method is precisely a non-uniform variance scaling only in the dimension of $\mathbf{v} - \mathbf{v}^*_\infty$. This method reduces the computational growth back to the 1-dimensional case. (One would still have to deal with the condition (4.12) which would play the same role as the forbidden set condition of Proposition 2.) However, this approach gives up the major advantages of uniform variance scaling over the mean translation, namely, the robustness and ease of implementation. In addition, mean-translation biasing is still more efficient. Taking $n = 1$ in (4.9) and (4.13), we have the computational cost of variance scaling grows like γ compared to the $\gamma^{1/2}$ growth for the mean translation. In our opinion, if we are to go to the trouble of computing $\mathbf{v}$, then we might as well apply this information to the more efficient estimator — mean translation, possibly in conjunction with a moderate degree of variance scaling for sake of robustness.

In Propositions 2 and 3 we have assumed the existence of a dominating point. The analysis of Sadowsky and Bucklew in [49] demonstrates that exponential efficiency can be achieved in the case of multiple minimum rate points by taking $f^*_{\mathbf{Y}}(\cdot)$ to be a convex combination of Gaussian mean-translation densities. (For example, decision regions for a QAM demodulator will have multiple minimum rate points.) We further remark that in comparison to [49], here we have considered a more restricted problem

formulation, but within this context we have obtained some deeper results (combined mean translation and variance scaling, and the conclusions about robustness obtained from the forbidden set condition). Extensions of these ideas to some exponentially efficient non-Gaussian simulation distributions are given by Schulebusch in [52].

4.2 Variance Scaling vs. Mean Translation

The uniform variance-scaling biasing and the optimized mean-translation biasing have been the two mainstreams along the line of the IS development and are also the methods appear to be most practical. In the high SNR region, (4.9) and (4.13) show that the computational cost of the variance scaling increases more rapidly than the mean translation biasing relative to the increased SNR. Therefore, the mean translation is more efficient than the variance scaling at a large SNR. Another important factor in (4.9) and (4.13) is the dimension n. $L_\epsilon(\gamma)$ for the optimized mean translation with no variance scaling is independent of n. Equation (4.13), on the other hand, suggests that the uniform variance scaling is very inefficient at a large n and would break down at some point.

For a moderate SNR, the above asymptotic behavior may not apply. We are interested in knowing how the two biasings compare to each other in this case. However, it is difficult to obtain general conclusions such as what we have obtained for large SNR's in the last section. (This is why we go to the asymptotic analysis in the first place.) In this section, we will examine the uniform variance-scaling biasing and the mean-translation biasing using some heuristic arguments which apply to all SNR's. This approach also helps to explain more clearly the dimensionality effect of the uniform variance scaling. At the end of this section, we will use a simple matched filter example to compare the two biasings numerically.

Consider Figure 4.5 which shows the $n = 2$ model density $f_{\mathbf{Y}}(y_1, y_2)$, an error set (on the y_1–y_2 plane) and a dominating point. This figure can be seen as a side view of Figure 4.2 which is viewed from the top. Recall from Chapter 2 the criterion we have established for a good simulation density which we restate here:

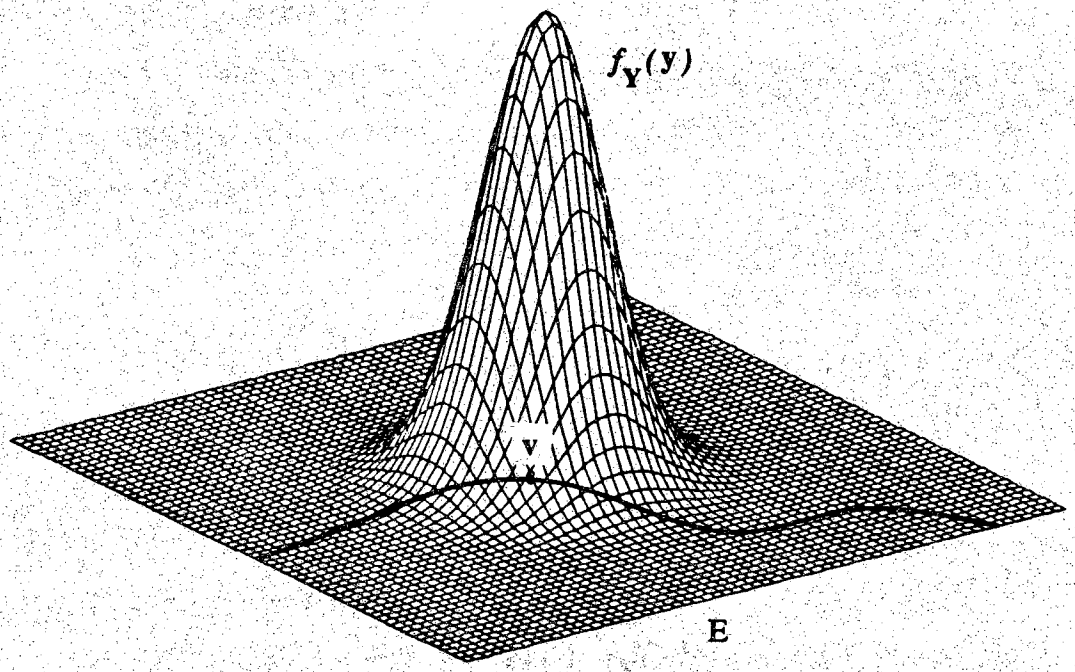
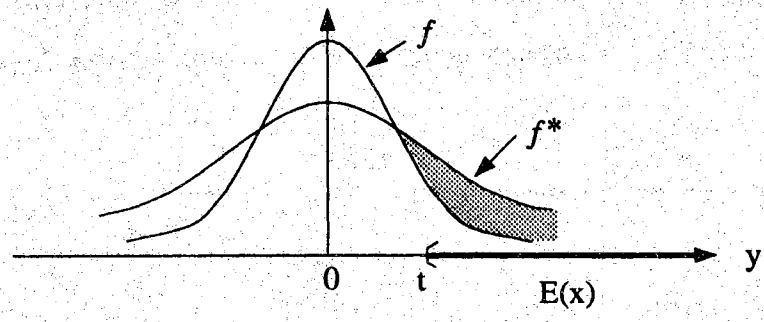


Figure 4.5 Dominating point is the most important sample.

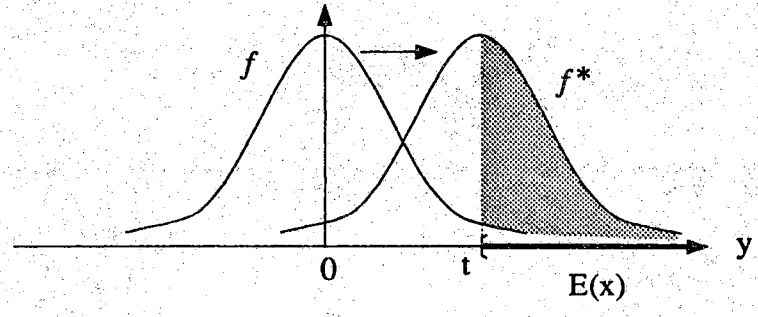
> An efficient simulation density $f_{\mathbf{Y}}(\mathbf{y})$ should sample $\mathbf{y}$ (important samples) from the important error regions where $1_E(\mathbf{y})f_{\mathbf{Y}}(\mathbf{y})$ are relatively large.

Since the Gaussian probability density $f_{\mathbf{Y}}(\mathbf{y})$ decreases as the distance between $\mathbf{y}$ and the origin (the mean of the Gaussian distribution) increases, it is obvious that the dominating point $\mathbf{v}$ has the largest probability density $f_{\mathbf{Y}}(\mathbf{v})$ among all points in the error set $E(\mathbf{x})$, i.e., $f_{\mathbf{Y}}(\mathbf{v}) = \max_{\mathbf{y}\in E(\mathbf{x})} f_{\mathbf{Y}}(\mathbf{y})$. Hence, the dominating point is the "most important sample." This is the physical interpretation of (4.1) which also explains why we perform the minimization in Example 4.1 to find the dominating point. An efficient simulation density therefore should be able to sample more often from the regions that are close to the origin in the error set. In particular, the dominating point should have a probability density that is the peak of $f^*_{\mathbf{Y}}(\mathbf{y})$, i.e., should be the point that is most likely to be sampled.

Recall that the model density is $f_{\mathbf{Y}}(\mathbf{y}) \sim N(\mathbf{0}, \sigma^2\mathbf{I})$, where $N(\mathbf{m}, \mathbf{C})$ denotes a Gaussian distribution with a mean vector $\mathbf{m}$ and covariance matrix $\mathbf{C}$. To increase the relative frequency of sampling those important samples, the strategy of variance scaling is to use $f^*_{\mathbf{Y}}(\mathbf{y}) \sim N(\mathbf{0}, c\sigma^2\mathbf{I})$; whereas for mean translation it is $f^*_{\mathbf{Y}}(\mathbf{y}) \sim N(\mathbf{v}, \sigma^2\mathbf{I})$. Therefore, intuitively, the variance-scaling method pushes the probability mass away from the mean and spreads it into the error set. The mean translation, on the other hand, "moves" the entire probability distribution toward the error set. The difference between these two approaches can be seen easily in the one-dimensional picture of Figure 4.6. The shaded areas in Figure 4.6(a) and (b) are the increased probability mass in the error set due to biasings compared to original model distribution functions. The superiority of mean translations is evident from this illustration. The mean-translation biasing causes more probability mass increase in the error set, especially in the (important) error region close to the threshold t. Note that in Figure 4.6(a), a significant amount of probability mass has been spread in the negative y-axis direction — the "wrong" direction by the variance scaling. Figure 4.7 shows the variance scaling for the above $n = 2$ example by which the



(a) Variance scaling.



(b) Mean translation.

Figure 4.6 One-dimensional variance scaling vs. mean translation.

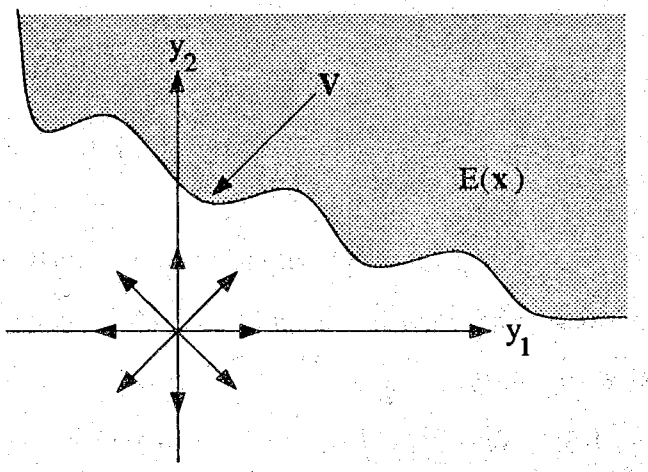


Figure 4.7 Dimensionality effect of variance scaling.

probability mass is pushed out of the origin in "all" directions. Clearly, as in the one-dimensional case, only part of the probability mass really goes in the "right" directions and ends up in the error set. As the dimension increases, there will be more and more wrong directions. This explains the rapid decrease of efficiency for uniform variance scalings, i.e., the dimensionality effect, documented in the literature as well as the discussion followed (4.13). Even with the non-uniform variance scaling, which is equivalent to the $n = 1$ case, mean-translation biasing is still more efficient as seen from Figure 4.6.

It is also important to notice from Figure 4.6(b) that the mean-translation biasing is signal-dependent. If the transmitted signal is the other polarity (e.g., $+1$) and hence the error set is $E(x) = (-\infty, -t]$ instead of $[t, \infty)$, then the probability distribution should be moved toward the negative y-axis. In general, because the error set $E(\mathbf{x})$ and thus the dominating point is strongly signal-dependent, the optimal mean translation has to be tailored for each $\mathbf{x}$. In contrast, the variance scaling is independent of the transmitted signal as is obvious from Figure 4.6(a). (As noted in Section 1.2, we can still have signal-dependent variance scalings. But because of

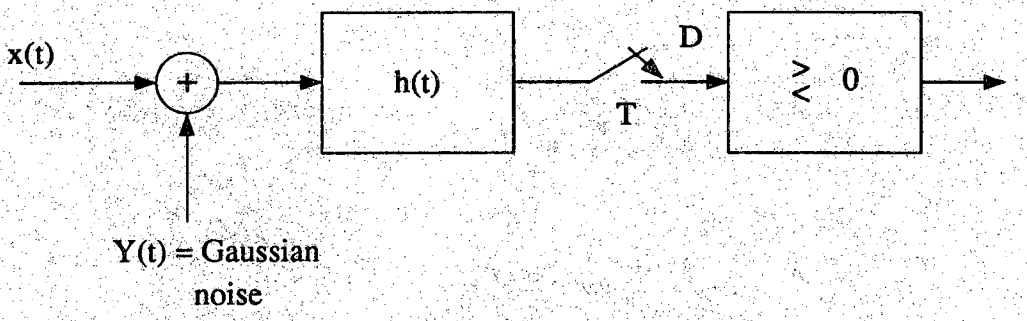


Figure 4.8 A matched filter example.

biasing conflicts, the optimal variance scaling can't be used and hence the biasing can only be sub-optimal.)

In summary, the optimal mean translation has a much higher efficiency gain than the variance scaling, especially at a high SNR and/or a large n. But because of its signal dependency it is also more difficult to implement and computationally more costly in making each decision (due to the need of computing the dominating point for each $\mathbf{x}$). In the next section, we will see that this disadvantage of mean translation can be reduced. When the mean-translation biasing is used in conjunction with the event simulation method, its efficiency can be boosted by also biasing the signal distribution which is called the conditional importance sampling technique. This "signal biasing" can not be used in the stream simulation where the variance scaling has been mostly used.

We now use a simple example to illustrate how the model dimension issue impacts the problem of discrete-time modeling of a continuous time system. The optimal mean translation and uniform variance scaling will be compared. The dimensionality effect of the variance scaling will be shown. Figure 4.8 is a receiver block diagram of a binary baseband communication system. The matched filter impulse response function $h(t)$ is matched to $-x(t)$, i.e., $h(T-t) \propto -x(t)$, where $x(t)$, the transmitted signal, is a pulse of duration $[0, T]$. Assume that $\int_0^T h(t)^2\,dt = 1$ and $\int_0^T h(T-t)x(t)\,dt < 0$.

Suppose that we are interested in estimating the decision error probability $P_\gamma(D \geq 0)$ where the decision statistic is the convolution $D = \int_0^T h(T-t)(x(t)+Y(t))\,dt$ and $Y(t)$ is a white Gaussian noise process with power spectral density $N_0/2$. Define $\mathcal{E} = (\int_0^T h(T-t)x(t)\,dt)^2$ to be the received signal energy. Let $\gamma = 2\mathcal{E}/N_0$. Then the exact expression for the decision error probability can be found, which is $P_\gamma(D \geq 0) = Q(\sqrt{\gamma})$.

For the purpose of simulation, a discrete-time model can be obtained by first approximating $Y(t)$ with $\sqrt{\frac{n\mathcal{E}}{T}}\,Y_n(t)$ where $Y_n(t)$ is a bandlimited white Gaussian noise process with power spectral density $N_0T/(2n\mathcal{E})$ and bandwidth $n/2T$. (Recall from Section 3.1 that infinite bandwidth white noise can not be sampled directly.) Then the discrete-time noise samples $\mathrm{Y}_k = Y_n(kT/n)$, $k = 1,..,n$, are i.i.d. zero mean Gaussian random variables with variance $\sigma^2 = (2\mathcal{E}/N_0)^{-1}\gamma^{-1}$. Next, we approximate the convolutional integral as a finite sum. Let $a = \int_0^T h(T-t)x(t)\,dt/\sqrt{\mathcal{E}}$, n = the number of samples in the interval [0,T], and $\triangle = T/n$ be the sampling period. Then we get

$$\begin{aligned}
&\int_0^T h(T-t)(x(t)+Y(t))\,dt \\
&\qquad \approx \sqrt{\mathcal{E}}\,a \;+\; \sum_{k=1}^{n} h(T-k\triangle)\sqrt{\frac{n\mathcal{E}}{T}}\,Y_n(k\triangle)\,\triangle \\
&\qquad = \sqrt{\mathcal{E}}\left\{ a \;+\; \sum_{k=1}^{n} \sqrt{\frac{T}{n}}\,h(T-k\triangle)\,\mathrm{Y}_k \right\}.
\end{aligned}$$

The above approximation leads to a linear discrete-time model of the form $\xi(\mathbf{y}) = a + \mathbf{b}^t\mathbf{y}$ where $\mathrm{b}_k = \sqrt{\frac{T}{n}}\,h(T-k\triangle)$. This is precisely the format of the linear system example in the last section except that the signal factors have been simplified because here we are considering only a fixed transmitted signal $x(t)$. We will also make one more approximation. Notice that $\sum_{k=1}^{n}(\sqrt{\frac{T}{n}}\,h(T-k\triangle))^2 \approx \int_0^T h(T-t)^2\,dt = 1$. We will set $\mathrm{b}_k \propto h(T-k\triangle)$ where the constant of proportionality is set so that $\|\mathbf{b}\|^2 = \sum_{k=1}^{n} b_k^2 = 1$. Provided that $x(t)$ and $h(t)$ are lowpass, these approximations are insignificant for large n. The reason for this normalization is that $P_\gamma(\mathbf{Y} \in E) =$

$P_\gamma(D \geq 0)$, that is, there is no modeling error introduced by the discrete-time model. (Of course, this is only true because of the linearity of the problem.) Finally, notice that the dominating point is, from the example in the last section, $\mathbf{v} = \mathbf{b}$, and hence, $\|\mathbf{v}\|^2 = 1$ for all n.

Figure 4.9 compares the computational costs of three estimators for various values of the model dimension n. $L_1(\gamma)$ is plotted, but for any $\epsilon > 0$ we can easily compute $L_\epsilon(\gamma) = L_1(\gamma)/\epsilon^2$. For the plotted range of γ, 4 to 16 dB, the probability $P_\gamma(D \geq 0)$ ranges from 5.6×10^{-2} down to 1.4×10^{-10}. The three estimators are the pure mean translation ($\mathbf{v}_\gamma^* = \mathbf{v}$ and $\mathbf{C}_\gamma^* = \gamma^{-1}\mathbf{I}$) which is denoted MT, the optimized CIS ($\mathbf{v}_\gamma^* = 0$ and $\mathbf{C}_\gamma^* = c^\gamma \gamma^{-1}\mathbf{C}_\infty^*$), and the unbiased Monte Carlo ($\mathbf{v}_\gamma^* = \mathbf{0}$ and $\mathbf{C}_\gamma^* = \gamma^{-1}\mathbf{I}$) which is denoted MC. By optimized CIS we mean that the scaling factor c_γ has been numerically optimized for each $\gamma < \infty$. In some cases this could be substantially better than the asymptotically optimal solution $c_\gamma \sim \gamma/n$. In contrast, the mean translation in Figure 4.9 is the asymptotically optimal solution $\mathbf{v}_\gamma^* = \mathbf{v}$, which is close, but not identical to the finite γ optimal mean translation which is derived by Lu and Yao in [36]. We also remark that the performance of Davis' non-uniform variance-scaling scheme [13] is equivalent to the $n = 1$ CIS curve for all n. Apparently, while the mean-translation suffers from no dimensionality effect, the computational cost for CIS increases exponentially. At large n, the performance of CIS is almost no better than the ordinary Monte Carlo method.

4.3 Conditional Importance Sampling

So far, we haven't talked much about the signal inputs $\mathbf{x}$ except it was defined at the beginning of this chapter to represent "anything else but noise" that have impacts on the output decision. We will keep this vagueness in this section to develop the general concept of conditional importance sampling. The context of $\mathbf{x}$ is system-dependent and will not be fully clear until we formally introduce the event simulation method in the next chapter. In this section, we will fit the mean-translation biasing

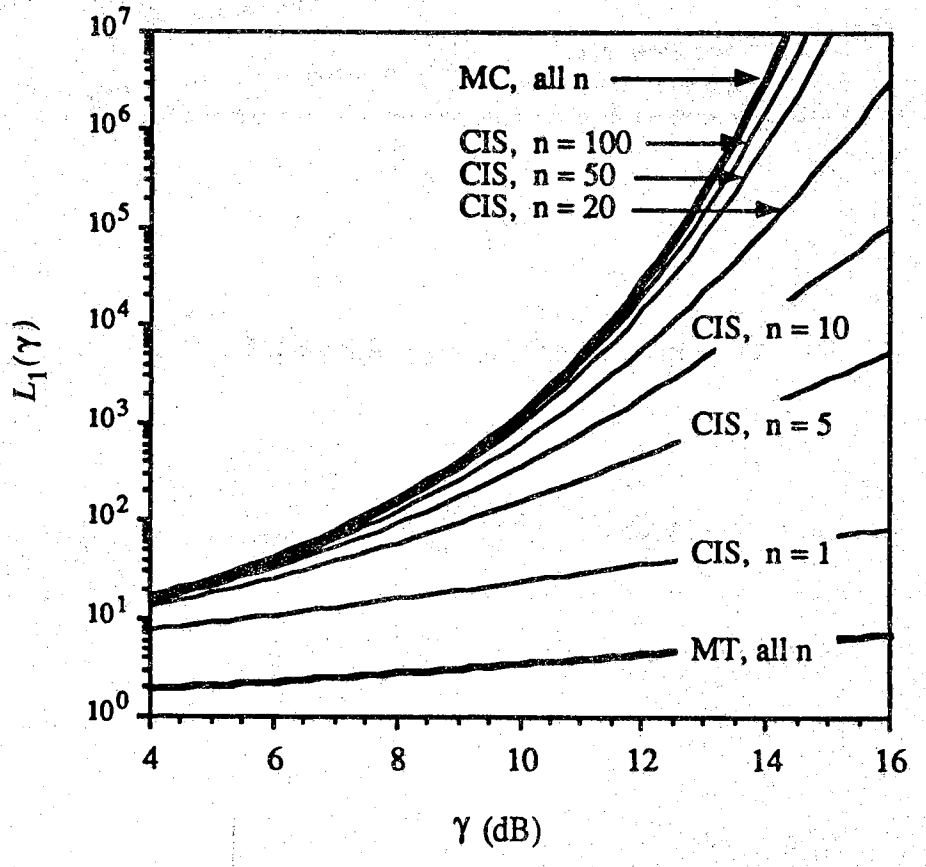


Figure 4.9 $L_1(\gamma)$ for three estimators.

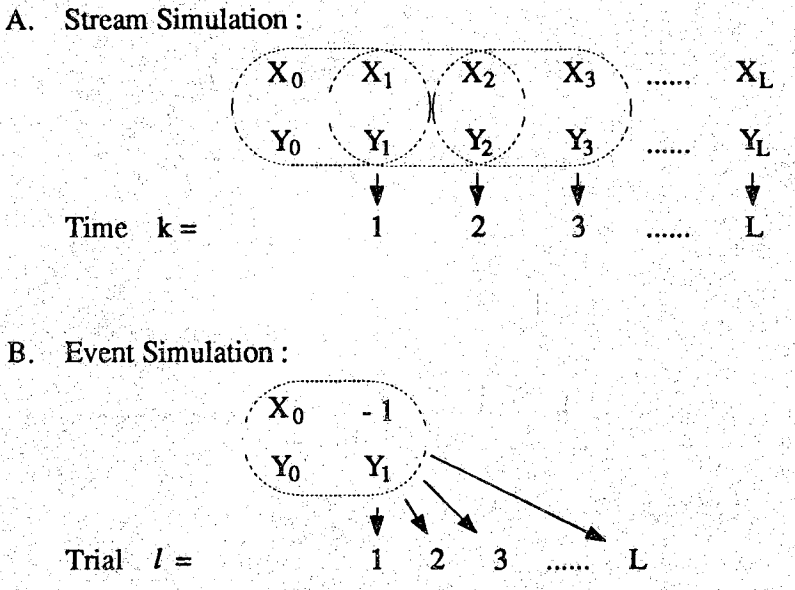


Figure 4.10 Event simulation vs. stream simulation.

into a conditional mean estimation environment and introduce the conditional importance sampling technique which links the mean-translation biasing of the noise with the signal biasing.

The following example provides a motivation for developing the conditional importance sampling. Also, we will be able to be a little more specific about $\mathbf{x}$, which may clear some cloud over the discussion we have had in this chapter.

Example 4.2:

Let us revisit the simple example in Section 1.2 which illustrates the difference between event simulation and stream simulation. Figure 1.3 is redrawn here for reference. Recall that in the event simulation we put $X_1 = -1$ and $\mathbf{Z} = (X_0, -1, Y_0, Y_1)$. The bit error probability is $P_b = P(\xi(\mathbf{Z}) \geq 0)$. Instead of generating L sets of i.i.d. random vectors $\mathbf{Z}^{(\ell)}, \ell = 1, \ldots, L$, and directly estimating P_b as is done in (1.3), we can first partition P_b as follows.

$$P_b = P(\xi(\mathbf{Z}) \geq 0)$$

$$= \sum_{x_0} P(\xi(\mathbf{Z}) \geq 0 | X_0 = x_0) P_{X_0}(x_0) \tag{4.14}$$

$$= \frac{1}{2} P(\xi(\mathbf{Z}) \geq 0 | X_0 = 1) + \frac{1}{2} P(\xi(\mathbf{Z}) \geq 0 | X_0 = -1),$$

where $P_{X_0}(x_0)$ is the marginal probability mass function of the random variable X_0. We have assumed that $P_{X_0}(X_0 = 1) = P_{X_0}(X_0 = -1) = 1/2$. Suppose that Y_0 and Y_1 are i.i.d. zero mean Gaussian random variables with variance σ^2. Then the random variable $\xi(\mathbf{Z}) = -1 + Y_1 + \lambda(X_0 + Y_0)$ has a conditional mean $(-1 + \lambda)$ if $X_0 = 1$ and $(-1 - \lambda)$ if $X_0 = -1$. Apparently, if $\lambda > 0$, we have $P(\xi(\mathbf{Z}) \geq 0 | X_0 = 1) > P(\xi(\mathbf{Z}) \geq 0 | X_0 = -1)$. That is, some "ISI patterns" are more likely to cause a bit error than others. In the language of importance sampling, the error event $\{\xi(\mathbf{Z}) \geq 0$, given $X_0 = 1\}$ is a more important error event than $\{\xi(\mathbf{Z}) \geq 0$, given $X_0 = -1\}$. It is then natural that we would like to devote more of our computational effort to the estimation of dominant terms of P_b, which is $P(\xi(\mathbf{Z}) \geq 0 | X_0 = 1)$ here. This can be done by biasing the probability mass function of X_0 such that $X_0 = 1$ would be sampled more often. The biased probability mass function can be $P^*_{X_0}(X_0 = 1) = p > 1/2$. Similar to (2.6), (4.14) can then be rewritten as

$$P_b = \sum_{x_0} P(\xi(\mathbf{Z}) \geq 0 | X_0 = x_0) \frac{P_{X_0}(x_0)}{P^*_{X_0}(x_0)} P^*_{X_0}(x_0).$$

After sampling X_0, we have the system signal inputs $\mathbf{x} = (x_0, x_1)$ and Gaussian noise inputs $\mathbf{Y} = (Y_0, Y_1)$ which is exactly the formulation of Section 4.1 and the optimal 2-dimensional Gaussian mean-translation simulation density can be applied to the noise inputs. ¶

We now formally introduce the conditional importance sampling. Consider Figure 4.1 except now we also let the signal inputs be a random vector $\mathbf{X}$. Let $(\mathbf{X}, \mathbf{Y})$ be jointly distributed as specified by the conditional density $f_{\mathbf{Y}|\mathbf{X}}(\cdot|\mathbf{x})$ and the marginal density $f_{\mathbf{X}}(\cdot)$. (We use the term "density" in a generic sense. In our application, $\mathbf{Y}$ will be a continuous random vector while $\mathbf{X}$ will be discrete, so $f_{\mathbf{X}}(\cdot)$ will actually be a probability mass function.) We wish to estimate $\alpha = \mathrm{E}[g(\mathbf{X}, \mathbf{Y})]$. As in Chapter 2, the special case $g(\cdot, \cdot) = 1_E(\cdot, \cdot)$, i.e., the decision error probability is what we are

most interested in. Often, it is convenient to think of this joint expectation in terms of the successive conditioning formula $\alpha = \mathrm{E}[\beta(\mathbf{X})]$ where $\beta(\mathbf{x}) = \mathrm{E}[g(\mathbf{x}, \mathbf{Y})|\mathbf{X} = \mathbf{x}]$. Successive conditioning suggests the "conditional importance sampling estimator" described as follows [5]. Independent samples $\mathbf{X}^{(\ell)}$, $\ell = 1, \ldots, L_X$, are sampled from a marginal simulation density $f^*_{\mathbf{X}}(\cdot)$. For each fixed ℓ, (conditionally) independent samples $\mathbf{Y}^{(\ell,\ell')}$, $\ell' = 1, .., L_Y$, are sampled from the conditional simulation density $f^*_{\mathbf{Y}|\mathbf{X}}(\cdot\,|\mathbf{X}^{(\ell)})$. It will be convenient to denote these samples as one big vector $\mathbf{Z}^{(\ell)} = (\mathbf{Y}^{(\ell,1)}, \ldots, \mathbf{Y}^{(\ell,L_Y)})$. For each $\ell = 1, .., L_X$, we compute a conditional estimate

$$\hat{\beta}\left(\mathbf{X}^{(\ell)}, \mathbf{Z}^{(\ell)}\right) \;=\; \frac{1}{L_Y} \sum_{\ell'=1}^{L_Y} g\left(\mathbf{X}^{(\ell)}, \mathbf{Y}^{(\ell,\ell')}\right) w_{\mathbf{Y}|\mathbf{X}}\left(\mathbf{Y}^{(\ell,\ell')}|\mathbf{X}^{(\ell)}\right) \tag{4.15}$$

where $w_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = f_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})/f^*_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$. It is easy to show that

$$\begin{aligned} \mathrm{E}^*\left[\hat{\beta}\left(\mathbf{X}^{(\ell)}, \mathbf{Z}^{(\ell)}\right)|\mathbf{X}^{(\ell)}\right] \;&=\; \mathrm{E}^*\left[g(\mathbf{X}^{(\ell)}, \mathbf{Y}^{(\ell,1)})\, w_{\mathbf{Y}|\mathbf{X}}(\mathbf{Y}^{(\ell,1)}|\mathbf{X}^{(\ell)})|\mathbf{X}^{(\ell)}\right] \\ &=\; \beta(\mathbf{X}^{(\ell)}), \end{aligned} \tag{4.16}$$

i.e., $\hat{\beta}\left(\mathbf{X}^{(\ell)}, \mathbf{Z}^{(\ell)}\right)$ is an unbiased estimator for the conditional expectation $\beta\left(\mathbf{X}^{(\ell)}\right)$. The estimator for α is

$$\hat{\alpha} \;=\; \frac{1}{L_X} \sum_{\ell=1}^{L_X} \hat{\beta}\left(\mathbf{X}^{(\ell)}, \mathbf{Z}^{(\ell)}\right) w_{\mathbf{X}}\left(\mathbf{X}^{(\ell)}\right) \tag{4.17}$$

where $w_{\mathbf{X}}(\mathbf{x}) = f_{\mathbf{X}}(\mathbf{x})/f^*_{\mathbf{X}}(\mathbf{x})$. Again, we have an unbiased estimator:

$$\begin{aligned} \mathrm{E}^*[\hat{\alpha}] \;&=\; \mathrm{E}^*[\hat{\beta}(\mathbf{X}, \mathbf{Z}) w_{\mathbf{X}}(\mathbf{X})] \\ &=\; \mathrm{E}^*\left[\; \mathrm{E}^*[\hat{\beta}(\mathbf{X}, \mathbf{Z}) w_{\mathbf{X}}(\mathbf{X})|\mathbf{X}]\;\right] \\ &=\; \mathrm{E}^*\left[\; w_{\mathbf{X}}(\mathbf{X})\;\; \mathrm{E}^*[\hat{\beta}(\mathbf{X}, \mathbf{Z})|\mathbf{X}]\;\right] = \alpha. \end{aligned} \tag{4.18}$$

Notice that the total number of samples is $L = L_X L_Y$.

Proposition 4: The variance of the estimator (4.17) is

$$\mathrm{var}^*[\hat{\alpha}] = \frac{1}{L_X L_Y}(v_1 + L_Y v_2), \tag{4.19}$$

where

$$v_1 = \mathrm{E}^*\left[\,\mathrm{var}^*\left[g(\mathbf{X},\mathbf{Y})w_{\mathbf{Y}|\mathbf{X}}(\mathbf{Y}|\mathbf{X})|\mathbf{X}\right]\; w_{\mathbf{X}}(\mathbf{X})^2\,\right] \tag{4.20}$$

and

$$v_2 = \mathrm{var}^*\left[\,\beta(\mathbf{X})w_{\mathbf{X}}(\mathbf{X})\,\right]. \tag{4.21}$$

Proof: Since the pairs $(\mathbf{X}^{(\ell)},\mathbf{Z}^{(\ell)})$, $\ell = 1,..,L_X$, are i.i.d. random vectors, it follows that $\mathrm{var}^*[\hat{\alpha}] = \mathrm{var}^*[\hat{\beta}(\mathbf{X},\mathbf{Z})w_{\mathbf{X}}(\mathbf{X})]/L_X$. Next we have

$$\begin{aligned}
&\mathrm{var}^*[\hat{\beta}(\mathbf{X},\mathbf{Z})w_{\mathbf{X}}(\mathbf{X})] \\
&\quad = \mathrm{E}^*\left[\,\mathrm{var}^*[\hat{\beta}(\mathbf{X},\mathbf{Z})w_{\mathbf{X}}(\mathbf{X})|\mathbf{X}]\,\right] + \mathrm{var}^*\left[\,\mathrm{E}^*[\hat{\beta}(\mathbf{X},\mathbf{Z})w_{\mathbf{X}}(\mathbf{X})|\mathbf{X}]\,\right] \\
&\quad = \mathrm{E}^*\left[\,w_{\mathbf{X}}^2(\mathbf{X})\mathrm{var}^*[\hat{\beta}(\mathbf{X},\mathbf{Z})|\mathbf{X}]\,\right] + \mathrm{var}^*\left[\,w_{\mathbf{X}}(\mathbf{X})\mathrm{E}^*[\hat{\beta}(\mathbf{X},\mathbf{Z})|\mathbf{X}]\,\right].
\end{aligned}$$

The second term in the last summation can be reduced to $\mathrm{var}^*[w_{\mathbf{X}}(\mathbf{X})\beta(\mathbf{X})] = v_2$ by (4.16). Furthermore, because $\mathbf{Y}^{(\ell,1)},\ldots,\mathbf{Y}^{(\ell,L_Y)}$ are conditionally independent given $\mathbf{X}^{(\ell)}$, $\mathrm{var}^*[\hat{\beta}(\mathbf{X},\mathbf{Z})|\mathbf{X}]$ in the first term is simply

$$\mathrm{var}^*[\hat{\beta}(\mathbf{X},\mathbf{Z})|\mathbf{X}] = \frac{1}{L_Y}\,\mathrm{var}^*[g(\mathbf{X},\mathbf{Y})w_{\mathbf{Y}|\mathbf{X}}(\mathbf{Y}|\mathbf{X})|\mathbf{X}].$$

Therefore the first term is v_1/L_Y.

We have used in the proof a conditional variance formula which is

$$\mathrm{var}[f(\mathbf{X},\mathbf{Z})] = \mathrm{E}\left[\,\mathrm{var}[f(\mathbf{X},\mathbf{Z})|\mathbf{X}]\,\right] + \mathrm{var}\left[\,\mathrm{E}[f(\mathbf{X},\mathbf{Z})|\mathbf{X}]\,\right].$$

This can be shown as follows. We have

$$\mathrm{var}[f(\mathbf{X},\mathbf{Z})] = \mathrm{E}[f^2(\mathbf{X},\mathbf{Z})] - (\mathrm{E}[f(\mathbf{X},\mathbf{Z})])^2$$

where

$$\begin{aligned}
\mathrm{E}[f^2(\mathbf{X},\mathbf{Z})] &= \mathrm{E}\left[\,\mathrm{E}[f^2(\mathbf{X},\mathbf{Z})|\mathbf{X}]\,\right] \\
&= \mathrm{E}\left[\,\mathrm{var}[f(\mathbf{X},\mathbf{Z})|\mathbf{X}]\,\right] + \mathrm{E}\left[\,(\mathrm{E}[f(\mathbf{X},\mathbf{Z})|\mathbf{X}])^2\,\right].
\end{aligned}$$

The second term in the last equation can be expressed as

$$\begin{aligned} \mathrm{E}\left[\,(\mathrm{E}[f(\mathbf{X},\mathbf{Z})|\mathbf{X}])^2\,\right] &= \mathrm{var}\left[\,\mathrm{E}[f(\mathbf{X},\mathbf{Z})|\mathbf{X}]\,\right] + (\mathrm{E}\left[\,\mathrm{E}[f(\mathbf{X},\mathbf{Z})|\mathbf{X}]\,\right])^2 \\ &= \mathrm{var}\left[\,\mathrm{E}[f(\mathbf{X},\mathbf{Z})|\mathbf{X}]\,\right] + (\mathrm{E}[f(\mathbf{X},\mathbf{Z})])^2 \end{aligned}$$

which completes the proof. ¶

In Section 4.1, we have considered the optimization of the estimator by minimizing L_ϵ of (4.2) via the minimization of the one sample variance. We now have two parameters to select: L_X and L_Y. One might first think that we should set L_Y sufficiently large so that each $\hat{\beta}\left(\mathbf{X}^{(\ell)},\mathbf{Z}^{(\ell)}\right)$ forms a precise estimate of the conditional expectation $\beta\left(\mathbf{X}^{(\ell)}\right)$. However, it turns out that this is not a good strategy. Suppose that we want to select L_Y to minimize $\mathrm{var}^*[\hat{\alpha}] = (v_1 + L_Y v_2)/(L_X L_Y)$ for a fixed total number of samples $L = L_X L_Y$. Clearly, the best choice in this sense is $L_Y = 1$ because both v_1 and $v_1 \geq 0$.

More generally, the choice of L_Y should be influenced by per sample computational cost. Let C_X and C_Y denote the costs of sampling and computing the associated weighting functions for single samples of, respectively, $\mathbf{X}^{(\ell)}$ and $\mathbf{Y}^{(\ell,\ell')}$. Then the total cost is $C = L_X(C_X + L_Y C_Y)$. We should set L_X and L_Y to minimize C subject to the precision constraint $\mathrm{var}^*[\hat{\alpha}] \leq \epsilon^2\alpha^2$. That is, L_X and L_Y are solutions to the constrained optimization problem:

$$\begin{cases} \min \quad L_X(C_X + L_Y C_Y) \\ \text{subject to} \quad (v_1 + L_Y v_2)/(L_X L_Y) \leq \epsilon^2\alpha^2. \end{cases}$$

Applying the Kuhn-Tucker conditions [35], we have the following linear system equations

$$\begin{cases} C_X + L_Y C_Y - \lambda L_Y \epsilon^2\alpha^2 = 0 \\ L_X C_Y + \lambda(v_2 - L_X\epsilon^2\alpha^2) = 0 \\ v_1 + L_Y v_2 - L_X L_Y \epsilon^2\alpha^2 = 0 \end{cases}$$

where λ is the Lagrange multiplier. We thus get

$$L_Y = \left\lceil \sqrt{\frac{C_X v_1}{C_Y v_2}} \right\rceil \tag{4.22}$$

$$L_X = \left\lceil \frac{1}{\epsilon^2 \alpha^2}(v_1/L_Y + v_2) \right\rceil . \tag{4.23}$$

Note that L_Y does not depend on ϵ.

As a practical matter, the solution (4.22) cannot be used to numerically set L_Y. While computational costs C_X and C_Y can be determined experimentally, we will not know the factors v_1 and v_2 which are constants once we have selected the simulation densities $f^*_{\mathbf{X}}(\cdot)$ and $f^*_{\mathbf{Y}|\mathbf{X}}(\cdot\,|\mathbf{x})$. (Although we can empirically estimate them, no good strategy has been found which can effectively determine the ratio C_X/C_Y for the purpose of setting L_Y.) In our application to coded digital communication systems the cost ratio C_X/C_Y will be usually large (e.g., 100) due to the computation of the optimal mean translation, hence, setting $L_Y > 1$ will be justified. For uncoded systems, the difference between C_X and C_Y narrows (e.g., $C_X/C_Y \sim 10^0$). We will have a more concrete idea about C_X and C_Y in the examples of Chapter 6. In any cases, we should not be tempted subjectively set L_Y extremely large because of the square root in (4.22). Once L_Y is set, from (4.23), the required L_X depends on the standard error ϵ and α, or empirically, on the relative precision ε and the estimate $\hat{\alpha}$.

There is also the issue of selecting the simulation distributions as represented by $f^*_{\mathbf{X}}(\cdot)$ and $f^*_{\mathbf{Y}|\mathbf{X}}(\cdot\,|\mathbf{x})$. The minimum total cost, using (4.22) and (4.23), is

$$C_{min} \approx \frac{1}{\epsilon^2 \alpha^2}\left(\sqrt{v_1 C_Y} + \sqrt{v_2 C_X}\right)^2 .$$

Therefore, we should choose $f^*_{\mathbf{X}}(\cdot)$ and $f^*_{\mathbf{Y}|\mathbf{X}}(\cdot\,|\mathbf{x})$ to minimize v_1 and v_1. See (4.20). The latter density impacts only v_1, and, regardless of the choice of $f^*_{\mathbf{X}}(\cdot)$, v_1 is minimized by minimizing the conditional variance $\mathrm{var}^*[g(\mathbf{x},\mathbf{Y})w_{\mathbf{Y}|\mathbf{X}}(\mathbf{Y}|\mathbf{x})|\mathbf{X} = \mathbf{x}]$ for each $\mathbf{x}$. In particular, if $\mathbf{Y}$ is Gaussian distributed then the biasing strategies developed in Section 4.1 apply directly to the design of the conditional simulation density $f^*_{\mathbf{Y}|\mathbf{X}}(\cdot\,|\mathbf{x})$. Next, consider the selection of $f^*_{\mathbf{X}}(\cdot)$ which involves joint minimization of both v_1 and v_2. The factor v_2 is just the variance expression that one would

have for estimating $\alpha = \mathrm{E}[\beta(\mathbf{X})]$, and hence, to minimize this factor we should seek a simulation density that approximates $f^*_{\mathbf{X}}(\mathbf{x}) \propto \beta(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})$. Minimization of v_1 is not so clear because this factor also depends on our choice of $f^*_{\mathbf{Y}|\mathbf{X}}(\cdot\,|\mathbf{x})$. However, if we have done a good job selecting $f^*_{\mathbf{Y}|\mathbf{X}}(\cdot\,|\mathbf{x})$, then the conditional variance $\mathrm{var}^*[g(\mathbf{X},\mathbf{Y}) w_{\mathbf{Y}|\mathbf{X}}(\mathbf{Y}|\mathbf{X})|\mathbf{X}]$ will be roughly proportional to $\beta(\mathbf{X})^2$. In the case of the exponentially efficient Gaussian simulation schemes developed in Section 3, for example, the exponential factor (as a function of the SNR parameter γ) of the variance does in fact satisfy this proportionality. Thus, v_1 is approximately proportional to $\mathrm{E}^*[\beta(X)^2 w_{\mathbf{X}}(X)^2]$ which is again minimized by $f^*_{\mathbf{X}}(\mathbf{x}) \propto \beta(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})$.

Finally we remark that we may estimate the empirical precision of our estimates using a sample variance estimator as in Section 2.2. Since $\left(\mathbf{X}^{(\ell)}, \mathbf{Z}^{(\ell)}\right)$, $\ell = 1, .., L_X$, are i.i.d., an appropriate estimator for $\mathrm{var}[\hat{\beta}(\mathbf{X},\mathbf{Z}) w_{\mathbf{X}}(\mathbf{X})]$ is

$$\hat{S}^2 \;=\; \frac{1}{L_X} \sum_{\ell=1}^{L_X} \hat{\beta}\left(\mathbf{X}^{(\ell)}, \mathbf{Z}^{(\ell)}\right)^2 w_{\mathbf{X}}\left(\mathbf{X}^{(\ell)}\right)^2 \;-\; \hat{\alpha}^2. \tag{4.24}$$

And the variance of the estimator (4.17), $\mathrm{var}[\hat{\alpha}]$ is estimated by $\hat{S}^2/L_X$.

5. EVENT SIMULATIONS AND AWGN CHANNEL EXAMPLES

5.1 Event Simulations for Uncoded Systems

In this section we will discuss the event simulation method for uncoded systems and channels with memory. This method was originally presented by Lu and Yao in [36]. It is, in fact, simply an enlarged version of the event simulation we have described in Section 1.2. Consider a baseband binary digital communication system with memory as shown in Figure 5.1. This is also the system model used in [36] and by Shanmugan and Balaban in [53]. We use this simple system model to illustrate the formulation of the event simulation method. Its generalization to more complicated systems will follow.

The transmitted signal in Figure 5.1 is $X(t) = \sum_k X_k\, p(t - kT)$, where X_k takes on the value +1 or -1 with equal probability (this equal *a priori* probability actually is not required), $p(t)$ is a rectangular signaling pulse over the duration $[0, T]$ and T is the signaling period. The additive noise process $Y(t)$ is a white Gaussian noise process with zero mean and two-sided p.s.d. $N_0/2$. The sampler output R_k is obtained by taking one sample every T seconds from the system output waveform $R(t)$. The decision device then decides on X_k by imposing some decision rules on R_k.

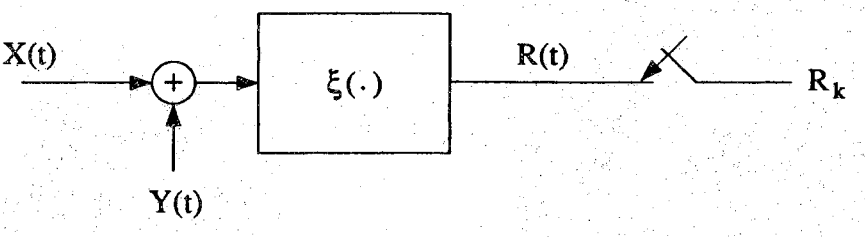


Figure 5.1 A binary communication system with memory.

To simplify present discussion, we assume that the system is linear so that we can represent the above continuous time model with only one sample per signaling interval T for both the signal and noise. Therefore, the signal samples can be represented by the sequence $\{X_k\}$. The discrete-time system with memory incorporating the effects of all the relevant filtering and the sampler can be modeled by

$$R_k = \xi\left(X_k, X_{k-1}, \ldots, X_{k-(n-1)}, Y_k, Y_{k-1}, \ldots, Y_{k-(n-1)}\right) \tag{5.1}$$

where $\xi(\cdot)$ is the discrete-time system response function, n is the system memory length and $\{Y_k\}$ are i.i.d. noise samples with zero mean and variance $\sigma^2 = N_0/2$.

Equation (5.1) is really the format we only need for deriving the event simulation method. Any uncoded communication system whose decision statistic can be expressed as a function of the current data sample plus the ISI samples and the noise samples can be simulated in an event-simulation fashion. If the system is time invariant, a more general formulation than (5.1) can be written as

$$\begin{aligned} R &= \xi\left(X_0, X_1, \ldots, X_{m-1}, Y_0, Y_1, \ldots, Y_{n-1}\right) \\ &= \xi(\mathbf{X}, \mathbf{Y}), \end{aligned} \tag{5.2}$$

where X_0 denotes the current data sample and $(X_1, \ldots, X_{m-1})$ is called an "ISI pattern" which is the influence on R coming from adjacent signal samples. Note that this representation has also removed the causality implied by the time indices in (5.1), hence the system can have noncausal filtering. The random variables $\{X_k\}$, $\{Y_k\}$ and R can be complex, and the dimension of $\mathbf{X}$, m, need not be the same as that of $\mathbf{Y}$, n, i.e., the signal memory does not have to be equal to the noise memory. The latter situation may happen when, say, $p(t)$ is a partial-response function or the noise process is not added to the signal at the same point in the system, that is, they go through different filtering. Furthermore, if $\xi(\cdot)$ is nonlinear, we may have many samples per transmitted symbol. In that case, every X_k (and Y_k) is itself a vector.

We now work on the general form (5.2). The ideas of the event simulation method and the conventional stream simulation method depart from here. While the stream

simulation sequentially generates R_k as in (5.1), the event simulation can be considered as taking only a snap shot at (any) one decision statistic. Suppose that $X_0 = x_0$ is transmitted and the decision rules are such that an incorrect decision on X_0 will be made if $R \in E(x_0) = E$, where E is the decision error region of x_0. Then the decision error probability $P = P(R \in E | X_0 = x_0) = \int_E f_{R|X_0}(r|x_0)\, dr$ can be expressed as

$$\begin{aligned} P &= P(\xi(\mathbf{X}, \mathbf{Y}) \in E | X_0 = x_0) \\ &= \int 1_E(\xi(\mathrm{x}, \mathrm{y}))\, f_{\mathbf{X},\mathbf{Y}|X_0}(\mathrm{x}, \mathrm{y}|x_0)\, d\mathrm{x}d\mathrm{y} \qquad (5.3) \\ &= \int 1_E(\xi(\mathrm{x}, \mathrm{y}))\, f_{\mathbf{X}|X_0}(\mathrm{x}|x_0) f_{\mathbf{Y}}(\mathrm{y})\, d\mathrm{x}d\mathrm{y} \qquad (5.4) \\ &= \frac{1}{J}\sum_{j=1}^{J} \int 1_E\left(\xi(\mathrm{x}^{(j)}, \mathrm{y})\right) f_{\mathbf{Y}}(\mathrm{y})\, d\mathrm{y} \qquad (5.5) \\ &= \frac{1}{J}\sum_{j=1}^{J} P(\mathrm{x}^{(j)}), \end{aligned}$$

where the first component of $\mathrm{x}^{(j)}, j = 1, \ldots, J$, is x_0, and

$$\begin{aligned} P(\mathrm{x}^{(j)}) &= \int 1_E\left(\xi(\mathrm{x}^{(j)}, \mathrm{y})\right) f_{\mathbf{Y}|\mathbf{x}^{(j)}}(\mathrm{y}|\mathrm{x}^{(j)})\, d\mathrm{y} \\ &= \mathrm{E}\left[1_E\left(\xi(\mathrm{x}^{(j)}, \mathbf{Y})\right) | \mathrm{x}^{(j)}\right] \\ &= \mathrm{E}\left[1_E(\xi(\mathbf{X}, \mathbf{Y})) | \mathrm{x}^{(j)}\right] \qquad (5.6) \end{aligned}$$

is the conditional decision error probability given a specific ISI pattern $\mathbf{X} = \mathrm{x}^{(j)}$. We have assumed that $f_{\mathbf{X},\mathbf{Y}|X_0}(\mathrm{x}, \mathrm{y}|x_0) = f_{\mathbf{X}|X_0}(\mathrm{x}|x_0) f_{\mathbf{Y}}(\mathrm{y})$, i.e., the signal and noise samples are independent, and that all of the ISI patterns (total number $= J$) are equally likely and therefore $f_{\mathbf{X}|X_0}(\mathrm{x}|x_0) = 1/J$. For the earlier binary system example, we would have $P = P_b$ and $J = 2^{(m-1)}$.

Lu and Yao's importance sampling event simulation then empirically estimates (5.5) by the following estimator:

$$\hat{P}^* = \frac{1}{L}\sum_{j=1}^{J}\sum_{\ell=1}^{L/J} 1_E\left(\xi(\mathrm{x}^{(j)}, \mathbf{Y}^{(j,\ell)})\right) w_{\mathbf{Y}}\left(\mathbf{Y}^{(j,\ell)}\right), \qquad (5.7)$$

where $w_{\mathbf{Y}}(\mathbf{y}) = f_{\mathbf{Y}}(\mathbf{y})/f^*_{\mathbf{Y}}(\mathbf{y})$ is the IS weighting function and $\mathbf{Y}^{(j,\ell)}$ is the ℓth sample of the noise $\mathbf{Y}$ for each of the N/J simulations conditioned on each $\mathbf{x}^{(j)}$ sample of $\mathbf{X}$. We note that $\mathrm{E}^*[\hat{P}^*] = \frac{1}{J}\sum_{j=1}^{J} P(\mathbf{x}^{(j)}) = P$, i.e., $\hat{P}^*$ is an unbiased estimator of P.

In the importance sampling simulation of the form (5.7), only the marginal noise density function $f_{\mathbf{Y}}(\mathbf{Y})$ is biased. Recall from Section 4.3 that we can bias the marginal signal density function $f_{\mathbf{X}|X_0}(\mathbf{x}|x_0)$ in (5.4) too. The advantage of having a signal biasing is twofold. Firstly, recall from the discussion following (4.14) that some ISI patterns $\mathbf{X} = \mathbf{x}^{(j)}$ are more likely to cause a decision error than others. That is, see (5.5), those $\mathbf{x}^{(j)}$ have more significant corresponding conditional decision error probabilities $P(\mathbf{x}^{(j)})$. Therefore, we can design a signal biasing $f^*_{\mathbf{X}|X_0}(\mathbf{x}|x_0)$ to "encourage" the generation of more of these "important" ISI patterns. As long as we do this in a reversible way as we do for the noise distribution, we can make use of our computational resource more efficiently. Secondly, if the total number of ISI patterns (J) is a large number, which is the case for most practical systems with memory, the number of simulation runs, L/J, devoted to each conditional decision error probability will be small, or conversely L must be large. In this case, similar to the noise, we should randomly generate an ISI pattern in each simulation run rather than distributing the total number of simulation runs evenly among all possible ISI patterns. Thus, if we can bias the sampling of the noise, there is no reason why we can't do so to the signal. (Signal biasings have not been a common practice in the importance sampling literature, or for that matter, any simulation study. This is because the conventional stream simulation method has been the norm for which the signal biasing is not very productive. As mentioned in Section 1.2, the biasing for signal or noise in the stream simulation environment would create biasing conflicts due to correlated decision statistics in successive simulation runs.)

Looking at (5.3), one might immediately think of a straightforward form of estimator which incorporates the signal biasing. That is, we employ a joint simulation density $f^*_{\mathbf{X},\mathbf{Y}|X_0}(\mathbf{x},\mathbf{y}|x_0) = f^*_{\mathbf{Y}|\mathbf{X},X_0}(\mathbf{y}|\mathbf{x},x_0) f^*_{\mathbf{X}|X_0}(\mathbf{x}|x_0)$, and the estimator which empirically evaluates (5.3) becomes

$$\hat{P}^* = \frac{1}{L}\sum_{\ell=1}^{L} 1_E\left(\xi(\mathbf{X}^{(\ell)},\mathbf{Y}^{(\ell)})\right) w_{\mathbf{X}|X_0}\left(\mathbf{X}^{(\ell)}|x_0\right) w_{\mathbf{Y}|\mathbf{X},X_0}\left(\mathbf{Y}^{(\ell)}|\mathbf{X}^{(\ell)},x_0\right), \tag{5.8}$$

where

$$w_{\mathbf{X}|X_0}(\mathbf{x}|x_0) = \frac{f_{\mathbf{X}|X_0}(\mathbf{x}|x_0)}{f^*_{\mathbf{X}|X_0}(\mathbf{x}|x_0)} = \frac{1}{Jf^*_{\mathbf{X}|X_0}(\mathbf{x}|x_0)}$$

$$w_{\mathbf{Y}|\mathbf{X},X_0}(\mathbf{y}|\mathbf{x},x_0) = \frac{f_{\mathbf{Y}}(\mathbf{y})}{f^*_{\mathbf{Y}|\mathbf{X},X_0}(\mathbf{y}|\mathbf{x},x_0)}$$

are IS weighting functions for signal and noise respectively. (Note that if a signal-independent noise biasing is used, we would have $f^*_{\mathbf{Y}|\mathbf{X},X_0}(\mathbf{y}|\mathbf{x},x_0) = f^*_{\mathbf{Y}}(\mathbf{y})$. However, we have demonstrated in Chapter 4 that signal-dependent noise biasings generally yield higher efficiency.) Therefore, in each of the L simulation runs, we randomly generate a signal sample $\mathbf{X}^{(\ell)}$ and a noise sample $\mathbf{Y}^{(\ell)}$ from the biased joint density function $f^*_{\mathbf{X},\mathbf{Y}|X_0}(\mathbf{x},\mathbf{y}|x_0)$. It is easy to show that $\hat{P}^*$ is an unbiased estimator: $\mathrm{E}^*[\hat{P}^*] = \mathrm{E}\left[1_E\left(\xi(\mathbf{X},\mathbf{Y})\right)|X_0 = x_0\right] = P$.

Equation (5.8) actually corresponds to the $L_Y = 1$ case in Section 4.3 where we have shown that $L_Y = 1$ may not be the optimal choice to minimize the total computational cost. Hence, the appropriate formulation for this problem is exactly what we have presented in Section 4.3, i.e., we want to estimate

$$P = \mathrm{E}[1_E\left(\xi(\mathbf{X},\mathbf{Y})\right)|X_0 = x_0] = \mathrm{E}\left[\mathrm{E}[1_E\left(\xi(\mathbf{X},\mathbf{Y})\right)|\mathbf{X},X_0 = x_0]\right]. \tag{5.9}$$

The estimator is therefore the combination of (4.15) and (4.17):

$$\hat{P}^* = \frac{1}{L_X}\sum_{\ell=1}^{L_X} w_{\mathbf{X}|X_0}\left(\mathbf{X}^{(\ell)}|x_0\right) \times \left\{\frac{1}{L_Y}\sum_{\ell'=1}^{L_Y} 1_E\left(\xi(\mathbf{X}^{(\ell)},\mathbf{Y}^{(\ell,\ell')})\right) w_{\mathbf{Y}|\mathbf{X},X_0}\left(\mathbf{Y}^{(\ell,\ell')}|\mathbf{X}^{(\ell)},x_0\right)\right\}. \tag{5.10}$$

The proof of the unbiasness of (5.10) proceeds as in (4.18) and we omit here.

Since $Y_k, k = 1,\ldots,n$, are conditionally i.i.d. Gaussian random variables, the optimal Gaussian simulation density obtained in Section 4.1 applies readily to the design of $f^*_{\mathbf{Y}|\mathbf{X},X_0}(\mathbf{y}|\mathbf{x},x_0)$. The choice of $f^*_{\mathbf{X}|X_0}(\mathbf{x}|x_0)$, as discussed in Section 4.3,

should be proportional to $\beta(\mathbf{x})f_{\mathbf{X}|X_0}(\mathbf{x}|x_0) = P(\mathbf{x})/J$ where $\beta(\mathbf{x}) = P(\mathbf{x})$ is the conditional decision error probability defined by (5.6). $P(\mathbf{x})$ is unknown, but, for the purpose of designing $f^*_{\mathbf{X}|X_0}(\mathbf{x}|x_0)$ we can obtain an approximation by appealing to the ideal channel behavior. Examples of optimal designs for $f^*_{\mathbf{Y}|\mathbf{X},X_0}(\mathbf{y}|\mathbf{x},x_0)$ for linear systems can be found in Lu and Yao [36]. The design of $f^*_{\mathbf{Y}|\mathbf{X},X_0}(\mathbf{y}|\mathbf{x},x_0)$ as well as $f^*_{\mathbf{X}|X_0}(\mathbf{x}|x_0)$ for a more general case — nonlinear systems with memory will be given in Chapter 6 via the example of an MSK modulation operating on a satellite channel.

5.2 Event Simulations for Trellis Codes

5.2.1 Event-Simulation Estimator

Recall from Chapter 1 that in the stream simulation of a Viterbi decoder, the decoder's branching decisions are correlated which includes scattered error events and intervals of correct paths lying in between. The bit error probability P_b is easily estimated by the relative frequency of erroneous decoded information bits. Accompanying with this simulation method, however, are the diminishing dimensionality effect when the importance sampling technique is applied and the difficulty to compute the estimator's variance.

Similar to the development of the event simulation method for uncoded systems in the last section, let us do some analysis on the decision error probability before jumping to the intuitive stream simulation. (The whole idea of an efficient importance sampling is to construct an analysis-based simulation and the analysis will pay off.) In Section 3.2.2, we have demonstrated that the first event error probability P_e and the bit error probability P_b for a trellis code can be expressed as (3.1) and (3.4) respectively which are rewritten as

$$P_e = \sum_{\mathbf{e}\neq\mathbf{0},\mathbf{e}\in\mathbf{C}} \mathrm{E}\left[P(\mathbf{X}+\mathbf{e}|\mathbf{X})\right] \tag{5.11}$$

and

$$P_b = \frac{1}{b}\sum_{\mathbf{e}\in\mathbf{C}} n_b(\mathbf{e})\,\mathrm{E}\left[P(\mathbf{X}+\mathbf{e}|\mathbf{X})\right] \tag{5.12}$$

where we have moved the expectations (with respect to the random signal selector sequence $\mathbf{X}$) inside the summations assuming that the infinite sums do converge. Also, the random variable $\mathrm{N}_b(\mathbf{X}, \mathbf{X}+\mathbf{e})$ in the summand of (3.4) has been replaced by $n_b(\mathbf{e})$ due to the linearity of the convolutional code. Note that (5.11) and (5.12) are also true for block codes. Thus, most of the following discussion also applies, at least in principle, to block codes.

See (5.11) and (5.12). It is logical for us to suspect that there are some signal selector error sequences $\mathbf{e}$ whose corresponding "expected specific decoding error probability" $\mathrm{E}[P(\mathbf{X}+\mathbf{e}|\mathbf{X})] = P(\mathbf{e}$ is decoded) are the dominant terms of the values of P_e and P_b. That is, some error sequences are more likely to be decoded than others. Hence, it would be nice if we can estimate P_e and P_b by identifying those "important" $\mathbf{e}$'s and summing only their estimated $\mathrm{E}[P(\mathbf{X}+\mathbf{e}|\mathbf{X})]$. This will inevitably introduce an estimation "truncation bias." But we will show that if the pool of $\mathbf{e}$ is large enough and/or the SNR is high, the truncation bias will be negligible.

For regular codes and the AWGN channel, the task of identifying important error sequences $\mathbf{e}$ is relatively easy. Recall from Section 3.2.1 that for regular codes, the Euclidean distance between two codewords depends only on the Hamming distance between their input information sequences (or signal selector sequences if the discrete encoder is linear). Also, on the AWGN channel, the Euclidean distance between two codewords determines the likelihood of decoding one codeword while the other one is sent, using a maximum likelihood decoder. Hence, $P(\mathbf{X}+\mathbf{e}|\mathbf{X}) = P(\mathbf{e}|\mathbf{0})$, where $\mathbf{0}$ is the all zero signal selector sequence, and thus important error sequences are those $\mathbf{e}$ with small Hamming weights (number of non-zero bits).

For non-regular codes, the Euclidean distance between two codewords depends not only on the Hamming distance between them but also what the two codewords actual are. Furthermore, for channels with memory, $P(\mathbf{X}+\mathbf{e}|\mathbf{X})$ depends also on the ISI pattern, i.e., adjacent signal selectors. Therefore, instead of looking for important error sequences $\mathbf{e}$, we need to find important error sequence and signal selector pairs,

$(\mathbf{x}, \mathbf{e})$, with $\mathbf{x}$ taking into account the intersymbol interference. To do this, we require more general and explicit formulas than (5.11) and (5.12).

Let M^- and M^+ denote the "backward" and "forward" memory lengths respectively of the channel, measured in numbers of symbol durations T. More specifically, because of ISI, the demodulator output symbol R_k is affected by signal selectors $(X_{k-M^-}, \ldots, X_{k+M^+})$. The total channel memory length is thus $M = M^- + M^+$. We allow $M^- \neq M^+$ here and in the simulation program for sake of generality. If the system is causal, we have $M^+ = 0$, and hence R_k depends only on the current and M^- previous signal selectors. For each $\ell = 1, 2, \ldots$, let $\mathcal{X}(\ell)$ denote the set of signal selector sequences of length $\ell + M$ beginning in a random initial state at trellis stage $k = -M^-$. Consider a fixed error sequence $\mathbf{e}$ of length $\ell(\mathbf{e})$. The expected specific decoding error probability $\mathrm{E}[P(\mathbf{X} + \mathbf{e}|\mathbf{X})]$ for $\mathbf{e}$ can be expressed as

$$\begin{aligned}
\mathrm{E}[P(\mathbf{X} + \mathbf{e}|\mathbf{X})] &= P(\mathbf{e} \text{ is decoded}) \\
&= \mathrm{E}_{\tilde{\mathbf{X}},\mathbf{Y}}\left[1_{\{\mathbf{e} \text{ is decoded}\}}(\tilde{\mathbf{X}}, \mathbf{Y})\right] \\
&= \mathrm{E}_{\tilde{\mathbf{X}}}\left[\mathrm{E}_{\mathbf{Y}}\left[1_{\{\mathbf{e} \text{ is decoded}\}}(\tilde{\mathbf{X}}, \mathbf{Y})|\tilde{\mathbf{X}}\right]\right] \\
&= \mathrm{E}_{\tilde{\mathbf{X}}}\left[P(\mathbf{e} \text{ is decoded})|\tilde{\mathbf{X}}\right] \\
&= \sum_{\tilde{\mathbf{x}} \in \mathcal{X}(\ell(\mathbf{e}))} P(\mathbf{e} \text{ is decoded}|\tilde{\mathbf{x}}) P_{\tilde{\mathbf{X}}}(\tilde{\mathbf{x}}) \\
&= \sum_{\tilde{\mathbf{x}} \in \mathcal{X}(\ell(\mathbf{e}))} P(\tilde{\mathbf{x}} + \mathbf{e}|\tilde{\mathbf{x}})\, 2^{-b(\ell(\mathbf{e})+M)} \qquad (5.13)
\end{aligned}$$

where $\mathbf{Y}$ is the Gaussian noise samples vector. We have used a subscript for the expectation to indicate the underlying probability measure. The factor $2^{-b(\ell(\mathbf{e})+M)}$ is just the marginal probability $P(\tilde{\mathbf{X}} = \tilde{\mathbf{x}})$. The single term probability $P(\tilde{\mathbf{x}} + \mathbf{e}|\tilde{\mathbf{x}})$ in (5.13) is determined by the Gaussian noise distribution, and, since the Viterbi decoder has a random decoding delay, $P(\tilde{\mathbf{x}} + \mathbf{e}|\tilde{\mathbf{x}})$ is also averaged with respect to the "tail sequence" $X_{\ell(\mathbf{e})+M^++1}, X_{\ell(\mathbf{e})+M^++2}, \ldots$. More precisely,

$$P(\tilde{\mathbf{x}} + \mathbf{e}|\tilde{\mathbf{x}}) = \mathrm{E}\left[\, P(\mathbf{X} + \mathbf{e}|\mathbf{X}) \,|\, \tilde{\mathbf{X}} = \tilde{\mathbf{x}} \,\right]$$

where $\tilde{\mathbf{X}} = (X_{1-M^-}, \ldots, X_{\ell(\mathbf{e})+M^+}) \in \mathcal{X}(\ell(\mathbf{e}))$ is a finite subsequence of the infinite sequence $\mathbf{X}$. Combining (5.13) and (5.12) we have

$$P_b = \sum_{\mathbf{e}\in C} \sum_{\tilde{\mathbf{x}}\in\mathcal{X}(\ell(\mathbf{e}))} \frac{n_b(\mathbf{e})}{b} P(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}})\, 2^{-b(\ell(\mathbf{e})+M)}. \tag{5.14}$$

For a fixed $\mathbf{e}$, it will be convenient to write $\tilde{\mathbf{X}} = (\mathbf{X}^-, \mathbf{X}, \mathbf{X}^+)$ where $\mathbf{X}^- = (X_{1-M^-}, \ldots, X_0)$, $\mathbf{X} = (X_1, \ldots, X_{\ell(\mathbf{e})})$, and $\mathbf{X}^+ = (X_{\ell(\mathbf{e})+1}, \ldots, x_{\ell(\mathbf{e})+M^+})$. In the case of a memoryless channel, i.e., no ISI, we have $M = 0$ and $\tilde{\mathbf{X}} = \mathbf{X}$.

We are now ready to design an event simulation based on the expression (5.14). Define $\mathcal{D} = \{\, (\mathbf{e}, \tilde{\mathbf{x}}) : \mathbf{e} \in C \text{ and } \tilde{\mathbf{x}} \in \mathcal{X}(\ell(\mathbf{e}))\,\}$. Let $P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e}, \tilde{\mathbf{x}})$ be a discrete distribution on $\mathcal{D}$, and let $(\mathbf{E}^{(\ell)}, \tilde{\mathbf{X}}^{(\ell)}), \ell = 1, \ldots, L_X$, be independent samples from $P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e}, \tilde{\mathbf{x}})$. The estimator of P_b which evaluates (5.14) is

$$\hat{P}_b = \frac{1}{L_X} \sum_{l=1}^{L_X} \hat{P}\left(\tilde{\mathbf{X}}^{(\ell)} + \mathbf{E}^{(\ell)} | \tilde{\mathbf{X}}^{(\ell)}\right) w_{\mathbf{E},\tilde{\mathbf{X}}}\left(\mathbf{E}^{(\ell)}, \tilde{\mathbf{X}}^{(\ell)}\right) \tag{5.15}$$

where

$$w_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e}, \tilde{\mathbf{x}}) = \frac{n_b(\mathbf{e})\, 2^{-b(\ell(\mathbf{e})+M)}}{b\; P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e}, \tilde{\mathbf{x}})}$$

is the importance sampling weighting function for the signal $\tilde{\mathbf{X}}$ and error sequence $\mathbf{E}$, and $\hat{P}(\tilde{\mathbf{X}}^{(\ell)} + \mathbf{E}^{(\ell)}|\tilde{\mathbf{X}}^{(\ell)})$ is a conditional estimate of the first event error probability given $(\mathbf{E}^{(\ell)}, \tilde{\mathbf{X}}^{(\ell)})$, $P(\tilde{\mathbf{X}}^{(\ell)} + \mathbf{E}^{(\ell)}|\tilde{\mathbf{X}}^{(\ell)})$, which is

$$\hat{P}(\tilde{\mathbf{X}}^{(\ell)} + \mathbf{E}^{(\ell)}|\tilde{\mathbf{X}}^{(\ell)}) = \frac{1}{L_Y} \sum_{\ell'=1}^{L_Y} 1_{\{\mathbf{E}^{(\ell)} \text{ is decoded}\}}\left(\tilde{\mathbf{X}}^{(\ell)}, \mathbf{Y}^{(\ell,\ell')}\right) \times$$

$$w_{\mathbf{Y}|\mathbf{E},\tilde{\mathbf{X}}}\left(\mathbf{Y}^{(\ell,\ell')}|\mathbf{E}^{(\ell)}, \tilde{\mathbf{X}}^{(\ell)}\right) \tag{5.16}$$

where

$$w_{\mathbf{Y}|\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{y}|\mathbf{e}, \tilde{\mathbf{x}}) = \frac{f_{\mathbf{Y}}(\mathbf{y})}{f^*_{\mathbf{Y}|\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{y}|\mathbf{e}, \tilde{\mathbf{x}})}.$$

It is easy to show that $\mathrm{E}^*\left[\hat{P}(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}})\right] = P(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}})$. Hence, provided that $P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e}, \tilde{\mathbf{x}}) > 0$ for all $(\mathbf{e}, \tilde{\mathbf{x}}) \in \mathcal{D}$, it follows from (5.14) and (4.18) that $\mathrm{E}^*[\hat{P}_b] = \mathrm{E}^*\left[\mathrm{E}^*[\hat{P}_b|\mathbf{E}, \tilde{\mathbf{X}}]\right] = P_b$, that is, $\hat{P}_b$ is an unbiased estimator for P_b. In practice, since

$\mathcal{D}$ is an infinite set we will have to restrict the support of $P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e},\tilde{\mathbf{x}})$ to a finite subset. This will introduce a truncation bias. However, if the support of $P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e},\tilde{\mathbf{x}})$ includes the dominant terms in (5.14), then this truncation bias will be insignificant.

The optimal Gaussian simulation density developed in Section 4.1 again applies here for $f^*_{\mathbf{Y}|\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{y}|\mathbf{e},\tilde{\mathbf{x}})$. Note that $\mathbf{e}$ and $\tilde{\mathbf{x}}$ together decides the error set, and thus the error set notation in Chapter 4 is really $E(\mathbf{e},\tilde{\mathbf{x}})$. Therefore, the optimal simulation density will depend on both the signal $\tilde{\mathbf{x}}$ and the specific error $\mathbf{e}$. The design of the signal biasing $P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e},\tilde{\mathbf{x}})$ is more complicated than the uncoded case, especially for non-regular codes and channels with memory. We postpone its discussion until we study some examples later in this chapter and the next chapter.

5.2.2 The Error Event Simulation Method

To estimate $P(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}})$, which is the probability that the specific error event $\tilde{\mathbf{x}}+\mathbf{e}$ is decoded while $\tilde{\mathbf{x}}$ is the correct path, a new Viterbi decoder simulation method which is error-event oriented is in order. The error event (simulation) method for Viterbi decoders developed by Sadowsky in [46] is an answer to this call. For a detailed description of this method, the readers are directed to [46]. The advantages of this method, as mentioned in Chapter 1 and have become clear through the discussion so far, are the independence between simulation runs, allowing signal-dependent noise biasings, and that the code distance information can be utilized to design a signal biasing to further speed up the simulation.

The error event simulation method for Viterbi decoders can be summarized as follows. First the decoder is initialized in a known state, randomly selected each time, at time index (stage) $k = 0$. This initialization is because we are interested in $\tilde{\mathbf{x}} + \mathbf{e}$ is a first event error path which diverges from the correct path $\tilde{\mathbf{x}}$ at stage 0. (The $\mathbf{x}^-$ part of $\tilde{\mathbf{x}}$ is to simulate the ISI, and since $e_k = 0, k < 1$, we don't initialize the decoder at stage $-M^-$ although $\tilde{\mathbf{x}}$ starts from there.) The Viterbi algorithm then proceeds until the first merger of the error event path with the correct path after stage 0 is detected. We call this a "simulation run." In the case that the first branching

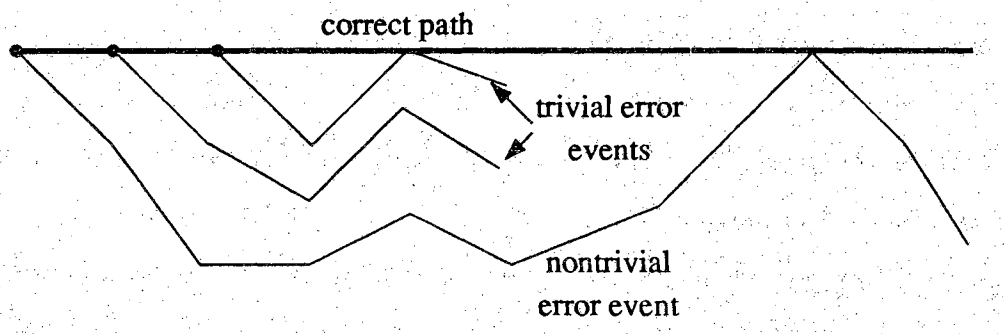


Figure 5.2 Trivial and nontrivial error events in the event simulation.

decision is correct, and thus the error event cannot be a first event error, we call it a "trivial error event." Therefore, each simulation run simulates one and only one error event. Whenever a trivial error event or the completion of a nontrivial error event is detected, that simulation run is terminated. See Figure 5.2 for examples of trivial and nontrivial error events. Note that the correct path need not be the all zero path.

In this manner, the Viterbi decoder is simulated in an event fashion similar to the event simulation method for uncoded systems discussed in the last section where, by conditioning on an ISI pattern, the receiver in each simulation run makes a binary correct/error decision. More specifically, the receiver decides in each run if the event "test bit error" has occurred. Decisions in consecutive runs are independent because different sets of random vectors $\mathbf{Y}$ are used. Here, by conditioning on the correct path, the decoder decides in each run whether or not an error event has occurred. A trivial error event corresponds to a "correct" decision in the uncoded case, while a nontrivial error event is an "error." The Viterbi decoder also makes independent trivial/nontrivial error event decisions in successive runs, again because the use of independent sets of random vectors.

Without importance sampling, the simulation runs will consist of mostly trivial error events. The objective of IS then is to increase the frequency of nontrivial error events, more precisely, to cause the particular nontrivial error event $\tilde{\mathbf{x}}+\mathbf{e}$. In the event simulation for a binary uncoded system, every decision error is counted in computing

the estimated bit error probability P_b. Here, the Viterbi decoder has more than two decoding options. Some nontrivial error events may not be the desired specific error $\tilde{\mathbf{x}} + \mathbf{e}$ and will be discarded in computing $P(\tilde{\mathbf{x}} + \mathbf{e}|\tilde{\mathbf{x}})$.

The independence between simulation runs requires some explanation. First though we review briefly the mechanism of the Viterbi algorithm [17, 62, 63]. Consider the example shown in Figure 5.3. At every stage, the Viterbi decoder performs, for each of the 2^ν states, ACS (Add, Compare and Select) functions and keeps track of the survivor path and its accumulated metric. (A programming trick which can reduce the memory size required for storing survivor paths and back-tracing by using a size 2^ν pointer array is given in [46].) The key principle of the Viterbi decoding process is that the survivor path for any state at stage $k+1$ is a one-branch extension of some survivor path at stage k. Note that the decoder does not discard paths entering a state with smaller metrics until after the transient stages or when candidate paths have reached all states. For the example in Figure 5.3, the transient stages are $k = 1, 2$.

Let $T_D(i)$ = the time that ith branch is decoded = the time all survivor paths agree on the ith branch, J = the time of the completion of an error event, and T_M = the time that a complete error event is decoded = length of simulation run. In the example, we have $T_D(1) = T_D(2) = 4, T_D(3) = 5$, and $T_D(4) = T_D(5) = T_M = 7$, and hence $J = 5$. The subscripts D and M stand for "decoded" and "merger" respectively. The minimum simulation length is $T_M = 3$ for the earliest possible detection of a trivial error event.

Because of the random noise, the first time that the maximum metric path remerges with the correct path, J, and the time of the detection of this event, T_M, are random variables. Furthermore they are "stopping times" [11]. That is, the event $\{T_M = t\}$ is measurable by the noise statistics up to time $t + M^+$. Note that T_M is also the time that the decoder decides whether or not the specific error event $\tilde{\mathbf{x}} + \mathbf{e}$ has been decoded. Therefore, the decision in the ℓ'th simulation run of the estimator (5.16) depends only on noise samples, generated by the simulation density

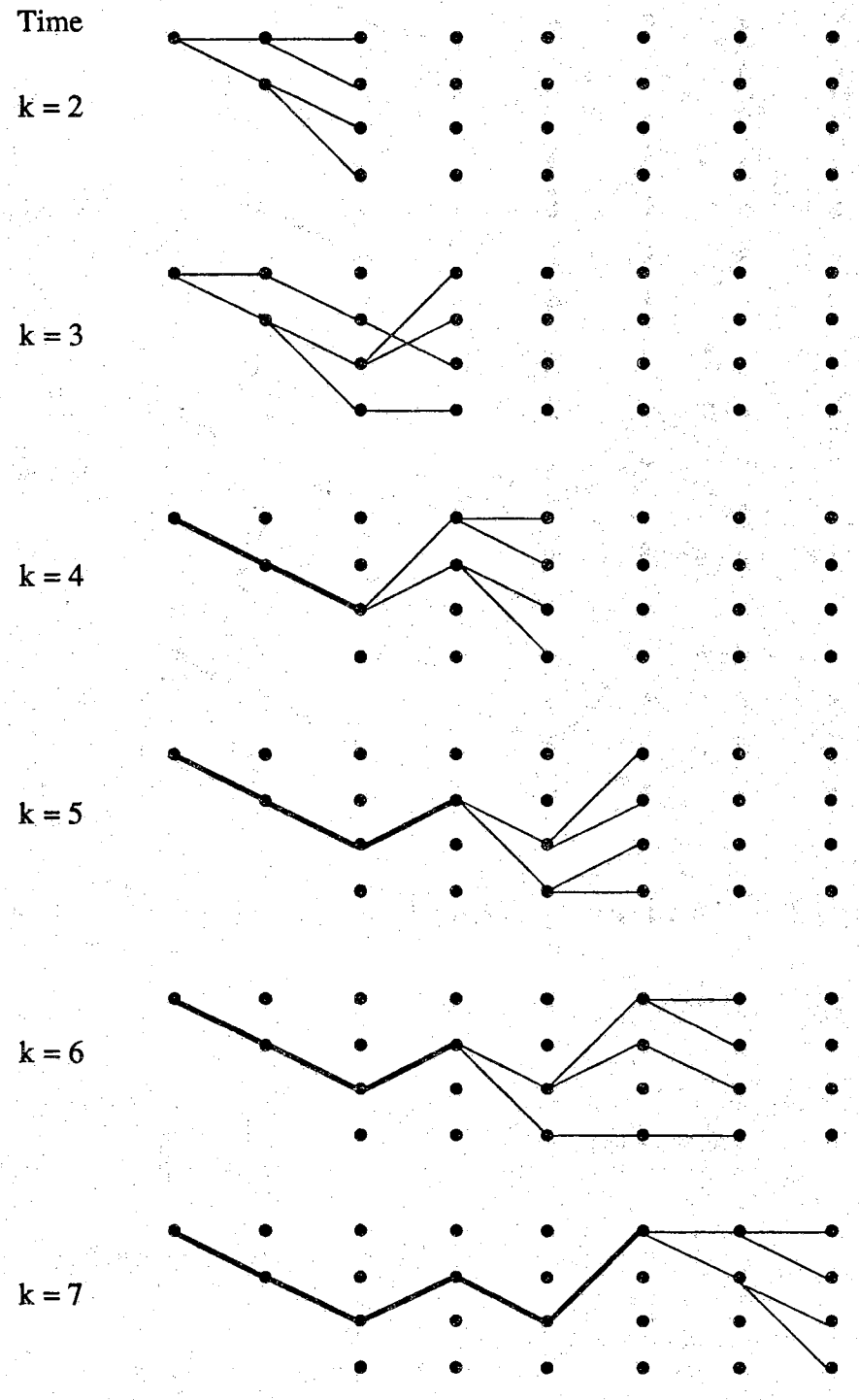


Figure 5.3 An example of event simulation. Correct path = all 0 path.

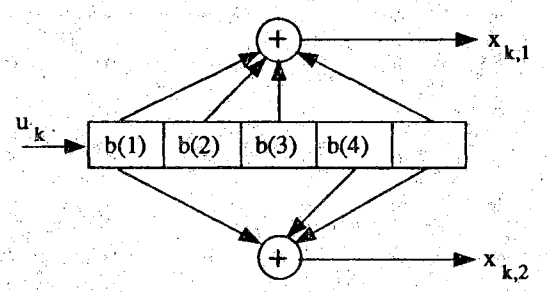
$f^*_{\mathbf{Y}|\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{y}|\mathbf{e},\tilde{\mathbf{x}})$, up to time $T_M + M^+$. This confirms the claim that simulation runs are independent. Also, it is necessary to generate signal selector $\tilde{x}_k$ and noise samples in each simulation run only up to time $T_M + M^+$.

5.3 Ideal Channel Simulations

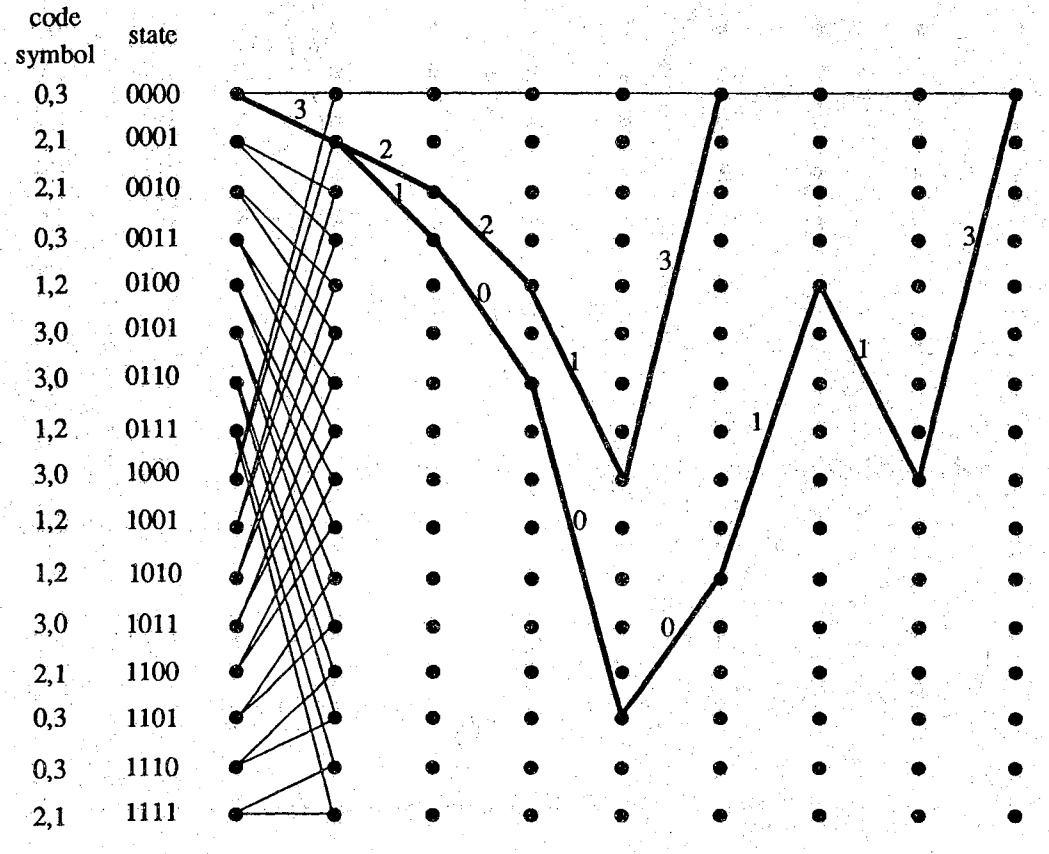
In this section, we will put together those principles for efficient IS simulations we have been discussing so far — event simulation, conditional importance sampling and Gaussian mean-translation biasing and use two trellis code examples to demonstrate the techniques and procedures in carrying out the simulation. The channel model will be the ideal linear memoryless AWGN channel, while the nonlinear satellite channel with memory case will be covered in the next chapter. Simulation results and analytical error bounds will be given.

5.3.1 A Convolutional Code

Consider a 16-state, code rate $R = 1/2$ convolutional code with BPSK or QPSK (with Gray coding) modulation operating on the linear memoryless AWGN channel. Figure 5.4(a) shows the discrete convolutional encoder configuration whose code generator is (35,23) [12]. The left-most shift register element $b(1)$ contains the most recent incoming information bit and is also the least significant bit of the encoder state index. The 2-bit encoder output $x_k = (x_{k,1}, x_{k,2})$ responding to an information bit input u_k is called a code symbol or a signal selector as defined in Chapter 3. For notational convenience, we often use quaternary representations for code symbols, i.e., $(0,0) = 0, (0,1) = 1, (1,0) = 2, (1,1) = 3$. The code's trellis diagram is shown in Figure 5.4(b) where trellis branches are labeled with quaternary representations of code symbols. The only two d_{min} paths are also shown with the correct path being the all zero path. By definition, d_{min} is the minimum Hamming distance between all possible codewords (signal selector sequences). This code appears in Heller and



(a) Encoder configuration.



(b) Trellis diagram.

Figure 5.4 A 16-state, $R = 1/2$ and $d_{min} = 7$ convolutional code.

Jacobs' simulation study [26] and is known to be optimal for $R = 1/2$, 16-state codes in the sense that it achieves the maximum $d_{min} = 7$ [12].

A. Union Bounds:

As mentioned in Section 3.2.1, this is a regular code. Performance evaluation can be greatly simplified by assuming that the all 0 sequence is the transmitted codeword. Therefore, letting $\mathbf{X} = \mathbf{0}$ in (3.1) and (3.4), we have

$$\begin{aligned} P_e &= P_e(\mathbf{0}) = \sum_{\mathbf{e}\neq\mathbf{0},\mathbf{e}\in C} P(\mathbf{e}|\mathbf{0}) \qquad (5.17) \\ &\leq \sum_{d=d_{min}} A_d P_d \end{aligned}$$

and

$$\begin{aligned} P_b &= P_b(\mathbf{0}) = \frac{1}{b}\sum_{\mathbf{e}\in C} n_b(\mathbf{e}) P(\mathbf{e}|\mathbf{0}) \qquad (5.18) \\ &\leq \sum_{d=d_{min}} B_d P_d, \end{aligned}$$

where A_d = number of codewords at distance d from $\mathbf{0}$, and B_d = total information weight for codewords at distance d from $\mathbf{0}$. The distance measure d here is the Hamming distance which is a linear function of the Euclidean distance. P_d is the probability that $\mathbf{e}$ is decoded while $\mathbf{0}$ is transmitted, assuming $\mathbf{0}$ and $\mathbf{e}$ are the only two decoding options, which is an upper bound of $P(\mathbf{e}|\mathbf{0})$. Note that $b = 1$ for this code. The multiplicities A_d and B_d are obtained by counting the output of the modified RC Algorithm which is a list of $\mathbf{e}$ such that $7 \leq d(\mathbf{0},\mathbf{e}) \leq 17$, where $d(\mathbf{0},\mathbf{e})$ is also the Hamming weight of $\mathbf{e}$. The algorithm takes less 5 mimutes CPU time on a SUN SPARC 1 station to obtain this list. Table 7 in Appendix A shows the beginning part of the list for $d(\mathbf{0},\mathbf{e}) \leq 10$. The resulting distance (weight) spectrum is shown in Table 5.1. We will compare our simulation results against union bounds computed from this table.

Table 5.1 A convolutional code weight spectrum.

d	A_d	B_d	d	A_d	B_d
7	2	4	13	176	1324
8	3	12	14	432	3680
9	4	20	15	925	8967
10	16	72	16	1966	19686
11	37	225	17	3003	30017
12	68	500			

B. IS Simulation:

As mentioned in the last section, we can estimate $P(\mathbf{e}|\mathbf{0})$ for those important $\mathbf{e}$ only, which are $\mathbf{e}$'s with small Hamming weights. See Table 5.1. In the simulation, we choose two cutoff distances — $d_{max} = 10$ which consists of 25 $\mathbf{e}$'s and $d_{max} = 11$ for which there are 62 terms in the summations of (5.17) and (5.18). For each $\mathbf{e}$, $P(\mathbf{e}|\mathbf{0})$ is estimated by (5.16) which is reduced to

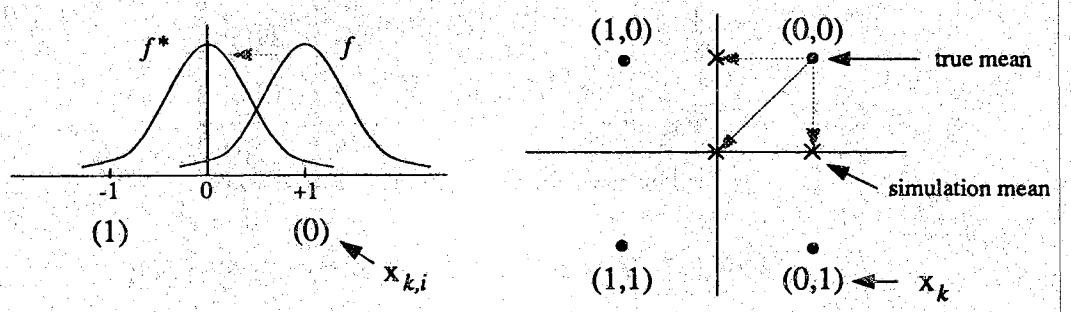
$$\hat{P}^*(\mathbf{e}|\mathbf{0}) = \frac{1}{L_Y} \sum_{\ell=1}^{L_Y} 1_{\{\mathbf{e}\}} \left(\mathbf{0}, \mathbf{Y}^{(\ell)}\right) w_{\mathbf{Y}|\mathbf{e},\mathbf{0}} \left(\mathbf{Y}^{(\ell)}|\mathbf{e}, \mathbf{0}\right), \tag{5.19}$$

where $1_{\{\mathbf{e}\}}(\mathbf{0}, \mathbf{y})$ is the indicator function which is unity if $\mathbf{e}$ is decoded when $\mathbf{0}$ is transmitted and $\mathbf{y}$ is the noise vector, and 0 otherwise. The evaluation of $1_{\{\mathbf{e}\}}(\mathbf{0}, \mathbf{y})$ in the ℓth run is nothing but one execution of the decoding process and checking if $\mathbf{e}$ is decoded. The variance of (5.19) is estimated by a sample variance estimator of the form (2.11), $\hat{S}^2(\mathbf{e})/L_Y$, where $\hat{S}^2(\mathbf{e})$ is

$$\hat{S}^2(\mathbf{e}) = \frac{1}{L_Y} \sum_{\ell=1}^{L_Y} 1_{\{\mathbf{e}\}} \left(\mathbf{0}, \mathbf{Y}^{(\ell)}\right) w^2_{\mathbf{Y}|\mathbf{e},\mathbf{0}} \left(\mathbf{Y}^{(\ell)}|\mathbf{e}, \mathbf{0}\right) - \hat{P}^*(\mathbf{e}|\mathbf{0})^2. \tag{5.20}$$

Finally, the IS estimator for P_b is

$$\hat{P}_b^* = \frac{1}{b} \sum_{\mathbf{e}:d(\mathbf{0},\mathbf{e})=d_{min}}^{d_{max}} n_b(\mathbf{e}) \hat{P}^*(\mathbf{e}|\mathbf{0}), \tag{5.21}$$



(a) BPSK. (b) QPSK.

Figure 5.5 BPSK and QPSK signal mappings and mean translations.

and the variance of $\hat{P}_b$ is estimated as

$$\text{Est. var}^*[\hat{P}_b] = \frac{1}{b^2} \sum_{\mathbf{e}:d(\mathbf{0},\mathbf{e})=d_{min}}^{d_{max}} n_b^2(\mathbf{e}) \hat{S}^2(\mathbf{e}) / L_Y. \tag{5.22}$$

The remaining problem is to design the noise biasing $f^*_{\mathbf{Y}|\mathbf{e},\mathbf{x}}(\mathbf{y}|\mathbf{e},\mathbf{x})$, given $\mathbf{x} = \mathbf{0}$ and $\mathbf{e}$. Following the formulation in Chapter 4, we need to first find the decision statistic D in terms of the system response function $\xi(\cdot)$. Recall that the code rate is $1/2$, i.e., $x_k = (x_{k,1}, x_{k,2})$. If x_k is transmitted by a QPSK signal pulse, the signal pulse amplitude $s(x_k)$ selected by x_k is a complex number, or equivalently a 2-dimensional vector. If the modulation is BPSK, x_k is transmitted by two BPSK signal pulses and hence $s(x_k)$ is still complex with the real part $s(x_k)^I = s(x_{k,1})$ and the imaginary part $s(x_k)^Q = s(x_{k,2})$. Both cases can be reduced to an equivalent discrete coding channel with $s(x_k) = (\pm 1, \pm 1)$. Figure 5.5 shows the BPSK and QPSK signal mappings in the signal space. Recall from Chapter 3 that the ML metric function for comparing a demodulator output symbol r_k and a signal pulse amplitude $s(x_k)$ is $Re[s(x_k)^* r_k] = s(x_k)^I r_k^I + s(x_k)^Q r_k^Q$. Thus, given that $\mathbf{x}$ and $\mathbf{x}+\mathbf{e}$ are the only two decoding options, the Viterbi decoder will favor the first event error path $\mathbf{x}+\mathbf{e}$ over the correct path $\mathbf{x}$ if and only if the decision statistic is

$$D = \sum_{k=1}^{\ell(\mathbf{e})} \left(s(x_k + e_k)^I - s(x_k)^I\right) r_k^I + \left(s(x_k + e_k)^Q - s(x_k)^Q\right) r_k^Q \geq 0$$

where for the AWGN channel we have $r_k = s(x_k) + y_k$. To simplify the following discussion, we represent a $\ell(\mathbf{e})$-dimensional complex vector by a $2\ell(\mathbf{e})$-dimensional real vector and define

$$\begin{aligned}
\mathbf{s} &= \left[s(x_1)^I, s(x_1)^Q, \ldots, s(x_{\ell(\mathbf{e})})^I, s(x_{\ell(\mathbf{e})})^Q\right]^t \\
\hat{\mathbf{s}} &= \left[s(x_1 + e_1)^I, s(x_1 + e_1)^Q, \ldots, s(x_{\ell(\mathbf{e})} + e_{\ell(\mathbf{e})})^I, s(x_{\ell(\mathbf{e})} + e_{\ell(\mathbf{e})})^Q\right]^t \\
\mathbf{y} &= \left[y_1^I, y_1^Q, \ldots, y_{\ell(\mathbf{e})}^I, y_{\ell(\mathbf{e})}^Q\right]^t.
\end{aligned}$$

Note that with the BPSK or the QPSK signal mapping in Figure 5.5, we have $\mathbf{s} = [1, 1, \ldots, 1]^t$ for $\mathbf{x} = \mathbf{0}$. The decision statistic D can then be expressed as

$$\begin{aligned}
D &= \xi(\mathbf{x}, \mathbf{y}) = (\hat{\mathbf{s}} - \mathbf{s})^t \, \mathbf{s} + (\hat{\mathbf{s}} - \mathbf{s})^t \, \mathbf{y} \\
&= a(\mathbf{x}) + \mathbf{b}(\mathbf{x})^t \mathbf{y} \qquad (5.23)
\end{aligned}$$

which is an affine function of $\mathbf{y}$, and hence this is a linear system by the definition in Section 4.1. Note that both $\mathbf{s}$ and $\hat{\mathbf{s}}$ are treated as constant vectors at this level of the simulation.

Therefore, recall from Example 4.1, the dominating point $\mathbf{v} = [v_i]_{i=1}^{2\ell(\mathbf{e})}$ is

$$\begin{aligned}
\mathbf{v} &= -\frac{a(\mathbf{x})}{\|\mathbf{b}(\mathbf{x})\|^2} \, \mathbf{b}(\mathbf{x}) \\
&= -\frac{(\hat{\mathbf{s}} - \mathbf{s})^t \, \mathbf{s}}{\|\hat{\mathbf{s}} - \mathbf{s}\|^2} \, (\hat{\mathbf{s}} - \mathbf{s}) \qquad (5.24)
\end{aligned}$$

which is also the simulation mean vector. Note that the multiplicative factor in (5.24), $-(\hat{\mathbf{s}} - \mathbf{s})^t \mathbf{s} / \|\hat{\mathbf{s}} - \mathbf{s}\|^2$, is a real number. Furthermore, if $\|\mathbf{s}\|^2 = \|\hat{\mathbf{s}}\|^2$, that is, a constant envelope modulation scheme is used (e.g, an M-ary PSK or an M-ary FSK), we get

$$\frac{(\hat{\mathbf{s}} - \mathbf{s})^t \mathbf{s}}{\|\hat{\mathbf{s}} - \mathbf{s}\|^2} = \frac{(\hat{\mathbf{s}} - \mathbf{s})^t \mathbf{s}}{2\|\mathbf{s}\|^2 - 2\hat{\mathbf{s}}^t \mathbf{s}} = -\frac{1}{2},$$

i.e., the multiplicative factor is always 1/2 in this case. Let us now examine each component of $\mathbf{v}$. Denote s_i and $\hat{s}_i$ as ith components, $i = 1, \ldots, 2\ell(\mathbf{e})$, of $\mathbf{s}$ and

$\hat{\mathbf{s}}$ respectively. If $e_k = 0$, and consequently $s(x_k)^I = s_{2k-1} = \hat{s}_{2k-1}$ and $s(x_k)^Q = s_{2k} = \hat{s}_{2k}$, we have $v_{2k-1} = v_{2k} = 0$. That is, the means of the Gaussian random variables y_k^I and y_k^Q are not shifted if $e_k = 0$. On the other hand, if $e_k \neq 0$, we have $v_{2k-1} = 0.5 * \left(s(x_k + e_k)^I - s(x_k)^I\right)$ and $v_{2k} = 0.5 * \left(s(x_k + e_k)^Q - s(x_k)^Q\right)$. Hence, the mean of y_k is one-half the difference between $s(x_k + e_k)$ and $s(x_k)$ in each dimension. Note that with this biasing the mean of the transmitted signal plus the zero-mean random Gaussian noise, $\mathbf{s}+\mathbf{y}$, has been translated from $\mathbf{s}$ to $\mathbf{s}+\mathbf{v} = \frac{1}{2}(\mathbf{s}+\hat{\mathbf{s}})$, which is the midpoint between the two signal points representing $\mathbf{s}$ and $\hat{\mathbf{s}}$. This is also true in each individual dimension. Figure 5.5 shows this asymptotically optimal Gaussian mean translation for BPSK and QPSK. Finally, we remark that the above derivation and conclusion are not restricted to $\mathbf{x} = \mathbf{0}$ or $\leq$ 2-dimensional modulations.

The non-uniform, signal-dependent mean-translation biasing is therefore

$$f^*_{\mathbf{Y}|\mathbf{e},\mathbf{x}}(\mathbf{y}|\mathbf{e},\mathbf{x}) = \prod_{k=1}^{T_M} f^*_{Y_k^I|e_k,x_k}(y_k^I|e_k,x_k) f^*_{Y_k^Q|e_k,x_k}(y_k^Q|e_k,x_k)$$

where T_M is the random simulation length (the time of detection of a complete first event error), and

$$f^*_{Y_k^I|e_k,x_k}(y_k^I|e_k,x_k) = \begin{cases} N\left(0.5 * \left(s(x_k + e_k)^I - s(x_k)^I\right), \sigma^2\right), & e_k \neq 0; \\ N(0,\sigma^2), & e_k = 0, \end{cases}$$

and similarly

$$f^*_{Y_k^Q|e_k,x_k}(y_k^Q|e_k,x_k) = \begin{cases} N\left(0.5 * \left(s(x_k + e_k)^Q - s(x_k)^Q\right), \sigma^2\right), & e_k \neq 0; \\ N(0,\sigma^2), & e_k = 0. \end{cases}$$

For example, if $x_k = (0,0)$ and $e_k = (0,1)$, then the probability distribution of the real part (I-channel) of the Gaussian noise is $N(0,\sigma^2)$, and $N(-1,\sigma^2)$ for the imaginary part (Q-channel). As mentioned in the last paragraph, this biasing has the effect of moving the mean of the transmitted signal from $s(x_k) = (1,1)$ to the midpoint between $s(x_k)$ and $s(x_k + e_k) = (1,-1)$ as shown in Figure 5.5(b). The importance sampling weighting function $w^*_{\mathbf{Y}|\mathbf{e},\mathbf{x}}(\mathbf{y}|\mathbf{e},\mathbf{x})$ is simply $f_{\mathbf{Y}}(\mathbf{y})/f^*_{\mathbf{Y}|\mathbf{e},\mathbf{x}}(\mathbf{y}|\mathbf{e},\mathbf{x})$, and since $f_{\mathbf{Y}}(\mathbf{y}) = \prod_{k=1}^{T_M} f_{Y_k^I}(y_k^I) f_{Y_k^Q}(y_k^Q)$ is a product form density, the weighting function is also in product form.

The above mean-translation biasing design can be also derived from a simple union bound argument given in Sadowsky [46]. Recall that the goal of IS is to "trick" the decoder into decoding the specific error event $\mathbf{x}+\mathbf{e}$. Given that $\mathbf{x}$ and $\mathbf{x}+\mathbf{e}$ are the only two decoding options, we don't have to bias transmitted signal samples if they are the same as those of the error event path. For those dimensions in which they differ, the variance of a Bernoulli trial is minimized when the success rate p is $1/2$. This explains why we translate the mean of the Gaussian random variable to the midpoint. Note that we use this union (upper) bound argument or the binary decision statistic D of (5.23) only to design the noise biasing. In the simulation, the actual Viterbi algorithm is performed which admits all possible decoding options. Some nontrivial error events will be decoded which are not the desired $\mathbf{x}+\mathbf{e}$. Therefore the biasing we have designed can be considered suboptimal in the sense that the overall "success rate" (of decoding $\mathbf{x}+\mathbf{e}$) will be less than $1/2$. In any way, with our event simulation method and the optimal mean-translation biasing, we are not estimating an upper bound of $P(\mathbf{x}+\mathbf{e}|\mathbf{x})$ but just use it to derive the biasing.

See the trellis diagram of Figure 5.4(b). Notice that the two admissible transition branches leaving a node always differ in two bits, i.e., it is either the pair (0,1) and (1,0) or the pair (0,0) and (1,1). Suppose that $e_k = (0,1)$. Then, see Figure 5.5(b), with the mean being shifted to (1,0) — the midpoint between $s(x_k) = (1,1)$ and $s(x_k + e_k) = (1,-1)$, the chance of that branch being decoded as $e_k = (1,0)$ is much smaller then $e_k = (0,1)$. (This is particularly true for a high SNR and/or a long error sequence.) Therefore, although the Viterbi decoder has many decoding options, it can be predicted that approximately 50% of the decoded error events will be trivial. And with high probability the remaining 50% will be the attempted $\mathbf{e}$. In simulations, it is verified that less than 5% of the total runs result in nontrivial error events other than $\mathbf{e}$ for SNR $\geq$ 5 dB.

The simulation algorithm can be described as follows.

The IS Simulation Algorithm for a Regular Convolutional Code:

- Input: Convolutional encoder configuration, SNR, number of simulated error sequences and L_Y.
- Output: $\hat{P}^*(\mathbf{e}|\mathbf{0})$ for all simulated $\mathbf{e}$, percentages of desired error events decoded, $\hat{P}_b^*$ and relative precisions for estimators.
- Method:

  1. Construct tables containing trellis diagram information.
  2. Read error sequence $\mathbf{e}$, length $\ell(\mathbf{e})$, and weight $n_b(\mathbf{e})$ from the data file obtained from the RC Algorithm. If all $\mathbf{e}$'s have been simulated, go to 7.
  3. Put $\ell = 0$.
  4. $\ell = \ell + 1$. If $\ell > L_Y$, go to 2.
  5. Compute received data. Initialize the Viterbi decoder.
  6. Compute received data. Perform the Viterbi decoder error event simulation. When an error event is decoded, update simulation data accumulators then go to 4.
  7. Output simulation results.

A sample simulation output of the above algorithm is given in Appendix A Table 7. Simulated P_b for various SNR's and their union bounds are shown in Figure 5.6. We can see that our estimates agree closely to union bounds at moderate-to-high SNR's. The truncation bias is the main reason for the under-estimation at small SNR's. (Recall from Section 3.2.2 that the union bound is not tight loose in this region either.) For comparison, also shown in Figure 5.6 are simulation results for the case in which we employ a hard, rather than soft, decision Viterbi decoder (this in effect reduces the channel model to the binary symmetric channel or BSC). The result confirms that there is about a 2 dB loss due to hard quantization. The number of simulation runs for every error sequence $\mathbf{e}$ is $L_Y = 1000$. For each SNR, the simulation is terminated if the relative precision ε for the estimate of P_b is less than 10%.

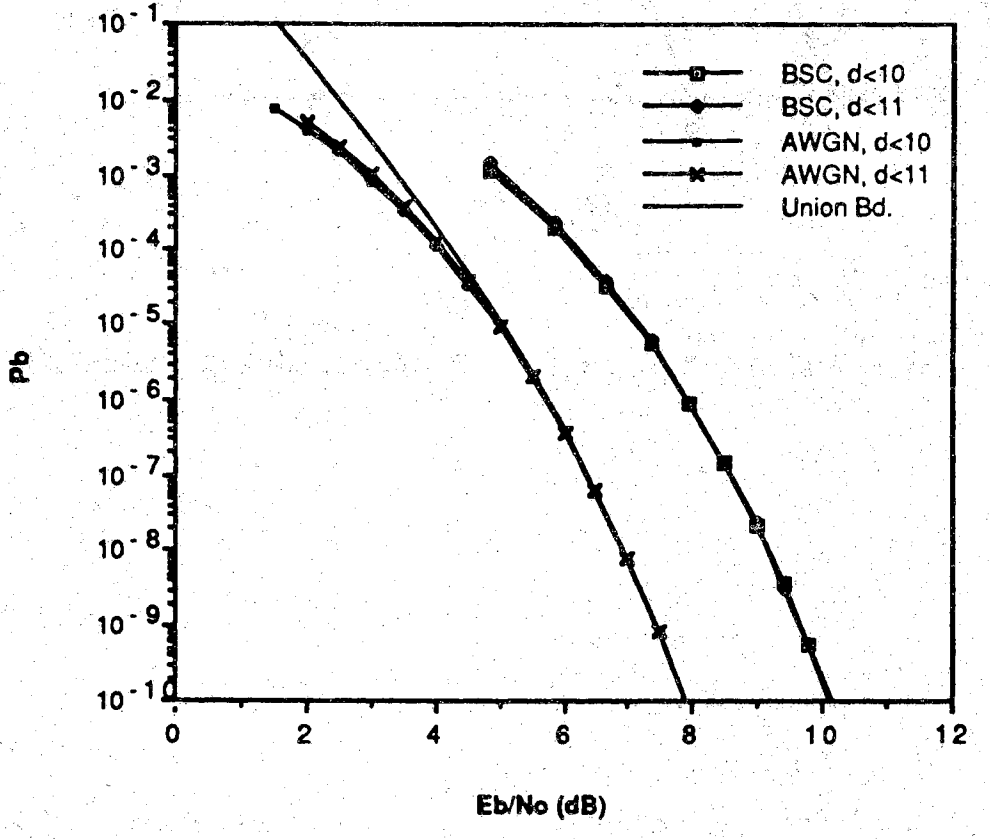


Figure 5.6 P_b of the 16-state, $R = 1/2$, $d_{min} = 7$ convolutional code.

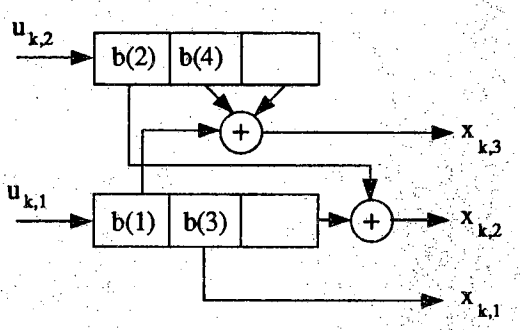
5.3.2 A Quasi-Regular TCM Code

Next we consider a 16-state, $R = 2/3$, 8-PSK, $d_{free} = 2.27$ TCM code given in Ungerboeck's original paper [57]. The encoder configuration and trellis diagram with a d_{free} path (correct path $\mathbf{x} = \mathbf{0}$) are shown in Figure 5.7. The signal mapping function (modulation) is as described in Section 3.2.1 and Figure 3.5. Shift register elements $b(1)$ and $b(2)$ contain the most recent incoming information bits and are also the two least significant bits of the new encoder state.

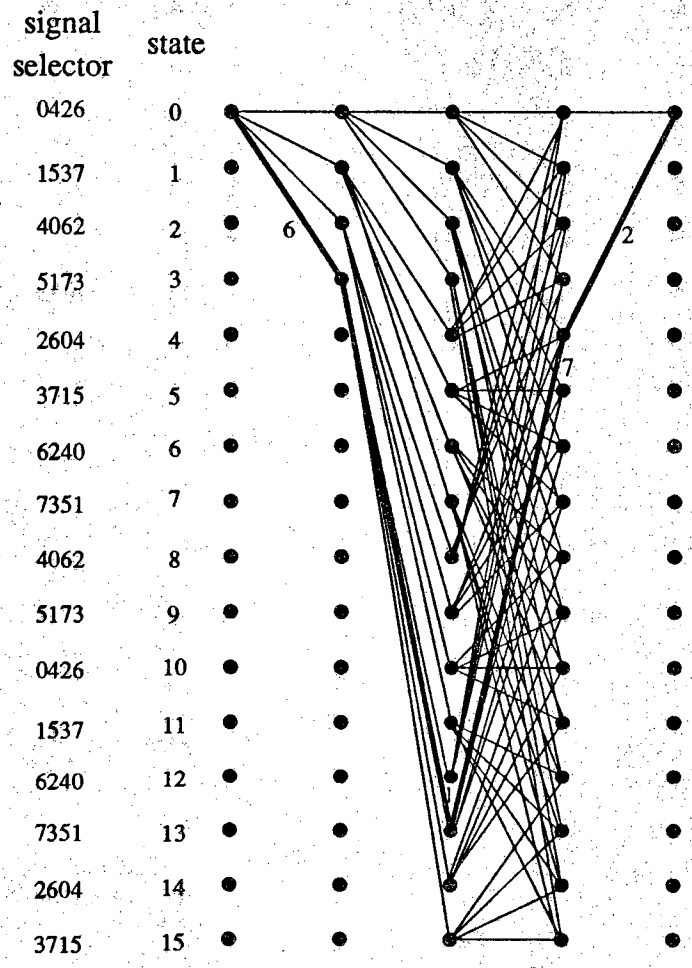
The fact that this is a non-regular code makes the union bound computation and the IS simulation more difficult than the convolutional code example discussed in the last subsection. The major additional complexity arises from that $d_{HD}(\mathbf{0}, \mathbf{e}) = d_{HD}(\mathbf{x}, \mathbf{x} + \mathbf{e})$, for all signal selector sequences $\mathbf{x}$, but $d_{ED}(\mathbf{0}, \mathbf{e}) \neq d_{ED}(\mathbf{x}, \mathbf{x} + \mathbf{e})$, where as defined in Chapter 3 $d_{HD}(\cdot, \cdot)$ and $d_{ED}(\cdot, \cdot)$ are the Hamming and Euclidean distance measures repectively. Consequently, we may have $P(\mathbf{x} + \mathbf{e}|\mathbf{x}) \neq P(\hat{\mathbf{x}} + \mathbf{e}|\hat{\mathbf{x}}), \mathbf{x} \neq \hat{\mathbf{x}}$. Therefore, we can no longer assume that $\mathbf{x} = \mathbf{0}$ is the transmitted codeword in computing P_b.

A. Union Bounds:

Fortunately, for many practical TCM codes that can be classified as quasi-regular, the average multiplicities A_d and B_d in (3.3) and (3.5) are computable by assuming the all 0 sequence is transmitted. We have demonstrated this using the RC Algorithm and the distance polynomial information in Section 3.2.2. The distance polynomials in (3.7) apply directly to this code. Table 5.2 shows the distance spectrum of this code for squared distance up to 10. The modified RC Algorithm takes 3.4 hours CPU time on a Gould PN9080 to obtain the list of error sequences $\mathbf{e}$ used in computing this table. Table 7 in Appendix B shows the beginning part of the modified RC Algorithm output for squared distance ≤ 5.7574 (18 error sequences). There are a total of 22 possible distances and 49,506 signal selector error sequences $\mathbf{e}$ within this range. Recall from Table 5.1 that, for the regular convolutional code, there are only



(a) Encoder configuration.



(b) Trellis diagram.

Figure 5.7 A 16-state, $R = 2/3$ and $d_{free} = 2.27$ TCM code.

Table 5.2 A TCM code distance spectrum.

d^2	# of e	A_d	B_d	d^2	# of e	A_d	B_d
5.1716	6	2.50	10.00	8.3431	248	50.31	446.56
5.7574	12	3.75	22.50	8.5858	25	17.75	94.50
6.3431	42	8.13	64.88	8.6863	3282	128.26	2052.34
6.5858	4	3.50	9.00	8.9289	948	127.72	1384.69
6.9289	120	16.19	161.88	9.1716	93	55.00	381.25
7.1716	12	5.00	26.00	9.2721	9840	255.95	4607.20
7.4142	2	2.00	4.00	9.4142	0	0.50	2.00
7.5147	366	31.78	381.41	9.5147	3364	303.58	3900.90
7.7574	68	21.13	143.00	9.7574	437	146.81	1317.63
8.0000	1	4.00	14.00	9.8579	29526	511.77	10234.89
8.1005	1092	63.92	894.75	10.000	18	19.50	85.00

4 possible distances and 25 $\mathbf{e}$'s for distance d^2 within the range of $d^2_{min} \leq d^2 \leq 2d^2_{min}$. In general, TCM codes have very dense distance spectra. Again, we will compare our simulation results against union bounds computed from (3.3) and (3.5) using the information in Table 5.2.

B. IS Simulation:

The importance sampling simulation algorithm we propose for this code is a refined version of that for the convolutional code. We will estimate the expected specific error probability $P(\mathbf{e}) \triangleq \mathrm{E}[P(\mathbf{X}+\mathbf{e}|\mathbf{X})]$ in (5.11) and (5.12) for only a few number of "important" error sequences $\mathbf{e}$, assuming their existence and the resulting truncation bias is insignificant at a moderate-to-high SNR. We will see from the following discussion and verify through simulation results that this assumption generally holds in the case of the AWGN channel. When there are too many $\mathbf{e}$'s to be considered, the above technique becomes very inefficient. This is the case when we consider the satellite channel transmission in the next chapter. There, a similar but more sophisticated algorithm and its rigorous reasoning will be presented. In this section, our estimator for P_b is thus still a specialized form of (5.15) in the sense that we don't jointly sample $\mathbf{x}$ and $\mathbf{e}$ in each simulation run.

The major difference between the simulation of a TCM code and a convolutional code is that we cannot assume the all 0 sequence is the transmitted signal selector sequence. Also, there will be more $\mathbf{e}$'s needed to be included in the summations of (5.11) and (5.12) because of the dense distance spectrum. More specifically, since $\mathrm{E}[P(\mathbf{X}+\mathbf{e}|\mathbf{X})] = \sum_{\mathbf{x}} P(\mathbf{x}+\mathbf{e}|\mathbf{x})P_{\mathbf{X}}(\mathbf{x})$, we will have to estimate $P(\mathbf{x}+\mathbf{e}|\mathbf{x})$ and average over all $\mathbf{x}$. To do this efficiently, note that $P(\mathbf{e}) = \mathrm{E}[\mathbf{X},\mathbf{Y}]$ and hence we can employ the conditional importance sampling technique of Section 4.3 with a signal biasing $P^*_{\mathbf{X}}(\mathbf{x})$. Following the discussion in Section 4.3, the conditional IS estimator for

$$P(\mathbf{e}) = \sum_{\mathbf{x}} P(\mathbf{x}+\mathbf{e}|\mathbf{x}) \frac{P_{\mathbf{X}}(\mathbf{x})}{P^*_{\mathbf{X}}(\mathbf{x})} P^*_{\mathbf{X}}(\mathbf{x})$$

thus becomes

$$\hat{P}^*(\mathbf{e}) = \frac{1}{L_X} \sum_{\ell=1}^{L_X} 1_{\{\mathbf{e}\}} \left(\mathbf{X}^{(\ell)}, \mathbf{Y}^{(\ell)}\right) w_{\mathbf{Y}|\mathbf{X}} \left(\mathbf{Y}^{(\ell)}|\mathbf{X}^{(\ell)}\right) w_{\mathbf{X}}\left(\mathbf{X}^{(\ell)}\right), \tag{5.25}$$

where we have let $L_Y = 1$ because per sample costs for the same length $\mathbf{X}$ and $\mathbf{Y}$ are approximately equal, and $w_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = f_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})/f^*_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$ and $w_{\mathbf{X}}(\mathbf{x}) = P_{\mathbf{X}}(\mathbf{x})/P^*_{\mathbf{X}}(\mathbf{x})$ are IS weighting functions for the noise and signal respectively. The variance of (5.25) can be estimated by $\hat{S}^2(\mathbf{e})/L_X$ where $\hat{S}^2(\mathbf{e})$ is

$$\hat{S}^2(\mathbf{e}) = \frac{1}{L_X} \sum_{\ell=1}^{L_X} 1_{\{\mathbf{e}\}} \left(\mathbf{X}^{(\ell)}, \mathbf{Y}^{(\ell)}\right) w^2_{\mathbf{Y}|\mathbf{X}} \left(\mathbf{Y}^{(\ell)}|\mathbf{X}^{(\ell)}\right) w^2_{\mathbf{X}}\left(\mathbf{X}^{(\ell)}\right) - \hat{P}^*(\mathbf{e})^2. \tag{5.26}$$

Finally, the estimator for P_b is

$$\hat{P}^*_b = \frac{1}{b} \sum_{\mathbf{e}:d_w(\mathbf{e})=d_{free}}^{d_{max}} n_b(\mathbf{e})\hat{P}^*(\mathbf{e}), \tag{5.27}$$

and the variance of $\hat{P}^*_b$ is estimated as

$$\text{Est. var}^*[\hat{P}^*_b] = \frac{1}{b^2} \sum_{\mathbf{e}:d_w(\mathbf{e})=d_{free}}^{d_{max}} n^2_b(\mathbf{e})\hat{S}^2(\mathbf{e})/L_X. \tag{5.28}$$

Another issue is how do we identify those $\mathbf{e}$'s with significant values of $P(\mathbf{e})$ and estimate for as many of them as possible. Equation (5.27) indicates that we sum over those $\mathbf{e}$ with worst-case distances $\leq d_{max}$ and we also propose to let the number of simulation runs for each $\mathbf{e}$, L_X, be a function of $\mathbf{e}$ which decreases as $P(\mathbf{e})$ decrease. We will explain these later. First though, let us consider the design of the signal biasing $P^*_{\mathbf{X}}(\mathbf{x})$.

1. The signal biasing $P^*_{\mathbf{X}}(\mathbf{x})$:

In the following discussion $\mathbf{e}$ is a fixed signal selector error sequence. Consider the expected specific error probability $P(\mathbf{e}) = \mathrm{E}[P(\mathbf{X} + \mathbf{e}|\mathbf{X})]$ where the expectation is with respect to the random signal selector sequence $\mathbf{X}$. Recall that $n_{\mathcal{E}}(\mathbf{e})$ is the number of appearances of $e_k, k = 1, \ldots, \ell(\mathbf{e})$, such that $e_k \in \mathcal{E} = \{011, 111\}$. (More generally, for a quasi-regular code $\mathcal{E}$ is the set of signal selector errors whose distance

polynomials are not monomials.) It is easy to show that there are $n_{\mathcal{E}}(\mathbf{e})+1$ possible distances associated with $\mathbf{e}$. The smallest or the worst-case distance is $d_w(\mathbf{e}) = \min_{\mathbf{x}} d(\mathbf{x}, \mathbf{x}+\mathbf{e})$. If we denote the worst-case distance as d_0 and the next smallest distance d_1, and so on. We find that

$$d_i^2 = d_w^2(\mathbf{e}) + i(\delta_2^2 - \delta_0^2), \tag{5.29}$$

for $i = 0, \ldots, n_{\mathcal{E}}(\mathbf{e})$, where $\delta_2^2 = 3.4142$ and $\delta_0^2 = 0.5858$ for a normalized signal constellation of Figure 3.5. Then we can express $P(\mathbf{e})$ as

$$\begin{aligned} P(\mathbf{e}) &= \sum_{\mathbf{x}} P(\mathbf{x}+\mathbf{e}|\mathbf{x}) P_{\mathbf{X}}(\mathbf{x}) \\ &= \sum_{d=d_0}^{d_{n_{\mathcal{E}}(\mathbf{e})}} \sum_{\mathbf{x}: d(\mathbf{x}+\mathbf{e})=d} P(\mathbf{x}+\mathbf{e}|\mathbf{x}) P_{\mathbf{X}}(\mathbf{x}) \\ &\leq 2^{-n_{\mathcal{E}}(\mathbf{e})} \left\{ \begin{pmatrix} n_{\mathcal{E}}(\mathbf{e}) \\ n_{\mathcal{E}}(\mathbf{e}) \end{pmatrix} Q(\frac{d_w(\mathbf{e})}{2\sigma}) + \begin{pmatrix} n_{\mathcal{E}}(\mathbf{e}) \\ n_{\mathcal{E}}(\mathbf{e})-1 \end{pmatrix} Q(\frac{d_1}{2\sigma}) + \cdots \right. \\ &\qquad \left. + \begin{pmatrix} n_{\mathcal{E}}(\mathbf{e}) \\ 0 \end{pmatrix} Q(\frac{d_{n_{\mathcal{E}}(\mathbf{e})}}{2\sigma}) \right\} \end{aligned} \tag{5.30}$$

where $Q(d_i/2\sigma)$ is the union bound for $P(\mathbf{x}+\mathbf{e}|\mathbf{x})$ on the AWGN channel given $d(\mathbf{x}, \mathbf{x}+\mathbf{e}) = d_i$. For example, suppose that $n_{\mathcal{E}}(\mathbf{e}) = 2$, then

$$P(\mathbf{e}) = \frac{1}{4}Q(\frac{d_0}{2\sigma}) + \frac{1}{2}Q(\frac{d_1}{2\sigma}) + \frac{1}{4}Q(\frac{d_2}{2\sigma}). \tag{5.31}$$

For all practical operating SNR's, we find that the first term of the summation in (5.30) or (5.31) is more significant than all other terms in spite of its small proportionality $2^{-n_{\mathcal{E}}(\mathbf{e})}$. That is, for $i \geq 1$, we have

$$\frac{Q(d_0/2\sigma)}{Q(d_i/2\sigma)} \approx \frac{d_i}{d_0} \exp(\frac{1}{8\sigma^2} 2.8284 i) \gg 1. \tag{5.32}$$

However, if the distribution of the random signal selector sequence $\mathbf{X}$ is not biased, only a small portion, $2^{-n_{\mathcal{E}}(\mathbf{e})}$, of the sampled $\mathbf{X}$ will result in this worst-case distance error event. As $n_{\mathcal{E}}(\mathbf{e})$ increases, this percentage decreases exponentially. Therefore,

in order to focus our computational resource on the first few (important) terms of (5.30), in particular, the worst-case distance term, we want to bias $P_{\mathbf{X}}(\mathbf{x})$ in such a way that signal selector sequences for which the distance $d(\mathbf{x}, \mathbf{x}+\mathbf{e})$ is small are sampled more often and later properly weight the result.

The true information process is a random walk through the trellis diagram. Leaving any encoder state, the probability of entering either one of the 2^b possible successor states, p, is the same, i.e., $p = 2^{-b}$. Hence, $P_{\mathbf{X}}(\mathbf{x}) = 2^{-b\ell(\mathbf{e})}$ is a uniformly distribution. In our 8-PSK example, if $e_k \in \mathcal{E}$, the information process will walk into a successor state which produces a code symbol x_k such that $d(x_k, x_k + e_k) = \delta_0$ or δ_2 with equal probability 1/2 (2^{b-1} transitions for each case, this information is provided by distance polynomials). We wish to "trick" the information process into generating more of those important x_k's which result in δ_0 instead of δ_2. Our strategy is to change the random walk parameter p. That is, with probability $p^* > p$, the information process will generate a code symbol x_k such that $d(x_k, x_k + e_k) = \delta_0$ and with probability $q^* = 1 - p^*$ choosing the greater (less favorable) distance δ_2.

To find an appropriate p^*, recall that $P(\mathbf{e}) = \sum_{\mathbf{x}} P(\mathbf{x}+\mathbf{e}|\mathbf{x})P_{\mathbf{X}}(\mathbf{x})$. Thus, we would like to have the probability of choosing $\mathbf{x}$ proportional to its "importance." That is, $P^*_{\mathbf{X}}(\mathbf{x}) \propto P(\mathbf{x}+\mathbf{e}|\mathbf{e})2^{-b\ell(\mathbf{e})} \propto P(\mathbf{x}+\mathbf{e}|\mathbf{e})$, because $P_{\mathbf{X}}(\mathbf{x}) = 2^{-b\ell(\mathbf{e})}$ is a constant for a fixed $\mathbf{e}$. Now consider two signal selector sequences $\mathbf{x}$ and $\hat{\mathbf{x}}$ such that $d(\mathbf{x}, \mathbf{x}+\mathbf{e}) = d_w(\mathbf{e}) = d_0$ and $d(\hat{\mathbf{x}}, \hat{\mathbf{x}}+\mathbf{e}) = d_1$. That is, the random walk of $\mathbf{x}$ and that of $\hat{\mathbf{x}}$ differ only in one choice in which $\mathbf{x}$ produces δ_0 while $\hat{\mathbf{x}}$ results in δ_2. Therefore,

$$\begin{aligned} \frac{P^*_{\mathbf{X}}(\mathbf{x})}{P^*_{\mathbf{X}}(\hat{\mathbf{x}})} &= \frac{(p^*)^{\ell(\mathbf{e})}}{(p^*)^{\ell(\mathbf{e})-1}q^*} = \frac{p^*}{q^*} \\ &\propto \frac{Q(d_0/2\sigma)}{Q(d_1/2\sigma)} \approx \frac{\exp(-d_w^2(\mathbf{e})/8\sigma^2)}{\exp(-d_1^2/8\sigma^2)} \end{aligned} \tag{5.33}$$

where we have used the approximation $Q(\alpha) \approx \sqrt{2\pi}\exp(-\alpha^2/2)$ for the Gaussian Q function. Recall from (5.29) that $d_1^2 = d_w^2(\mathbf{e}) + (\delta_2^2 - \delta_0^2)$. Therefore we get

$$\frac{p^*}{q^*} = \frac{\exp(-\delta_0^2/8\sigma^2)}{\exp(-\delta_2^2/8\sigma^2)} = \frac{c_0}{c_2} > 1$$

$$\Longrightarrow \quad p^* = \frac{1}{2}\frac{c_0}{c_0 + c_2} > \frac{1}{4} = 2^{-b} \tag{5.34}$$

which satisfies the condition $2^{b-1}p^* + 2^{b-1}q^* = 1$. We note that p^* is a function of δ_0, δ_2 and the SNR. Had we used the tighter upper bound $Q(\alpha) = \sqrt{2\pi}\alpha \exp(-\alpha^2/2)$, p^* would have been also dependent on the worst-case distance $d_w(\mathbf{e})$. But for all practical SNR's, we find that the difference is negligible because of the dominance of the exponential factor in (5.34). Therefore in the simulation we use the same random walk parameter p^* for all error sequences $\mathbf{e}$ in consideration.

In summary, the quasi-regularity of the code allows us to do the signal biasing on the per branch basis. The biased probability mass function of the random signal selector sequence $\mathbf{X}$ is hence a product form $P^*_{\mathbf{X}}(\mathbf{x}) = \prod_{k=1}^{\ell(\mathbf{e})} P^*_{X_k}(x_k)$ where $P^*_{X_k}(x_k)$ is

$$P^*_{X_k}(x_k) = \begin{cases} 2^{-b} = P_{X_k}(x_k), & e_k \notin \mathcal{E} \\ p^*, & e_k \in \mathcal{E}, d(x_k, x_k + e_k) = \delta_0 \\ q^*, & e_k \in \mathcal{E}, d(x_k, x_k + e_k) = \delta_2 \end{cases} . \tag{5.35}$$

The IS weight in the ℓth run $w_{\mathbf{X}}(\mathbf{X}^{(\ell)})$ is simply

$$w_{\mathbf{X}}(\mathbf{X}^{(\ell)}) = \frac{P_{\mathbf{X}}(\mathbf{X}^{(\ell)})}{P^*_{\mathbf{X}}(\mathbf{X}^{(\ell)})} = \frac{2^{-bn_{\mathcal{E}}(\mathbf{e})}}{(p^*)^{n_{\mathcal{E}}(\mathbf{e})-i}(q^*)^i}$$

if in i out of $n_{\mathcal{E}}(\mathbf{e})$ decisions an x_k is chosen such that $d(x_k, x_k + e_k) = \delta_2$ (which results in $d(\mathbf{x}, \mathbf{x} + \mathbf{e}) = d_i$). For non-quasi-regular codes whose distance polynomials are also functions of the originating pair of states (see (3.7)), the signal biasing will also have to depend on the encoder state.

2. Let the number of simulation runs for $\mathbf{e}$, L_X, be a function of $d_w(\mathbf{e})$.

It can be seen from (5.30) and (5.32) that the value of $P(\mathbf{e})$ is generally dominated by the worst-case distance term $2^{-bn_{\mathcal{E}}(\mathbf{e})}P(\mathbf{x} + \mathbf{e}|\mathbf{x})$ given $d(\mathbf{x}, \mathbf{x} + \mathbf{e}) = d_w(\mathbf{e})$. Therefore, the "importance" of $\mathbf{e}$ can be roughly measured by its worst-case distance $d_w(\mathbf{e})$ (or more precisely by the factor $2^{-bn_{\mathcal{E}}(\mathbf{e})}P(\mathbf{x} + \mathbf{e}|\mathbf{x})$). It is then just natural that we should devote most of the computation to the estimation of $P(\mathbf{x} + \mathbf{e}|\mathbf{x})$ for those $\mathbf{e}$'s with small worst-case distances so that they have smaller relative precisions.

On the other hand, a smaller value of L_X can be assigned to error sequences $\mathbf{e}$ with relatively large $d_w(\mathbf{e})$.

Let $L_X(\mathbf{e})$ be the number of simulation runs assigned to the error sequence $\mathbf{e}$. In particular, $L_X(\mathbf{e}_f)$ is the preset number of runs for $\mathbf{e}_f$, where $\mathbf{e}_f$ denotes some error sequence $\mathbf{e}$ whose worst-case distance is the free distance d_{free}. Our objective is to design $L_X(\mathbf{e})$ to be decreasing from the peak value $L_X(\mathbf{e}_f)$ with respect to the increased $d_w(\mathbf{e})$. Suppose that for all $\mathbf{e}$ a fixed variance for the estimator $\hat{P}^*(\mathbf{e})$ is desired. For example, we choose the variance of the estimator for $\mathbf{e}_f$, $\text{var}^*[\hat{P}^*(\mathbf{e}_f)]$. (Note that the relative precision will increase with the decreased $\hat{P}^*(\mathbf{e})$, meaning poorer accuracy.) We next approximate $\text{var}^*[\hat{P}^*(\mathbf{e})]$ by

$$\text{var}^*[\hat{P}^*(\mathbf{e})] \leq \frac{1}{L_X(\mathbf{e})}\left(P(\mathbf{e}) - P^2(\mathbf{e})\right) \approx \frac{1}{L_X(\mathbf{e})}P(\mathbf{e}), \tag{5.36}$$

where the upper bound, recall from (2.3), is the variance for the Monte Carlo estimator, and the last approximation holds if $P(\mathbf{e}) \gg P^2(\mathbf{e})$, which is generally the case. Furthermore, we approximate $P(\mathbf{e})$ by $Q(d_w(\mathbf{e})/2\sigma)$. Then, it is found that

$$L_X(\mathbf{e}) \approx L_X(\mathbf{e}_f)\frac{d_{free}}{d_w(\mathbf{e})}\exp\left(-\frac{1}{8\sigma^2}(d_w^2(\mathbf{e}) - d_{free}^2)\right) \tag{5.37}$$

which drops exponentially fast relative to the increased worst-case distance $d_w(\mathbf{e})$.

It is noted that (5.36) is the worst-case approximation. With the optimal mean-translation biasing for Gaussian noise samples, the estimator variance actually can be approximated by $(1/L_X(\mathbf{e}))P^2(\mathbf{e})$. (Recall from Chapter 4 that the variance of the optimal estimator decreases as fast as $P^2(\mathbf{e})$.) Therefore, the actual number of simulation runs required for $\mathbf{e}$, $\mathbf{e} \neq \mathbf{e}_f$ is less than that computed by (5.37). We remark that this technique can be also used in the previous convolutional code example, and in fact more accurately. For convolutional codes, $d(\mathbf{x}, \mathbf{x}+\mathbf{e}) = d_w(\mathbf{e})$ for all $\mathbf{x}$, i.e., (5.30) consists of only one term.

Finally, given $\mathbf{x}$ and $\mathbf{e}$, the Gaussian noise biasing is exactly the same as that for the convolutional code except now the signal set is 8-PSK. The dominating point obtained in (5.24) still holds here because we did not require $\mathbf{x}$ to be $\mathbf{0}$ in the derivation. Therefore we have as before

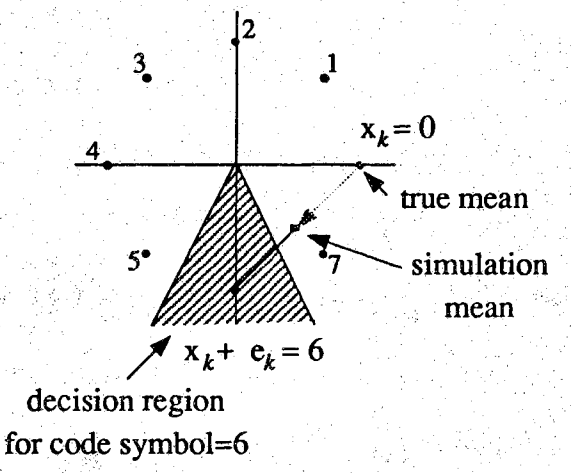


Figure 5.8 An example of the optimal mean translation for 8-PSK.

$$\mathbf{s} + \mathbf{v} = \frac{1}{2}(\mathbf{s} + \hat{\mathbf{s}})$$

where $\mathbf{s}$ and $\hat{\mathbf{s}}$ are samples of channel symbol sequences representing $\mathbf{x}$ and $\mathbf{x} + \mathbf{e}$ respectively. The channel noise simulation density is

$$f^*_{\mathbf{Y}|\mathbf{e},\mathbf{x}}(\mathbf{y}|\mathbf{e},\mathbf{x}) = \prod_{k=1}^{T_M} f^*_{Y_k^I|e_k,x_k}(y_k^I|e_k,x_k) f^*_{Y_k^Q|e_k,x_k}(y_k^Q|e_k,x_k),$$

where T_M is the random simulation length and

$$f^*_{Y_k^I|e_k,x_k}(y_k^I|e_k,x_k) = \begin{cases} N\left(0.5 * \left(s(x_k+e_k)^I - s(x_k)^I\right), \sigma^2\right), & e_k \neq 0; \\ N(0,\sigma^2), & e_k = 0, \end{cases}$$

$$f^*_{Y_k^Q|e_k,x_k}(y_k^Q|e_k,x_k) = \begin{cases} N\left(0.5 * \left(s(x_k+e_k)^Q - s(x_k)^Q\right), \sigma^2\right), & e_k \neq 0; \\ N(0,\sigma^2), & e_k = 0. \end{cases}$$

An example is given in Figure 5.8 where the signal selector is $x_k = 0 = (0,0,0)$, and $e_k = 6 = (1,1,0)$. Again, the simulation mean of y_k is located at the midpoint between the two signal points representing x_k and $x_k + e_k$. Note that the admissible signal selectors leaving any decoder state are either the set $\{0,2,4,6\}$ or the set $\{1,3,5,7\}$. Therefore, with high probability, the Viterbi decoder will choose between x_k or $x_k + e_k$ even the biased mean seems to fall into other decision regions. It can also be predicted that approximately 50% of the decoded error events will be trivial.

The simulation algorithm can be described as follows.

Quasi-Regular TCM Codes IS Simulation Algorithm:

- Input: Convolutional encoder configuration, signal mapping function, SNR, number of simulated error sequences and $L_X(\mathbf{e}_f)$.
- Output: $\hat{P}^*(\mathbf{e})$ for all simulated $\mathbf{e}$, percentages of trivial and desired error events, $\hat{P}_e^*$, $\hat{P}_b^*$, their relative precisions and CPU time.
- Method:
  1. Construct tables containing trellis diagram information.
  2. Compute noise variance, random walk parameter p^*.
  3. Read error sequence $\mathbf{e}$, length $\ell(\mathbf{e})$, weight $n_b(\mathbf{e})$ and worst-case distance $d_w(\mathbf{e})$ from the data file obtained from the RC Algorithm. If all $\mathbf{e}$'s have been simulated, go to 8.
  4. Compute $L_X(\mathbf{e})$. Put $\ell = 0$.
  5. $\ell = \ell + 1$. If $\ell > L_X(\mathbf{e})$, go to 3. Select a random initial state and correct path $\mathbf{x}$. Compute the signal biasing weight $w_{\mathbf{X}}(\mathbf{x})$.
  6. Compute received data. Initialize the Viterbi decoder.
  7. Compute received data. Perform the Viterbi decoder error event simulation. When an error event is decoded, update simulation data accumulators then go to 5.
  8. Output simulation results.

A more detailed simulation program flow chart, especially on the part of the Viterbi Algorithm, is given in Appendix B Figure 7.

A sample simulation output of the above algorithm is given in Appendix B Table 7. Estimates $\hat{P}_e^*$ and $\hat{P}_b^*$ for various SNR's are shown in Figure 5.9 and Figure 5.10

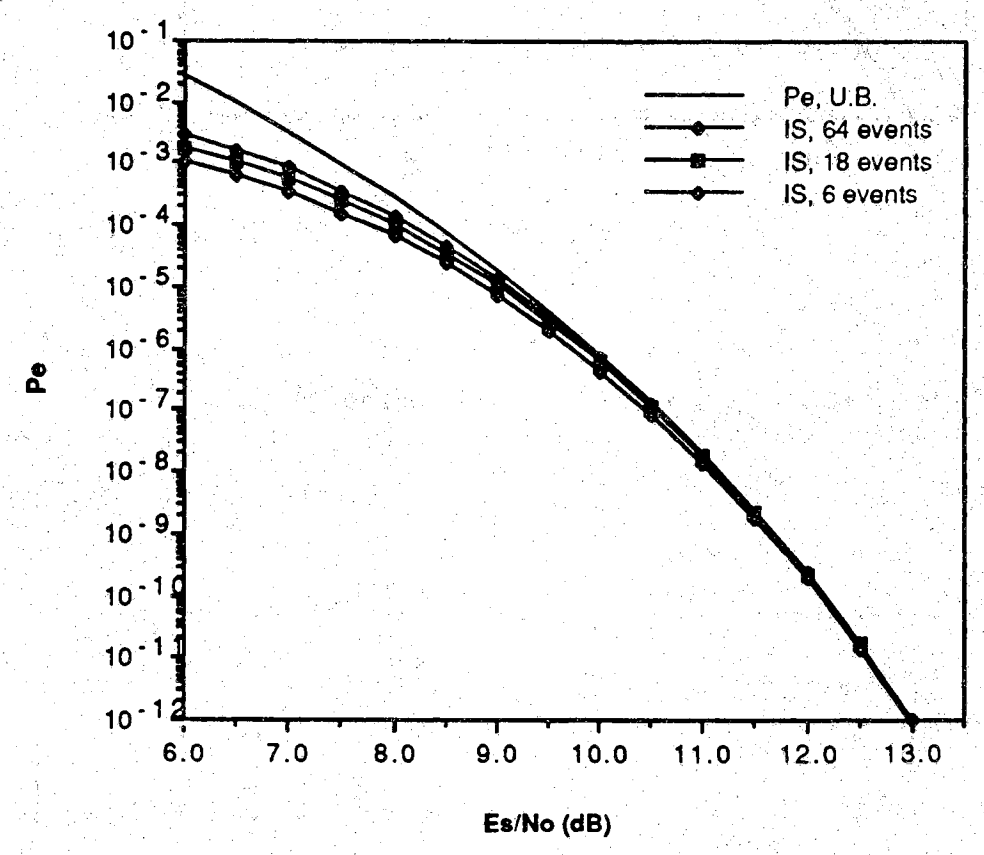


Figure 5.9 P_e for the 16-state, $R = 2/3$, $d_{free} = 2.27$ TCM code.

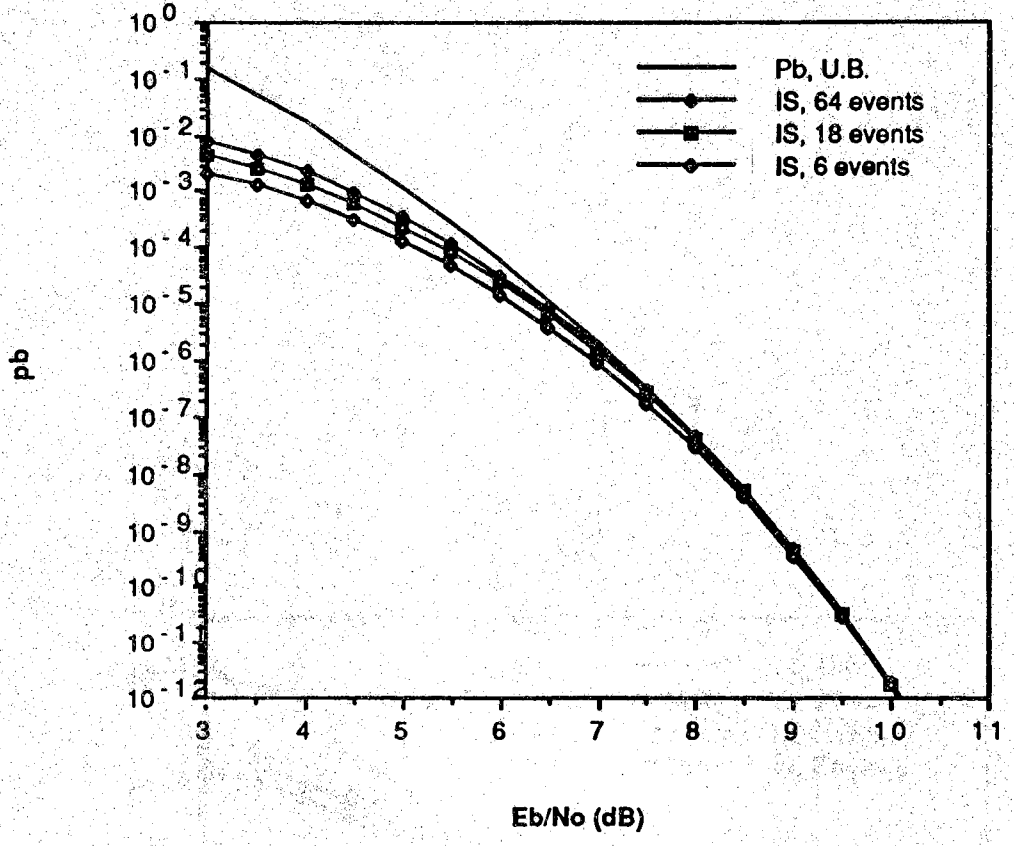


Figure 5.10 P_b for the 16-state, $R = 2/3$, $d_{free} = 2.27$ TCM code.

Table 5.3 Average CPU time required per SNR for TCM simulations.

# of e	CPU (Gould 9080)	CPU (Sun 3/50)
6	7 min	118 min
18	11 min	181 min
64	23 min	303 min

respectively. In all simulations, $L_X(\mathbf{e}_f) = 1500$ and $\varepsilon < 10\%$. We have also preset the minimum number of simulation runs for any $\mathbf{e}$ to be 200 to ensure the reliability of estimates. We can see that our IS estimates agree with the union bounds at moderate-to-high SNR's. Table 5.3 lists average CPU times required to estimate a pair of P_e and P_b on a Gould PN9080 and a SUN 3/50 (with 4MB memory) for various numbers of error sequences $\mathbf{e}$. In particular, for P_b on the order of 10^{-6}, it takes CPU time 400 seconds and 115 minutes for Gould PN9080 and SUN 3/50 respectively. These CPU times are very close to the averages. This is expected because the IS computational cost is theoretically independent of the P_b being estimated. In contrast, the computational cost for the Monte Carlo estimator is inversely proportional to the true value.

6. SIMULATIONS FOR THE SATELLITE CHANNEL

6.1 The Complexity in the Satellite Channel Simulation

Simulations for a satellite channel such as that described in Figure 3.7 is more complicated than for the ideal linear memoryless channel. First of all, there are two noise processes, usually having different signal-to-noise ratios. The uplink noise $Y_u(t)$ is passed through the satellite nonlinear TWTA while the downlink noise $Y_d(t)$ is not. Hence, the contribution from $Y_d(t)$ to the receiver decision statistic is a linear function of $Y_d(t)$. Secondly, we need many samples per signaling interval T for both the signal and the (uplink) noise to model the nonlinear TWTA. As far as the Monte Carlo simulation is concerned, these are the major differences between the satellite channel simulation and the ideal channel simulation. For an efficient importance sampling simulation, which is strongly system-dependent, there are more.

Two problems needed to be addressed before we can implement an efficient importance sampling simulation such as what we have done for ideal linear memoryless channels (the AWGN channel and the BSC) in Section 5.3. As is clear from the discussion so far, the efficiency of our importance sampling algorithm is obtained by the optimal Gaussian mean-translation biasing for the channel noise and the signal biasing for the distribution of ISI patterns in the uncoded system case or the distribution of random signal selector sequences in the trellis-coded system case. For trellis code simulations, we can also utilize the code distance spectrum information to emphasize those "important" signal selector sequences $\mathbf{e}$, i.e., those with a high expected specific error probability $P(\mathbf{e}) = \mathrm{E}[P(\mathbf{X} + \mathbf{e}|\mathbf{X})]$. Therefore, a question arises naturally here is "How do we find the optimal Gaussian mean-translation biasing and an efficient

signal biasing scheme for the satellite channel simulation?" Also, for trellis codes, how the distance property can be used to facilitate the simulation?

First we consider the signal biasing. For uncoded systems, important ISI patterns (those that are more likely to cause a test bit error) can be found analytically if the channel is linear and the memory length is short. In the satellite channel simulation, the channel is nonlinear and the channel memory length is usually long. Therefore, we must find an efficient way to identify important ISI patterns and plan the signal biasing according to their "importance," more specifically, their conditional bit error probabilities, $\mathrm{E}[1_E(\xi(\mathbf{X},\mathbf{Y}))|\mathbf{X}, X_0 = x_0]$, in (5.9), where $X_0 = x_0$ represents the test bit, $\mathbf{X}$ is the ISI pattern and the indicator function takes on the value 1 if the pair $(\mathbf{X},\mathbf{Y})$ results in a test bit error and 0 otherwise. We will demonstrate our approach in the next section when we consider an MSK modulation example. For quasi-regular trellis codes, recall that we used the quasi-regularity of the code and the Gaussian Q function approximation to design the signal biasing $P^*_{\mathbf{X}}(\mathbf{x})$ (see (5.30)–(5.34)) and to decide on the important $\mathbf{e}$'s which should be estimated with small relative precisions (see (5.37)). Note that the Gaussian Q function $Q(d/2\sigma)$ is a union bound for $P(\mathbf{x}+\mathbf{e}|\mathbf{x})$, given $d(\mathbf{x},\mathbf{x}+\mathbf{e}) = d$, which is accurate only for the AWGN channel at a large SNR. In Section 6.3 the same TCM code operating on the satellite channel will be studied. We will show our sampling scheme for $\mathbf{e}$ and $\mathbf{x}$ and justify the technique by simulation data.

Next we consider the optimal Gaussian mean-translation biasing for nonlinear channels with memory. Memory is really not the part that causes problem because we already know how to deal with linear systems with memory. The analysis in Chapter 4 did not place a limit on the dimension of the signal inputs $\mathbf{x}$ or the noise inputs $\mathbf{Y}$. In fact, the two examples in Section 5.3 are linear systems with memory where the memory results from the channel encoding rather than from the ISI. No matter where the memory comes from, as long as the decision statistic D can be expressed as an affine function in the noise vector $\mathbf{Y}$, the dominating point is what is shown in (5.24). For nonlinear systems, our derivations in Chapter 4 still hold. It

is just that the system response function $\xi(\mathbf{x}, \mathbf{Y})$ is a nonlinear (and not an affine) function in $\mathbf{Y}$. However, now the challenge is to find the dominating point. Note that, for moderately nonlinear systems, we will assume the existence of a unique dominating point.

It is difficult to find the dominating point analytically for nonlinear systems. Instead, we can solve the original minimization problem (see Example 4.1)

$$\begin{cases} \min \ \|\mathbf{y}\|^2, \\ \text{subject to} \quad \xi(\mathbf{x}, \mathbf{y}) = 0. \end{cases} \tag{6.1}$$

for the dominating point numerically. However, (6.1) needs to be modified for our satellite channel model which has two noise sources. Recall in the discussion of Section 4.1, the covariance matrix of the random noise vector $\mathbf{Y}$ is $\mathbf{C} = \sigma^2\mathbf{I}$ and we showed that the decision error probability $P_\gamma(\mathbf{Y} \in E)$ decays with an exponential rate $\|\mathbf{v}\|^2/2$, where $\mathbf{v}$ is the dominating point, because the joint probability density function of $\mathbf{Y}$ is $f_\mathbf{Y}(\mathbf{y}) \propto \exp(-\|\mathbf{y}\|^2/2\sigma^2)$. Now we have two noise processes. Suppose that the complex baseband samples of the uplink noise process $Y_u(t)$, $Y_{u,i}, i = 1, \ldots, n_u$, are i.i.d. complex Gaussian random variables with zero mean and variance σ_u^2 in each dimension. Similarly, $Y_{d,i}, i = 1, \ldots, n_d$, denote samples of the downlink noise $Y_d(t)$ with a common variance σ_d^2 in each dimension. Let $\mathbf{Y}_u = (Y_{u,i})_{i=1}^{n_u}$ and $\mathbf{Y}_d = (Y_{d,i})_{i=1}^{n_d}$ be the uplink and downlink noise vectors respectively. Then $\mathbf{Y} = [\mathbf{Y}_u^t, \mathbf{Y}_d^t]^t$ is an $n = n_u + n_d$ dimensional zero mean complex Gaussian random vector. The covariance matrix of $\mathbf{Y}$, $\mathbf{C}$, is an $n \times n$ diagonal matrix whose n_u upper left diagonal components are $2\sigma_u^2$ and the lower right n_d components are $2\sigma_d^2$. Therefore, we have $f_\mathbf{Y}(\mathbf{y}) \propto \exp(-\mathbf{y}^t\mathbf{C}^{-1}\mathbf{y}/2)$. Consequently, what should be minimized in the minimization problem of (6.1), instead of $\|\mathbf{y}\|^2$, is

$$\frac{1}{2}\mathbf{y}^t\mathbf{C}^{-1}\mathbf{y} = \frac{1}{4}\left(\frac{\|\mathbf{y}_u\|^2}{\sigma_u^2} + \frac{\|\mathbf{y}_d\|^2}{\sigma_d^2}\right). \tag{6.2}$$

Equation (6.2) is called the large deviations rate function by Sadowsky and Bucklew in [49]. It is also noted that it would make no difference to the minimization problem

if we consider $\mathbf{Y}$ as a $2n$-dimensional real vector, which we will do in the following discussions unless otherwise stated.

Since the downlink noise is processed only by linear devices before arriving at the decision device, the contribution to the decision statistic from the downlink noise is a linear function of $\mathbf{Y}_d$. Therefore, we can write the decision statistic D as

$$D = \xi(\mathbf{Y}) = \psi(\mathbf{Y}_u) + \mathbf{a}^t\mathbf{Y}_d, \tag{6.3}$$

where $\psi : \mathbf{R}^{2n_u} \rightarrow \mathbf{R}$ is a nonlinear function in $\mathbf{Y}_u$ and the $2n_d$-dimensional vector $\mathbf{a}$ summarizes the linear operation on $\mathbf{Y}_d$. To avoid confusion in notations when taking derivatives, we have dropped the dependency of the system response function on the signal inputs $\mathbf{x}$ because at this level of the simulation $\mathbf{x}$ is a constant vector. Then the dominating point $\mathbf{v} = [\mathbf{v}_u^t, \mathbf{v}_d^t]^t$, if it exists, is the solution to the following constrained minimization problem:

$$\begin{cases} \min & \|\mathbf{y}_u\|^2/\sigma_u^2 + \|\mathbf{y}_d\|^2/\sigma_d^2 \\ \text{s.t.} & \psi(\mathbf{y}_u) + \mathbf{a}^t\mathbf{y}_d = 0 \end{cases} \tag{6.4}$$

Applying the Lagrange method to the above optimization problem, the dominating point $\mathbf{v}$ is found to satisfy

$$\begin{cases} \mathbf{v}_u = \lambda\sigma_u^2 \bigtriangledown \psi(\mathbf{v}_u) \\ \mathbf{v}_d = \lambda\sigma_d^2\mathbf{a} \\ \psi(\mathbf{v}_u) + \mathbf{a}^t\mathbf{v}_d = 0 \end{cases} \tag{6.5}$$

where λ is the Lagrange multiplier and $\bigtriangledown$ denotes the vector gradient. To solve (6.5) numerically, the Newton's method can be used which is easy to implement and has a second order convergence rate. The method can be described as follows. We define a $(2n+1)$ dimensional vector $\mathbf{z} = [\mathbf{v}_u^t, \mathbf{v}_d^t, \lambda]^t$ and a vector function $G : \mathbf{R}^{2n+1} \rightarrow \mathbf{R}^{2n+1}$,

$$G(\mathbf{z}) = \begin{bmatrix} \lambda\sigma_u^2 \bigtriangledown \psi(\mathbf{v}_u) - \mathbf{v}_u \\ \lambda\sigma_d^2\mathbf{a} - \mathbf{v}_d \\ \psi(\mathbf{v}_u) + \mathbf{a}^t\mathbf{v}_d \end{bmatrix}.$$

Then the Newton's method which solves $G(\mathbf{z}) = 0$ by numerical iterations can be expressed as

$$\mathbf{z}_{k+1} = \mathbf{z}_{k+1} - \left[\frac{\partial G}{\partial \mathbf{z}}(\mathbf{z}_k)\right]^{-1} G(\mathbf{z}_k).$$

One disadvantage of this method is that it requires the computation of $\nabla^2\psi(\mathbf{v}_{u,k})$ and the inversion of a high dimensional, non-diagonal matrix $\partial G/\partial \mathbf{z}$. Both can be done but are computationally costly.

A modified Newton's method will be used which does not require the inverse matrix or the knowledge of $\nabla^2\psi(\cdot)$. We can fix $\nabla\psi(\mathbf{v}_u)$ in the first equation of (6.5) and solve for λ first. Thus we have

$$\psi\left(\lambda\sigma_u^2 \nabla \psi(\mathbf{v}_u)\right) + \lambda\sigma_d^2\|\mathbf{a}\|^2 = 0. \tag{6.6}$$

Applying the Newton's method to the last equation, we get

$$\lambda_{k+1} = \lambda_k - \frac{\psi\left(\lambda_k\sigma_u^2 \nabla \psi(\mathbf{v}_{u,k})\right) + \lambda_k\sigma_d^2\|\mathbf{a}\|^2}{\sigma_u^2\| \nabla \psi\left(\lambda_k\sigma_u^2 \nabla \psi(\mathbf{v}_{u,k})\right)\|^2 + \sigma_d^2\|\mathbf{a}\|^2} \tag{6.7}$$

$$\mathbf{v}_{u,k+1} = \lambda_{k+1}\sigma_u^2 \nabla \psi(\mathbf{v}_{u,k}) \tag{6.8}$$

$$\mathbf{v}_{d,k+1} = \lambda_{k+1}\sigma_d^2\mathbf{a} \tag{6.9}$$

One refinement of the above method is to run many iterations on (6.7) to obtain λ_{k+1}.

A convergence is said to have reached if $\|\mathbf{v}_{k+1} - \mathbf{v}_k\|$ is less than some preset value, which we set to be $0.01\|\mathbf{v}_k\|$ in the simulation program. Two initial values are needed for the above iteration method: $\mathbf{v}_{u,0}$ and λ_0. We will set $\mathbf{v}_{u,0} = \phi_0(\hat{\mathbf{s}} - \mathbf{s})$, where ϕ_0 is an input parameter, and $\lambda_0 = -\psi(\mathbf{v}_{u,0})/(\sigma_d^2\|\mathbf{a}\|^2)$. The parameter $\phi_0 = 1/2$ corresponds to the linear system solution (5.24). Depending on the "importance ratio" between the uplink noise and the downlink noise and the severity of nonlinearity, the optimal value of ϕ_0 can be determined experimentally at the beginning of the simulation. Generally, if $\sigma_u^2 > \sigma_d^2$, that is, the uplink is noisier than the downlink, then intuitively the uplink noise is the "important" noise (which has a greater impact on the decision statistic) and we should bias the uplink noise more than the downlink noise, and vice versa. A good choice of the initial value will speed up the convergence of the

iteration. The value of λ_0 is simply the solution to (6.6) with $\mathbf{v}_u = \mathbf{v}_{u,0}$. (Or we can approximate the system by a linear system of the form $D = a + \mathbf{b}^t\mathbf{y}$, and the solution is $\lambda = 2a/\|b\|^2$.) If the system is highly nonlinear, many iterations will be necessary for a convergence. In contrast, if the channel is linear, the dominating point is expected to converge with only one iteration and we did have confirmed this. The notations $\mathbf{s}$ and $\hat{\mathbf{s}}$, as in Chapter 5, denote the signal vectors which are sampled values of $s(\mathbf{x})$ and $s(\mathbf{x}+\mathbf{e})$, i.e., the channel symbol sequences representing $\mathbf{x}$ and $\mathbf{x}+\mathbf{e}$ respectively. For uncoded systems, we can view $\mathbf{x}+\mathbf{e}$ as a "one-branch" error event. More will be said about this in following sections.

It is therefore obvious that whether or not the channel has memory has no effect on the dominating point other than the dimension n. For a large n, the modified Newton's method may require many iterations to reach a convergence. Also, we can see from (6.5) that the dominating point is a function of all factors affecting the decision statistic: σ_u^2, σ_d^2, the filtering operation on $\mathbf{y}_d$, i.e., $H_4(f)$ in Figure 3.7, and the system response function $\psi(\cdot)$. The value of $\psi(\mathbf{y}_u)$ in turn depends on the transmitted signal vector $\mathbf{s}$, the desired error event vector $\hat{\mathbf{s}}$, transponder filters $H_2(f)$ and $H_3(f)$, TWTA, and the receive filter $H_4(f)$. Note that $\mathbf{s}$ and $\hat{\mathbf{s}}$ are signal vectors at the input of the satellite transponder where the uplink noise $\mathbf{y}_u$ is added. Therefore, the transmitting earth station HPA and the BPF $H_1(f)$ in Figure 3.7 can be anything. In our simulation study, we will conveniently set the HPA to be a linear amplifier with a gain of unity and $H_1(f)$ be an all-pass filter. Finally, we remark that with the Newton's method the system response function $\psi(\cdot)$ is required to be differentiable. Other methods can be used to solve (6.5) which may not have this restriction.

6.2 MSK Modulation on the Satellite Channel

In this section, we will study the P_b estimation problem for an uncoded system with MSK modulation operating on the satellite channel of Figure 3.7.

6.2.1 Construction of the Discrete IS Simulation Model

The unfiltered transmitted signal at the satellite transponder input is of the following form

$$s(t) = s^I(t)\cos(2\pi f_c t) - s^Q(t)\sin(2\pi f_c t)$$

where f_c is the carrier frequency of the bandpass signal, and $s^I(t)$ and $s^Q(t)$ are the information bearing lowpass quadrature signals. Let the signaling pulse be $p(t) = A\sin(\pi t/T)$ for $0 \le t \le T$ and $p(t) = 0$ for $t < 0$ and $t > T$, where $T = 2T_b$ is called the symbol (signaling) interval and $1/T_b$ is the information bit rate. Then $s^I(t) = \sum_k X_k^I p(t - kT)$ and $s^Q(t) = \sum_k X_k^Q p(t - kT + T/2)$ where $X_k^I = \pm 1$ and $X_k^Q = \pm 1$ are i.i.d. with values $+1$ and -1 being equally likely as determined by the information bits. An MSK signal is equivalent to two BPSK signals modulated in quadrature with half-sine pulse waveforms and with the Q-channel signal being staggered by $T/2$. The average power of an MSK signal is $A^2/2$.

We will employ the event simulation method for uncoded systems as described in Section 5.1 which considers the specific event of a bit error. For example, we choose the 0'th inphase bit as our test bit. By the symmetry of the channel, we can set $X_0^I = -1$ which we take to correspond to a bit value 0. Let D denote the demodulator output which is the decision statistic for the test bit. That is, the decision device performs a threshold test $D \gtrless 0$ and decides on 1 if $D \ge 0$ and 0 if $D < 0$. Hence, the decision error event is $\{D \ge 0\}$ and the bit error probability can be expressed as $P_b = P(D \ge 0)$.

In order to apply the techniques developed in Chapter 4 and Section 5.1, we need to first derive a discrete-time model of the form $D = \xi(\mathbf{X}, \mathbf{Y})$ where $\mathbf{X}$ is a random ISI pattern and $\mathbf{Y}$ is a finite dimensional vector which models the Gaussian noise inputs.

Figure 6.1 shows a complex baseband equivalent model of Figure 3.7 and notations for signals and filters which we will use throughout this chapter. $Y_u^I(t)$ and $Y_u^Q(t)$ are the inphase (I-channel) and quadrature (Q-channel) lowpass components respectively

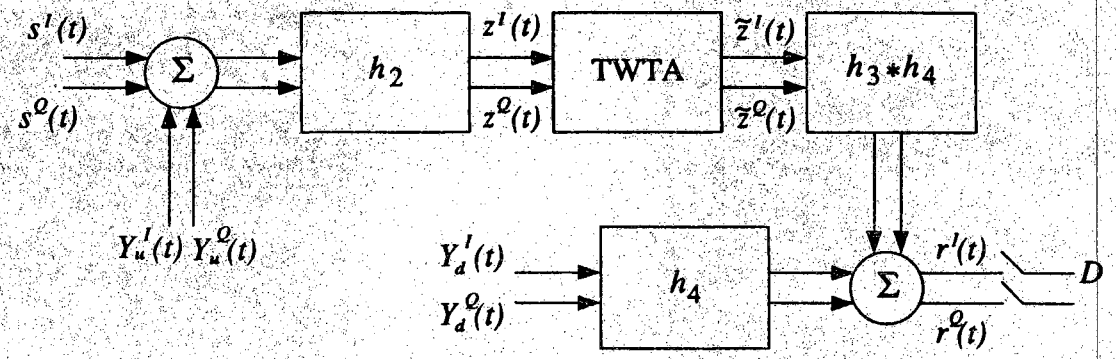


Figure 6.1 A complex baseband equivalent of the satellite channel model.

of the uplink noise process $Y_u(t)$ in Figure 3.7. Similarly, $Y_d^I(t)$ and $Y_d^Q(t)$ are the I and Q channel components of the downlink noise $Y_d(t)$. The complex impulse response function $h_2(t) = h_2^I(t) + jh_2^Q(t)$ is the lowpass equivalent of the satellite input filter $H_2(f)$. This notational rule also applies to filters $H_3(f)$ and $H_4(f)$. We have also combined filters $H_3(f)$ and $H_4(f)$ for the signal plus uplink noise path where $h_3 * h_4$ in Figure 6.1 denotes the convolution of $h_3(t)$ and $h_4(t)$. For notational convenience, we will denote the combined filter $h_{34}(t)$. The input signal to the TWTA is denoted $z(t)$ and the output is $\tilde{z}(t)$. The demodulator output is the complex signal $r(t)$.

Now, let N_s be the number of samples per symbol interval T, i.e., the sampling rate is N_s/T and the sampling period is $\Delta = T/N_s$. We will use the subscript k as the index for information bits X_k^I and X_k^Q, and k' as the sample numbering index. For convenience, we line up the time axis and the "sampling axis" such that the sample at time $t = 0$ is the $k' = 0$ sample. This can be explained by Figure 6.2 which shows the transmitted signals $s^I(t)$ and $s^Q(t)$. The dots on the time axes represent sampling times. Thus, the I and Q channel samples at time $t = T$ are the N_s'th samples. The uplink noise samples of $Y_u^I(t)$ and $Y_u^Q(t)$ are denoted $\{Y_{u,k'}^I\}_{k'}$ and $\{Y_{u,k'}^Q\}_{k'}$ respectively. They are i.i.d. zero mean Gaussian random variables with variance σ_u^2. Similarly, $\{Y_{d,k'}^I\}_{k'}$ and $\{Y_{d,k'}^Q\}_{k'}$ are samples of $Y_d^I(t)$ and $Y_d^Q(t)$ with a common variance σ_d^2.

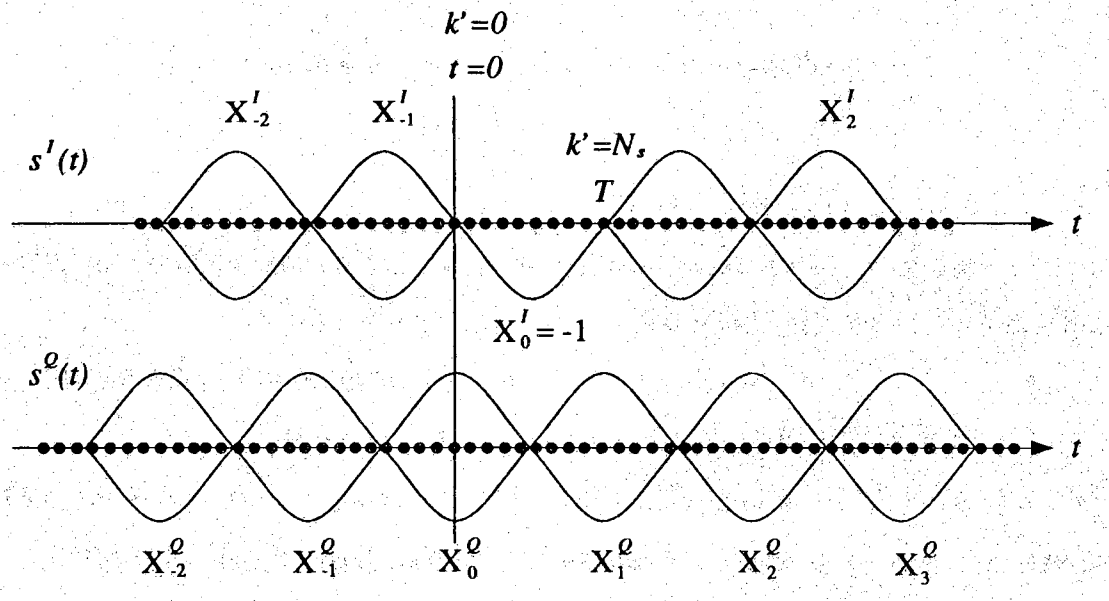


Figure 6.2 $s^I(t)$, $s^Q(t)$ and sampling indices.

Suppose that the decision statistic D is the sampled value at time $t = t_D$ of the inphase demodulator output $r^I(t)$, which corresponds to the K'th sample on the sampling axis. Recall from Chapter 3 that we have assumed all filters have finite discrete impulse response functions. Let $h_{2,i} = h_2(i\triangle), m_2^- \leq i \leq m_2^+$, and $h_{2,i} = 0$ for $i < m_2^-$ or $i > m_2^+$. If the filter is causal, we have $m_2^- \geq 0$. However, as mentioned in Chapter 3, we can use noncausal filters without loss of generality . A time delay can be added to the received waveform if we have causal filters. It is not required that the filter is symmetric about $t = 0$, hence, we can have $m_2^- \neq -m_2^+$. Similarly, $h_{4,i} \neq 0$ for $m_4^- \leq i \leq m_4^+$, and $h_{34,i} \neq 0$ when $m_{34}^- \leq i \leq m_{34}^+$. If all filters are noncausal, symmetric and the channel is linear, the optimal sampling time would be $t_D = T/2$ or equivalently $K = N_s/2$. In the following discussion and the simulation program, we will consider the general case and let K be an input parameter. Or, with little modification, the program can search for the optimal sampling time within a signaling interval which is a common practice in satellite simulations.

We are now ready to find the signal vector $\mathbf{X}$ and the noise vector $\mathbf{Y}$ in the expression $D = \xi(\mathbf{X}, \mathbf{Y})$. See the signal flow diagram in Figure 6.3. The decision statistic $D = r^I(t_D) = r^I_K$ depends on the $h_4(t)$-filtered downlink noise process and the $h_{34}(t)$-filtered TWTA outputs. That is, the contribution to D from the downlink noise samples $\{Y^I_{d,k'}\}_{k'}$ and $\{Y^Q_{d,k'}\}_{k'}$ is

$$rn^I_K = \triangle \sum_{i=m_4^-}^{m_4^+} Y^I_{d,K-i} h^I_{4,i} - Y^Q_{d,K-i} h^Q_{4,i}.$$

The above display is simply a discrete approximation of the real part of the complex convolution

$$\left(x^I(t)+jx^Q(t)\right) * \left(h^I(t)+jh^Q(t)\right) =$$

$$\left(x^I(t)*h^I(t)-x^Q(t)*h^Q(t)\right) + j\left(x^I(t)*h^Q(t)+x^Q(t)*h^I(t)\right). \tag{6.10}$$

Therefore, the downlink noise samples that are relevant to the decision are $Y_{d,k'}, K - m_4^+ \le k' \le K - m_4^-$. This is illustrated in the lower part of Figure 6.3. Similarly, the decision statistic depends on the TWTA output complex samples $\tilde{z}_{k'}, K - m_{34}^+ \le k' \le K - m_{34}^-$. The TWTA is memoryless, i.e., $\tilde{z}_{k'}$ is a function of the input at the same instance, $z_{k'}$, only. The input sample $z_{k'}$ is in turn the output of $h_2(t)$-filtered transmitted signal and uplink noise samples. Hence each $z_{k'}$, $K-m_{34}^+ \le k' \le K-m_{34}^-$, depends on signal samples s_i and uplink noise samples $Y_{u,i}$ for $k'-m_2^+ \le i \le k'-m_2^-$. Consequently, the relevant transmitted signal and uplink noise samples are $s_{k'}$ and $Y_{u,k'}, K - m_{34}^+ - m_2^+ \le k' \le K - m_{34}^- - m_2^-$, respectively.

Therefore, if we define the uplink noise vector

$$\mathbf{Y}_u = \left(Y^I_{u,k'}, Y^Q_{u,k'}\right), \quad K-m_{34}^+-m_2^+ \le k' \le K-m_{34}^--m_2^- \tag{6.11}$$

and the downlink noise vector

$$\mathbf{Y}_d = \left(Y^I_{d,k'}, Y^Q_{d,k'}\right), \quad K-m_4^+ \le k' \le K-m_4^-, \tag{6.12}$$

then the noise vector $\mathbf{Y}$ in the decision statistic expression $D = \xi(\mathbf{X}, \mathbf{Y})$ is $\mathbf{Y} = [\mathbf{Y}_u^t, \mathbf{Y}_d^t]^t$. On the other hand, the transmitted signal samples which have impacts

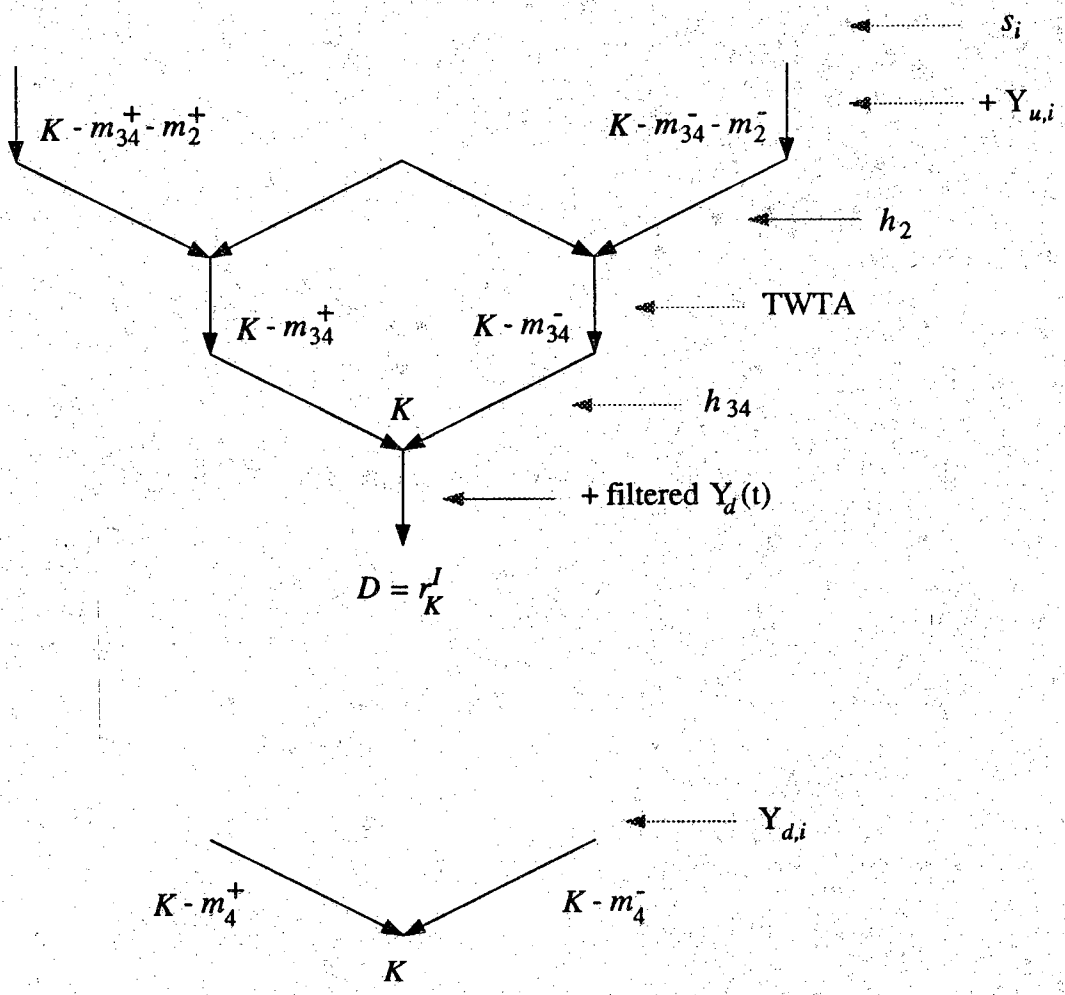


Figure 6.3 MSK signal flow diagram in the satellite channel.

on D are sampled values of the transmitted waveforms $s^I(t) = \sum_k X_k^I p(t - kT)$ and $s^Q(t) = \sum_k X_k^Q p(t - kT + T/2)$ for some k. It then is found that the information bits which need to be considered in the event simulation are

$$\mathbf{X}^I = \left(X_k^I\right), \; -\left\lceil \frac{m_{34}^+ + m_2^+ - K}{N_s} \right\rceil \le k \le \left\lceil \frac{|m_{34}^- + m_2^-| - N_s + K}{N_s} \right\rceil \tag{6.13}$$

and

$$\mathbf{X}^Q = \left(X_k^Q\right), \; -\left\lceil \frac{m_{34}^+ + m_2^+ - K - N_s/2}{N_s} \right\rceil \le k \le \left\lceil \frac{|m_{34}^- + m_2^-| + K - N_s/2}{N_s} \right\rceil. \tag{6.14}$$

Hence the signal vector is $\mathbf{X} = \left[(\mathbf{X}^I)^t, (\mathbf{X}^Q)^t\right]^t$. For example, if $N_s = 16$, $K = 8$, $m_2^+ = m_{34}^+ = 31$ and $m_2^- = m_{34}^- = -32$, then we have $\mathbf{X} = \left[X_{-4}^I, \ldots, X_4^I, X_{-3}^Q, \ldots, X_4^Q\right]$ and recall that $X_0^I = -1$ is fixed.

Now we have the decision statistic of the form $D = \xi(\mathbf{X}, \mathbf{Y})$ and we know precisely what $\mathbf{X}$ and $\mathbf{Y}$ are. The event simulation method which we has developed in Section 5.1 states that to estimate

$$\begin{aligned} P_b &= P(D \ge 0) = \mathrm{E}[1_E(\xi(\mathbf{X}, \mathbf{Y}))\,|X_0^I = x_0] \\ &= \mathrm{E}\left[\mathrm{E}[1_E(\xi(\mathbf{X}, \mathbf{Y}))\,|\mathbf{X}, X_0^I = x_0]\right]. \end{aligned} \tag{6.15}$$

the estimator is

$$\hat{P}^* = \frac{1}{L_X} \sum_{\ell=1}^{L_X} w_{\mathbf{X}|X_0^I}\left(\mathbf{X}^{(\ell)}|x_0\right) \times \left\{ \frac{1}{L_Y} \sum_{\ell'=1}^{L_Y} 1_E\left(\xi(\mathbf{X}^{(\ell)}, \mathbf{Y}^{(\ell,\ell')})\right) w_{\mathbf{Y}|\mathbf{X}, X_0^I}\left(\mathbf{Y}^{(\ell,\ell')}|\mathbf{X}^{(\ell)}, x_0\right) \right\}, \tag{6.16}$$

where $x_0 = -1$ and $E = \{(\mathbf{x}, \mathbf{y}) : \xi(\mathbf{x}, \mathbf{y}) \ge 0\}$ is the error set which, although the notation does not show, is a function of $\mathbf{x}$.

The simulation program thus is executed in a double-do-loop structure. First we select an ISI pattern $\mathbf{x}$ according to the biased signal probability mass function $P^*_{\mathbf{X}|X_0^I}(\mathbf{x}|x_0)$. We then have the same format $D = \xi(\mathbf{Y}) = \psi(\mathbf{Y}_u) + \mathbf{a}^t\mathbf{Y}_d$ as in (6.3) of Section 6.1. Therefore, a dominating point can be computed by the modified Newton's method described in that section. For this ISI pattern and its dominating

point, L_Y samples of the random noise vector $\mathbf{Y}$ will be taken from the optimally mean-translated Gaussian simulation distribution as represented by the joint density function $f^*_{\mathbf{Y}|\mathbf{X},X_0^I}(\mathbf{y}|\mathbf{x},x_0) = f^*_{\mathbf{Y}_u|\mathbf{X},X_0^I}(\mathbf{y}_u|\mathbf{x},x_0) f^*_{\mathbf{Y}_d|\mathbf{X},X_0^I}(\mathbf{y}_d|\mathbf{x},x_0)$. The signal and noise samples are then passed through the channel and a threshold test is carried out at the demodulator sampler output to see if the event of a test bit error has occurred. If yes, the indicator function in (6.16) registers a one and weights it by the importance sampling weight product $w_{\mathbf{X}|X_0^I}\left(\mathbf{x}^{(\ell)}|x_0\right) w_{\mathbf{Y}|\mathbf{X},X_0^I}\left(\mathbf{y}^{(\ell,\ell')}|\mathbf{x}^{(\ell)},x_0\right)$ for the ℓ'th sample of $\mathbf{X}$ and (ℓ,ℓ')'th sample of $\mathbf{Y}$. Note that we need to compute the IS weight $w_{\mathbf{X}|X_0^I}\left(\mathbf{x}^{(\ell)}|x_0\right)$ only once for each sample of $\mathbf{X}$ in L_Y runs. Another ISI pattern then is chosen and the above procedures repeated until the relative precision of the bit error estimate falls below some pre-determined value which we set to be 10%. We next demonstrate techniques in finding the dominating point and the design of an efficient signal biasing scheme.

6.2.2 Finding the Dominating Point

Recall the discussion following (6.3) and the solution to the minimization problem in (6.7) – (6.9). To find the dominating point, we need to know the downlink noise linear transformation vector $\mathbf{a}$, the uplink noise transfer function $\psi(\cdot)$ and and its gradient $\nabla\psi(\cdot)$. The test statistic D can be expressed as:

$$\begin{aligned} D &= r^I(t_D) = r_K^I \\ &= rx_K^I + rn_K^I \end{aligned}$$

where rx_K^I denotes the contribution from the signal plus the uplink noise samples $\mathbf{s}+\mathbf{y}_u$ while rn_K^I is solely from the downlink noise samples $\mathbf{y}_d$.

Hence, if we express the decision statistic D as $D = \psi(\mathbf{y}_u) + \mathbf{a}^t\mathbf{y}_d$ we have

$$\mathbf{a}^t\mathbf{y}_d = rn_K^I = \sum_{i=m_4^-}^{m_4^+} y_{d,K-i}^I h_{4,i}^I - Y_{d,K-i}^Q h_{4,i}^Q, \tag{6.17}$$

where for convenience we have incorporated the sampling period factor Δ into the filter impulse response function. This practice will be also used for all other filters.

It is then easy to find $\mathbf{a} = \left(a_{k'}^I, a_{k'}^Q\right)_{k'}$ where

$$a_{k'}^I = h_{4,K-k'}^I \tag{6.18}$$

$$a_{k'}^Q = -h_{4,K-k'}^Q \tag{6.19}$$

for $K - m_4^+ \leq k' \leq K - m_4^-$.

Next we will find the uplink noise transfer function $\psi(\mathbf{y}_u)$. We have

$$\begin{aligned}\psi(\mathbf{y}_u) &= rx_K^I \\ &= \sum_{i=m_{34}^-}^{m_{34}^+} \tilde{z}_{K-i}^I h_{34,i}^I - \tilde{z}_{K-i}^Q h_{34,i}^Q \\ &= \sum_{k'=K-m_{34}^+}^{K-m_{34}^-} \tilde{z}_{k'}^I b_{k'}^I + \tilde{z}_{k'}^Q b_{k'}^Q \end{aligned} \tag{6.20}$$

where

$$\begin{aligned} b_{k'}^I &= h_{34,K-k'}^I \\ b_{k'}^Q &= -h_{34,K-k'}^Q \end{aligned}$$

for $K - m_{34}^+ \leq k' \leq K - m_{34}^-$, and $\tilde{z}_{k'}$ is the output of the TWTA responding to an input sample $z_{k'}$.

Recall from Section 3.3 in which we have shown that the memoryless nonlinearity of the TWTA results in an input-output relationship as described by (3.11) and (3.12) which are rewritten below for they will be of use later:

$$\tilde{z}_{k'}^I = c_1\eta_0(c_2A^2)z_{k'}^I - s_1\eta_1(s_2A^2)z_{k'}^Q \tag{6.21}$$

$$\tilde{z}_{k'}^Q = s_1\eta_1(s_2A^2)z_{k'}^I + c_1\eta_0(c_2A^2)z_{k'}^Q \tag{6.22}$$

where c_1, c_2, s_1, s_2 are constants, $\eta_0(\alpha) = e^{-|\alpha|}\mathrm{I}_0(\alpha)$ and $\eta_1(\alpha) = e^{-|\alpha|}\mathrm{I}_1(\alpha)$ are exponentially scaled modified Bessel functions of the first kind of order 0 and 1 respectively, and $A^2 = (z_{k'}^I)^2 + (z_{k'}^Q)^2$ is the instantaneous input squared amplitude. Or, if we use

the spline function approximation also described in Section 3.3, the instantaneous I/O relationship is

$$\tilde{z}_{k'}^I = \left(C_3A^2+C_2A+C_1+\frac{C_0}{A}\right) z_{k'}^I - \left(S_3A^2+S_2A+S_1+\frac{S_0}{A}\right) z_{k'}^Q \tag{6.23}$$

$$\tilde{z}_{k'}^Q = \left(S_3A^2+S_2A+S_1+\frac{S_0}{A}\right) z_{k'}^I + \left(C_3A^2+C_2A+C_1+\frac{C_0}{A}\right) z_{k'}^Q \tag{6.24}$$

where C_i and $S_i, i = 0, \ldots, 3$, are constants.

The uplink noise vector $\mathbf{y}_u$ and the TWTA input vector $\mathbf{z}$ are related by

$$z_{k'}^I = c_{k'}^I + \sum_{i=m_2^-}^{m_2^+} h_{2,i}^I y_{u,k'-i}^I - h_{2,i}^Q y_{u,k'-i}^Q \tag{6.25}$$

$$z_{k'}^Q = c_{k'}^Q + \sum_{i=m_2^-}^{m_2^+} h_{2,i}^Q y_{u,k'-i}^I + h_{2,i}^I y_{u,k'-i}^Q \tag{6.26}$$

where

$$c_{k'}^I = \sum_{i=m_2^-}^{m_2^+} h_{2,i}^I s_{k'-i}^I - h_{2,i}^Q s_{k'-i}^Q \tag{6.27}$$

$$c_{k'}^Q = \sum_{i=m_2^-}^{m_2^+} h_{2,i}^Q s_{k'-i}^I + h_{2,i}^I s_{k'-i}^Q \tag{6.28}$$

and $\mathbf{s} = (s_{k'}^I, s_{k'}^Q)$, $K - m_{34}^+ - m_2^+ \le k' \le K - m_{34}^- - m_2^-$, is the transmitted signal vector which has the same dimension as the uplink noise vector $\mathbf{y}_u$.

In summary, the uplink noise transfer function is

$$\begin{aligned}
\psi(\mathbf{y}_u) &= \sum_{k'=K-m_{34}^+}^{K-m_{34}^-} \tilde{z}_{k'}^I b_{k'}^I + \tilde{z}_{k'}^Q b_{k'}^Q \\
&= \sum_{k'=K-m_{34}^+}^{K-m_{34}^-} \left(\rho_1 z_{k'}^I - \rho_2 z_{k'}^Q\right) b_{k'}^I + \left(\rho_2 z_{k'}^I + \rho_1 z_{k'}^Q\right) b_{k'}^Q \\
&= \sum_{k'=K-m_{34}^+}^{K-m_{34}^-} \left(\rho_1 b_{k'}^I + \rho_2 b_{k'}^Q\right) z_{k'}^I + \left(\rho_1 b_{k'}^Q - \rho_2 b_{k'}^I\right) z_{k'}^Q,
\end{aligned} \tag{6.29}$$

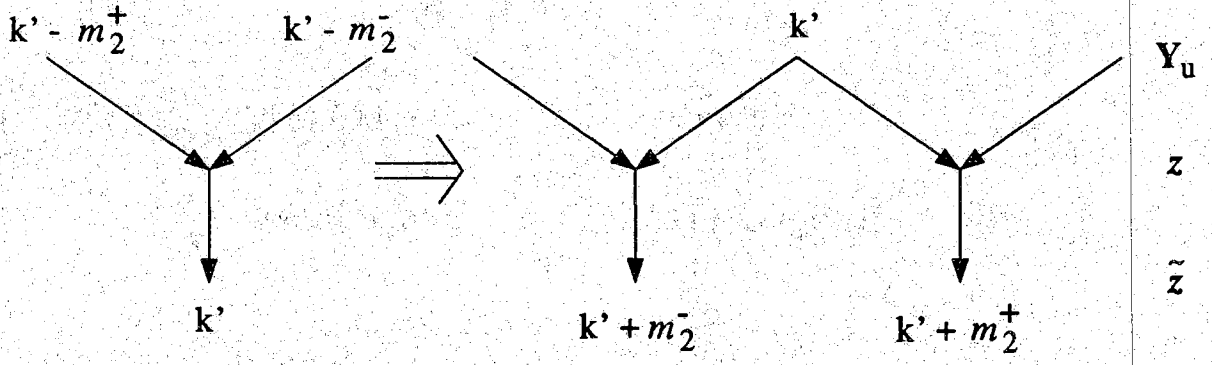


Figure 6.4 The range of impact of a uplink noise sample $y_{u,k'}$.

where

$$\rho_1 = c_1\eta_0(c_2A^2) \tag{6.30}$$

$$\rho_2 = s_1\eta_1(s_2A^2) \tag{6.31}$$

if the Bessel approximation is used, and

$$\rho_1 = C_3A^2 + C_2A + C_1 + \frac{C_0}{A} \tag{6.32}$$

$$\rho_2 = S_3A^2 + S_2A + S_1 + \frac{S_0}{A} \tag{6.33}$$

if the spline approximation is used. Finally, $(z^I_{k'}, z^I_{k'})$ is computed by (6.25) – (6.28).

The last task is to compute the gradient of $\psi(\cdot)$,

$$\nabla\psi(\mathbf{y}_u) = \left(\frac{\partial\psi}{\partial y^I_{u,k'}}, \frac{\partial\psi}{\partial y^Q_{u,k'}}\right)_{k'=K-m^+_{34}-m^+_2}^{K-m^-_{34}-m^-_2}$$

assuming it is everywhere differentiable with respective to $\mathbf{y}_u$. We note that $y_{u,k'}$ affects the value of z_i for $k' + m_2^- \le i \le k' + m_2^+$. This is illustrated in Figure 6.4. Let us first consider the real part of the last display. From (6.29), we have

$$\frac{\partial\psi}{\partial y^I_{u,k'}} = \sum_{i=k'+m_2^-}^{k'+m_2^+} b^I_i \frac{\partial\tilde{z}^I_i}{\partial y^I_{u,k'}} + b^Q_i \frac{\partial\tilde{z}^Q_i}{\partial y^I_{u,k'}}. \tag{6.34}$$

Note that $\partial \tilde{z}_i^I / \partial y_{u,k'}^I = \partial \tilde{z}_i^Q / \partial y_{u,k'}^I = 0$, if $i < K - m_{34}^+$ or $i > K - m_{34}^-$ because $\psi(\mathbf{y}_u)$ depends only on z_i for $K - m_{34}^+ \leq i \leq K - m_{34}^-$ by (6.29). This is also clear from Figure 6.4. For any one TWTA output sample z_i, because the nonlinearity is memoryless, we get

$$\frac{\partial \tilde{z}_i^I}{\partial y_{u,k'}^I} = \frac{\partial \tilde{z}_i^I}{\partial z_i^I} \frac{\partial z_i^I}{\partial y_{u,k'}^I} + \frac{\partial \tilde{z}_i^I}{\partial z_i^Q} \frac{\partial z_i^Q}{\partial y_{u,k'}^I} \tag{6.35}$$

where, from (6.25) and (6.26), it is easy to show that

$$\frac{\partial z_i^I}{\partial y_{u,k'}^I} = h_{2,i-k'}^I$$

$$\frac{\partial z_i^Q}{\partial y_{u,k'}^I} = h_{2,i-k'}^Q$$

The derivation is similar for the imaginary part $\partial \psi / \partial y_{u,k'}^Q$. Then, after grouping terms, we have

$$\frac{\partial f}{\partial y_{u,k'}^I} = \sum_{i=k'+m_2^-}^{k'+m_2^+} w_1 h_{2,i-k'}^I + w_2 h_{2,i-k'}^Q \tag{6.36}$$

$$\frac{\partial f}{\partial y_{u,k'}^Q} = \sum_{i=k'+m_2^-}^{k'+m_2^+} -w_1 h_{2,i-k'}^Q + w_2 h_{2,i-k'}^I \tag{6.37}$$

for $K - m_{34}^+ - m_2^+ \leq k' \leq K - m_{34}^- - m_2^-$, where

$$w_1 = b_i^I \frac{\partial \tilde{z}_i^I}{\partial z_i^I} + b_i^Q \frac{\partial \tilde{z}_i^Q}{\partial z_i^I}$$

$$w_2 = b_i^I \frac{\partial \tilde{z}_i^I}{\partial z_i^Q} + b_i^Q \frac{\partial \tilde{z}_i^Q}{\partial z_i^Q}.$$

Again, terms are omitted in the summations of (6.36) and (6.37) if $i < K - m_{34}^+$ or $i > K - m_{34}^-$.

The derivatives in the above w_1 and w_2 displays are the I and Q channel TWTA outputs taken with respect to the I and Q channel inputs. Taking the derivatives of (6.21) – (6.24), we find that, after some manipulations,

$$\frac{\partial \tilde{z}_i^I}{\partial z_i^I} = \rho_1 + \rho_3 z_i^I z_i^I + \rho_4 z_i^I z_i^Q \tag{6.38}$$

$$\frac{\partial \tilde{z}_i^I}{\partial z_i^Q} = -\rho_2 + \rho_3 z_i^I z_i^Q - \rho_4 z_i^Q z_i^Q \tag{6.39}$$

$$\frac{\partial \tilde{z}_i^Q}{\partial z_i^I} = \rho_2 + \rho_3 z_i^I z_i^Q + \rho_4 z_i^I z_i^I \tag{6.40}$$

$$\frac{\partial \tilde{z}_i^Q}{\partial z_i^Q} = \rho_1 + \rho_3 z_i^Q z_i^Q + \rho_4 z_i^I z_i^Q. \tag{6.41}$$

As mentioned in Chapter 3, useful rules are $I_0'(x) = I_1(x)$ and $I_1'(x) = I_0(x) - \frac{1}{x} I_1(x)$ for the derivatives of Bessel functions [1]. With the Bessel function approximation to the AM/AM and AM/PM nonlinearities, ρ_1 and ρ_2 are as defined in (6.30) and (6.31), and

$$\rho_3 = 2c_1c_2\left[-\eta_0(c_2A^2) + \eta_1(c_2A^2)\right]$$

$$\rho_4 = 2s_1s_2\left[\eta_0(s_2A^2) - \left(1 + \frac{1}{s_2A^2}\right)\eta_1(s_2A^2)\right].$$

For the spline function approximation, ρ_1 and ρ_2 are as defined in (6.32) and (6.33), and

$$\rho_3 = 2C_3 + \frac{C_2}{A} - \frac{C_0}{A^3}$$

$$\rho_4 = 2S_3 + \frac{S_2}{A} - \frac{S_0}{A^3}.$$

We note that the computation of $\rho_1 - \rho_4$ for the spline approximation is simpler than for the Bessel approximation. This completes our discussion of finding the dominating point numerically. It is evident that this is a difficult and CPU-intensive part of the simulation algorithm, and it will get even more complicated when it comes to coded systems. However, we will show via simulation data that this hard work does pay off.

6.2.3 MSK Signal Biasing

In this subsection, we discuss the design of the signal biasing, i.e., the biasing of the probability mass function of the ISI patterns, $P_{\mathbf{X}|X_0^I}(\mathbf{x}|x_0)$. Recall from (6.13) and (6.14), the components of $\mathbf{X} = \left[(\mathbf{X}^I)^t, (\mathbf{X}^Q)^t\right]^t$ are the test bit $X_0^I = x_0 = -1$

and its adjacent bits in the I and Q channels whose number depends on the channel memory length and the demodulator output sampling time. The principle we have established in Section 4.3 for the signal biasing is that the biased signal probability mass function $P^*_{\mathbf{X}|X_0^I}(\mathbf{x}|x_0)$ should be proportional to $\beta(\mathbf{x})P_{\mathbf{X}|X_0^I}(\mathbf{x}|x_0) = P(\mathbf{x})/J$ where $\beta(\mathbf{x}) = P(\mathbf{x})$ is the conditional bit error probability given a particular ISI pattern $\mathbf{X} = \mathbf{x}$ is transmitted, i.e., $P(\mathbf{x}) = \mathrm{E}[1_E(\xi(\mathbf{X}, \mathbf{Y}))\,|\mathbf{X} = \mathbf{x}, X_0^I = x_0]$, and J is the total number of ISI patterns. We have $J = 2^M$ here if there are M ISI bits.

For our simulation study in the next subsection, we will have $M = 16$ and hence $J = 2^{16}$ ISI patterns. It will be quite a task to design an efficient biasing scheme for all of them, and in fact is unnecessary. In general, the most adjacent bits of X_0^I have greater impacts on the decision statistic than those that are farther away. That is, they are the "important ISI bits," and thus we can bias their distributions only. Suppose that, among the 16 ISI bits, we will only bias the two most adjacent bits in both the I channel and the Q channel. Then we can write $\mathbf{X}$ as

$$\mathbf{X} = (X_0^I, \tilde{\mathbf{X}}, \hat{\mathbf{X}})$$

where $\tilde{\mathbf{X}} = (X_1^I, X_{-1}^I, X_0^Q, X_1^Q)$ represents the 4 closest bits to X_0^I (see Figure 6.2) and $\hat{\mathbf{X}}$ is the collection of all other 12 ISI bits whose probability distributions will not be biased. Therefore, our simulation signal density function is

$$P^*_{\mathbf{X}|X_0^I}(\mathbf{x}|x_0) = P^*_{\tilde{\mathbf{X}}|X_0^I}(\tilde{\mathbf{x}}|x_0)P_{\hat{\mathbf{X}}|X_0^I}(\hat{\mathbf{x}}|x_0). \tag{6.42}$$

The importance sampling weighting function is

$$w_{\mathbf{X}|X_0^I}(\mathbf{x}|x_0) = \frac{P_{\mathbf{X}|X_0^I}(\mathbf{x}|x_0)}{P^*_{\mathbf{X}|X_0^I}(\mathbf{x}|x_0)} = \frac{P_{\tilde{\mathbf{X}}|X_0^I}(\tilde{\mathbf{x}}|x_0)}{P^*_{\tilde{\mathbf{X}}|X_0^I}(\tilde{\mathbf{x}}|x_0)},$$

and here we have $P_{\tilde{\mathbf{X}}|X_0^I}(\tilde{\mathbf{x}}|x_0) = 1/16$.

The problem thus reduces to finding $P^*_{\tilde{\mathbf{X}}|X_0^I}(\tilde{\mathbf{x}}|x_0) \propto P(\tilde{\mathbf{x}})$. For the nonlinear satellite channel, the conditional bit error probability $P(\tilde{\mathbf{x}})$ is difficult to obtain analytically. However, for the purpose of designing the signal biasing, we can approximate the channel with a linear channel and obtain an approximation of $P(\tilde{\mathbf{x}})$. Recall the linear

system example Example 4.1 in Chapter 4. If the system is linear (D is an affine function in the noise vector $\mathbf{Y}$), we have $P(\mathbf{Y} \in E) = P(\text{decision error}) = Q\left(\sqrt{\|\mathbf{v}\|^2}/\sigma\right)$ where $\mathbf{v}$ is the dominating point. Here, suppose that $\mathbf{x} = (x_0, \tilde{\mathbf{x}}, \hat{\mathbf{x}})$ is transmitted and the dominating point is $\mathbf{v}(\tilde{\mathbf{x}})$. Then $Q\left(\sqrt{\|\mathbf{v}(\tilde{\mathbf{x}})\|^2}/\sigma\right)$ is the approximated conditional bit error probability $P(\tilde{\mathbf{x}})$. (Note that the above $P(\tilde{\mathbf{x}})$ actually should be averaged over all possible $\hat{\mathbf{x}}$. However, for designing the biasing, we can choose $\hat{\mathbf{x}}$, say, a fixed alternating ± 1 sequence.) Therefore, we get

$$P^*_{\tilde{\mathbf{X}}|X_0^I}(\tilde{\mathbf{x}}|x_0) = \frac{Q\left(\sqrt{\|\mathbf{v}(\tilde{\mathbf{x}})\|^2}/\sigma\right)}{\sum_{\tilde{\mathbf{x}}} Q\left(\sqrt{\|\mathbf{v}(\tilde{\mathbf{x}})\|^2}/\sigma\right)} \tag{6.43}$$

We can compute $P^*_{\tilde{\mathbf{X}}|X_0^I}(\tilde{\mathbf{x}}|x_0)$ at the beginning of the simulation by using the numerical iteration method to find the dominating point for all 16 ISI patterns. The noise variance σ^2 can be approximated by the linear channel formula (3.8).

6.2.4 Simulation Parameters and Results

In our simulation study, we use $N_s = 16$ (a typical number for digital telecommunication satellites) samples per symbol interval $T = 2T_b$ for both I and Q channels. The demodulator output sampling time is set at the middle of the signaling pulse, i.e., the 8'th sample ($K = 8$). All filters are specified in the baseband of the frequency domain as the ideal "brick-wall" filters with bandwidth B such that $BT = 1.5$. The inverse FFT then computes finite impulse response functions of the filters with the time domain truncation occurs at $\pm 2T$. The combined downlink filter h_{34} is also truncated to be within $\pm 2T$ although theoretically it spans over $\pm 4T$. Therefore we have $m_2^- = m_{34}^- = m_4^- = -32$ and $m_2^+ = m_{34}^+ = m_4^+ = 31$.

Consequently, the set of information bits which are of concern in our event simulation, i.e., the ISI pattern $\mathbf{X}$, is $\left(X_{-4}^I, \ldots, X_4^I, X_{-3}^Q, \ldots, X_4^Q\right)$ where $X_0^I = -1$ is fixed. Thus, the total number of ISI bits is 16. As discussed in the last subsection, only the probability distributions of the four bits which are closest to $X_0^I = -1$ will be biased in the signal biasing. On the other hand, the dimension of the downlink noise vector $\mathbf{y}_d$ (and that of the vector $\mathbf{a}$) is $2 \times \left(m_4^+ - m_4^- + 1\right) = 128$, and for the uplink

noise vector $\mathbf{y}_u$ it is $2\times\left(m_2^+ + m_{34}^+ - m_2^- - m_{34}^- + 1\right) = 254$. Thus the noise vector $\mathbf{y}$ is a 382-dimensional real (or 191-dimensional complex) vector. In simulations, we found that $L_y = 5$ is an appropriate inner do-loop number for all SNR's.

The satellite TWTA input backoff power is IBO = 1 dB. To compensate for the phase rotation introduced by the TWTA AM/PM conversion, a phase shift corresponding to the phase of the operating point is added to the impulse response function h_{34}. The uplink and downlink noise samples are i.i.d. Gaussian random variables with zero mean and the variance is computed as (recall from Section 3.1)

$$\begin{aligned}\sigma^2 &= N_0 N_s R = N_s \frac{1}{2x} E_s R \\ &= N_s \frac{1}{2x} \frac{A^2}{2}\end{aligned}$$

where $R = 1/T = 1/2T_b$ is the symbol rate, $E_b = E_s/2$ is the energy per bit, x is the given specification of E_b/N_0 and $E_s R = A^2/2$ is the carrier power which is $A^2/2$. The value of A is calculated from the TWTA IBO and the AM/AM conversion characteristic. The spline function approximation is used.

Table 6.1 shows simulation results for the equal uplink and downlink SNR case. The columns labeled "MC" present data from a conventional stream, unbiased Monte Carlo simulation. That is, data are continuously injected into the channel and P_b is estimated by the relative frequency of bit decision errors. (We omit a description of the algorithm because it is straightforward. The only thing that requires special caution is keeping the timing and the sampling indices straight because, for both convenience and efficiency, the signal and noise samples are generated by long blocks instead of one symbol interval at a time.) A Monte Carlo simulation is terminated whenever 100 bit errors are detected which amounts to roughly a 10% relative precision. These Monte Carlo simulation results can serve as baseline data for comparison with our importance sampling data. The importance sampling simulation is run until a 10% relative precision is achieved for the P_b estimate. The CPU-time data are for a Sun SPARC 1 workstation. We note that our importance sampling estimator is very accurate and the computational cost is stable with respect to P_b. In contrast, the

Table 6.1 MSK P_b estimates and CPU time comparisons.

SNR	Estimates of P_b		CPU Time (Min.)	
(dB)	MC	IS	MC	IS
9.0	3.71×10^{-3}	3.56×10^{-3}	50	18
9.5	2.12×10^{-3}	2.00×10^{-3}	87	20
10.0	1.07×10^{-3}	1.04×10^{-3}	173	24
10.5	5.43×10^{-4}	5.07×10^{-4}	345	23
11.0	2.77×10^{-4}	2.39×10^{-4}	674	27
11.5	1.12×10^{-4}	1.02×10^{-4}	1674	31
12.0	*	3.89×10^{-5}	*	38
13.0	*	4.22×10^{-6}	*	55
14.0	*	2.67×10^{-7}	*	45

CPU time required for the Monte Carlo estimator increases with $1/P_b$. Figure 6.5 shows P_b vs. downlink SNR curves with the uplink SNR as a parameter. As the downlink SNR $\rightarrow \infty$, P_b becomes dominated by the uplink noise and approaching a constant.

6.3 A TCM Code on the Satellite Channel

In this section we estimate P_b for the quasi-regular TCM code of Figure 5.7 operating on the satellite channel of Figure 3.7.

6.3.1 The signal vector $\mathbf{X}$ and the noise vector $\mathbf{Y}$

The unfiltered transmitted 8-PSK signal at the satellite transponder input is of the form

$$s(t) = s^I(t)\cos(2\pi f_c t) - s^Q(t)\sin(2\pi f_c t).$$

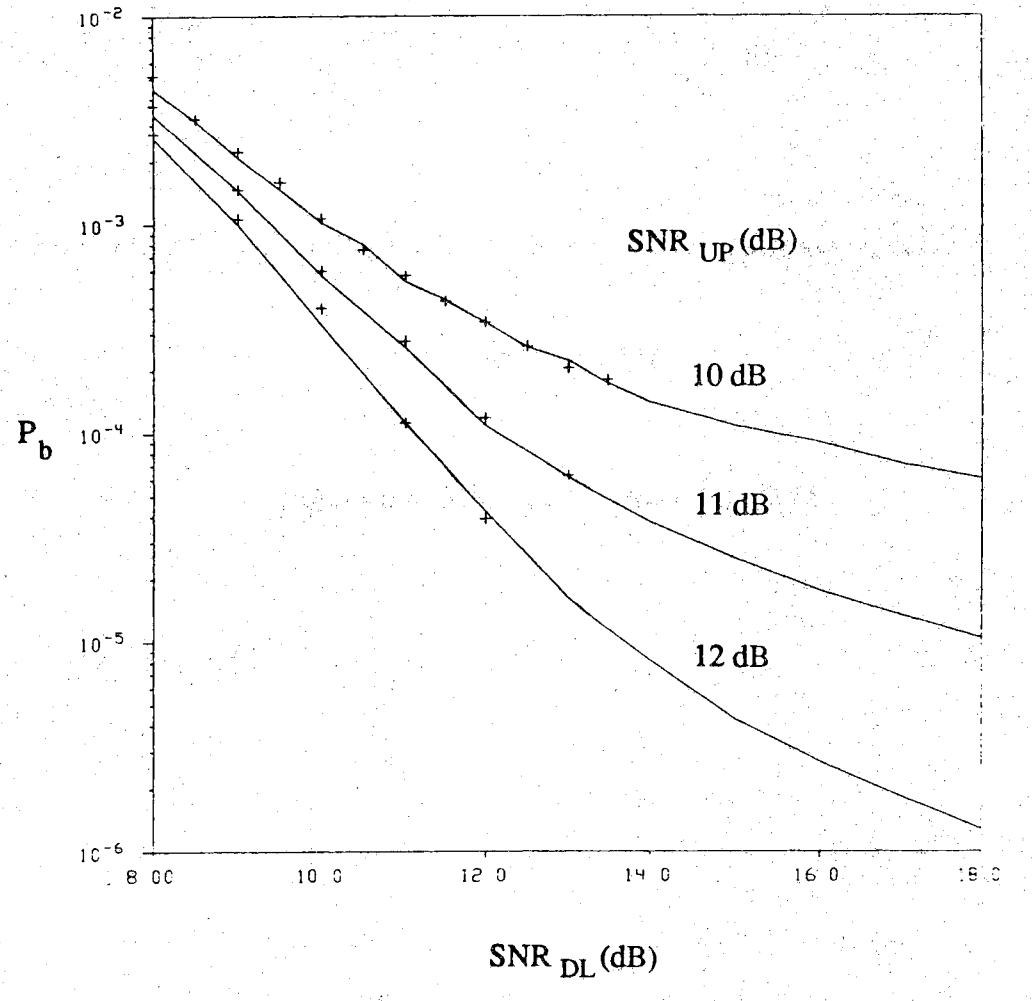


Figure 6.5 MSK P_b vs. downlink E_b/N_0.

The information bearing lowpass quadrature signals are $s^I(t) = \sum_k s(X_k)^I p(t - kT)$ and $s^Q(t) = \sum_k s(X_k)^Q p(t - kT)$, where $s(X_k) = (s(X_k)^I, s(X_k)^Q)$ is the complex 8-PSK signal pulse amplitude selected by the signal selector X_k (see Figure 3.5), $p(t)$ is the signaling pulse and T is the symbol (signaling) interval. For this code rate $R = 2/3$, 8-PSK code, we have $T = 2T_b$. The discretizations of the transmitted signal and the channel are identical to that described in Section 6.2.1. We will also use same notations.

The event simulation method for trellis codes presented in Section 5.2 will be used. In particular, the estimator is of the form (5.15) and (5.16) which for convenience we rewrite here

$$\hat{P}_b = \frac{1}{L_X} \sum_{l=1}^{L_X} \hat{P}\left(\tilde{\mathbf{X}}^{(\ell)} + \mathbf{E}^{(\ell)} | \tilde{\mathbf{X}}^{(\ell)}\right) w_{\mathbf{E},\tilde{\mathbf{X}}}\left(\mathbf{E}^{(\ell)}, \tilde{\mathbf{X}}^{(\ell)}\right), \tag{6.44}$$

where the estimate of the conditional first event error probability given $(\mathbf{E}^{(\ell)}, \tilde{\mathbf{X}}^{(\ell)})$, $P(\tilde{\mathbf{X}}^{(\ell)} + \mathbf{E}^{(\ell)} | \tilde{\mathbf{X}}^{(\ell)})$, is

$$\hat{P}(\tilde{\mathbf{X}}^{(\ell)} + \mathbf{E}^{(\ell)} | \tilde{\mathbf{X}}^{(\ell)}) = \frac{1}{L_Y} \sum_{\ell'=1}^{L_Y} 1_{\{\mathbf{E}^{(\ell)} \text{ is decoded}\}}\left(\tilde{\mathbf{X}}^{(\ell)}, \mathbf{Y}^{(\ell,\ell')}\right) \times$$

$$w_{\mathbf{Y}|\mathbf{E},\tilde{\mathbf{X}}}\left(\mathbf{Y}^{(\ell,\ell')} | \mathbf{E}^{(\ell)}, \tilde{\mathbf{X}}^{(\ell)}\right), \tag{6.45}$$

and

$$w_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e}, \tilde{\mathbf{x}}) = \frac{n_b(\mathbf{e})\, 2^{-b(\ell(\mathbf{e})+M)}}{b\, P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e}, \tilde{\mathbf{x}})}$$

$$w_{\mathbf{Y}|\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{y}|\mathbf{e}, \tilde{\mathbf{x}}) = \frac{f_{\mathbf{Y}}(\mathbf{y})}{f^*_{\mathbf{Y}|\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{y}|\mathbf{e}, \tilde{\mathbf{x}})}.$$

are the importance sampling weighting functions for the signal and noise respectively.

The core of the above method is that we choose a pair of signal selector sequence $\tilde{\mathbf{x}} = (\mathbf{x}^-, \mathbf{x}, \mathbf{x}^+)$ and error sequence $\mathbf{e}$ then find the optimal mean-translation biasing for the distribution of the noise vector $\mathbf{Y}$. The objective is to estimate the probability of a specific error event that $\tilde{\mathbf{x}} + \mathbf{e}$ is decoded while $\tilde{\mathbf{x}}$ is the transmitted signal selector sequence, i.e., $P(\tilde{\mathbf{x}} + \mathbf{e}|\tilde{\mathbf{x}})$. As in the MSK case, let us first clarify what are $\tilde{\mathbf{x}}$ and $\mathbf{Y}$.

For a fixed $\mathbf{e}$, recall that $\mathbf{X}^- = (X_{1-M^-}, \ldots, X_0)$, $\mathbf{X} = (X_1, \ldots, X_{\ell(\mathbf{e})})$ and $\mathbf{X}^+ = (X_{\ell(\mathbf{e})+1}, \ldots, x_{\ell(\mathbf{e})+M^+})$ from Section 5.2 where $\ell(\mathbf{e})$ is the length of $\mathbf{e}$. That is, $\tilde{\mathbf{x}}$ is the transmitted signal selector sequence $\mathbf{x}$ plus its adjacent signal selectors. To understand this and to find M^+ and M^-, consider the signal flow diagram in Figure 6.6. By the definition of signal selector error sequence $\mathbf{e}$, the trellis path representing $\tilde{\mathbf{x}} + \mathbf{e}$ diverges from the path of $\tilde{\mathbf{x}}$ at stage $k = 0$ and later remerges for the first time at stage $k = \ell(\mathbf{e})$. Now, if the Viterbi decoder is to choose between $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}} + \mathbf{e}$ (assuming for the moment they are the only decoding options), the error event $\tilde{\mathbf{x}} + \mathbf{e}$ will be decoded if the decision statistic D is

$$\begin{aligned} D &= \sum_{k=1}^{\ell(\mathbf{e})} Re[s(x_k + e_k)^* r_k] - Re[s(x_k)^* r_k] \\ &= \sum_{k=1}^{\ell(\mathbf{e})} Re[(s(x_k + e_k) - s(x_k))^* r_k] \geq 0 \end{aligned} \tag{6.46}$$

where r_k is the k'th demodulator sampler output. Note that the summation is only over the range of the error event length $\ell(\mathbf{e})$ because $s(x_k + e_k) = s(x_k)$ for $k < 1$ and $k > \ell(\mathbf{e})$.

Therefore, D depends on r_k for $1 \leq k \leq \ell(\mathbf{e})$. Let k' still be the sample numbering index as in the last section and the demodulator sampler takes the K'th sample of a transmitted pulse, $0 \leq K \leq N_s - 1$. Also, we align the time axis and the sampling axis such that the sample at time $t = 0$ (stage $k = 0$) corresponds to the 0'th sample. Hence, r_k is the $k' = K + (k-1)N_s$ sample of the demodulator output (before the sampler) and we can write $r_k = rn_{K+(k-1)N_s} + rx_{K+(k-1)N_s}$, where $rn_{K+(k-1)N_s}$ is the contribution from the $h_4(t)$-filtered downlink noise and $rx_{K+(k-1)N_s}$ is from the $h_{34}(t)$-filtered TWTA outputs. See the lower part of Figure 6.6. It is then clear that the downlink noise samples which have impacts on the decision statistic D are $Y_{d,k'}, K - m_4^+ \leq k' \leq K + (\ell(\mathbf{e}) - 1)N_s - m_4^-$. Similarly, the decision statistic depends on the TWTA output complex samples $\tilde{z}_{k'}, K - m_{34}^+ \leq k' \leq K + (\ell(\mathbf{e}) - 1)N_s - m_{34}^-$. The dimension of the TWTA output vector $\tilde{\mathbf{z}}$ is the same as the input $\mathbf{z}$ because the TWTA is memoryless. Each TWTA input sample $z_{k'}$ is in turn the output of $h_2(t)$-filtered

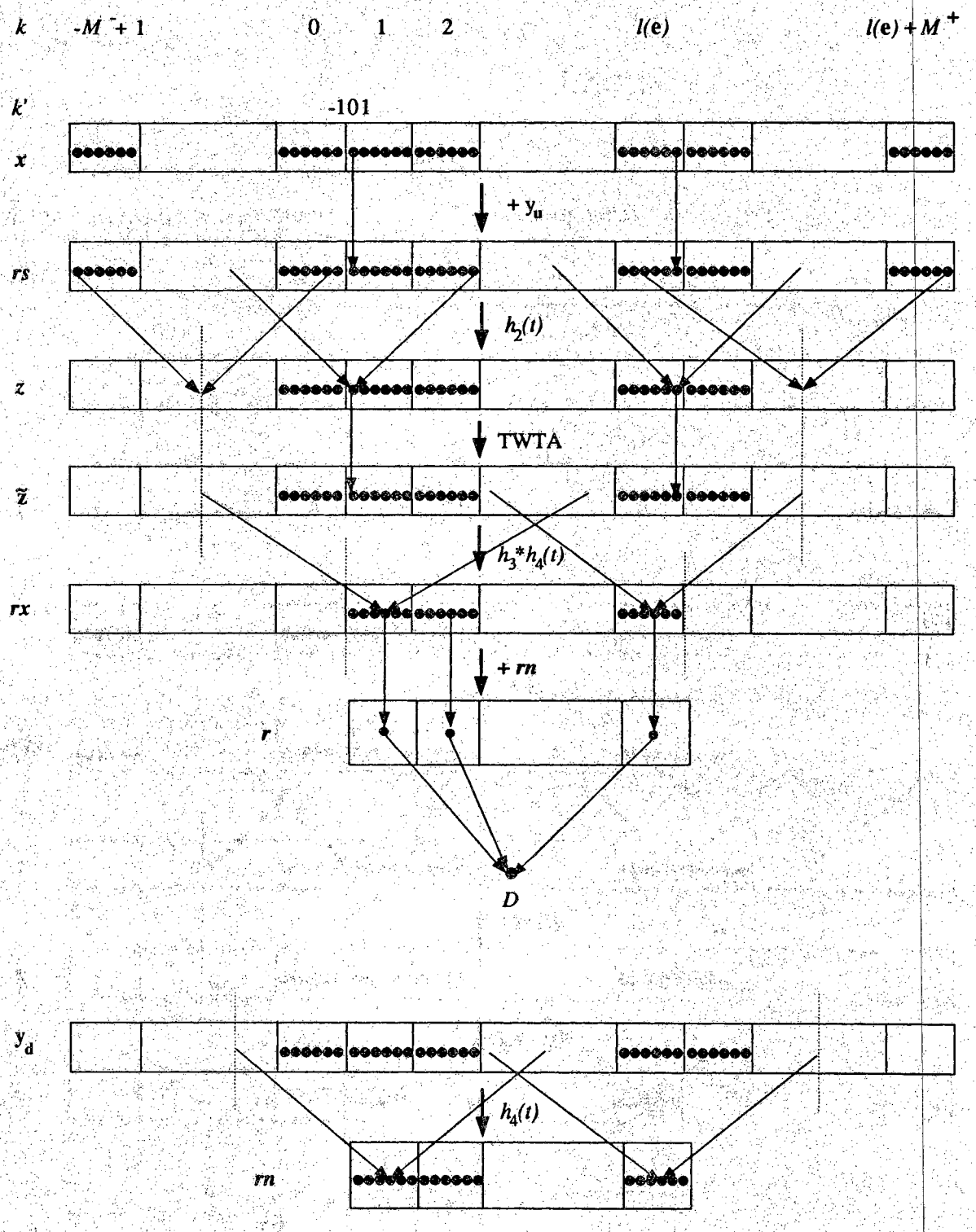


Figure 6.6 TCM signal flow diagram in the satellite channel.

transmitted signal samples plus the uplink noise samples. Consequently, the relevant transmitted signal and uplink noise samples are $s_{k'}$ and $Y_{u,k'}$ for $K - m_{34}^{+} - m_{2}^{+} \leq k' \leq K + (\ell(\mathbf{e}) - 1)N_s - m_{34}^{-} - m_{2}^{-}$ respectively. It is not difficult to see that Figure 6.6 is really an expanded version of Figure 6.3; or conversely Figure 6.3 is a special case of Figure 6.6 with $\ell(\mathbf{e}) = 1$.

In summary, the noise vector $\mathbf{Y}$ in this problem is $[\mathbf{Y}_u^t, \mathbf{Y}_d^t]^t$ where

$$\mathbf{Y}_u = \left(Y_{u,k'}^{I}, Y_{u,k'}^{Q}\right)_{k'=K-m_{34}^{+}-m_{2}^{+}}^{K+(\ell(\mathbf{e})-1)N_s-m_{34}^{-}-m_{2}^{-}} \tag{6.47}$$

and

$$\mathbf{Y}_d = \left(Y_{d,k'}^{I}, Y_{d,k'}^{Q}\right)_{k'=K-m_{4}^{+}}^{K+(\ell(\mathbf{e})-1)N_s-m_{4}^{-}} . \tag{6.48}$$

The signal selectors which need to be considered are

$$\mathbf{X} = (X_k)\,,\ 1 - \left\lceil \frac{m_{34}^{+}+m_{2}^{+}-K}{N_s} \right\rceil \leq k \leq \ell(\mathbf{e}) + \left\lceil \frac{|m_{34}^{-}+m_{2}^{-}|-N_s+K}{N_s} \right\rceil . \tag{6.49}$$

Thus, $M^{-} = \left\lceil (m_{34}^{+} + m_{2}^{+} - K)/N_s \right\rceil$, $M^{+} = \left\lceil (|m_{34}^{-} + m_{2}^{-}| - N_s + K)/N_s \right\rceil$, and the total signal memory length $M = M^{-} + M^{+}$ is a function of the filter memory and where the demodulator output samples are taken.

6.3.2 Finding the Dominating Point

As far as finding the dominating point is concerned, as the signal flow diagrams Figure 6.3 and Figure 6.6 indicate, the derivation in Section 6.2.2 for the MSK is in fact a special case of that for a trellis code with $\ell(\mathbf{e}) = 1$. We now show this. The test statistic D, which is the difference between the metrics of the error event path $\tilde{\mathbf{x}} + \mathbf{e}$ and the correct path $\tilde{\mathbf{x}}$, is given in (6.46). That is,

$$\begin{aligned} D &= \sum_{k=1}^{\ell(\mathbf{e})} r_k^I d_k^I + r_k^Q d_k^Q \\ &= \sum_{k=1}^{\ell(\mathbf{e})} \left(rx_{K+(k-1)N_s}^{I} + rn_{K+(k-1)N_s}^{I}\right) d_k^I + \left(rx_{K+(k-1)N_s}^{Q} + rn_{K+(k-1)N_s}^{Q}\right) d_k^Q \\ &= \xi(\mathbf{y}_u) + \mathbf{a}^t \mathbf{y}_d \end{aligned}$$

where $d_k = (d_k^I, d_k^Q) = \left(s(x_k + e_k)^I - s(x_k)^I, s(x_k + e_k)^Q - s(x_k)^Q\right)$. Therefore, we have

$$\mathbf{a}^t \mathbf{y}_d = \sum_{k=1}^{\ell(\mathbf{e})} rn^I_{K+(k-1)N_s} d_k^I + rn^Q_{K+(k-1)N_s} d_k^Q \tag{6.50}$$

and

$$\xi(\mathbf{y}_u) = \sum_{k=1}^{\ell(\mathbf{e})} rx^I_{K+(k-1)N_s} d_k^I + rx^Q_{K+(k-1)N_s} d_k^Q. \tag{6.51}$$

As before, in order to find the dominating point, we need to know $\mathbf{a}$, $\psi(\cdot)$ and $\nabla\psi(\cdot)$. We first show how to find the vector $\mathbf{a}$. From (6.50) and letting $k'(k) = K + (k-1)N_s$, it can be shown that

$$\mathbf{a}^t \mathbf{y}_d = \sum_{k=1}^{\ell(\mathbf{e})} \sum_{i=m_4^-}^{m_4^+} \left(h^I_{4,i} d_k^I + h^Q_{4,i} d_k^Q\right) y^I_{d,k'(k)-i} + \left(h^I_{4,i} d_k^Q - h^Q_{4,i} d_k^I\right) y^Q_{d,k'(k)-i}.$$

The objective is to express $\mathbf{a}^t \mathbf{y}_d$ as

$$\mathbf{a}^t \mathbf{y}_d = \sum_{k'} a^I_{k'} y^I_{d,k'} + a^Q_{k'} y^Q_{d,k'}. \tag{6.52}$$

First we consider $a^I_{k'}$, $K - m_4^+ \le k' \le K + (\ell(\mathbf{e})-1)N_s - m_4^-$. After a change of variable, we get

$$\sum_{k=1}^{\ell(\mathbf{e})} \sum_{k'=k'(k)-m_4^+}^{k'(k)-m_4^-} \left(h^I_{4,k'(k)-k'} d_k^I + h^Q_{4,k'(k)-k'} d_k^Q\right) y^I_{d,k'} = \sum_{k'} a^I_{k'} y^I_{d,k'}$$

It is then found that

$$a^I_{k'} = \sum_{k=1}^{\ell(\mathbf{e})} h^I_{4,K+(k-1)N_s-k'} d_k^I + h^Q_{4,K+(k-1)N_s-k'} d_k^Q \tag{6.53}$$

where terms in the summation are omitted if $K+(k-1)N_s-k' < m_4^-$ or $K+(k-1)N_s-k' > m_4^+$. Similarly, $a^Q_{k'}$ is

$$a^Q_{k'} = \sum_{k=1}^{\ell(\mathbf{e})} h^I_{4,K+(k-1)N_s-k'} d_k^Q - h^Q_{4,K+(k-1)N_s-k'} d_k^I \tag{6.54}$$

for $K - m_4^+ \le k' \le (\ell(\mathbf{e})-1)N_s + K - m_4^-$. With $\ell(\mathbf{e}) = 1$, $d_1^I = 1$ and $d_1^I = 0$, (6.53) and (6.54) reduce to (6.18) and (6.19). The uncoded MSK can be considered as having $s(\mathbf{x}) = s(0) = -1$, $s(\mathbf{x}+\mathbf{e}) = s(1) = 1$ and $\mathbf{e}$ is a one-branch error sequence.

Next we find $\psi(\mathbf{y}_u)$. Following the above derivation for $\mathbf{a}$, we can rewrite (6.51) in a form similar to (6.52). That is,

$$\psi(\mathbf{y}_u) = \sum_{k'=K-m_{34}^{+}}^{K+(\ell(\mathbf{e})-1)N_s-m_{34}^{-}} \tilde{z}_{k'}^{I} b_{k'}^{I} + \tilde{z}_{k'}^{Q} b_{k'}^{Q} \tag{6.55}$$

where

$$b_{k'}^{I} = \sum_{k=1}^{\ell(\mathbf{e})} h_{34,K+(k-1)N_s-k'}^{I} d_k^{I} + h_{34,K+(k-1)N_s-k'}^{Q} d_k^{Q} \tag{6.56}$$

$$b_{k'}^{Q} = \sum_{k=1}^{\ell(\mathbf{e})} h_{34,K+(k-1)N_s-k'}^{I} d_k^{Q} - h_{34,K+(k-1)N_s-k'}^{Q} d_k^{I}. \tag{6.57}$$

Similarly, terms are omitted in the summations of (6.56) and (6.57) if $K+(k-1)N_s-k' < m_{34}^{-}$ or $K+(k-1)N_s-k' > m_{34}^{+}$. The relations between $z_{k'}$ and $\tilde{z}_{k'}$ are given in (6.21) and (6.22) or (6.23) and (6.24) if using spline functions. The uplink noise vector $\mathbf{y}_u$ and the TWTA input vector $\mathbf{z}$ are also related by the same formulas as in the MSK case, i.e., Equations (6.25) – (6.28), except now the dimension of the uplink noise vector $\mathbf{y}_u = (y_{u,k}^{I}, y_{u,k}^{Q})_k$ and the transmitted signal vector $\mathbf{s} = (s_k)_k$ is $K-m_{34}^{+}-m_2^{+} \le k' \le K+(\ell(\mathbf{e})-1)N_s-m_{34}^{-}-m_2^{-}$.

In summary, the uplink noise transfer function is

$$\begin{aligned}\psi(\mathbf{y}_u) &= \sum_{k'=K-m_{34}^{+}}^{K+(\ell(\mathbf{e})-1)N_s-m_{34}^{-}} \tilde{z}_{k'}^{I} b_{k'}^{I} + \tilde{z}_{k'}^{Q} b_{k'}^{Q} \\ &= \sum_{k'=K-m_{34}^{+}}^{K+(\ell(\mathbf{e})-1)N_s-m_{34}^{-}} \left(\rho_1 b_{k'}^{I} + \rho_2 b_{k'}^{Q}\right) z_{k'}^{I} + \left(\rho_1 b_{k'}^{Q} - \rho_2 b_{k'}^{I}\right) z_{k'}^{Q},\end{aligned} \tag{6.58}$$

where ρ_1 and ρ_2 are defined in (6.30) – (6.33). Equation (6.58) reduces to the uncoded MSK case (6.29) for $\ell(\mathbf{e}) = 1$.

The computation of $\nabla\psi(\mathbf{y}_u) = (\partial f/\partial y_{u,k}^{I}, \partial f/\partial y_{u,k}^{Q})$ is also very similar to that for the uncoded MSK system except increased dimensions in the signal and up/downlink noise vectors. Hence we only state the result. For $K-m_{34}^{+}-m_2^{+} \le k' \le K+(\ell(\mathbf{e})-1)N_s-m_{34}^{-}-m_2^{-}$,

$$\frac{\partial f}{\partial y_{u,k'}^{I}} = \sum_{i=k'+m_2^{-}}^{k'+m_2^{+}} w_1 h_{2,i-k'}^{I} + w_2 h_{2,i-k'}^{Q} \tag{6.59}$$

$$\frac{\partial f}{\partial y_{u,k'}^{Q}} = \sum_{i=k'+m_2^{-}}^{k'+m_2^{+}} -w_1 h_{2,i-k'}^{Q} + w_2 h_{2,i-k'}^{I} \tag{6.60}$$

where terms are omitted in the summations if $i < K - m_{34}^{+}$ or $i > K + \ell(\mathbf{e}) - 1)N_s - m_{34}^{-}$, and

$$w_1 = b_i^I \frac{\partial \tilde{z}_i^I}{\partial z_i^I} + b_i^Q \frac{\partial \tilde{z}_i^Q}{\partial z_i^I}$$

$$w_2 = b_i^I \frac{\partial \tilde{z}_i^I}{\partial z_i^Q} + b_i^Q \frac{\partial \tilde{z}_i^Q}{\partial z_i^Q}.$$

The four derivatives $\partial \tilde{z}_i^I / \partial z_i^I$, $\partial \tilde{z}_i^I / \partial z_i^Q$, $\partial \tilde{z}_i^Q / \partial z_i^I$, and $\partial \tilde{z}_i^Q / \partial z_i^Q$ are given in (6.38) – (6.41).

6.3.3 Sampling $\mathbf{E}$ and $\mathbf{X}$

The event simulation estimator (6.44) calls for the joint sampling of the pair $(\mathbf{e}, \tilde{\mathbf{x}})$ according to the signal biasing density $P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e}, \tilde{\mathbf{x}})$. Recall the notation $\tilde{\mathbf{x}} = (\mathbf{x}^-, \mathbf{x}, \mathbf{x}^+)$ where $\mathbf{x}^-$ and $\mathbf{x}^+$, whose lengths are computed by (6.49), represent the ISI. We will first derive the criterion for a "good" design of $P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e}, \tilde{\mathbf{x}})$ and then propose a sampling scheme which satisfies this criterion.

Before proceeding, let us first recall the fundamental principle of importance sampling as spelled out in (2.10). That is, to estimate a quantity $\alpha = \mathrm{E}[g(\mathbf{U})]$, the optimal simulation density is $f^*(\mathbf{u}) \propto g(\mathbf{u}) f(\mathbf{u})$. The optimal mean-translation Gaussian noise simulation density satisfies this proportionality. Also, we have used it to design the signal biasing in Section 4.3 and found that $P^*_{\mathbf{X}}(\mathbf{x}) \propto \beta(\mathbf{x}) P_{\mathbf{X}}(\mathbf{x})$ where $\beta(\mathbf{x}) = \mathrm{E}[g(\mathbf{X}, \mathbf{Y})|\mathbf{x}]$. The signal biasing design in Section 6.2.3 for the MSk example follows precisely this rule.

Now let us apply this fundamental principle to the discrete sum of (5.14) with $\mathbf{U} = (\mathbf{E}, \tilde{\mathbf{X}})$. Suppose that we have an exact conditional estimate, that is, $\hat{P}(\tilde{\mathbf{x}} + \mathbf{e}|\tilde{\mathbf{x}}) = P(\tilde{\mathbf{x}} + \mathbf{e}|\tilde{\mathbf{x}})$. Then from (5.14) it is apparent that the optimal choice of $P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e}, \tilde{\mathbf{x}})$ is

$$P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e}, \tilde{\mathbf{x}}) \propto n_b(\mathbf{e})\, P(\tilde{\mathbf{x}} + \mathbf{e}|\tilde{\mathbf{x}})\, 2^{-b\ell(\mathbf{e})}. \tag{6.61}$$

One reason that we cannot implement this sampling distribution is that we do not have an exact formula for $P(\tilde{\mathbf{x}} + \mathbf{e}|\tilde{\mathbf{x}})$ and can only empirically estimate it by (6.45).

However, for the purpose of designing the signal biasing, we can approximate $P(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}})$ using the Euclidean distance information. Recalling the union bound $P(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}}) \leq Q\left(d(\tilde{\mathbf{x}},\tilde{\mathbf{x}}+\mathbf{e})/2\sigma\right)$ (assuming $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}+\mathbf{e}$ are the only decoding options) in Section 3.2.2 and using $Q(x) \leq \exp(-x^2/2)$, we have $P(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}}) \leq \exp(-d^2(\mathbf{x},\mathbf{x}+\mathbf{e})/8\sigma^2)$. (Note that $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}+\mathbf{e}$ differ only in the part of $\mathbf{x}$.) As previously noted, this bound holds only for the ideal AWGN channel. Nonetheless, provided that the ISI and nonlinearity are not too severe, to some degree $P(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}})$ should be roughly proportional to $\exp(-d^2(\mathbf{x},\mathbf{x}+\mathbf{e})/8\sigma^2)$. Hence, neglecting the less important factor $n_b(\mathbf{e})$ in (6.61), we will design a simulation distribution such that

$$P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e},\mathbf{x}) \;\propto\; 2^{-b\ell(\mathbf{e})}\,\exp\left(-\alpha d^2(\mathbf{x},\mathbf{x}+\mathbf{e})\right) \tag{6.62}$$

where α is a free parameter that will be roughly determined by SNR.

Next we describe a two-step sampling scheme for implementing a sampling distribution that does satisfy the proportionality (6.62). The basic idea is to first sample $\mathbf{E}$ from a precompiled list of error sequences and then sample $\tilde{\mathbf{X}}$ from a conditional distribution. Hence, the sampling distribution will be of the form $P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e},\tilde{\mathbf{x}}) = P^*_{\tilde{\mathbf{X}}|\mathbf{E}}(\tilde{\mathbf{x}}|\mathbf{e})P^*_{\mathbf{E}}(\mathbf{e})$.

Given a sampled error event $\mathbf{E}=\mathbf{e}$, the correct path $\tilde{\mathbf{X}}$ is sampled as that described in Section 5.3.2. A general formulation is presented below. First a random initial state is generated for stage $k=0$. The branches X_k, $k=1,..,\ell(\mathbf{e})$, are then sampled sequentially as a "biased random walk" through the trellis. At each stage there are 2^b branching possibilities. When $e_k \notin \mathcal{E}$, the 2^b branching possibilities are given equal probability, so the random walk is not biased on these branches. (Recall that $\mathcal{E}$ is the set of single branch signal selector errors e such that $d(x,x+e) > w(e)$ for some x, i.e., their distance polynomials are not monomials. For example, for the 8-PSK signal set with the natural mapping, we have $\mathcal{E} = \{(011),(111)\}$.) Now suppose that $e_k \in \mathcal{E}$ and that at stage $k-1$ the sampled path $\mathbf{X}$ is in state s and path $\mathbf{X}+\mathbf{e}$ is in state s'. Notice that the set of branching possibilities for X_k is indicated by the distance polynomial $p_{s,s',e_k}(z)$ of e_k as defined in (3.6). Let us denote

this set of branching possibilities $\mathcal{X}(s,s',e_k)$. We bias these branching probabilities in proportion to $\exp(-\alpha d^2(x,x+e_k))$. Hence, after normalizing to a probability distribution, the probability of selecting a particular $x \in \mathcal{X}(s,s',e_k)$ is

$$\frac{\exp(-\alpha d^2(x,x+e_k))}{\sum_{x' \in \mathcal{X}(s,s',e_k)} \exp(-\alpha d^2(x',x'+e_k))} = \frac{\exp(-\alpha d^2(x,x+e_k))}{2^b\, p_{s,s',e_k}(e^{-\alpha})}. \tag{6.63}$$

Note that $P(x|s) = 2^{-b}$ in (3.6). For a quasi-regular code the normalizing factor becomes $\left(2^b p_{e_k}(e^{-\alpha})\right)^{-1}$ which does not depend on the particular pair of states (s,s'). This is the key property of quasi-regular codes that allows us to ultimately realize the proportionality (6.62) via a sequential sampling procedure for $\mathbf{X}$. For non-quasi-regular codes, the samplg will be state-dependent. For the 8-PSK signal set with the natural mapping, (6.63) results in

$$P^*_{X_k|E_k}(x_k|e_k) = \begin{cases} 2^{-b} = P_{X_k|E_k}(x_k|e_k), & e_k \notin \mathcal{E} \\ p^*, & e_k \in \mathcal{E}, d(x_k, x_k+e_k) = \delta_0 \\ q^*, & e_k \in \mathcal{E}, d(x_k, x_k+e_k) = \delta_2 \end{cases}$$

which is precisely what we have obtained in (5.35) for the signal biasing of this TCM code operating on the AWGN channel. In that case, we have $\alpha = 1/8\sigma^2$. Since the relationship between the union bound and the probability $P(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}})$ becomes weaker when the ISI and nonlinearilty are present, we should have $\alpha \leq 1/8\sigma^2$ in the satellite channel case.

The sampling of $\tilde{\mathbf{X}}$ is completed by adding the extensions $\mathbf{X}^-$ and $\mathbf{X}^+$ which are sampled sequentially as unbiased random walks. Therefore, this procedure implements the conditional sampling distribution

$$P^*_{\tilde{\mathbf{X}}|\mathbf{E}}(\tilde{\mathbf{x}}|\mathbf{e}) = 2^{-b(\ell(\mathbf{e})+M)} \prod_{1 \leq k \leq \ell(\mathbf{e}); e_k \in \mathcal{E}} \frac{\exp(-\alpha d^2(x_k, x_k+e_k))}{p_{e_k}(e^{-\alpha})}. \tag{6.64}$$

To complete the procedure we must describe the sampling of the random error event $\mathbf{E}$. We wish to realize a distribution $P^*_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e},\tilde{\mathbf{x}}) = P^*_{\tilde{\mathbf{X}}|\mathbf{E}}(\tilde{\mathbf{x}}|\mathbf{e}) P^*_{\mathbf{E}}(\mathbf{e})$ which satisfies the proportionality (6.62). Using $d_w^2(\mathbf{e}) = \sum_{k=1}^{\ell(\mathbf{e})} d_w^2(e_k)$ and $d^2(x_k, x_k+e_k) = d_w^2(e_k)$ whenever $e_k \notin \mathcal{E}$, it follows from (6.62) and (6.64) that

$$P_{\mathbf{E}}^{*}(\mathbf{e}) = K\left[\prod_{k;e_k\in E} p_{e_k}(e^{-\alpha})\right]\exp\left(-\alpha\left(d_w^2(\mathbf{e}) - \sum_{k;e_k\in\mathcal{E}} d_w^2(e_k)\right)\right) \tag{6.65}$$

where K is a constant normalizing factor. To implement (6.65), we simply sampled $\mathbf{E}$ from a precompiled list that includes all error sequences with minimum distance $d_w^2(\mathbf{e}) \leq d_{max}^2$. In the simulation, we use the list of 49,506 $\mathbf{e}$'s with $d_w^2(\mathbf{e}) \leq 10$ which we have used to compute Table 5.2. Since this is a subset of the infinite set $\mathcal{C}$ of all possible error sequences, there will be a truncation bias as mentioned previously. However, we can expect that this truncation bias will be negligible at moderate-to-high SNR's, and we will verify this in the next subsection's simulation data.

6.3.4 Simulation Algorithm, Parameters and Results

The simulation algorithm can be described as follows.

Quasi-Regular TCM Codes IS Simulation Algorithm:

- Input: Convolutional encoder configuration, signal mapping function, error sequences list, N_s, product BT, uplink SNR, TWTA IBO, downlink SNR, demodulator sampling time K, L_y, dominating point initial value ϕ_0 and signal biasing factor α.
- Output: $\hat{P}_b^*$, its relative precision and CPU time.
- Method:
  1. Construct tables containing trellis diagram information.
  2. Compute noise variance, random walk parameter p^*.
  3. Read error sequence $\mathbf{e}$, length $\ell(\mathbf{e})$, weight $n_b(\mathbf{e})$ and worst-case distance $d_w(\mathbf{e})$ from the data file obtained from the RC Algorithm.
  4. Compute $P_{\mathbf{E}}^{*}(\mathbf{e})$.
  5. Sample $\mathbf{E}^{(\ell)}$. Select a random initial state and sample the correct path $\tilde{\mathbf{X}}^{(\ell)}$. Compute the signal biasing weight $w_{\mathbf{E},\tilde{\mathbf{X}}}(\mathbf{e},\tilde{\mathbf{x}})$. Put $\ell' = 0$.

6. $\ell' = \ell' + 1$. If $\ell' > L_Y$, compute $\hat{P}(\tilde{\mathbf{X}}^{(\ell)} + \mathbf{E}^{(\ell)} | \tilde{\mathbf{X}}^{(\ell)})$ then go to 5.

7. Compute the dominating point.

8. Perform the Viterbi decoder error event simulation. When an error event is decoded, update simulation data accumulators then go to 6.

As in the case of MSK complex baseband simulation of Section 6.2, we use $N_s = 16$ samples per symbol, brick-wall filters with bandwidth B ($BT = 1$), impulse response functions for all filters being truncated to be within $\pm 2T$ and the demodulator output sampling time $K = N_s/2$. The satellite TWTA input power backoff is IBO = 4 dB. A phase shift corresponding to the phase of the operating point is added to the impulse response function h_{34} to compensate for the phase rotation introduced by the TWTA AM/PM conversion. The variances of the uplink and downlink noise samples are computed as in the MSK case, that is, $\sigma^2 = N_s(N_0/2E_b)C$ where C is the carrier (pure sinusoid) power which is calculated from the TWTA IBO and the AM/AM conversion characteristic. The spline function approximation is used.

Table 6.2 shows simulation results for equal uplink and downlink SNR cases. Conventional stream, unbiased Monte Carlo simulations, whose data are presented in the columns labeled "MC," are performed for comparisons with our importance sampling algorithm. A Monte Carlo simulation is terminated whenever 100 independent error events are detected. This number translates into a 10% relative precision for the estimation of P_e. For a P_b estimate, the relative precision is slightly worse. As an aside, we remark that for the low SNR figures in the table our conventional Monte Carlo simulation did produce a significant number of error events with $d_w^2(\mathbf{e}) > 10$. Hence, a non-negligible truncation bias is to be expected for these low SNR values. We can reduce this bias by increasing the size of the list of error sequences. However, we note that the power of our importance sampling algorithm is in the moderate-to-high SNR region where truncation bias is negligible. Notice that the Monte Carlo simulation for 11 dB required 14 CPU days! The importance sampling simulation is run until a 10% relative precision is achieved for the P_b estimate. The CPU times data are for a Sun

Table 6.2 TCM P_b estimates and CPU time comparisons.

SNR	Estimates of P_b		CPU Time (Min.)	
(dB)	MC	IS	MC	IS
8.5	6.53×10^{-3}	4.38×10^{-3}	43	191
9.0	2.17×10^{-3}	1.37×10^{-3}	115	190
9.5	4.66×10^{-4}	4.20×10^{-4}	401	362
10.0	9.35×10^{-5}	8.85×10^{-5}	1,653	245
10.5	2.19×10^{-5}	2.03×10^{-5}	6,047	283
11.0	4.17×10^{-6}	4.17×10^{-6}	20,114	294
11.5	*	8.61×10^{-7}	*	347
12.0	*	1.45×10^{-7}	*	406

SPARC 1 workstation. Figure 6.7 shows P_b vs. downlink SNR curves with the uplink SNR fixed as a parameter. As the downlink SNR $\rightarrow \infty$, P_b becomes dominated by the uplink noise and approaching a constant.

Recall that our signal biasing scheme which achieves the proportionality (6.62) is based on the assumption that $P(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}})$ is roughly proportional to $\exp(-\alpha d^2(\mathbf{x}, \mathbf{x}+\mathbf{e}))$ where α is determined by some combination of the uplink and downlink SNR's. To test this hypothesis, we performed the following experiment. We randomly selected various pairs $(\mathbf{e}, \tilde{\mathbf{x}})$ and accurately estimated $P(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}})$ using the error event simulation algorithm with $L_Y = 2000$. This amount of simulation was sufficient to produce an empirical precision of less than 10%. The results of this experiment for SNR = 9/12 dB (uplink/downlink) are plotted in Figure 6.8. Since these probabilities are plotted on the logarithmic scale, we should expect to see a linear trend, and we do. The straight lines plotted in Figure 6.8 are least square fits of the exponential function $c\exp(-\hat{\alpha}d^2)$. For SNR = 9 db, $\hat{\alpha} = 1.92$, and for SNR = 12 dB, $\hat{\alpha} = 3.62$.

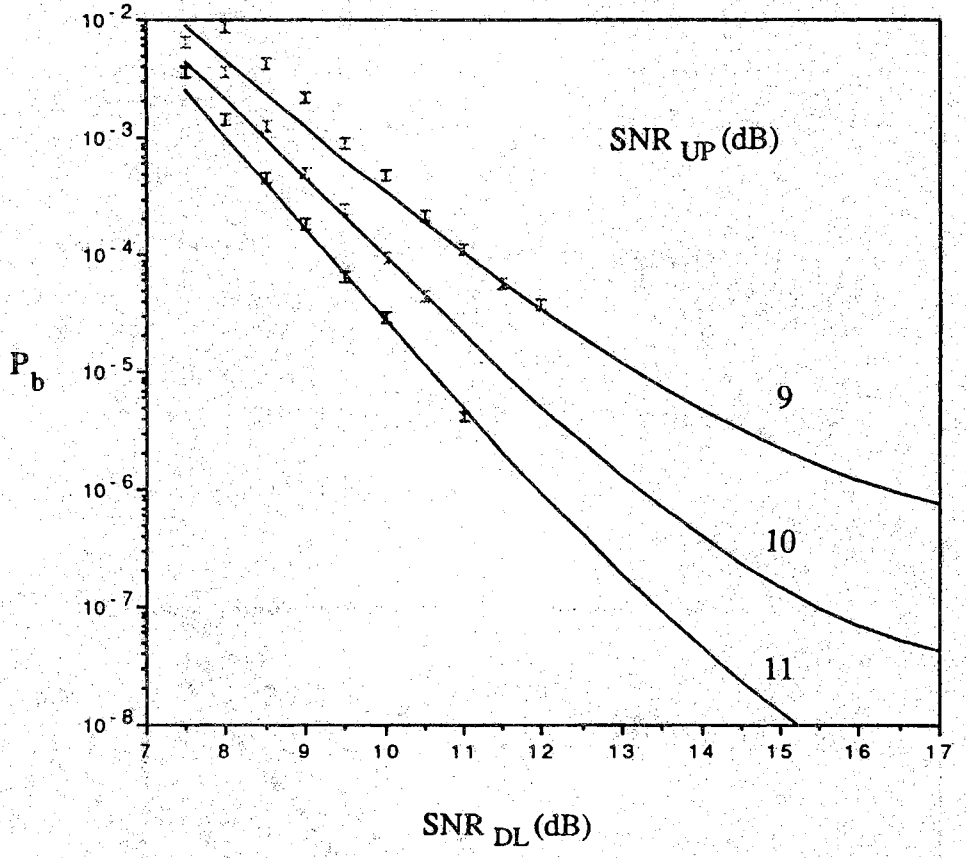


Figure 6.7 TCM P_b vs. downlink E_b/N_0.

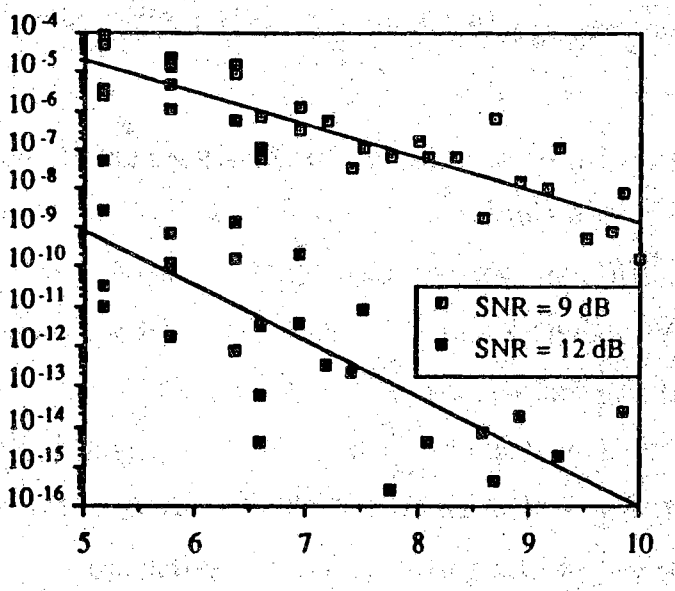


Figure 6.8 $P(\tilde{\mathbf{x}}+\mathbf{e}|\tilde{\mathbf{x}})$ vs. $d^2(\tilde{\mathbf{x}},\tilde{\mathbf{x}}+\mathbf{e})$ for some $(\mathbf{e},\tilde{\mathbf{x}})$.

In the overall simulation algorithm, α is a parameter which should be specified as a function of the SNR. To do this, we developed a formula $\alpha = f(\text{SNR}_{UL}, \text{SNR}_{DL}) =$ 0.8 times the value of α for a crude linear and memoryless AWGN approximation of the satellite channel model. The reason for the 0.8 "fudge factor" is as follows. Setting α on the small side of the (unknown) optimal value simply causes some less important terms (with higher distances) to be over-sampled. On the other hand, setting α too large will cause some important terms (with moderate distances) to be under-sampled. Hence, it is better to error by setting α to be less than the "optimal" value. For the two cases of Figure 6.8, our formula produced $\alpha = 1.59$ and $\alpha = 3.17$, respectively, for SNR = 9 and 12 dB. This compares well to the empirically measured optimal values $\hat{\alpha} = 1.92$ and $\hat{\alpha} = 3.62$ in the last paragraph.

7. CONCLUSIONS

We have presented a comprehensive development of the optimal Gaussian mean-translation biasing, event simulation and conditional importance sampling techniques and demonstrated their usage in efficiently estimating the bit error rate for digital communication systems. In particular, we study the simulation of uncoded and trellis-coded systems operating on linear memoryless channels and nonlinear channels with memory. Quasi-regularity of the code is utilized to facilitate the simulation as well as to design the signal biasing. Uncoded systems can be considered as special cases of coded systems from the viewpoint of our importance sampling algorithm. Simulation procedures and numerical results are presented which show the efficiency and accuracy of our algorithms. The techniques presented in this thesis can be readily applied to other system simulations as building blocks for their particular system configurations and channels.

Future work is foreseeable in many areas. We are interested in other coded systems whose decoding algorithms may not be describable by functionals, e.g., many block codes used in practice; or for which the system response function $\xi(\cdot)$ is discontinuous and thus not everywhere differentiable, e.g., Viterbi decoders with quantized demodulator outputs (instead of soft-decision decoders). The latter situation (discontinuity) also happens when the channel model contains components with discontinuous characteristics, e.g., a hard limiter. For these systems, we may have problems, both in the theory and in the implementation, in finding the dominating point because we may not be dealing with a moderately nonlinear system anymore.

Rotational-invariant and multi-dimensional TCM codes are useful in practical applications. It may happen that many of them are not quasi-regular and hence for

which we must explore their other properties to aid the simulation. Concatenated codes (combined block codes and trellis codes) are also widely used for which the implementation of the event simulation method and conditional importance sampling will be more complex than when only a single code is used. The event simulation method for some trellis codes is also yet to be further investigated, such as CPM (Continuous Phase Modulation) codes which do not have the linear convolutional encoder structure and thus we probably don't have a list of error sequences to sample from. For them, we may have to find a pure "random sampling" scheme which randomly generates the "important" information process and error event in each simulation run. Trellis codes whose distance spectra are computable and may even include the ISI effect as reported by Schlegel in [53] are also worth studying.

The type of carrier synchronization error for which our algorithm can readily handle is either a constant phase error or a phase jitter whose statistics are known. We would like to study the case where only the phase estimator is known, for example, a Costas loop or a nonlinear estimator as presented in [67]. An adaptive equalization at the receiver end is often a popular means to combat severe ISI. We have not look into such system configurations.

We have only considered intersymbol interference in our examples. The presence of ACI, CCI and/or multipath in the channel can often be modeled as an additive noise process. If the noise process is Gaussian, then we simply expand the dimension of the noise vector $\mathbf{Y}$. If the noise process is non-Gaussian or it is correlated with the signal process, which is often true for the multipath and CCI, the conditional importance sampling technique discussed in Section 4.3 can be used. That is, we will let the "signal input" vector $\mathbf{X}$ include samples of this noise process. In general, we can express $\mathbf{X}$ as $(\mathbf{X}_1, \mathbf{X}_2, \ldots)$. The task then is to design $f^*_{\mathbf{X}_1,\mathbf{X}_2,\ldots}(\mathbf{x}_1, \mathbf{x}_2, \ldots)$ or, we can appeal to successive conditioning again, e.g., if $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$, then $\alpha = \mathrm{E}[g(\mathbf{X}_1, \mathbf{X}_2, \mathbf{Y})] = \mathrm{E}[\mathrm{E}[\mathrm{E}[g(\mathbf{X}_1, \mathbf{X}_2, \mathbf{Y})|\mathbf{X}_1, \mathbf{X}_2]|\mathbf{X}_1]]$. This procedure is particularly useful if $\mathbf{X}_1$ and $\mathbf{X}_2$ are correlated. However, by doing this, we need to consider the joint minimization of the total computational cost as a function of L_Y, L_{X_1} and L_{X_2}.

as well as the design of simulation densities. The latter may become increasingly difficult as the number of levels of conditioning is increased.

Other channel and system models whose optimal simulation density designs are not covered by the method presented in this thesis can also be future research subjects. For example, in optical channels, the noise process is not Gaussian but often time-varying shot noise. In spread spectrum communications, the dominant distortion source is not the Gaussian noise but the channel interference and jamming. Finally, not much work has been done in applying the importance sampling to network-layer simulations which are event-driven (often with Poisson event arrival process) in contrast to the time-driven nature of data-link layer simulations. Network level analysis and simulations are becoming more important because of the increasing number, complexity and interconnection of data networks.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions.* Dover, New York, 1965.

[2] S. Benedetto, M. A. Marsan, G. Albertengo and E. Giachin, "Combined coding and modulation: Theory and applications," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 223–236, Mar. 1988.

[3] V. K. Bhargava, D. Haccoun, R. Matyas and P. P. Nuspl, *Digital Communications by Satellite.* Wiley, New York, 1981.

[4] J. A. Bucklew, P. Ney and J. S. Sadowsky, "Monte Carlo simulation and large deviations theory for uniformly recurrent Markov chains," *J. Appl. Proba.* , vol. 27, pp. 44–59, Mar. 1990.

[5] P. Bratley, B. L. Fox and L. E. Schrage, *A Guide to Simulation*, 2nd ed. Springer, New York, 1987.

[6] A. R. Calderbank and N. J. A. Sloane, "New trellis codes based on lattices and cosets," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 177–195, Mar. 1987.

[7] A. R. Calderbank and N. J. A. Sloane, "Four-dimensional modulation with an eight-state trellis code," *AT&T Tech. J.*, vol. 64, pp. 1005–1018, May–June 1985.

[8] M. Cedervall and R. Johannesson, "A fast algorithm for computing distance spectrum of convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-35, pp. 1146–1159, Nov. 1989.

[9] J. C. Chen and J. S. Sadowsky, "Simulation of trellis-coded modulation using importance sampling and error event simulation," 1991 Conference on Information Sciences and Systems, Baltimore, Maryland, March 20–22, 1991.

[10] J. C. Chen and J. S. Sadowsky, "Error event simulation for trellis-coded modulation," *Proc. 1991 IEEE Int. Symp. Inform. Theory*, p. 201, Budapest, Hungary, June 23–28, 1991.

[11] E. Cinlar, *Introduction to Stochastic Processes.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[12] G. C. Clark, Jr. and J. B. Cain, *Error Correction-Coding for Digital Communications.* Plenum Press, New York, 1981.

[13] B. R. Davis, "An improved importance sampling method for digital communications system simulation," *IEEE Trans. Commun.*, vol. COM-34, pp. 715–719, July 1986.

[14] A. Erdélyi, *Asymptotic Expansions.* Dover, New York, 1956.

[15] R. J. F. Fang, "A coded 8-PSK system for 140-Mbit/s information rate transmission over 80-MHz nonlinear transponders," *Proc. 7th Int. Conf. on Digital Satellite Communications*, ICDS-7, pp. 305–313, Munich, May 12–16, 1986.

[16] K. Feher, *Digital Communications: Satellite/Earth Station Engineering.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[17] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, March, 1973.

[18] G. D. Forney, Jr., R. G. Gallager, G. R. Lang, F. M. Longstaff and S. U. Qureshi, "Efficient modulation for band-limited channels," *IEEE J. Select. Areas Commun.*, vol. SAC-2, pp. 632–647, Sep. 1984.

[19] F. Fujino, Y. Moritani, M. Miyake, K. Murakami, Y. Sakato and H. Shiino, "A 120Mbit/s 8-PSK modem with soft-decision Viterbi decoder," *Proc. 7th Int. Conf. on Digital Satellite Communications*, ICDS-7, pp. 315–321, Munich, May 12–16, 1986.

[20] R. G. Gallager, *Information Theory and Reliable Communication.* Wiley, New York, 1968.

[21] P. W. Glynn and D. L. Iglehart, "Importance sampling for stochastic simulations," *Management Science*, vol. 35, pp. 1367–1392, 1989.

[22] S. A. Gronemeyer and A. L. McBride, "MSK and Offset QPSK Modulation," *IEEE Trans. Commun.*, vol. COM-24, pp. 809–820, Aug. 1976.

[23] T. T. Ha, *Digital Satellite Communications*, 2nd ed. McGraw-Hill, New York, 1990.

[24] J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods.* Chapman and Hall, New York, 1964.

[25] P. M. Hahn and M. C. Jeruchim, "Developments in the theory and application of importance sampling," *IEEE Trans. Commun.*, vol. COM-35, pp. 706–714, July 1987.

[26] A. Heller and I. M. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Trans. Commun.*, vol. COM-19, pp. 835–848, Oct. 1971.

[27] D. L. Hedderly and L. Lundquist, "Computer simulation of a digital satellite communications link," *IEEE Trans. Commun.*, vol. COM-21, pp. 321–325, April 1973.

[28] M. A. Herro and J. M. Nowack, "Simulated Viterbi decoding using importance sampling," *IEE Proceedings*, vol. 135, pt. F., no. 2, pp. 133–142, April 1988.

[29] P. Hetrakul and D. P. Taylor, "The effects of transponder nonlinearity on binary CPSK signal transmission," *IEEE Trans. Commun.*, vol. COM-24, pp. 546–553, May 1976.

[30] IMSL, Inc., *SFUN/LIBRARY User's Manual*, version 2.0. IMSL, Houston, 1987.

[31] M. C. Jeruchim, "Techniques for estimating the bit error rate in the simulation of digital communication systems," *IEEE J. Select. Areas Commun.*, vol. SAC-2, pp. 153–170, Jan. 1984.

[32] M. C. Jeruchim, P. M. Hahn, K. P. Smyntek and R. T. Ray, "An experimental investigation of conventional and efficient importance sampling," *IEEE Trans. Commun.*, vol. COM-37, pp. 578–587, June 1989.

[33] G. W. Lank, "Theoretical aspects of importance sampling applied to false alarms," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 73–82, Jan. 1983.

[34] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[35] D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Addison-Wesley, Reading, MA, 1984.

[36] D. Lu and K. Yao, "Improved importance sampling technique for efficient simulation of digital communication systems," *IEEE J. Select. Areas Commun.*, vol. SAC-6, pp. 67–75, Jan. 1988.

[37] D. Lu and K. Yao, "Estimation variance bounds of importance sampling simulations in digital communication systems," to appear in *IEEE Trans. Commun.*

[38] M. A. Marsan, S. Benedetto, E. Biglieri, V. Castellani M. Elia, L. L. Presti and M. Pent, "Digital simulation of communication systems with TOPSIM III," *IEEE J. Select. Areas Commun.*, vol. SAC-2, pp. 29–42, Jan. 1984.

[39] G. Orsak and B. Aazhang, "On the theory of importance sampling applied to the analysis of detection systems," *IEEE Trans. Commun.*, vol. COM-37, pp. 332–339, April 1989.

[40] G. Orsak and B. Aazhang, "Constrained solutions in importance sampling via robust statistics," *IEEE Trans. on Inform. Theory*, vol. IT-37, pp. 307–316, Mar. 1991.

[41] L. C. Palmer, "Computer modeling and simulation of communications satellite channels," *IEEE J. Select. Areas Commun.*, vol. SAC-2, pp. 89–102, Jan. 1984.

[42] S. Parekh and J. Walrand, "A quick simulation method for excessive backlogs in networks of queues," *IEEE Trans. Auto. Control*, vol. AC-34, pp. 54–66, Jan. 1989.

[43] J. G. Proakis, *Digital Communications*, 2nd ed. McGraw-Hill, New York, 1989.

[44] M. Rouanne and D. J. Costello, Jr., "An algorithm for computing the distance spectrum of trellis codes," *IEEE J. Select. Areas Commun.*, vol. SAC-7, pp. 929–940, Aug. 1989.

[45] R. Y. Rubinstein, *Simulation and the Monte Carlo Method.* Wiley, New York, 1981.

[46] J. S. Sadowsky, "A new method of Viterbi decoder simulation using importance sampling," *IEEE Trans. on Commun.*, vol. COM-38, pp. 1341–1351, Sep. 1990.

[47] J. S. Sadowsky, "Large deviations theory and efficient simulation of excessive backlogs in a G1/G1/m queue," to appear in *IEEE Trans. Auto. Control* special issue on queueing systems.

[48] J. S. Sadowsky and R. K. Bahr, "Direct-sequence spread-spectrum multiple-access communications with random signature sequences: a large deviations analysis," *IEEE Trans. Inform. Theory*, vol. IT-37, pp. 514–527, May 1991.

[49] J. S. Sadowsky and J. A. Bucklew, "On large deviations theory and asymptotically efficient Monte Carlo estimation," *IEEE Trans. on Inform. Theory*, vol. IT-36, pp. 579–588, Mar. 1990.

[50] S. M. Schetzen, "The Volterra and Wiener Theories of Nonlinear Systems," Wiley, New York, 1980.

[51] C. Schlegel, "Evaluating distance spectra and performance bounds of trellis codes on channels with intersymbol interference," *IEEE Trans. on Inform. Theory*, vol. IT-37, pp. 627–634, May 1991.

[52] H. J. Schulebusch, "On the asymptotical efficiency of importance sampling techniques," preprint, in review in *IEEE Trans. on Inform. Theory.*

[53] K. S. Shanmugan and P. Balaban, "A modified Monte Carlo simulation technique for the evaluation of error rate in digital communications systems," *IEEE Trans. Commun.*, vol. COM-28, pp. 1916–1924, Nov. 1980.

[54] Special issue on combined modulation and coding, *IEEE Trans. Commun.*, vol. COM-29, Mar. 1981.

[55] J. J. Spilker, *Digital Communications by Satellite.* Prentice-Hall, Englewood Cliffs, NJ, 1977.

[56] M. L. Steinberger, P. Balaban and K. S. Shanmugan, "On the effect of uplink noise on a nonlinear digital satellite channel," *Proc. Int. Conf. on Communications*, ICC-7, pp. 299–304, Munich, May 12–16, 1986.

[57] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55–67, Jan. 1982.

[58] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets — Part I: Introduction and Part II: State of the art," *IEEE Commun. Mag.*, vol. 25, pp. 5–21, Feb. 1987.

[59] G. Ungerboeck, J. Hagenauer and T. Abdel-Nabi, "Coded 8-PSK experimental modem for the INTELSAT SCPC system," *Proc. 7th Int. Conf. on Digital Satellite Communications*, ICDS-7, pp. 299–304, Munich, May 12–16, 1986.

[60] G. Ungerboeck and I. Csajka, "On improving data-link performance by increasing the channel alphabet and introducing sequence coding," *Proc. 1976 IEEE Int. Symp. Inform. Theory*, p. 53, Ronneby, Sweden, June 1976.

[61] H. L. Van Trees, *Satellite Communications.* IEEE Press, New York, 1979.

[62] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260–269, April, 1967.

[63] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. COM-19, Oct. 1971.

[64] A. J. Viterbi and J. K. Omura, *Principles of Digital Communications and Coding.* McGraw-Hill, New York, 1979.

[65] A. J. Viterbi and A. M. Viterbi, "Nonlinear estimation of PSK-modulated carrier phase with application to burst digital transmission," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 543–551, July, 1983.

[66] L. F. Wei, "Trellis-coded modulation with multidimensional constellations," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 483–501, July, 1987.

[67] L. F. Wei, "Rotationally invariant convolutional channel coding with expanded signal space — Part I: 180 degree and Part II: Nonlinear codes," *IEEE J. Select. Areas Commun.*, vol. SAC-2, pp. 659–686, Sep. 1984.

[68] S. B. Weinstein, "Estimation of small probabilities by linearization of the tail of the probability distribution function," *IEEE Trans. Commun. Technol.*, vol. COM-19, Dec. 1971.

[69] S. B. Weinstein, "Theory and application of some classical and generalized asymptotic distributions of extreme values," *IEEE Trans. Inform. Theory*, vol. IT-19, Mar. 1973.

[70] R. J. Wolfe, M. C. Jeruchim and P. M. Hahn, "On optimum and suboptimum biasing procedures for importance sampling in communication simulation," *IEEE Trans. Commun.*, vol. COM-38, pp. 639–647, May 1990.

[71] E. Zehavi and J. K. Wolf, "On the performance evaluation of trellis codes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 196–202, Mar. 1987.

[72] R. E. Ziemer and R. L. Peterson, *Digital Communications and Spread Spectrum Systems*. Macmillan, New York, 1985.

APPENDICES

A. Sample Simulation Data for the Convolutional Code

Table A.1 is an output of the modified RC Algorithm for the convolutional code example in Section 5.3.1 with the Hamming weight of codewords $d(\mathbf{0}, \mathbf{e}) \leq 10$. This is also the input data file required by the importance sampling simulation program. Table A.2 shows a sample importance sampling simulation output data for the above code at the signal-to-noise ratio $E_b/N_0 = 5.5$ dB. The final bit error rate estimate is computed by (5.21) and the result is $\hat{P}_b^* = 0.200562\text{E-}05$, Est. $\left(\text{var}^*[\hat{P}_b^*]\right)^{1/2} = 0.755697\text{E-}07$ and $1/\varepsilon = 26.54$.

Table A.1 The IS simulation input data file, $d(\mathbf{0}, \mathbf{e}) \leq 10$.

#	$n_b(\mathbf{e})$	$\ell(\mathbf{e})$	$d(\mathbf{0}, \mathbf{e})$	$\mathbf{e}$
1	1	5	7	1110100111
2	3	8	7	1101000000010111
3	2	6	8	110100111011
4	4	9	8	111001000001001011
5	6	12	8	110100000010000001001011
6	3	7	9	11011101001011
7	5	10	9	1101000011000 1001011
8	5	10	9	1110010000 1010001011
9	7	13	9	1101000000100000 1010001011
10	2	7	10	11100111010111
11	2	8	10	11101010011 00111
12	4	8	10	1101111010001011
13	2	9	10	111010010010100111
14	4	10	10	1101000001011010111
15	4	10	10	1110010011000 0010111
16	6	11	10	1110010000100110001011
17	4	11	10	1101000000011001100111
18	4	11	10	1110101010000000010111
19	6	11	10	1101000011001010001011
20	4	12	10	110100000001010010100111
21	4	12	10	111010010001000000010111
22	6	13	10	1101000000100011000 0010111
23	8	14	10	110100000010000010011 0001011
24	6	14	10	110100000001101000000010111
25	6	15	10	1101000000010100010000000 10111

Table A.2 A sample result of the convolutional code simulation.

#	$n_b(\mathbf{e})$	$d(\mathbf{0},\mathbf{e})$	% e decoded	$\hat{P}^*(\mathbf{e}\|\mathbf{0})$	Est. $\left(\text{var}^*[\hat{P}^*(\mathbf{e}\|\mathbf{0})]\right)^{1/2}$	$1/\varepsilon$
1	1	7	50	0.3226E-06	0.2341E-07	13.78
2	3	7	52	0.2886E-06	0.2177E-07	13.26
3	2	8	51	0.4909E-07	0.3927E-08	12.50
4	4	8	50	0.5273E-07	0.4073E-08	12.94
5	6	8	48	0.4662E-07	0.3831E-08	12.17
6	3	9	48	0.7068E-08	0.5906E-09	11.97
7	5	9	48	0.6773E-08	0.5946E-09	11.39
8	5	9	52	0.7000E-08	0.5756E-09	12.16
9	7	9	47	0.7338E-08	0.6116E-09	12.00
10	2	10	47	0.1411E-08	0.1173E-09	12.03
11	2	10	48	0.1127E-08	0.9939E-10	11.34
12	4	10	48	0.1164E-08	0.1055E-09	11.03
13	2	10	50	0.1150E-08	0.1019E-09	11.29
14	4	10	48	0.1199E-08	0.1036E-09	11.57
15	4	10	46	0.1141E-08	0.9965E-10	11.45
16	6	10	50	0.1380E-08	0.1110E-09	12.43
17	4	10	47	0.1082E-08	0.9783E-10	11.06
18	4	10	48	0.1080E-08	0.9313E-10	11.59
19	6	10	47	0.1316E-08	0.1094E-09	12.03
20	4	10	46	0.1158E-08	0.1011E-09	11.46
21	4	10	48	0.1311E-08	0.1091E-09	12.01
22	6	10	47	0.1217E-08	0.1034E-09	11.77
23	8	10	49	0.1165E-08	0.1023E-09	11.39
24	6	10	51	0.1222E-08	0.1061E-09	11.52
25	6	10	47	0.1137E-08	0.9735E-10	11.68

B. Sample Simulation Data for the Quasi-Regular TCM Code

Table B.1 is an output of the modified RC Algorithm for the quasi-regular TCM code example in Section 5.3.2 and the worst-case distance of signal selector error sequences $d_w^2(\mathbf{e}) \leq 5.7574$. Figure B.1 is a program flow chart for the importance sampling simulation which shows in more detail the Viterbi Algorithm and the error event simulation method. Table B.2 presents a sample importance sampling simulation output data for the above code at the signal-to-noise ratio $E_s/N_0 = 9.5$ dB. The index number in the first column for an error sequence $\mathbf{e}$ is a one-to-one correspondence to that in Table B.1. The final P_e and P_b estimates are

$\hat{P}_e^*$	=	0.280688E-05
$\left(\text{var}^*[\hat{P}_e^*]\right)^{1/2}$	=	0.622554E-07
$1/\varepsilon$	=	45.09
$\hat{P}_b^*$	=	0.637126E-05
$\left(\text{var}^*[\hat{P}_b^*]\right)^{1/2}$	=	0.148704E-06
$1/\varepsilon$	=	42.85

Table B.1 The IS simulation input data file with 18 $\mathbf{e}$'s and $d_w^2(\mathbf{e}) \leq 5.7574$.

#	$n_b(\mathbf{e})$	$\ell(\mathbf{e})$	$d_w^2(\mathbf{e})$	$\mathbf{e}$
1	3	4	5.1716	6172
2	4	4	5.1716	6336
3	3	5	5.1716	20132
4	4	5	5.1716	20376
5	5	6	5.1716	207016
6	5	7	5.1716	6700012
7	4	5	5.7574	61332
8	5	5	5.7574	61176
9	5	5	5.7574	63772
10	4	6	5.7574	201732
11	5	6	5.7574	203372
12	6	6	5.7574	203136
13	6	7	5.7574	2011016
14	7	7	5.7574	2070736
15	7	8	5.7574	67000776
16	7	8	5.7574	63100012
17	8	9	5.7574	670003016
18	8	10	5.7574	2070300012

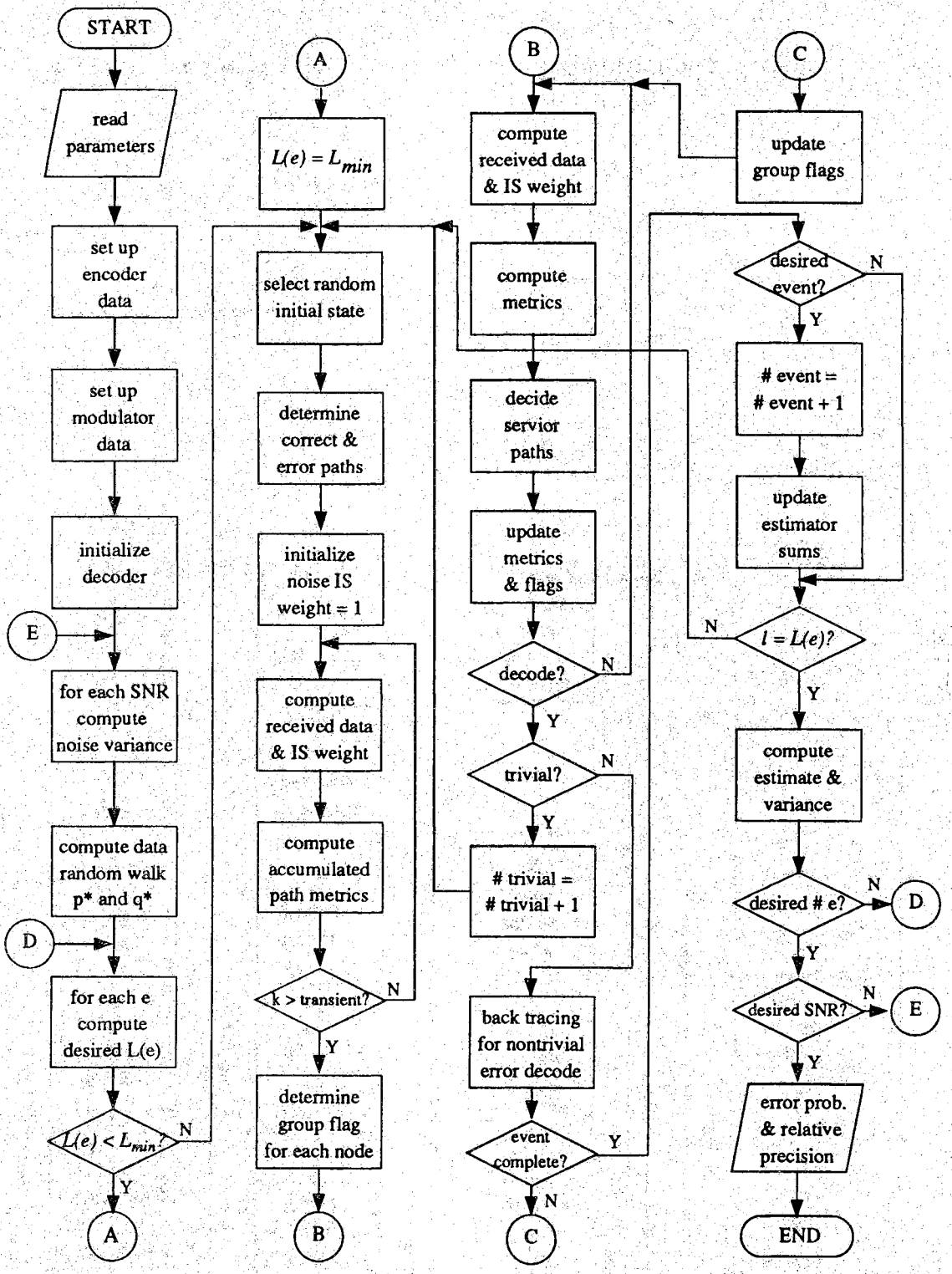


Figure B.1 A program flow chart for the TCM code IS simulation.

Table B.2 A sample result of the TCM code simulation.

#	$L_X(\mathbf{e})$	% e decoded	% trivial	$\hat{P}^*(\mathbf{e})$	$\left(\text{var}^*[\hat{P}^*(\mathbf{e})]\right)^{1/2}$	$1/\varepsilon$
1	1500	49	50	0.3829E-06	0.2414E-07	15.86
2	1500	51	48	0.2176E-06	0.1264E-07	17.21
3	1500	48	51	0.4426E-06	0.2595E-07	17.06
4	1500	49	49	0.2062E-06	0.1199E-07	17.20
5	1500	47	52	0.3697E-06	0.2299E-07	16.08
6	1500	51	48	0.4092E-06	0.2388E-07	17.13
7	385	50	49	0.5310E-07	0.6123E-08	8.67
8	385	49	50	0.9606E-07	0.1198E-07	8.02
9	385	49	50	0.2618E-07	0.3296E-08	7.94
10	385	48	50	0.5231E-07	0.6619E-08	7.90
11	385	52	45	0.2599E-07	0.3270E-08	7.95
12	385	51	48	0.5438E-07	0.6385E-08	8.52
13	385	51	48	0.2152E-06	0.2627E-07	8.19
14	385	54	45	0.2696E-07	0.3182E-08	8.47
15	385	51	48	0.2486E-07	0.3126E-08	7.95
16	385	46	53	0.9590E-07	0.1149E-07	8.35
17	385	45	54	0.4505E-07	0.6023E-08	7.48
18	385	52	47	0.6271E-07	0.7200E-08	8.71

VITA

VITA

Jyun-cheng Chen was born in April, 1959 in Tainan City, Taiwan, the Republic of China. He graduate from the National Chaio Tung University, Hsin-Chu, Taiwan, in June 1981 with a BS degree in Communication Engineering. From 1981 to 1983, he was a communication officer in a Chinese army radio corps. In 1983, he joined the United Microelectronics Corp. (UMC) of Hsin-chu, Taiwan as an engineer where he was responsible for telecommunication IC device testings. He obtained his MSEE degree from the New Jersey Institute of Technology, Newark, New Jersey, in 1986. He is now a Ph.D. candidate at the Purdue University. His research interest is in digital communication systems, coding, satellite communications, statistical analysis and simulations. His doctoral dissertation is on importance sampling simulations of digital communication systems.