An Efficient Technique for Bit Error Probability Estimation of MLSDA-Turbo Decoder

Prepared by Jia-Qin Pan

Advisory by Prof. Po-Ning Chen

In Partial Fulfillment of the Requirements

For the Degree of
Master of Science

Department of Communications Engineering

National Chiao Tung University

Hsinchu, Taiwan 300, R.O.C.

E-mail: u8813524@cc.nctu.edu.tw

June 14, 2001

Abstract

In this thesis, we applied the *Importance Sampling* (IS) method [1, 2, 3] to estimate the performance of the *MLSDA-turbo decoder* proposed in [4]. Unlike the brutal force *Monte Carlo* (MC) simulation [5], which often requires a very large number of simulation trials to achieve meaningful estimate of system performance, the IS technique can provide an efficient estimate with less trials. The success of the IS technique relies on the proper choice of the biased channel. Hence, the first task of the thesis is to find the right biased channel (or equivalently, the suitable error events) to ensure the efficiency of the IS simulation. By the simulation results, the best IS simulation among the channels we tried in this thesis yielded a moderate improvement in performance estimation. After the selection of the right biased channel, we investigated the time-unreachable performance region (of the MLSDA-turbo decoder) for MC simulation. We conclude that the best biased channel among all simulation channels we used indeed has the channel property similar to the mathematically optimal channel for IS technique.

Acknowledgements

I would like to thank Dr. Po-Ning Chen, as well as Dr. Yunghsiang S. Han, for their encouragement and guidance. This work would not have been possible without their advice and commitments. I would also like to thank my dear laboratory mates Zhao-Yuan, Yi-Ren, Web-Cheng for they provide a lot of fun on countless boring days. The members and friends of the Network Technology Laboratory are also grateful. Finally, I give the greatest respect to my family for their continuous support during my study.

Contents

List of Figures

Chapter 1

Introduction and Background

In 1993, codes with near-capacity performance, named turbo codes, were reported. Specifically, with parallel concatenated RSC convolutional codes and feedback decoding, its performance is only 0.7dB away from the capacity limit [6, 7]. Although the iterative decoding performed on the turbo codes [6, 7] is not an ML decoding algorithm, it provides a feasible decoding complexity to codewords with long block length. In [4], the authors proposed a soft-decision sequential decoding algorithm for time-invariant convolutional codes which are antipodally transmitted over the additive white Gaussian noise (AWGN) channel. In the same work, the authors defined a new metric for the sequential decoding algorithm, and presented a new *maximum-likelihood soft-decision sequential decoding algorithm* (MLSDA) based on the new metric. Later, it was applied to the decoding of turbo codes [17]. Unlike the iterative decoding algorithm proposed in [6, 7], which is a suboptimal decoding algorithm, the algorithm turns out to be a maximum-likelihood decoding algorithm for turbo codes.

Unfortunately, for this ML decoder, brutal force *Monte Carlo* (MC) simulation often requires a very large number of simulation trials in order to achieve a meaningful estimate of system performance. In general, the MC estimation of a binary decision error probability P_e within a 10% accuracy requires more than $100/P_e$ independent simulation trials. This num-

ber could be quite large for $P_e \leq 10^{-6}$, thereby making ordinary MC somewhat impractical in such case.

Importance Sampling (IS) is a modified MC technique which can efficiently reduce the number of simulation trials required to achieve a specified accuracy. The fundamental principle behind this method is to generate the random inputs from a "biased" sampling density. This new simulation density is chosen in a way that the error events of interest are "forced" to occur more often. Simulation data is then weighted by an appropriate likelihood ratio in order to obtain an unbiased estimate of the desired parameter. In recent years, there has been considerable interest in applying IS to digital communications systems. Some of them are applied to ML decoder, e.g., Sadowsky [2] developed an *error event simulation* (EES) method for the simulation of the Viterbi decoders; Khaled and Khurram [3] developed a *modified error event simulation* (MEES) method for the simulation of the ZJ decoders. In this thesis, a new application of the important sampling method for estimating BER of *MLSDA-Turbo* decoder is presented.

The thesis is organized in the following fashion. In Chapter 2, the preliminaries of *Importance Sampling* simulation on *MLSDA-Turbo* decoder is briefly presented. Specifically, definitions and notations for system models of fixed-length codes and maximum metric decoder are first given, and then some background on IS is introduced. In the end of Chapter 2, we introduce the *Maximum-likelihood Soft-Decision Sequential Decoding Algorithm* for *Parallel Concatenated Convolutional Code*.

Chapter 3 is devoted to the *Importance Sampling* method for *MLSDA-Turbo* decoders. The introduction of our approach is divided into two parts: "Simulation Environment" and "Channel Design."

Chapter 4 includes the simulation results of both stationary and nonstationary channels. In the same chapter, we also compare their performances, and find the best channel that

gives the most accurate estimate.

Chapter 5 concludes the thesis.

Chapter 2

Preliminaries

In this chapter, we introduce in sequence (1) the system model of fixed-length codes and maximum metric decoder, (2) the background of IS simulation, and then (3) the *Maximum-likelihood Soft-Decision Sequential Decoding Algorithm* for *Parallel Concatenated Convolutional Code*.

I Fixed-Length Codes and Maximum Metric Decoder

The following definitions and notations are defined similarly to [2].

For a fixed-length code $\mathcal{C}$, its codeword is a sequence of channel input symbols, denoted by $\boldsymbol{x} = (x_1, \ldots, x_i, \ldots, x_t)$. $\mathcal{C}$ is simply a collection of M codewords. The random sequence of channel output symbols is represented by $\boldsymbol{R} = (R_1, \ldots, R_i, \ldots, R_t)$. Here we adopt the notational convention that the lower case variables such as x_i denote deterministic quantities, the upper case variables such as R_i are random, and the boldface lower case variables such as $\boldsymbol{x}$ denote vectors. In addition, we use the subscript i to denote the time instance. The channel transition density is denoted by $f(\boldsymbol{r}|\boldsymbol{x})$ if $\boldsymbol{x}$ is transmitted. In case the channel is time-stationary and memoryless, $f(\boldsymbol{r}|\boldsymbol{x})$ can be reduced to:

$$f(\boldsymbol{r}|\boldsymbol{x}) = \sum_{i=1}^{t} p(r_i|x_i) \tag{2.1}$$

where $p(r|x)$ is the single-letter *channel transition probability.*

In a communication system, the decoder is to decide which of the M codewords was transmitted. It function is similar to partitioning all channel outputs $\boldsymbol{r}$ into different *decision regions* with respect to the M codewords. We denote the *decision region* corresponding to codeword $\boldsymbol{x}$ as $D(\boldsymbol{x})$. Specifically, $D(\boldsymbol{x})$ consists of all observation channel output sequences which are decoded to the codeword $\boldsymbol{x}$. Such decision regions can be defined through a maximum metric decoding rule. It can render a maximum likelihood decoding if a right metric definition is taken. The form of a *maximum metric decoder* with respect to metric $m(\cdot,\cdot)$ can be given as:

$$D(\boldsymbol{x}) = \left\{ \boldsymbol{r} : \sum_{i=1}^{t} m(x_i, r_i) \geq \max_{\boldsymbol{x}' \in \mathcal{C}} \sum_{i=1}^{t} m(x'_i, r_i) \right\} \tag{2.2}$$

Now based on the decision partitions, an important task for system evaluation is to find the average *bit error probability* P_b.

Let $P(\boldsymbol{x}'|\boldsymbol{x})$ denote the error probability of $\boldsymbol{x}'$ being decoded when $\boldsymbol{x}$ is transmitted. Equipped with the channel density $f(\boldsymbol{r}|\boldsymbol{x})$ and the decoding regions $D(\boldsymbol{x}')$, it can be described as:

$$P(\boldsymbol{x}'|\boldsymbol{x}) = \int_{D(\boldsymbol{x}')} f(\boldsymbol{r}|\boldsymbol{x}) d\boldsymbol{r} = \int_{\Re^t} I_{\boldsymbol{x}'}(\boldsymbol{r}) f(\boldsymbol{r}|\boldsymbol{x}) d\boldsymbol{r} \tag{2.3}$$

where

$$I_{\boldsymbol{x}'}(\boldsymbol{r}) \triangleq \begin{cases} 1, & \text{for } \boldsymbol{r} \in D(\boldsymbol{x}'); \\ 0, & \text{for } \boldsymbol{r} \notin D(\boldsymbol{x}') \end{cases}$$

is the *indicator function* of the decoding region $D(\boldsymbol{x}')$.

Denote by $N_b = N_b(\boldsymbol{x})$ the number of bit errors for the transmitted codeword $\boldsymbol{x}$. Then, the average bit error probability is given by:

$$P_b(\boldsymbol{x}) = \frac{1}{m} E[N_b | \boldsymbol{x} \text{ transmitted}] = \frac{1}{m} \sum_{\boldsymbol{x}' \in \mathcal{C}} n_b(\boldsymbol{x}, \boldsymbol{x}') P(\boldsymbol{x}'|\boldsymbol{x}), \tag{2.4}$$

where $m = \log_2(M)$, and $n_b(\boldsymbol{x}, \boldsymbol{x}')$ is the Hamming distance between binary codewords $\boldsymbol{x}$ and $\boldsymbol{x}'$. In the case of memoryless symmetric channels and linear codes, $P_b(\boldsymbol{x})$ will be independent of the transmitted codeword $\boldsymbol{x}$ [8, 9]; hence, one can just send the all-zero codeword to yield the system bit error probability $P_b = (1/M) \sum_{\boldsymbol{x} \in \mathcal{C}} P_b(\boldsymbol{x})$.

II Importance Sampling

In order to apply the IS technique to coded communication systems, we need to choose an appropriate biased channel density, denoted by $f^*(\boldsymbol{r}|\boldsymbol{x})$, which is called the *simulation density*. Consider the alternative expression of (2.3) in terms of dividing and multiplying $f^*(\boldsymbol{r}|\boldsymbol{x})$ in the integral of (2.3):

$$\begin{aligned} P(\boldsymbol{x}'|\boldsymbol{x}) &= \int_{\Re^t} \frac{f(\boldsymbol{r}|\boldsymbol{x})}{f^*(\boldsymbol{r}|\boldsymbol{x})} I_{\boldsymbol{x}'}(\boldsymbol{r}) f^*(\boldsymbol{r}|\boldsymbol{x}) d\boldsymbol{r} \\ &= \int_{\Re^t} w(\boldsymbol{r}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{r}) f^*(\boldsymbol{r}|\boldsymbol{x}) d\boldsymbol{r}, \end{aligned} \tag{2.5}$$

where

$$w(\boldsymbol{r}|\boldsymbol{x}) \stackrel{\triangle}{=} \frac{f(\boldsymbol{r}|\boldsymbol{x})}{f^*(\boldsymbol{r}|\boldsymbol{x})}. \tag{2.6}$$

It turns out that the importance sampling estimator can be written as:

$$\widehat{P}_J^*(\boldsymbol{x}'|\boldsymbol{x}) = \frac{1}{J} \sum_{j=1}^{J} w(\boldsymbol{r}^{(j)}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)}) \tag{2.7}$$

Here, $\boldsymbol{r}^{(j)} = (r_1^{(j)}, \ldots, r_i^{(j)}, \ldots, r_t^{(j)})$ denotes the channel output sequence for the jth simulation run, which is due to the biased channel density $f^*(\boldsymbol{r}|\boldsymbol{x})$. J represents the total number of simulation runs. The likelihood ratio $w(\boldsymbol{r}|\boldsymbol{x})$ is usually referred as the *importance sampling weight*. Note that when $f^*(\boldsymbol{r}|\boldsymbol{x}) = f(\boldsymbol{r}|\boldsymbol{x})$, i.e., $w(\boldsymbol{r}|\boldsymbol{x}) = 1$, the IS estimator reduces to the *Monte-Carlo* relative frequency estimator. The estimator for the bit error probability $P_b(\boldsymbol{x})$ can be obtained by substituting (2.7) termwisely into (2.4).

Two important statistical parameters of an estimator are the mean and variance. The expectation value of the IS estimator corresponding to the simulation density $f^*(\boldsymbol{r}|\boldsymbol{x})$ is:

$$\begin{aligned} E^*[\widehat{P}_J^*(\boldsymbol{x}'|\boldsymbol{x})] &= \frac{1}{J}\sum_{j=1}^{J} E^*\left[w(\boldsymbol{R}^{(j)}|\boldsymbol{x})I_{\boldsymbol{x}'}(\boldsymbol{R}^{(j)})\right] \\ &= E^*[w(\boldsymbol{R}|\boldsymbol{x})I_{\boldsymbol{x}'}(\boldsymbol{R})] \qquad (2.8) \\ &= \int_{\Re^t} I_{\boldsymbol{x}'}(\boldsymbol{r})w(\boldsymbol{r}|\boldsymbol{x})f^*(\boldsymbol{r}|\boldsymbol{x})d\boldsymbol{r} \\ &= \int_{\Re^t} I_{\boldsymbol{x}'}(\boldsymbol{r})\frac{f(\boldsymbol{r}|\boldsymbol{x})}{f^*(\boldsymbol{r}|\boldsymbol{x})}f^*(\boldsymbol{r}|\boldsymbol{x})d\boldsymbol{r} \\ &= P(\boldsymbol{x}'|\boldsymbol{x}), \qquad (2.9) \end{aligned}$$

where (2.8) follows from that $\{\boldsymbol{R}^{(j)}\}_{j=1}^{J}$ is independently and identically distributed. This result indicates the IS estimator is unbiased. Again, from the i.i.d. of $\{\boldsymbol{R}^{(j)}\}_{j=1}^{J}$, the variance of the IS estimator is given by:

$$\begin{aligned} \mathrm{Var}^*\left[\widehat{P}_J^*(\boldsymbol{x}'|\boldsymbol{x})\right] &= \frac{1}{J^2}\sum_{j=1}^{J}\mathrm{Var}^*\left[w(\boldsymbol{R}^{(j)}|\boldsymbol{x})I_{\boldsymbol{x}'}(\boldsymbol{R}^{(j)})\right] \\ &= \frac{1}{J}\mathrm{Var}^*[w(\boldsymbol{R}|\boldsymbol{x})I_{\boldsymbol{x}'}(\boldsymbol{R})] \\ &= \frac{1}{J}\left\{\int_{\Re^t} w^2(\boldsymbol{r}|\boldsymbol{x})I_{\boldsymbol{x}'}(\boldsymbol{r})f^*(\boldsymbol{r}|\boldsymbol{x})d\boldsymbol{r} - P^2(\boldsymbol{x}'|\boldsymbol{x})\right\}. \qquad (2.10) \end{aligned}$$

As anticipated, the larger the variance is, the less precision the estimator is. A common precision index used in IS techniques is the *relative precision estimate* (RPE) [2], which is defined as:

$$\mathrm{RPE}(\boldsymbol{x}'|\boldsymbol{x}) = \frac{\sqrt{\mathrm{Var}^*\left[\widehat{P}_J^*(\boldsymbol{x}'|\boldsymbol{x})\right]}}{E^*\left[\widehat{P}_J^*(\boldsymbol{x}'|\boldsymbol{x})\right]} \times 100(\%).$$

The RPE can be estimated empirically as:

$$\widehat{RPE}_J^*(\boldsymbol{x}'|\boldsymbol{x}) = \frac{\sqrt{\frac{1}{J}\left\{\frac{1}{J}\sum_{j=1}^{J} w^2(\boldsymbol{r}^{(j)}|\boldsymbol{x})I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)}) - \left(\frac{1}{J}\sum_{j=1}^{J} w(\boldsymbol{r}^{(j)}|\boldsymbol{x})I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)})\right)^2\right\}}}{\frac{1}{J}\sum_{j=1}^{J} w(\boldsymbol{r}^{(j)}|\boldsymbol{x})I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)})}$$

$$= \sqrt{\frac{\sum_{j=1}^{J} w^2(\boldsymbol{r}^{(j)}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)})}{\left(\sum_{j=1}^{J} w(\boldsymbol{r}^{(j)}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)})\right)^2} - \frac{1}{J}} \times 100(\%). \tag{2.11}$$

Apparently, the less the RPE is, the more accuracy the estimator can achieve[1].

Similarly, for the IS estimator of $P_b(\boldsymbol{x})$, we have:

$$\begin{aligned}
\widehat{P}_J^*(\boldsymbol{x}) &= \frac{1}{m} \sum_{\boldsymbol{x}' \in \mathcal{C}} n_b(\boldsymbol{x}, \boldsymbol{x}') \widehat{P}_J^*(\boldsymbol{x}'|\boldsymbol{x}) \\
&= \frac{1}{m} \sum_{\boldsymbol{x}' \in \mathcal{C}} n_b(\boldsymbol{x}, \boldsymbol{x}') \left[\frac{1}{J} \sum_{j=1}^{J} w(\boldsymbol{r}^{(j)}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)}) \right] \\
&= \frac{1}{mJ} \sum_{j=1}^{J} w(\boldsymbol{r}^{(j)}|\boldsymbol{x}) \left(\sum_{\boldsymbol{x}' \in \mathcal{C}} n_b(\boldsymbol{x}, \boldsymbol{x}') I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)}) \right).
\end{aligned}$$

Its mean and variance are respectively:

$$\begin{aligned}
E^*[\widehat{P}_J^*(\boldsymbol{x})] &= \frac{1}{mJ} \sum_{j=1}^{J} E^* \left[w(\boldsymbol{R}^{(j)}|\boldsymbol{x}) \left(\sum_{\boldsymbol{x}' \in \mathcal{C}} n_b(\boldsymbol{x}, \boldsymbol{x}') I_{\boldsymbol{x}'}(\boldsymbol{R}^{(j)}) \right) \right] \\
&= \frac{1}{m} E^* \left[w(\boldsymbol{R}|\boldsymbol{x}) \left(\sum_{\boldsymbol{x}' \in \mathcal{C}} n_b(\boldsymbol{x}, \boldsymbol{x}') I_{\boldsymbol{x}'}(\boldsymbol{R}) \right) \right] \\
&= \frac{1}{m} \sum_{\boldsymbol{x}' \in \mathcal{C}} n_b(\boldsymbol{x}, \boldsymbol{x}') E^* \left[w(\boldsymbol{R}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{R}) \right] \\
&= \frac{1}{m} \sum_{\boldsymbol{x}' \in \mathcal{C}} n_b(\boldsymbol{x}, \boldsymbol{x}') P(\boldsymbol{x}'|\boldsymbol{x}) \\
&= P_b(\boldsymbol{x}),
\end{aligned}$$

[1]Both the IS and MC simulations are based the statistical fact that the average of i.i.d. random variables will converge to its mean. Here, what the RPE concerns is actually the *rate-of-convergence*. For example, one may wish to know the number of simulation runs required such that the probability of the average of i.i.d. observations lying within $\mu(1+\beta)$ and $\mu(1-\beta)$ is larger than $1-\varepsilon$, where μ is the marginal mean of the i.i.d. process. Mathematically, it can be written as finding the minimum n satisfying:

$$Pr\left\{ \left| \frac{X_1 + X_2 + \cdots + X_n}{n} - \mu \right| < \mu\beta \right\} > 1 - \varepsilon,$$

or equivalently,

$$Pr\left\{ \left| \frac{(X_1/\mu) + (X_2/\mu) + \cdots + (X_n/\mu)}{n} - 1 \right| < \beta \right\} > 1 - \varepsilon.$$

It is then reasonable to say that for fixed β and ε, the larger the variance of (X_i/μ), the larger the minimum n is. Indeed, the variance of (X_i/μ) equals $\mathrm{Var}[X_i]/\mu^2$, which is exactly the square of the RPE.

and

$$\begin{aligned}
\mathrm{Var}^*[\widehat{P}_J^*(\boldsymbol{x})] &= \frac{1}{m^2J^2}\sum_{j=1}^{J}\mathrm{Var}^*\left[w(\boldsymbol{R}^{(j)}|\boldsymbol{x})\left(\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b(\boldsymbol{x},\boldsymbol{x}')I_{\boldsymbol{x}'}(\boldsymbol{R}^{(j)})\right)\right] \\
&= \frac{1}{m^2J}\mathrm{Var}^*\left[w(\boldsymbol{R}|\boldsymbol{x})\left(\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b(\boldsymbol{x},\boldsymbol{x}')I_{\boldsymbol{x}'}(\boldsymbol{R})\right)\right] \\
&= \frac{1}{m^2J}\left(E^*\left[w^2(\boldsymbol{R}|\boldsymbol{x})\left(\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b(\boldsymbol{x},\boldsymbol{x}')I_{\boldsymbol{x}'}(\boldsymbol{R})\right)^2\right] - m^2P_b^2(\boldsymbol{x})\right) \\
&= \frac{1}{m^2J}\left(E^*\left[w^2(\boldsymbol{R}|\boldsymbol{x})\left(\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b^2(\boldsymbol{x},\boldsymbol{x}')I_{\boldsymbol{x}'}(\boldsymbol{R})\right)\right] - m^2P_b^2(\boldsymbol{x})\right) \qquad (2.12) \\
&= \frac{1}{m^2J}\left(\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b^2(\boldsymbol{x},\boldsymbol{x}')E^*\left[w^2(\boldsymbol{R}|\boldsymbol{x})I_{\boldsymbol{x}'}(\boldsymbol{R})\right] - m^2P_b^2(\boldsymbol{x})\right),
\end{aligned}$$

where (2.12) holds since

$$\left(\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b(\boldsymbol{x},\boldsymbol{x}')I_{\boldsymbol{x}'}(\boldsymbol{R})\right)^2 = \sum_{\boldsymbol{x}'\in\mathcal{C}} n_b^2(\boldsymbol{x},\boldsymbol{x}')I_{\boldsymbol{x}'}(\boldsymbol{R}) + \sum_{\boldsymbol{x}'\neq\boldsymbol{x}''} n_b(\boldsymbol{x},\boldsymbol{x}')n_b(\boldsymbol{x},\boldsymbol{x}'')I_{\boldsymbol{x}'}(\boldsymbol{R})I_{\boldsymbol{x}''}(\boldsymbol{R})$$

and $I_{\boldsymbol{x}'}(\boldsymbol{R})I_{\boldsymbol{x}''}(\boldsymbol{R}) = 0$ whenever $\boldsymbol{x}' \neq \boldsymbol{x}''$. By defining its RPE as

$$\mathrm{RPE}(\boldsymbol{x}) = \frac{\sqrt{\mathrm{Var}^*[\widehat{P}_J^*(\boldsymbol{x})]}}{E^*[\widehat{P}_J^*(\boldsymbol{x})]},$$

we obtain[2]:

$$\widehat{\mathrm{RPE}}_J^*(\boldsymbol{x}) = \sqrt{\frac{\sum_{j=1}^{J} w^2(\boldsymbol{r}^{(j)}|\boldsymbol{x})\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b^2(\boldsymbol{x},\boldsymbol{x}')I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)})}{\left(\sum_{j=1}^{J} w(\boldsymbol{r}^{(j)}|\boldsymbol{x})\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b(\boldsymbol{x},\boldsymbol{x}')I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)})\right)^2} - \frac{1}{J}}, \qquad (2.13)$$

[2]An alternative but unsound way to derive $\widehat{\mathrm{RPE}}_J^*(\boldsymbol{x})$ could be:

$$\mathrm{RPE}(\boldsymbol{x}) = \frac{\sqrt{\mathrm{Var}^*\left[\widehat{P}_J^*(\boldsymbol{x})\right]}}{E\left[\widehat{P}_J^*(\boldsymbol{x})\right]} = \frac{\sqrt{\mathrm{Var}^*\left[\frac{1}{m}\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b(\boldsymbol{x},\boldsymbol{x}')\widehat{P}_J^*(\boldsymbol{x}'|\boldsymbol{x})\right]}}{\frac{1}{m}\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b(\boldsymbol{x}',\boldsymbol{x})E\left[\widehat{P}_J^*(\boldsymbol{x}'|\boldsymbol{x})\right]} = \frac{\sqrt{\frac{1}{m^2}\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b^2(\boldsymbol{x},\boldsymbol{x}')\mathrm{Var}^*[\widehat{P}_J^*(\boldsymbol{x}'|\boldsymbol{x})]}}{\frac{1}{m}\sum_{\boldsymbol{x}'\in\mathcal{C}} n_b(\boldsymbol{x}',\boldsymbol{x})E\left[\widehat{P}_J^*(\boldsymbol{x}'|\boldsymbol{x})\right]}.$$

Note that the last equality holds only when $\widehat{P}_J^*(\boldsymbol{x}'|\boldsymbol{x})$ and $\widehat{P}_J^*(\boldsymbol{x}''|\boldsymbol{x})$ are independent for $\boldsymbol{x}' \neq \boldsymbol{x}''$, which is not necessarily true. Its invalidness can also be substantiated by that the $\widehat{\mathrm{RPE}}_J^*(\boldsymbol{x})$ obtained based on the

which again can be empirically estimated.

When the code book $\mathcal{C}$ is large, it is not possible to yield the estimate $\widehat{P}_J^*(\boldsymbol{x})$, as well as $\widehat{\mathrm{RPE}}(\boldsymbol{x})$, in terms of the entire code book. In that case, a few selected codewords that dominates the error probability will be used instead. For example, one may focus on those codewords $\boldsymbol{x}'$ in $\mathcal{C}' \subset \mathcal{C}$, which have $n_b(\boldsymbol{x}, \boldsymbol{x}') = d_{\min}$, where $d_{\min}$ is the minimum pairwise Hamming distance for code $\mathcal{C}$, and yield:

$$\widehat{P}_J^*(\boldsymbol{x}) \approx \frac{d_{\min}}{mJ} \sum_{\boldsymbol{x}' \in \mathcal{C}'} \sum_{j=1}^{J} w(\boldsymbol{r}^{(j)}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)}),$$

and

$$\widehat{\mathrm{RPE}}_J^*(\boldsymbol{x}) \approx \sqrt{\frac{\sum_{\boldsymbol{x}' \in \mathcal{C}'} \sum_{j=1}^{J} w^2(\boldsymbol{r}^{(j)}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)})}{\left(\sum_{\boldsymbol{x}' \in \mathcal{C}'} \sum_{j=1}^{J} w(\boldsymbol{r}^{(j)}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)}) \right)^2} - \frac{1}{J}}.$$

As mentioned in the previous section, $P_b(\boldsymbol{x})$ is independent of $\boldsymbol{x}$ when a linear code and a symmetric channel are concerned. This is exactly the case considered in this thesis. Hence, we may choose to transmit the all-zero codeword, and take the above formula to estimate the bit error probability $\widehat{P}_J^*$ and $\widehat{\mathrm{RPE}}_J^*$.

In the end, we point out that we can also calculate the relative precision estimate of *Monte-Carlo* simulation by taking $f^*(\boldsymbol{r}|\boldsymbol{x}) = f(\boldsymbol{r}|\boldsymbol{x})$. We can then compare the numbers of simulation runs required respectively for IS and MC estimators under the same RPE.

above equation does not equal (2.13), which is

$$\widehat{\mathrm{RPE}}_J^*(\boldsymbol{x}) = \frac{\sqrt{\frac{1}{m^2} \sum_{\boldsymbol{x}' \in \mathcal{C}} n_b^2(\boldsymbol{x}, \boldsymbol{x}') \frac{1}{J} \left\{ \frac{1}{J} \sum_{j=1}^{J} w^2(\boldsymbol{r}^{(j)}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)}) - \left(\frac{1}{J} \sum_{j=1}^{J} w(\boldsymbol{r}^{(j)}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)}) \right)^2 \right\}}}{\frac{1}{m} \sum_{\boldsymbol{x}' \in \mathcal{C}} n_b(\boldsymbol{x}', \boldsymbol{x}) \frac{1}{J} \sum_{j=1}^{J} w(\boldsymbol{r}^{(j)}|\boldsymbol{x}) I_{\boldsymbol{x}'}(\boldsymbol{r}^{(j)})}.$$

III Maximum-likelihood Soft-Decision Sequential Decoding Algorithm for Parallel Concatenated Convolutional Codes

In this section, we briefly present *MLSDA* for PCCC.

III.1 Structure of Turbo Encoder

The following definitions and notations on convolutional codes are defined similarly to [10] and [11].

We denote $\mathcal{C}$ as a binary (n, k, q) convolutional code, in which q is the *memory order*. The memory order means the maximum number of shift-register stages from the encoder input to the encoder output. Denote by

$$R \triangleq k/n \quad \text{and} \quad t \triangleq n(L+q)$$

respectively the *code rate* and the *code length* of $\mathcal{C}$, where L represents the length of the information sequence. When a convolutional code is transmitted into its equivalent linear block code, its *effective code rate* [10] becomes

$$\tilde{R} \triangleq \frac{kL}{t} = \frac{kL}{n(L+q)}.$$

As anticipated, $\tilde{R} \approx R$ when $L \gg q$.

Based on the above notations, we consider a structural representation of codewords for convolutional codes. A *code tree* of an (n, k, q) convolutional code represents every codeword as a path over the tree. For an information sequence of length L, the code tree consists of $(L + q + 1)$ levels. The leftmost node at level 0 is referred to as the *origin node*. There are exactly 2^k branches leaving each node at the first L level. For those nodes at levels L through $(L + q)$, there is only one branch leaving each node. The 2^{kL} rightmost nodes at

level $(L+q)$ are called the *terminal nodes*. A path from the origin node to a terminal node is named a *code path*.

The turbo encoder is based on a special kind of convolutional encoder named recursive systematic convolutional (RSC) encoder. *Systematic* means that the input bits will be part of the encoded bits. *Recursive* means that the registers in the generator has a feedback loop. The encoder of turbo codes is a parallel concatenation of several RSC encoders, which are interconnected through interleavers. Figure 2.1 shows an example of a specific turbo encoder, consisting of two identical (37,21) RSC encoders, parallelly concatenated by an interleaver. This is the c

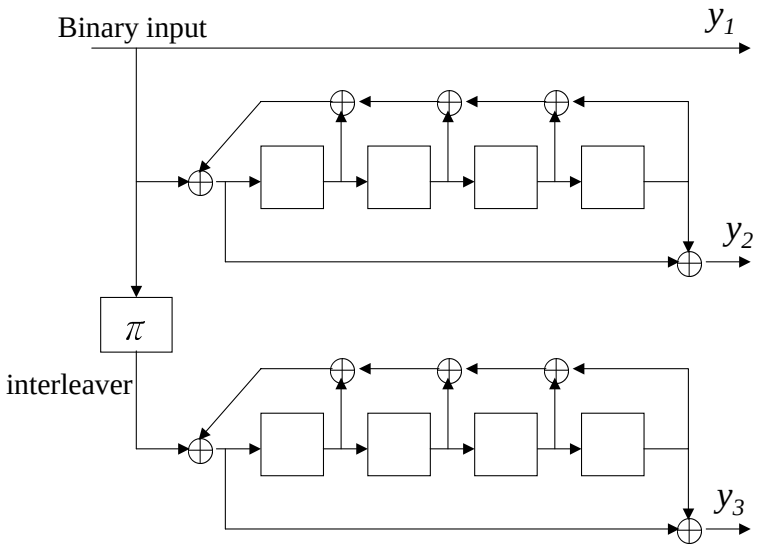

Figure 2.1: Turbo Code Encoder with (37,21) RSC component codes.

The interleaver is used to shuffle the bits in the input sequence. Hence, the two constituent RSC encoders operates on two input sequences consisting of the same information bits, but in different order. In the simulation model used in this thesis, a block interleaver is employed. A block interleaver can be described in term of a $P \times Q$ matrix. Such interleaver is characterized

by a process in which the data is written in along the rows of a matrix and read out along the columns [12]. In Figure 2.2, an example with $P = Q = 6$ is given to demonstrate how a block interleaver works. Indeed, this is the block interleaver used in our simulations.

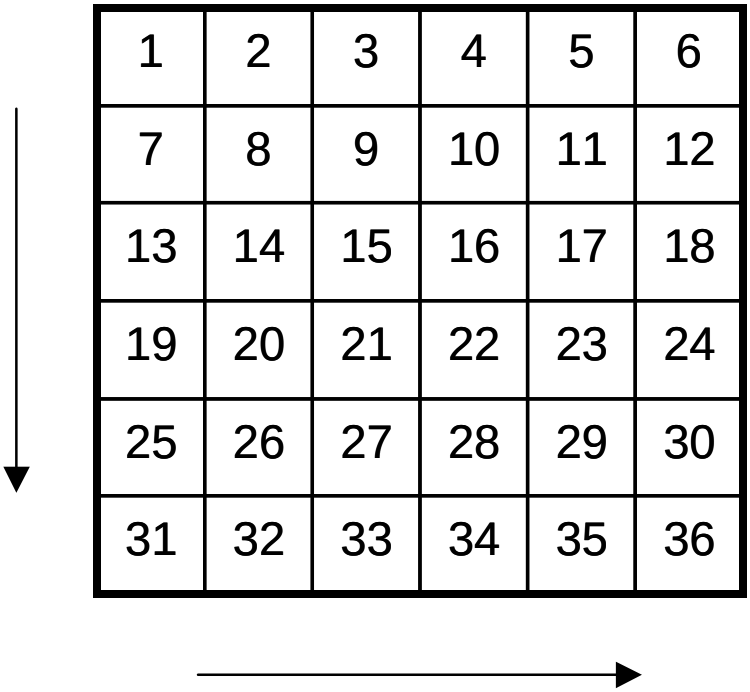

Figure 2.2: A 6×6 block interleaver.

III.2 Tree Representation of Turbo Codes

In this subsection, we present the code tree structure of a turbo code defined through the encoder in Figure 2.3. The code tree representation of a convolutional code is a well-known subject, which can be found in most coding textbook. Due to the effect of interleaver, the code tree representation of a turbo code is much more complex than that of a convolutional code. To facilitate the characterization of a code-tree structural turbo code, an interleaver of size 5 is taken instead of the simulated 6×6 one in the following paragraphs.

Now suppose that the interleaver rearranges the information sequence $I_1 I_2 I_3 I_4 I_5$ into

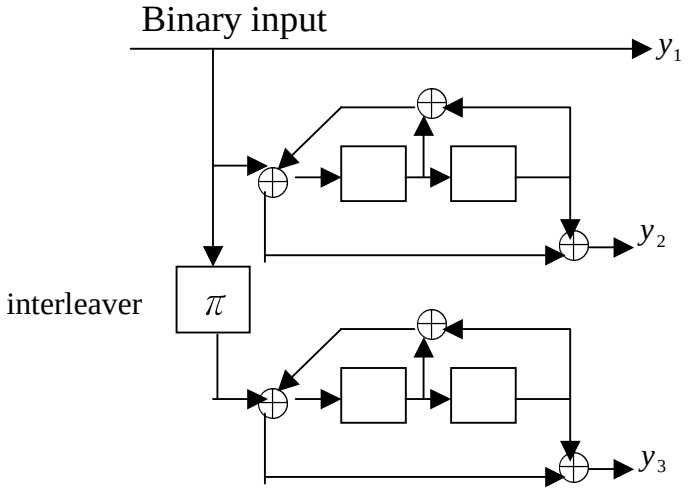

Figure 2.3: The turbo encoder with (7,5) component codes.

$\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_4\bar{I}_5 = I_3I_1I_4I_5I_2$. The two information sequence $\boldsymbol{I} = I_1I_2I_3I_4I_5$ and $\bar{\boldsymbol{I}} = I_3I_1I_4I_5I_2$ are respectively fed into the two encoders at the same time. As a consequence, we can represent the two sequences as ordered pairs like $(I_1I_3), (I_2I_1), (I_3I_4), (I_4I_5)$ and (I_5I_2). The codetree will then grow according to the paired input.

To be specific, at the origin node of the code tree, both components of the input pair (I_1I_3) are new to the system, so they could be either (00), (01), (10) or (11). Hence, there are four paths generated from the origin node. At the first level of the code tree, the new pair (I_2I_1) appears; however, because I_1 has appeared in the previous level, I_1 should be consistent with the previous determined one. So, the number of branches goes out from each node at the second level is two, decided by the new information bit I_2. As shown in Figure 2.4, the successor path generated from the node associated with the previous input pair (10) can only be marked by either (01) or (11). This explains why the number of successor paths could be either 1, 2 or 4, which is determined by the newly appearing information bit.

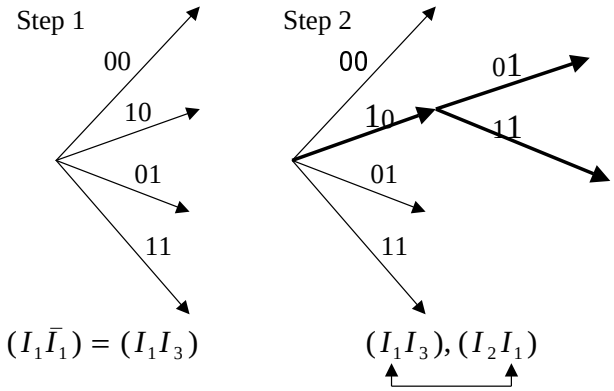

Figure 2.4: The generation of the turbo code tree.

For example, if both input bits are already assigned by previous levels, there will be one valid successor. In case that only one of the two input bits is determined, there will be two valid successors. There will be four valid successors if no input bits are assigned previously.

III.3 Tree-based MLSDA for Turbo Codes

In this subsection, we present how to apply the MLSDA proposed in [4] to turbo decoding. We denote by $\boldsymbol{x} \stackrel{\triangle}{=} (x_1, \ldots, x_i, \ldots, x_t)$ a binary codeword of an (n, k, q) convolutional code. $t \stackrel{\triangle}{=} n(L+q)$ is the code length and L is the length of the information sequence. We assume that a binary codeword is antipodally transmitted over the AWGN channel, and induce a received vector $\boldsymbol{r} \stackrel{\triangle}{=} (r_1, \ldots, r_i, \ldots, r_t)$. In other words, $r_i = (-1)^{x_i} + n_i$ for $1 \leq i \leq t$. Here, signal energy per channel bit is taken to be 1, and n_i is a Gaussian noise sample with single-sided noise power per hertz N_0, i.e., n_i is zero-mean Gaussian distributed with variance $N_0/2$. Let the signal-to-noise ratio be γ. For the code rate R, the signal-to-noise ratio per information bit, denoted by γ_b, is $\gamma_b = \gamma/R$.

During the MLSDA decoding, we use the new bit metric defined in [4]. Specifically,

for any path $\boldsymbol{x}_{(\ell n)} = (x_1, x_2, \ldots, x_{\ell n})$ ending at the ℓth level of the code tree, the metric associated with it is given by:

$$M\left(\boldsymbol{x}_{(\ell n)}\right) \stackrel{\triangle}{=} \sum_{i=1}^{\ell n} (y_i \oplus x_i)|\phi_i|, \tag{2.14}$$

where y_i is the hard decision of r_i and

$$\phi_i \stackrel{\triangle}{=} \ln \frac{Pr(r_i|0)}{Pr(r_i|1)}. \tag{2.15}$$

The term $(y_i \oplus x_i)|\phi_i|$ in (2.14) is called the *bit metric*, and is denoted by $M(x_i)$. Note that the path metrics defined here are equivalent to those given in [13] and [14], which were originally defined over code trees of block codes. Because of the non-negativity of the metric, a sequential decoding algorithm based on it can always find the path with the smallest metric. Hence, it was named the *maximum-likelihood sequential decoding algorithm* (MLSDA). For completeness, the tree-based MLSDA algorithm is quoted below.

<The tree-based MLSDA>

Step 1. Load the Stack with the origin node whose metric is assigned to be zero.

Step 2. Compute the metrics of the successors of the top path in the Stack, delete the top path from the Stack.

Step 3. Insert the new paths into the Stack, and reorder the Stack according to ascending metric values.

Step 4. If the top path in the Stack ends at a terminal node in the tree, the algorithm stops; otherwise go to Step 2.

The above algorithm can be applied to any code with code-tree structure, including the turbo codes described in the previous subsections. Because of a possible limitation on

computer memory, a threshold on the stack size has been set. In words, if the size of stack grows larger than the threshold, the path with the maximum metric in the stack will be deleted. To clearly illustrating the MLSDA for Turbo codes, both its algorithm and block diagram (cf. Figure 2.5) are quoted.

<The tree-based MLSDA for turbo Codes>

Step 1. Load the Stack with the origin node whose metric is assigned to be zero.

Step 2. Compute the metrics of the successors of the top path in the Stack, it might be one or two or four successors. Delete the top path from the Stack.

Step 3. Insert the new paths into the Stack, and reorder the Stack according to ascending metric values.

Step 4. If the stack size exceed the Threshold, delete the path with the maximum metric in the stack.

Step 5. If the top path in the Stack ends at a terminal node in the tree, the algorithm stops; otherwise go to Step 2.

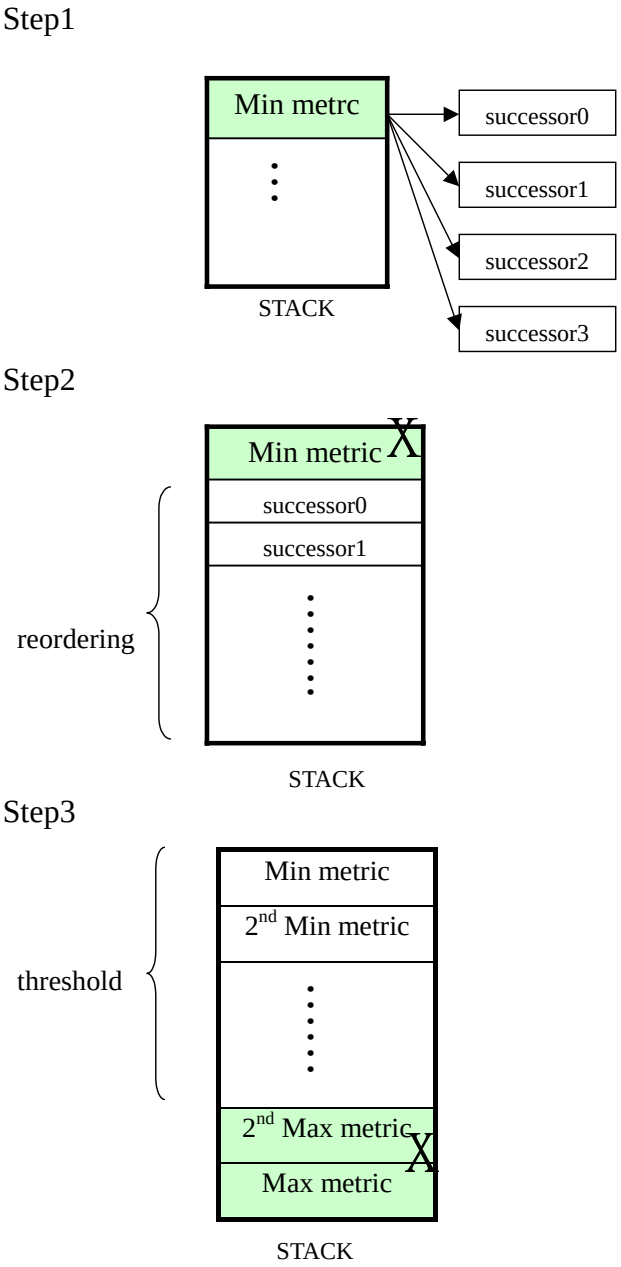

Figure 2.5: The block diagram of MLSDA decoding for tree-structural turbo codes.

Chapter 3

The Important Sampling Method on MLSDA-Turbo Decoder

In this section, we present how to apply the *Importance Sampling* method to *MLSDA-Turbo* decoder.

I Simulation Environment

Our simulation environment is illustrated in Figure 3.1.

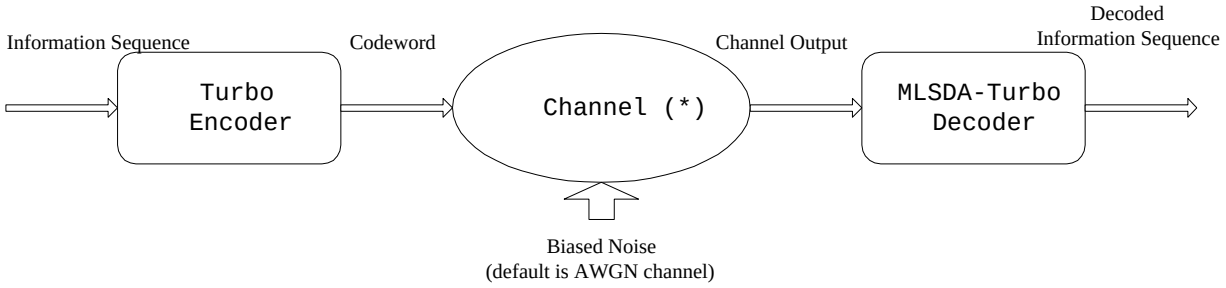

Figure 3.1: Simulation model.

In our simulations, the turbo component codes are chosen to be the 16-state RSC codes with generator $(37, 21)$, as shown in Figure 2.1. The information length L is taken to be 36. Consequently, for the $(3, 1)$ turbo codes, the codeword length is 108. As mentioned in the

previous chapter, we adopt a 6×6 block interleaver. Although we have set a stack limit in our simulation, no discard activities have ever occurred during our simulation; hence, the simulation results are identical to those obtained under unlimited stack.

One way to locate the most likely error events for AWGN channels is to calculate the distance spectrum of the codes, which is depicted in Table 3.1. In this table, H_d denotes the number of codewords with Hamming weight d, and W_d is the total information weight of all codewords with Hamming weight d. From the table, the free distance of this code (d_{free}) is apparently 3.

d	H_d	W_d
3	1	1
6	2	4
7	2	2
9	3	3
10	25	50
11	13	31
12	13	34

Table 3.1: The distance spectrum of the turbo encoder with L=36, block interleaver and (37,21) RSC component codes.

In order to find the dominant error events, we truncated some error events during simulations, and compared the resultant performance change. We found that the top 8 error events in Table 3.1 are sufficiently dominant for bit error estimations. We then performed simulation trials through the biased channel we designed. The error estimate was calculated by multiplying the *IS weight* through the importance sampling estimator in (2.7). The biased channels designed will be described in next section.

II Channel Designs

II.1 Observations and Discussions

The fundamental objective of the IS technique is to select $f^*_{\boldsymbol{x}'}(\boldsymbol{r}|\boldsymbol{x})$ in a way that the variance of $\widehat{P}^*_J(\boldsymbol{x}'|\boldsymbol{x})$ as given by (2.10) is minimized, where the erroneous decoding result $\boldsymbol{x}'$ is caused by the dominant error events. For convenience, we refer to the optimal $f^*_{\boldsymbol{x}'}(\boldsymbol{r}|\boldsymbol{x})$ as *optimal density function*, and name its corresponding channel as the *optimal channel*. Denote the *optimal density function* with respect to the erroneous decoding result $\boldsymbol{x}'$ as $f^*_{\boldsymbol{x}',\text{opt}}(\boldsymbol{r}|\boldsymbol{x})$. From (2.10), we found that $\text{Var}[\widehat{P}^*_J(\boldsymbol{x}'|\boldsymbol{x})] \equiv 0$, when

$$f^*_{\boldsymbol{x}',\text{opt}}(\boldsymbol{r}|\boldsymbol{x}) = \frac{f(\boldsymbol{r}|\boldsymbol{x})I_{\boldsymbol{x}'}(\boldsymbol{r})}{P(\boldsymbol{x}'|\boldsymbol{x})} \tag{3.1}$$

Hence, the *optimal IS density* $f^*_{\boldsymbol{x}',\text{opt}}(\boldsymbol{r}|\boldsymbol{x})$ is proportional to $f(\boldsymbol{r}|\boldsymbol{x})$. Unfortunately, this unconstrained optimal solution is not practical because of the lack of knowledge in the unknown parameter $P(\boldsymbol{x}'|\boldsymbol{x})$. Nonetheless, $f^*_{\boldsymbol{x}',\text{opt}}(\boldsymbol{r}|\boldsymbol{x})$ still indicates certain features a "good" IS density should have, which are listed below.

<Possible features of "good" IS densities>

1. *Concentrate probability mass on the decoding error region $D(\boldsymbol{x}')$ for some dominant error contributors $\boldsymbol{x}'$.*

2. *Within this region, the density is proportional to the true density $f(\boldsymbol{r}|\boldsymbol{x})$.*

3. *$f^*_{\boldsymbol{x}',\text{opt}}(\boldsymbol{r}|\boldsymbol{x})$ in (3.1) depends on the specific decoding error contributor $\boldsymbol{x}'$.*

4. *A channel model $f^*_{\boldsymbol{x}',\text{opt}}(\boldsymbol{r}|\boldsymbol{x})$ in (3.1) could be neither stationary nor memoryless. This density, however, must be practical in the sense that it can be generated by a digital computer.*

As a result, a "good" IS density should increase the relative frequency of these "important" error events. This is why it is named "importance" sampling.

The previous discussions indicate that the key issue in IS is to choose an appropriate IS density, which is similar to the unconstrained optimal IS density, to ensure the efficiency of the IS simulation, thereby reducing the simulation runs for a given accuracy.

In literatures [1, 2, 3, 15, 16], There are basically three methods to construct the IS density, i.e., *mean translation*, *variance scaling* and *exponential twisting*. In this thesis, we have tried both the mean-translation method and the variance-scaling method, and demonstrated that such methodologies indeed lead to a substantial saving in simulation time, as compared to MC simulations.

II.2 Simulation Channels

There will be several simulation channels tried in our simulations, which are introduced in sequence.

In the beginning, we test the *stationary biased channel* as shown in Figure 3.2. The all-zero codeword is BPSK-modulated, and induce the received vector $(-1+n_{\text{default}}, -1+n_{\text{default}}, -1+n_{\text{default}}, ..., -1+n_{\text{default}})_{108}$. Now instead of sending the codeword through the original channel, we used a *variance scaling* channel, which gives us the received vector $(-1+n_{\text{biasd}}, -1+n_{\text{biased}}, -1+n_{\text{biased}}, ..., -1+n_{\text{biased}})_{108}$, where n_{biased} is Gaussian distributed with zero mean and different variance from n_{default}. Usually, this biased channel is "noisier" than the default AWGN channel in the sense that $\text{Var}[n_{\text{default}}] < \text{Var}[n_{\text{biased}}]$. For convenience, we will refer to this variance-scaling channel as a biased *Stationary Channel* in the next chapter.

In terms of mean-shifts, we design three non-stationary channels. The first non-stationary channel is designed based on the error events that arise decision $\boldsymbol{x}' = (000...01)_{36}$ whose

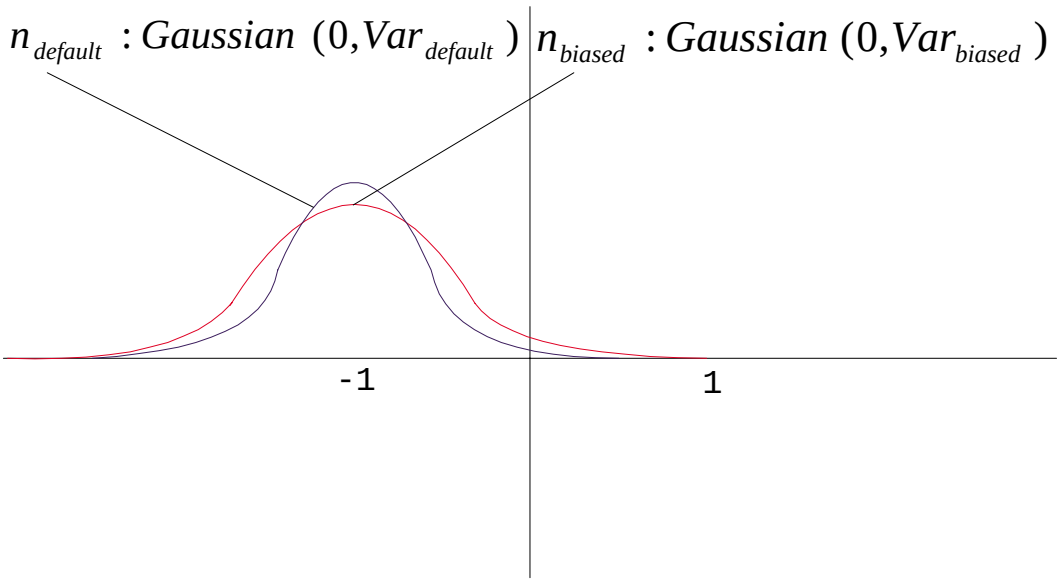

Figure 3.2: The densities of the original channel and the stationary biased channel.

codeword is $(000...0111)_{108}$ (note that the transmitted information bits are assumed to be all zeros.) Hence, in our simulations, we send the BPSK-modulated codeword "$(000...00)_{108}$" and yield the received vector

$$(-1+n_{\text{default}}, -1+n_{\text{default}}, \ldots, -1+n_{\text{default}}, -1+n_{\text{biased}}, -1+n_{\text{biased}}, -1+n_{\text{biased}})_{108}$$

instead of

$$(-1+n_{\text{default}}, -1+n_{\text{default}}, \ldots, -1+n_{\text{default}}, -1+n_{\text{default}}, -1+n_{\text{default}}, -1+n_{\text{default}})_{108},$$

where the zero-mean Gaussian distributed n_{biased} has larger variance than n_{default}. The key different between the original channel and the first non-stationary channel is that the noise variance has been enlarged for those positions with code bit being equal to 1. For convenience, we refer to this channel as "*The first non-stationary channel*" in the sequel.

Again, for the error events that arise decision $\boldsymbol{x}' = (000...01)_{36}$, we design the second non-stationary channel. To be specific, the received vector corresponding to the transmitted

all-zero information bit is $(-1+n_{\text{default}}, -1+n_{\text{default}}, -1+n_{\text{default}}, ..., -1+n_{\text{default}}, 0+n_{\text{biased}}, 0+n_{\text{biased}}, 0+n_{\text{biased}})_{108}$. In a word, there is a mean-shift 1 at those positions at which the code bit is 1. This can further increase the occurrence frequency of the error events corresponding to previously defined erroneous decision $\boldsymbol{x}'$. The resultant density is illustrated in Figure 3.3. For convenience, we name such a channel "*the second non-stationary channel.*"

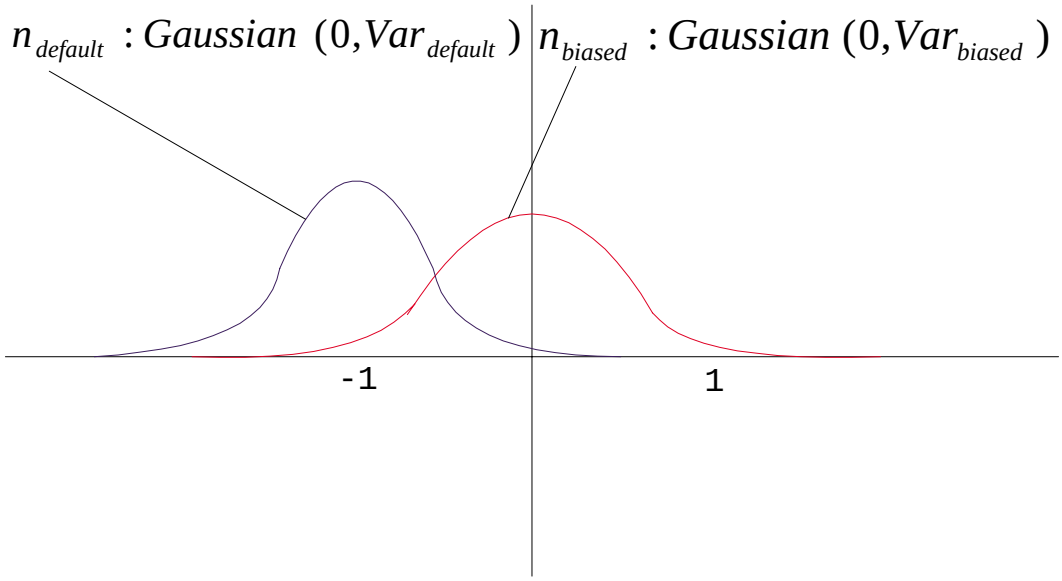

Figure 3.3: The density of the second non-stationary channel.

To further increase the occurrence of the previous error events, we design the third non-stationary channel as the received vector becomes $(-1+n_{\text{default}}, -1+n_{\text{default}}, -1+n_{\text{default}}, ..., -1+n_{\text{default}}, +1+n_{\text{biased}}, +1+n_{\text{biased}}, +1+n_{\text{biased}})_{108}$. In short, the mean-shift is increased up to 2. The density is shown in Figure 3.4. For convenience, we refer to such a channel as "*the third non-stationary channel.*"

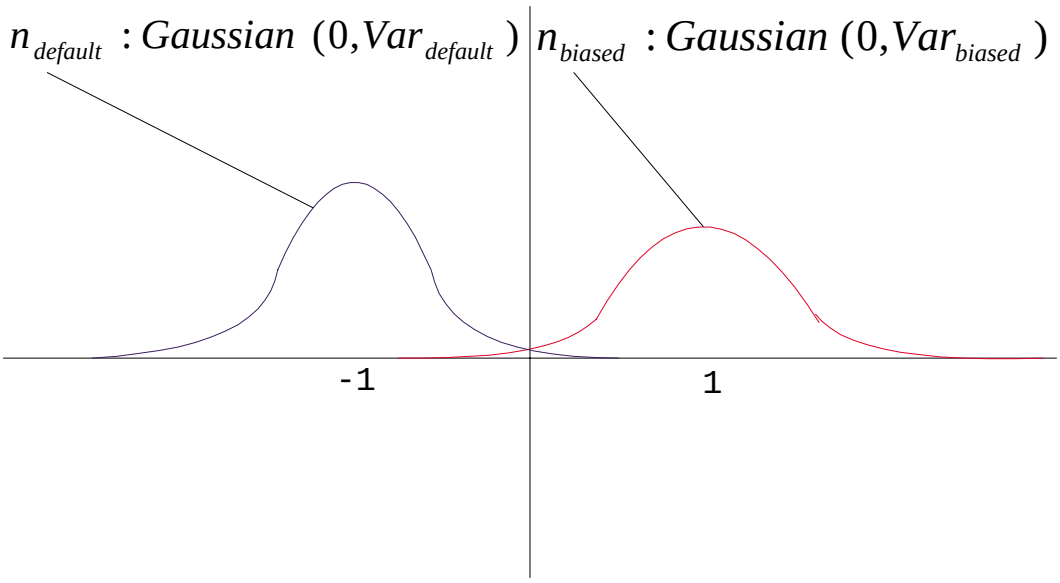

Figure 3.4: The density of the third non-stationary channel.

Chapter 4

Simulation Results

In this chapter, we present the simulation results of both stationary and non-stationary channels, and compare them to Monte Carlo simulations for the same codes.

I Stationary Channels

Figure 4.1 shows the estimation of bit error probability at SNR=8dB for different estimators including both IS and MC. The line marked "Monte Carlo (5.3e-06 5.80 161673235)" indicates the MC simulation without performing code truncation, and the quantities inside the parenthesis respectively represent P_b=5.3e-6, RPE=5.80, and the number of metric computations taken is 161673235 after collecting 300 error information bits. Note that the MC simulation stops when 300 error information bits are collected. The result of MC simulation will serve as a reference to the following IS simulations.

The line marked "Important Sampling SNR=5dB Var=0.4743" indicates the IS simulation results responding to the 8 dominant error events. All the remaining error events are ignored. From our experiments, the error probability is indeed determined by the top 8 dominant error events. Also, the noise variance of the IS channel is enlarged to SNR=5dB. The P_b is calculated (and re-calculated) in every 8000 simulation runs. From the figure,

we observe that the larger the noise variance of the biased channel, the slower the rate of convergence to P_b.

Figure 4.2 shows the *Relative Precision Estimations* for those simulations shown in Figure 4.1. Surely, the more the simulation runs, the smaller the RPE. Also, we observe that the larger the variance of the biased noise, the larger the RPE.

Because the MLSDA-Turbo decoder operating over the code tree, its computation time varies with the noise power. A more accurate estimate on the simulation time should be based on the number of metric computations taken during simulations. From Figure 4.3, we found that the larger the variance of the biased noise is, the more metric computations will be taken for IS simulations. This is because of the characteristic of MLSDA-Turbo decoder, i.e., a large noise variance will induce more metric computations. Hence, to reach the same RPE, the IS simulation based on stationary biased channels surprisingly require even more simulation time to reach the same RPE as the MC simulations.

Figures 4.4, 4.5 and 4.6 conceptually illustrates the same simulation quantities as Figure 4.1, 4.2 and 4.3, except that the true SNR becomes 3.0dB. Notably from Figure 4.4, the top 8 error events are not very dominant in error probabilities for SNR=3dB. That is why the IS estimation of P_b is less than that of the MC estimation. Similar to the case of true SNR=8dB, we need even more metric computations for IS estimators with stationary biased channels to reach the same RPE than the MC estimator.

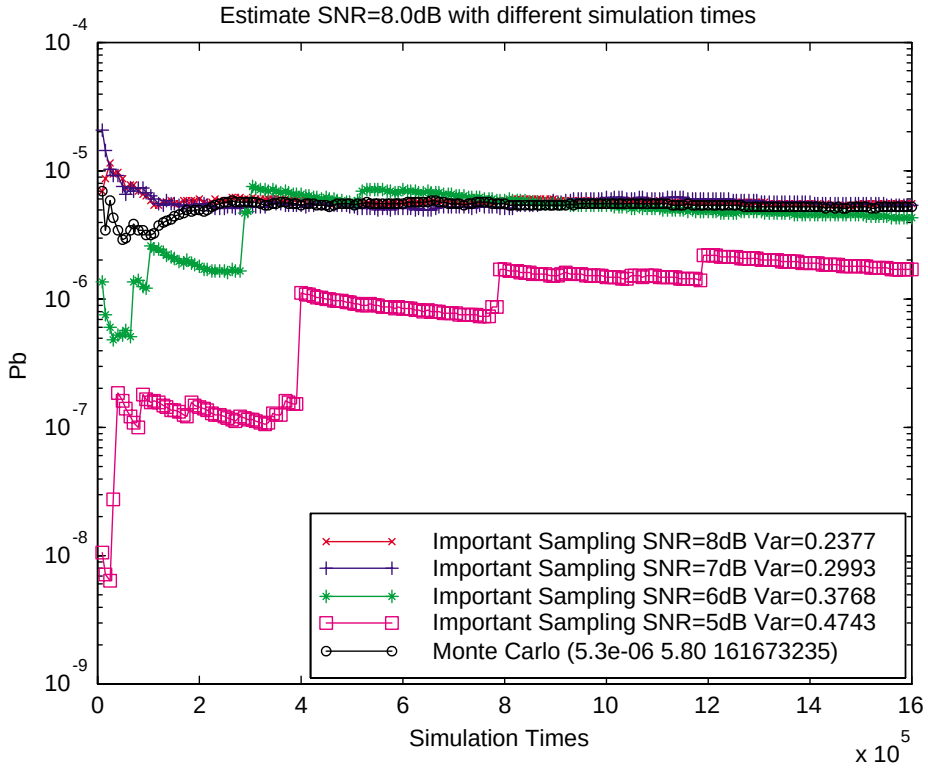

Figure 4.1: The estimation of bit error probabilities at SNR=8dB with respect to the "*stationary channels*."

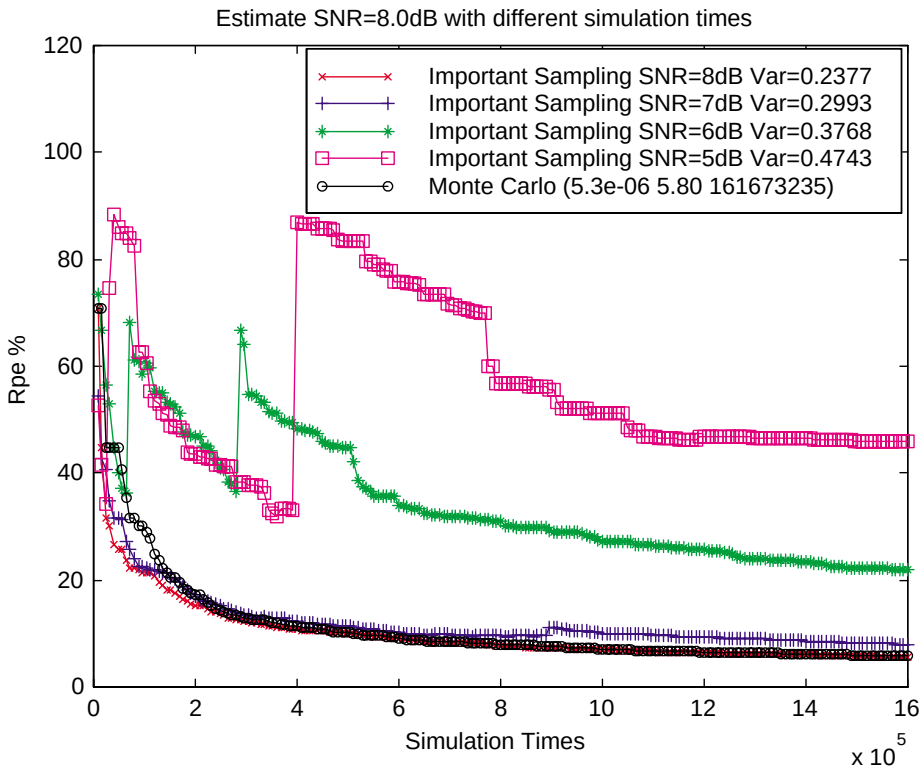

Figure 4.2: The Relative Precision Estimation at SNR=8dB with respect to the "*stationary channels*."

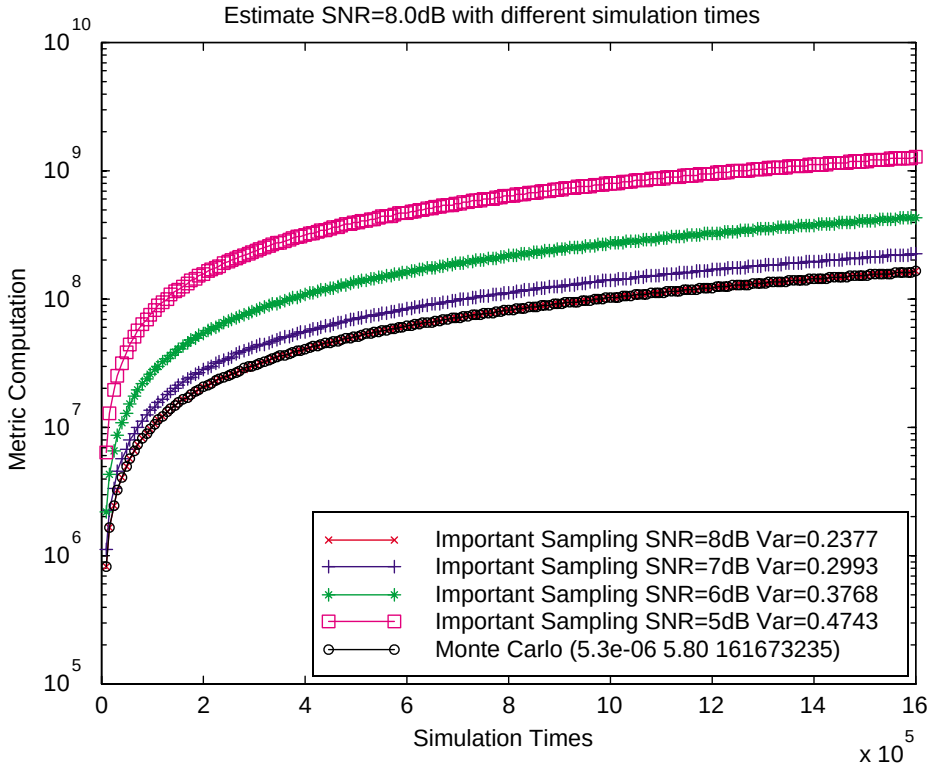

Figure 4.3: The number of metric computation taken at SNR=8dB with respect to the "*stationary channels*."

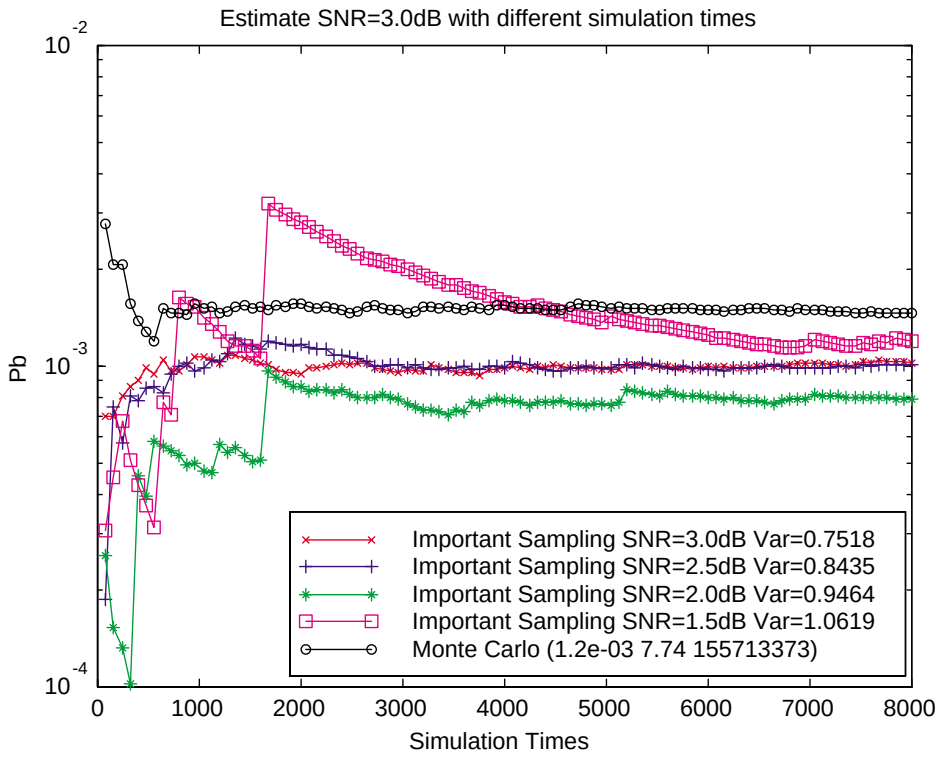

Figure 4.4: The estimation of bit error probabilities at SNR=3dB with respect to the "*stationary channels*."

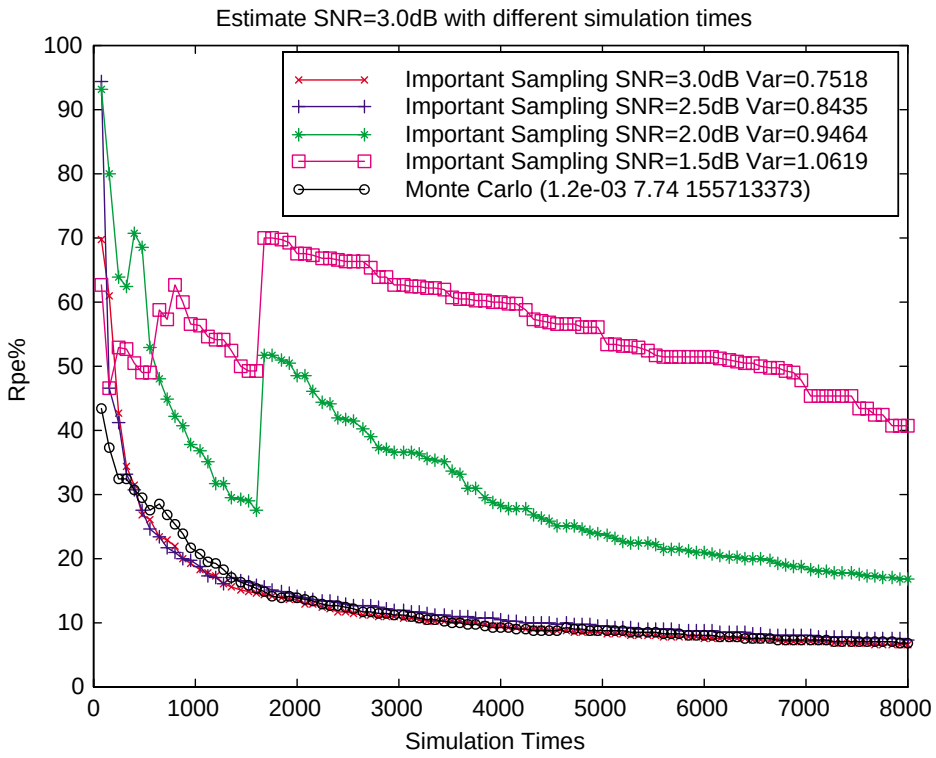

Figure 4.5: The Relative Precision Estimation at SNR=3dB with respect to the "*stationary channels*."

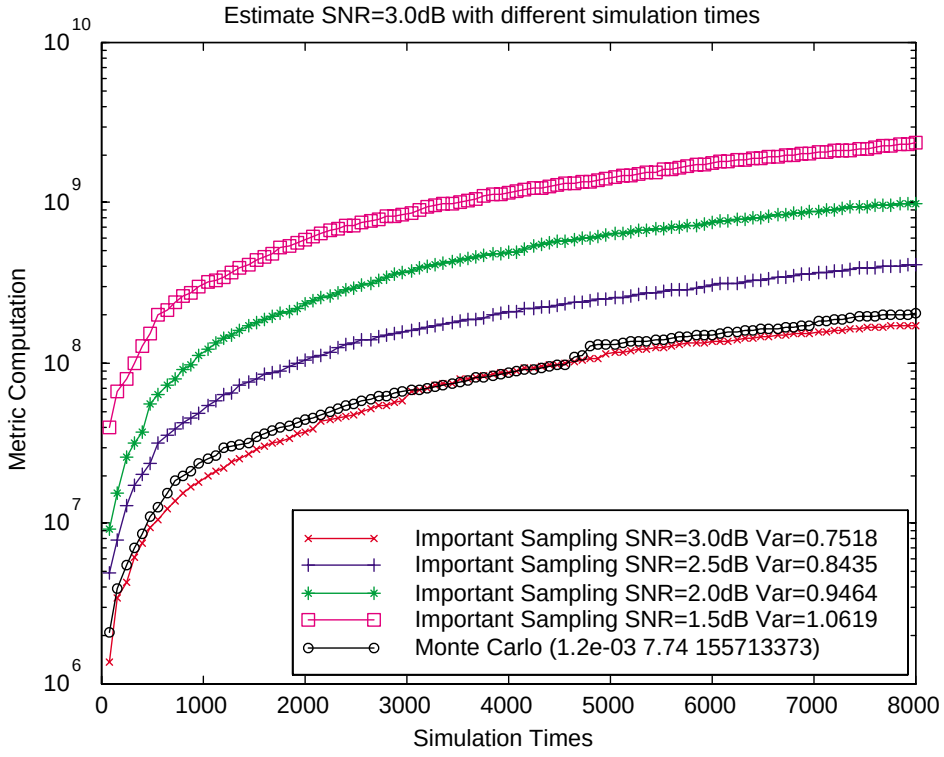

Figure 4.6: The number of metric computation taken at SNR=3dB with respect to the "*stationary channels*."

	$MC_{SNR=8dB}$	$IS_{SNR=6dB}$	$IS_{SNR=3dB}$
P_b	5.3×10^{-6}	6.2×10^{-6}	5.2×10^{-06}
RPE%	5.80	5.75	5.84
Complexity	161673235	125344768	47607830
Ratio	1	0.7753	0.2945

Table 4.1: The performance comparisons among MC simulator and IS simulators using the first non-stationary channel with different biased noise variance, subject to comparable RPE. The relative computational complexity with respect to MS simulator is also listed.

II Non-stationary Channels

In this section, we present the IS simulation results with respect to three non-stationary channels designed in the previous chapter.

II.1 The first non-stationary channel

The simulation results for true SNR=8dB are illustrated in Figures 4.7, 4.8, 4.9, which are respectively the estimated error probability, the RPE, and the computational complexity. Table 4.1 shows the computation efforts taken and their resultant estimated error probabilities for different simulators under comparable RPE.

From Table 4.1, we found that the IS simulation with larger biased noise variance is more efficient in estimating P_b than one with smaller noise variance. Nonetheless, the computational gain, defined as the relative computational complexity with respect to the MS simulator under the same RPE, is not big.

When reducing the true noise variance to SNR=3dB, we obtained the simulation results in Figures 4.10, 4.11 and 4.12. Again, Table 4.2 puts the computation efforts taken and their resultant estimated error probabilities for different simulators under comparable RPE.

From Table 4.2, we again found that the IS simulation with the larger biased noise

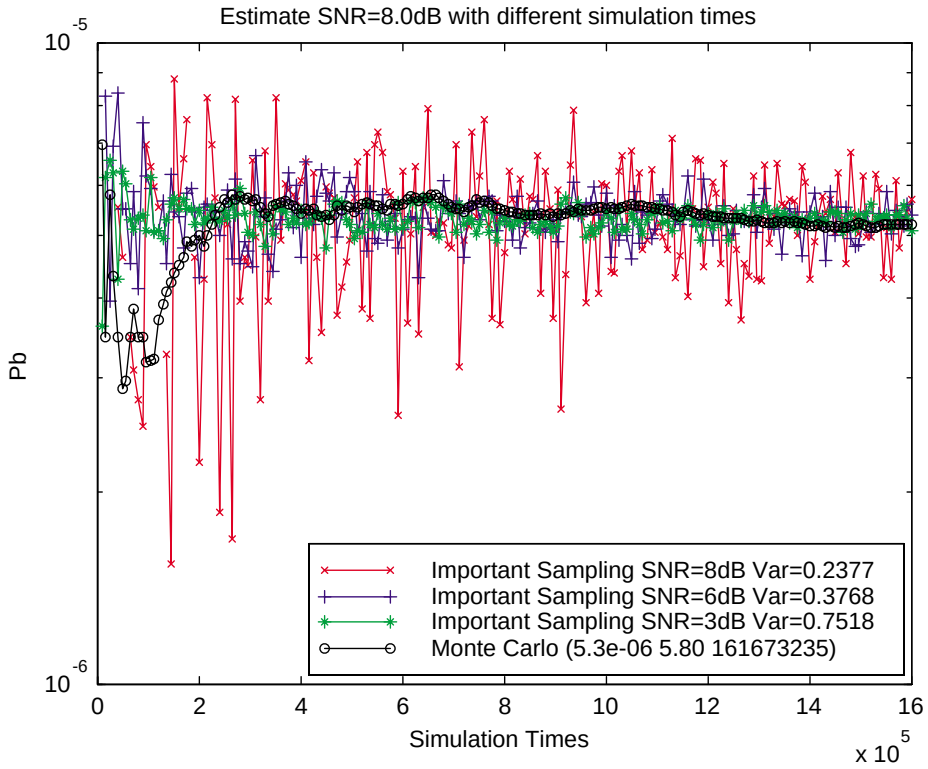

Figure 4.7: The estimation of bit error probabilities at SNR=8dB with respect to the "*first non-stationary channels.*"

variance is more efficient that the one with smaller biased noise variance. The computational gain is even lesser that the case of true SNR=8dB.

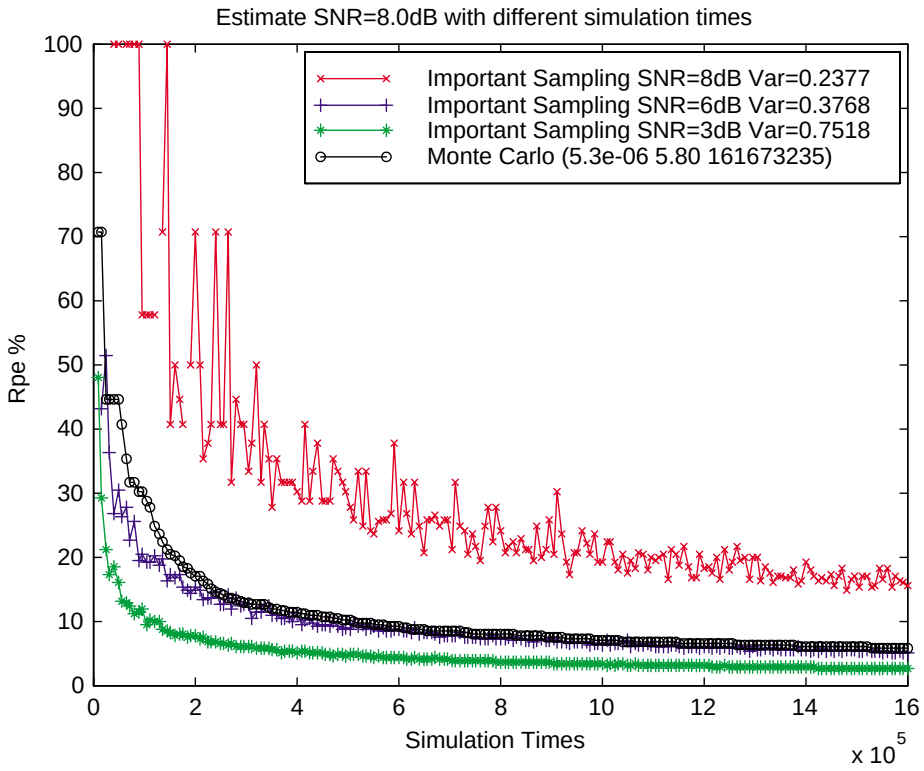

Figure 4.8: The Relative Precision Estimation at SNR=8dB with respect to the "*first non-stationary channels.*"

	$\text{MC}_{\text{SNR=3dB}}$	$\text{IS}_{\text{SNR=3dB}}$	$\text{IS}_{\text{SNR=1.5dB}}$
P_b	1.2×10^{-3}	1.1×10^{-3}	1.0×10^{-3}
RPE%	7.74	17.4	17.2
Complexity	155713373	182182136	113213118
Ratio	1	1.17	0.7271

Table 4.2: The performance comparisons among MC simulator and IS simulators using the first non-stationary channel with different biased noise variance, subject to comparable RPE. The relative computational complexity with respect to MS simulator is also listed.

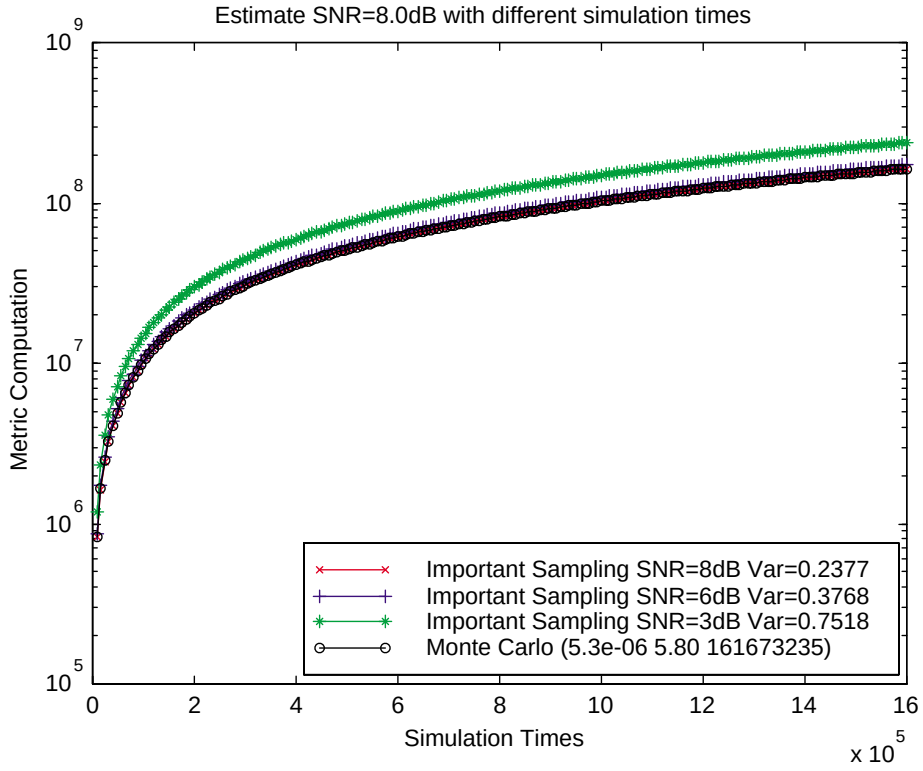

Figure 4.9: The number of metric computation taken at SNR=8dB with respect to the "*first non-stationary channel*."

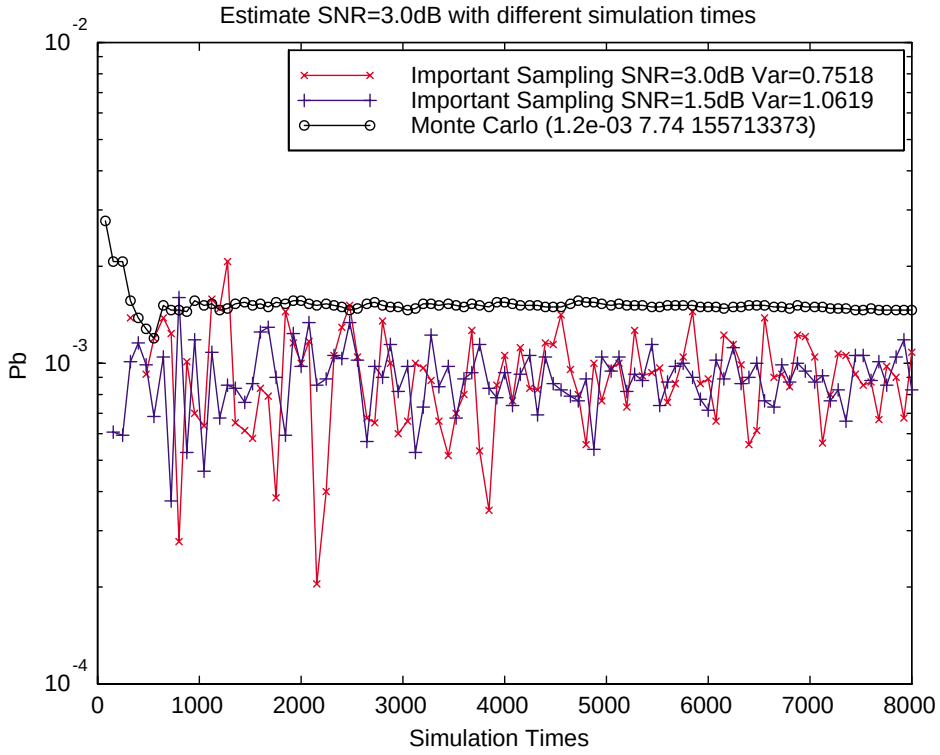

Figure 4.10: The estimation of bit error probabilities at SNR=3dB with respect to the "*first non-stationary channels.*"

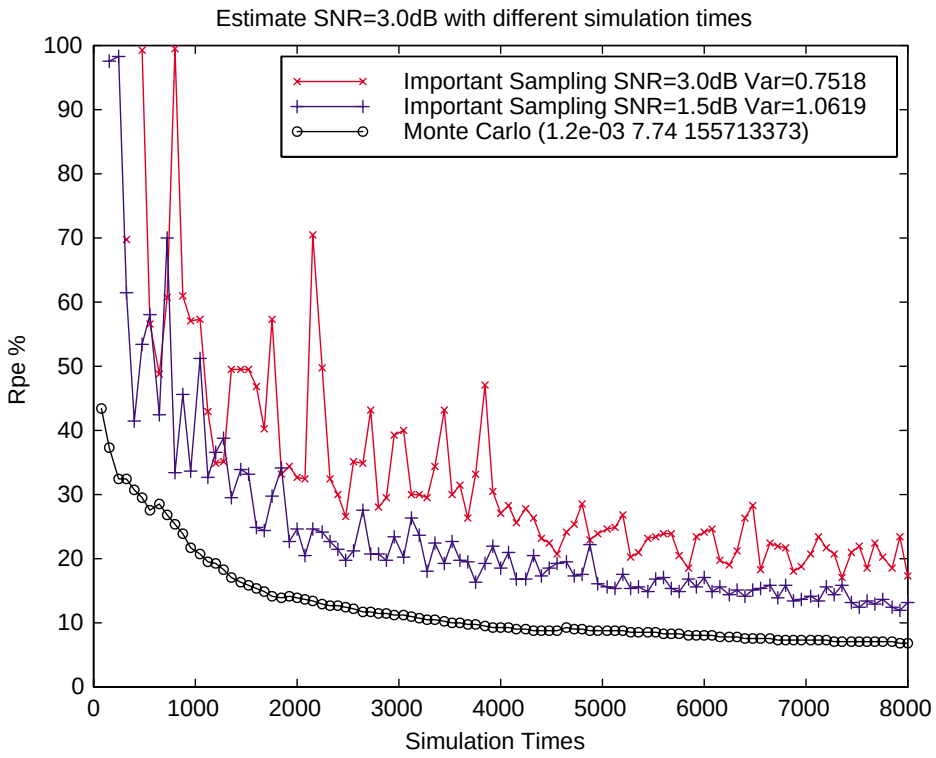

Figure 4.11: The Relative Precision Estimation at SNR=3dB with respect to the "*first non-stationary channels*."

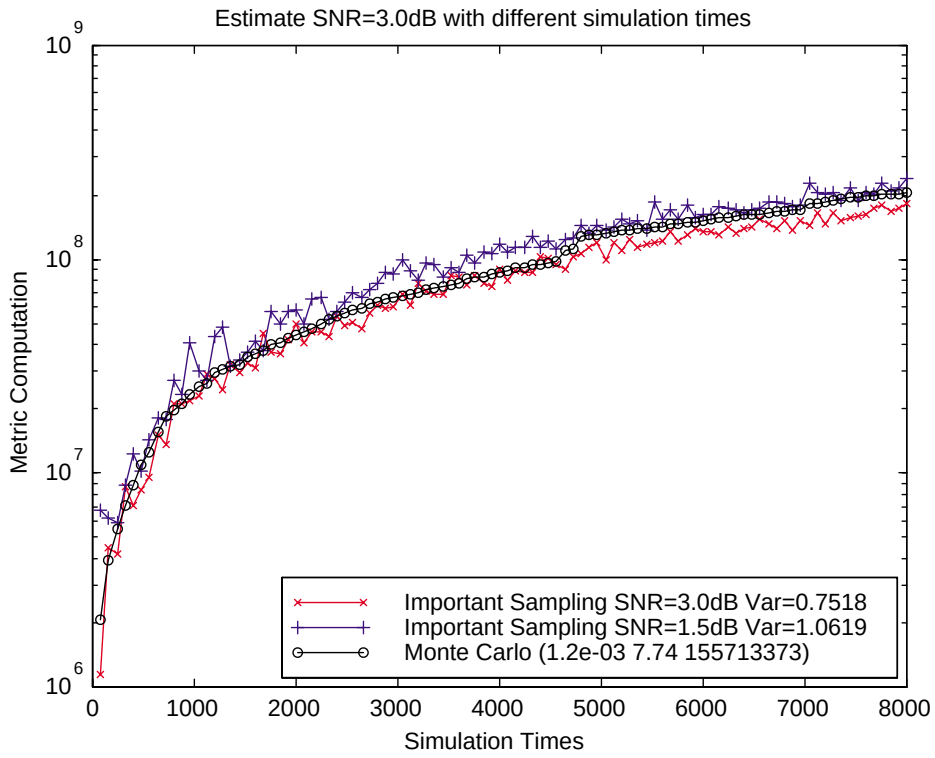

Figure 4.12: The number of metric computation taken at SNR=3dB with respect to the "*first non-stationary channel*."

	$\text{MC}_{\text{SNR=8dB}}$	$\text{IS}_{\text{SNR=8dB}}$	$\text{IS}_{\text{SNR=6dB}}$	$\text{IS}_{\text{SNR=3dB}}$
P_b	5.3×10^{-6}	5.2×10^{-6}	5.4×10^{-6}	5.2×10^{-6}
RPE%	5.80	5.78	5.77	5.76
Complexity	161673235	1947871	4107672	19117489
Ratio	1	0.01205	0.02541	0.11825

Table 4.3: The performance comparisons among MC simulator and IS simulators using the second non-stationary channel with different biased noise variance, subject to comparable RPE. The relative computational complexity with respect to MS simulator is also listed.

	$\text{MC}_{\text{SNR=3dB}}$	$\text{IS}_{\text{SNR=3dB}}$	$\text{IS}_{\text{SNR=1.5dB}}$
P_b	1.2×10^{-3}	1.0×10^{-3}	9.8×10^{-4}
RPE%	7.74	7.79	7.75
Complexity	155713373	76220279	139841615
Ratio	1	0.4895	0.8981

Table 4.4: The performance comparisons among MC simulator and IS simulators using the second non-stationary channel with different biased noise variance, subject to comparable RPE. The relative computational complexity with respect to MS simulator is also listed.

II.2 The second non-stationary channel

Figures 4.13, 4.14, 4.15 shows the IS simulation results for the second non-stationary channels. Also, under (almost) the same RPE, we quoted the computational efforts of MC simulator and IS simulators, as well as their estimated P_b in Table 4.3.

We conclude from Table 4.3 that the IS simulators with smaller biased noise variance causes is more efficient that the one with larger biased noise variance. This conclusion is different from the one drawn from the IS simulation results with respect to the first non-stationary channels. The computational gain in this case is improved. One can save around 98.9% computational efforts by using such an IS estimator.

Now we turn to the situation where the true noise gives SNR=3dB. The simulation results for estimated bit error rates, RPEs and computational efforts are respectively depicted in Figures 4.16, 4.17 and 4.18. Furthermore, Table 4.4 lists the computational efforts for different estimators subject to the same RPE.

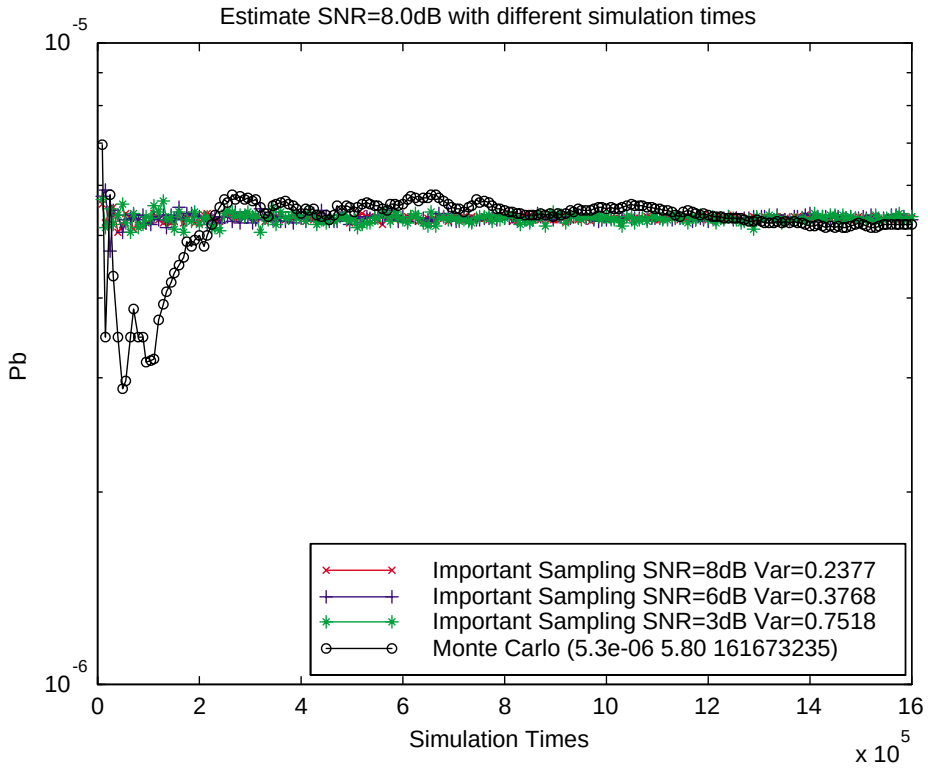

Figure 4.13: The estimation of bit error probabilities at SNR=8dB with respect to the "*second non-stationary channels.*"

The table shows that the IS simulation with small biased noise variance is more efficient that the one with large biased noise variance. Also note that the computational gains in Table 4.4 are less than those in Table 4.3. These results conclude that the IS estimator based on the second non-stationary channels is more efficient than the one based on the first non-stationary channels.

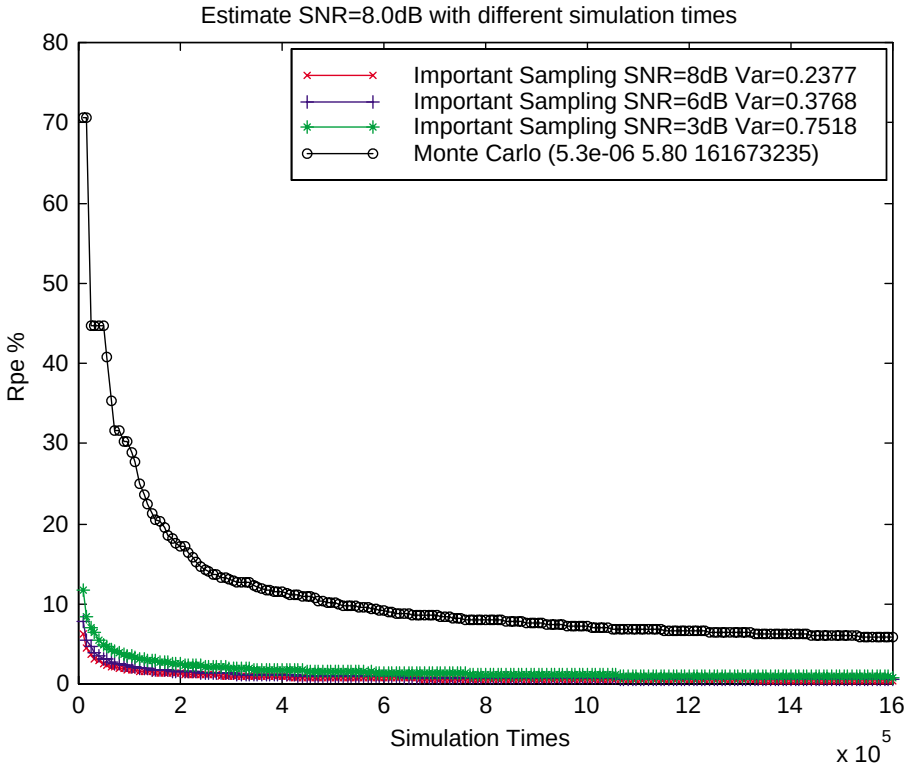

Figure 4.14: The Relative Precision Estimation at SNR=8dB with respect to the "*second non-stationary channels*."

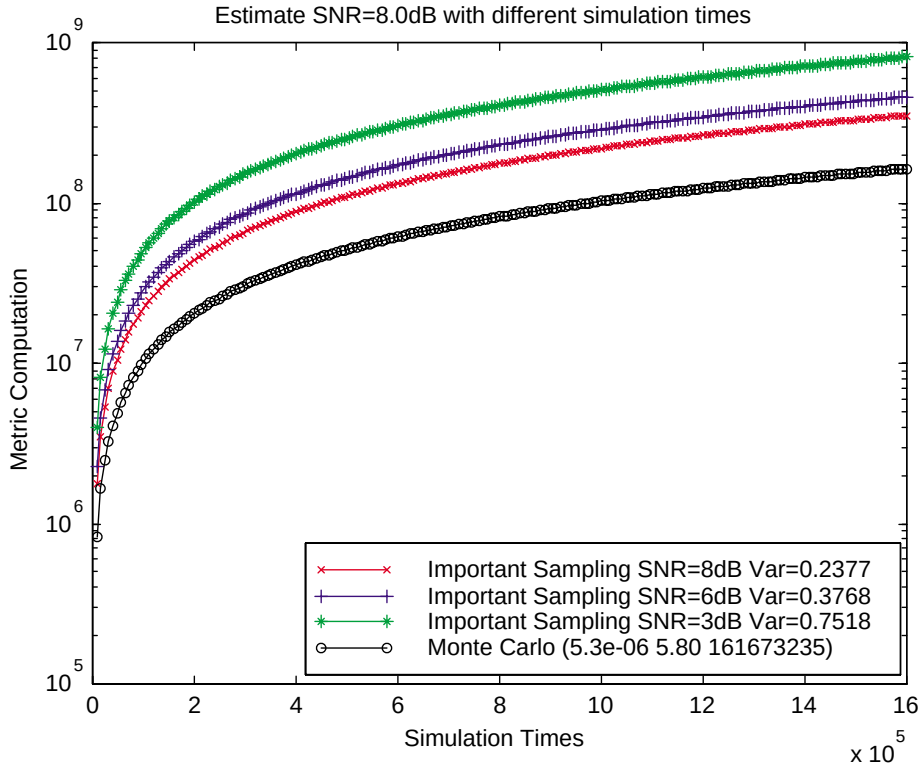

Figure 4.15: The number of metric computation taken at SNR=8dB with respect to the "*second non-stationary channel*."

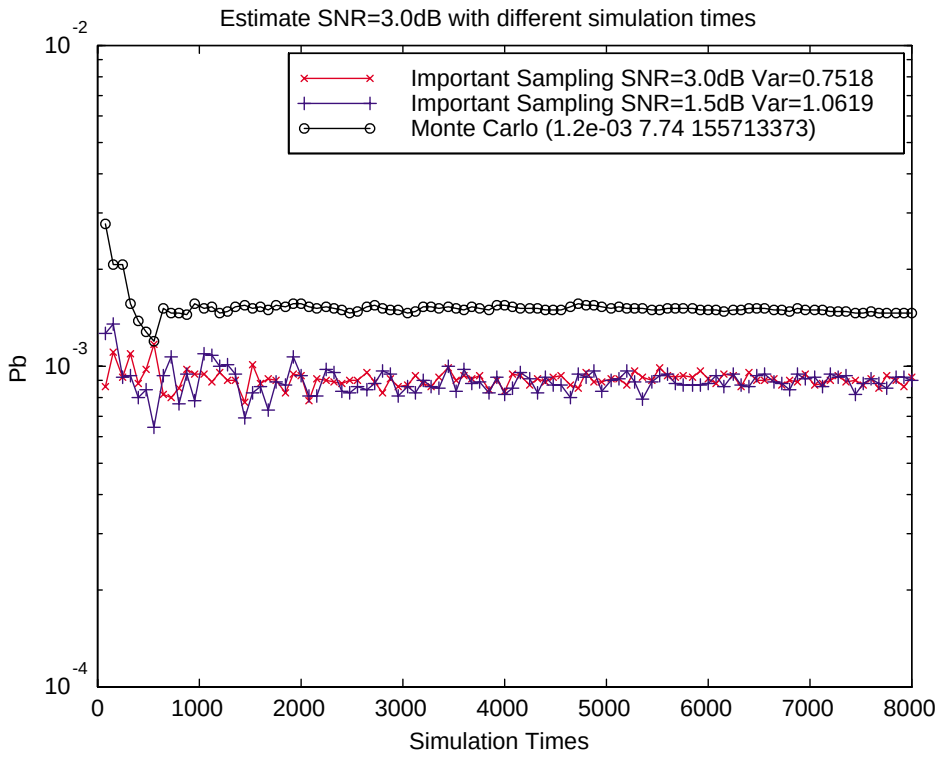

Figure 4.16: The estimation of bit error probabilities at SNR=3dB with respect to the "*second non-stationary channels*."

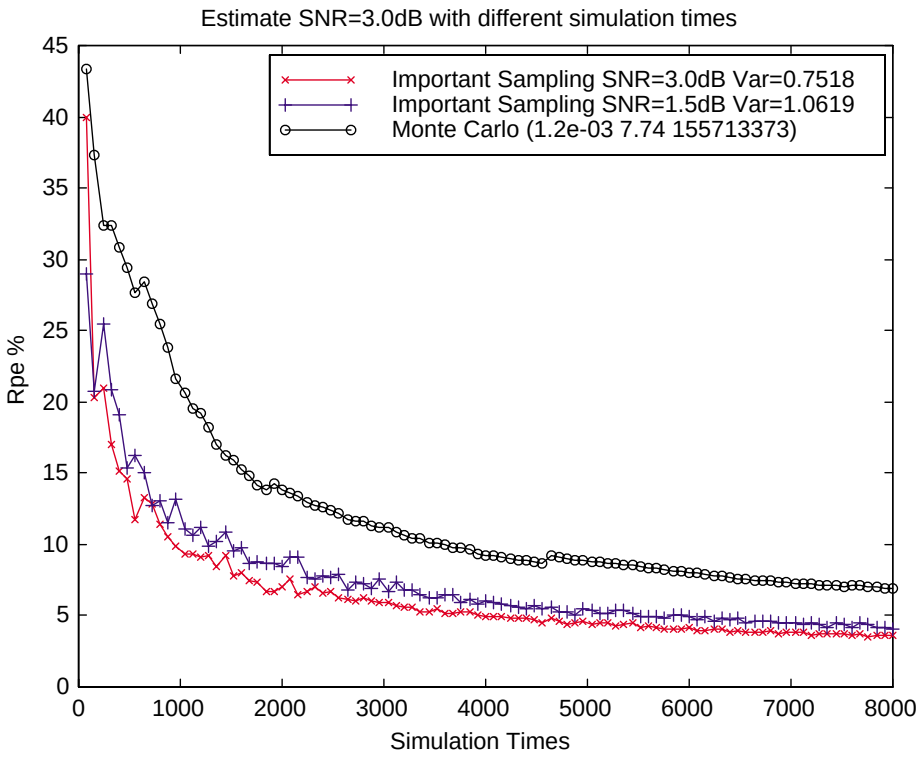

Figure 4.17: The Relative Precision Estimation at SNR=3dB with respect to the "*second non-stationary channels*."

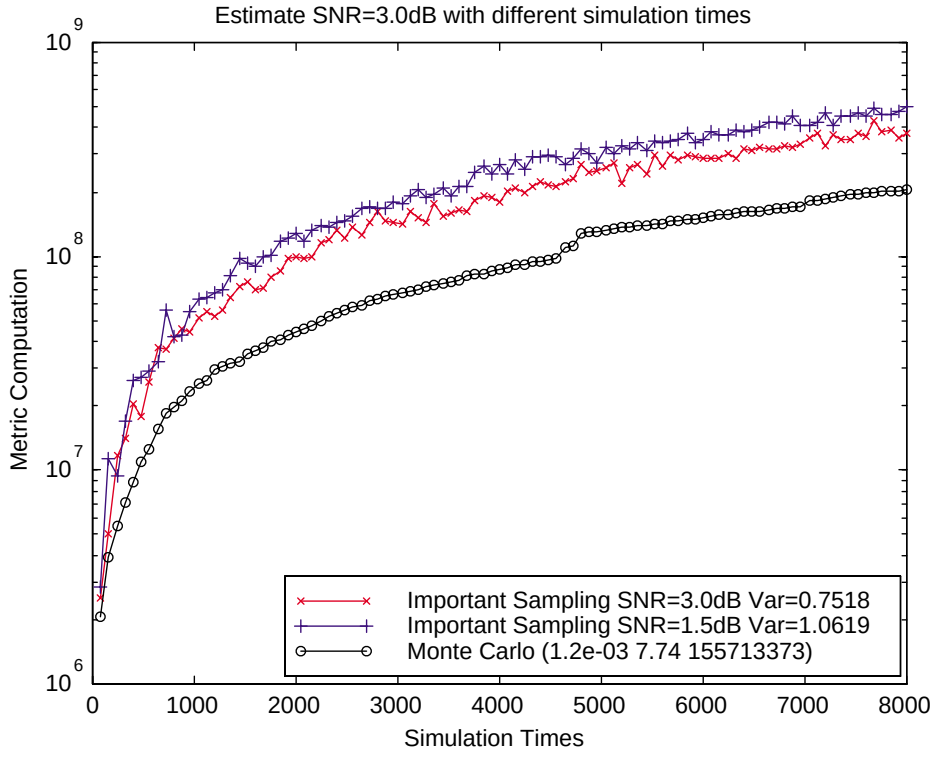

Figure 4.18: The number of metric computation taken at SNR=3dB with respect to the "*second non-stationary channel*."

	$\text{MC}_{\text{SNR=8dB}}$	$\text{IS}_{\text{SNR=6dB}}$	$\text{IS}_{\text{SNR=3dB}}$
P_b	5.3×10^{-6}	5.3×10^{-6}	5.4×10^{-6}
Rpe %	5.80	5.83	5.85
Complexity	161673235	76910196	36824995
Ratio	1	0.4757	0.2278

Table 4.5: The performance comparisons among MC simulator and IS simulators using the third non-stationary channel with different biased noise variance, subject to comparable RPE. The relative computational complexity with respect to MS simulator is also listed.

	$\text{MC}_{\text{SNR=3dB}}$	$\text{IS}_{\text{SNR=3dB}}$	$\text{IS}_{\text{SNR=1.5dB}}$
P_b	1.2×10^{-3}	9.5×10^{-4}	9.8×10^{-4}
Rpe %	7.74	10.9	10.7
Complexity	155713373	171978318	148068555
Ratio	1	1.1045	0.9509

Table 4.6: The performance comparisons among MC simulator and IS simulators using the third non-stationary channel with different biased noise variance, subject to comparable RPE. The relative computational complexity with respect to MS simulator is also listed.

II.3 The third non-stationary channel

Figures 4.19, 4.20, 4.21 respectively summarize the error probabilities, the RPEs and computational efforts for the IS estimators based on the third non-stationary channels. The true SNR equals 8dB. We also place in Table 4.5 that the computational efforts for different estimators subject to the same RPE.

We conclude from the table that find the IS simulator with large biased noise variance is more efficient that the one with small biased noise variance. In general, the computational gain using the third non-stationary channel is less than that using the second non-stationary channel.

Now after reducing the true SNR to 3dB, we re-do the simulations for the previous three figures, and yield Figures 4.22, 4.23 and 4.24. Also, the computational efforts for different estimators subject to the same RPE are summarized in Table 4.6.

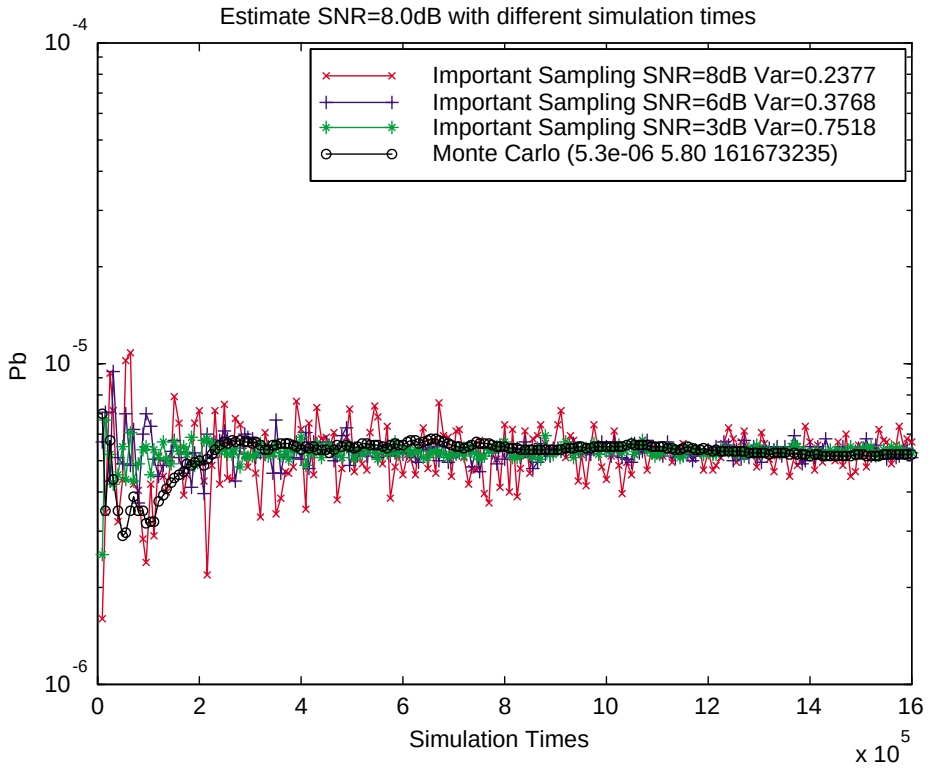

Figure 4.19: The estimation of bit error probabilities at SNR=8dB with respect to the "*third non-stationary channels.*"

Similar to the previous result under true SNR=8dB, we observe from the above table that the IS simulation with large biased noise variance is more efficient than the one with small biased noise variance. However, there is in general no computational gain in such case.

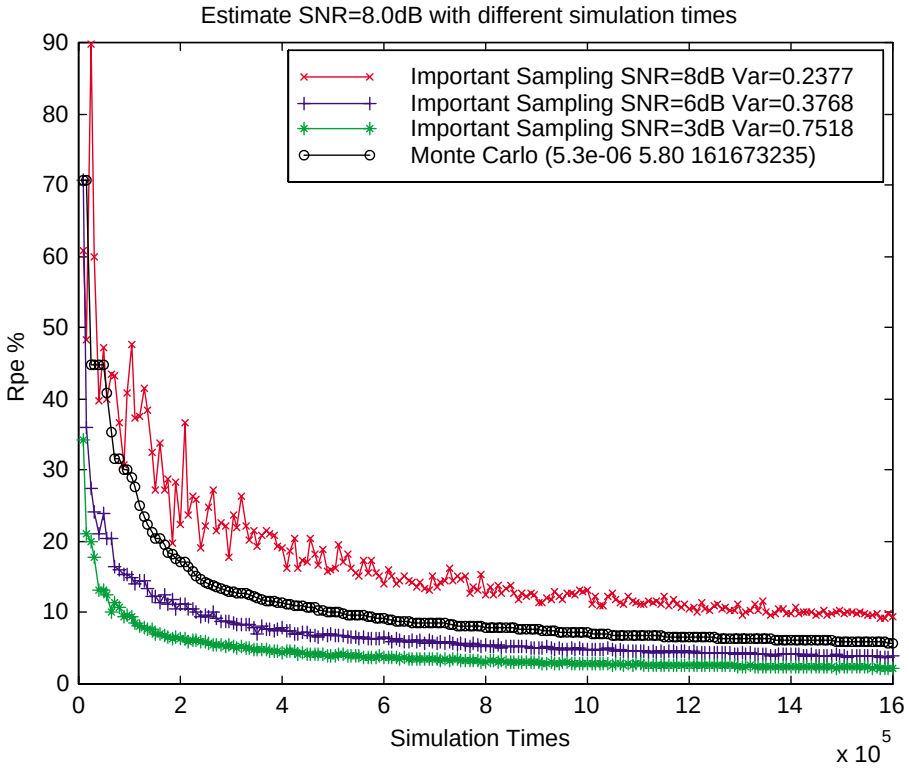

Figure 4.20: The Relative Precision Estimation at SNR=8dB with respect to the "*third non-stationary channels*."

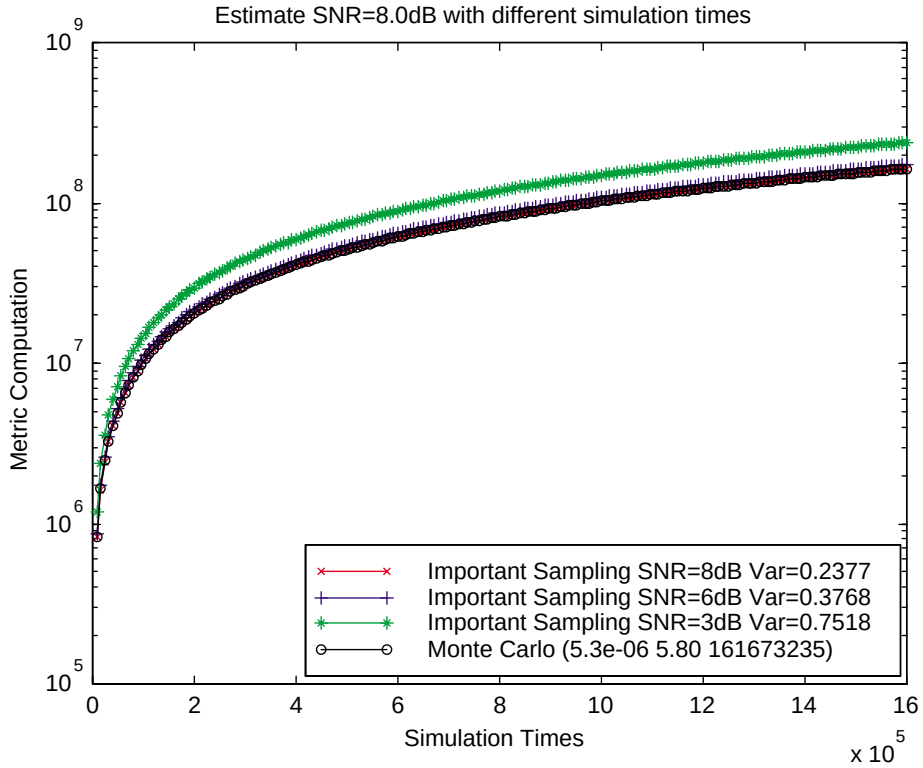

Figure 4.21: The number of metric computation taken at SNR=8dB with respect to the "*third non-stationary channel*."

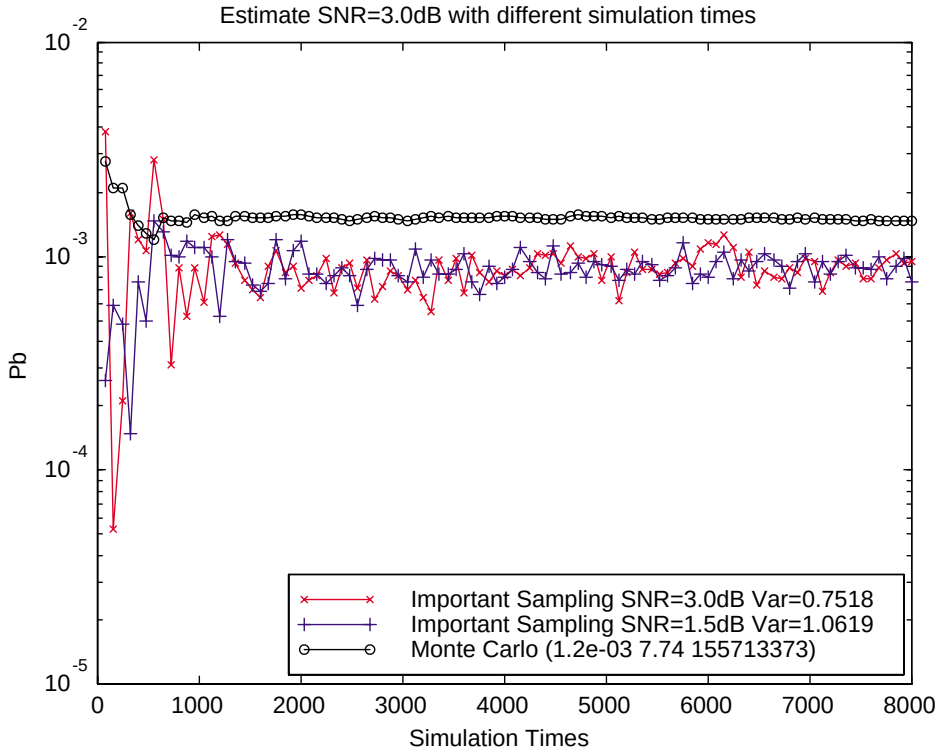

Figure 4.22: The estimation of bit error probabilities at SNR=3dB with respect to the "*third non-stationary channels*."

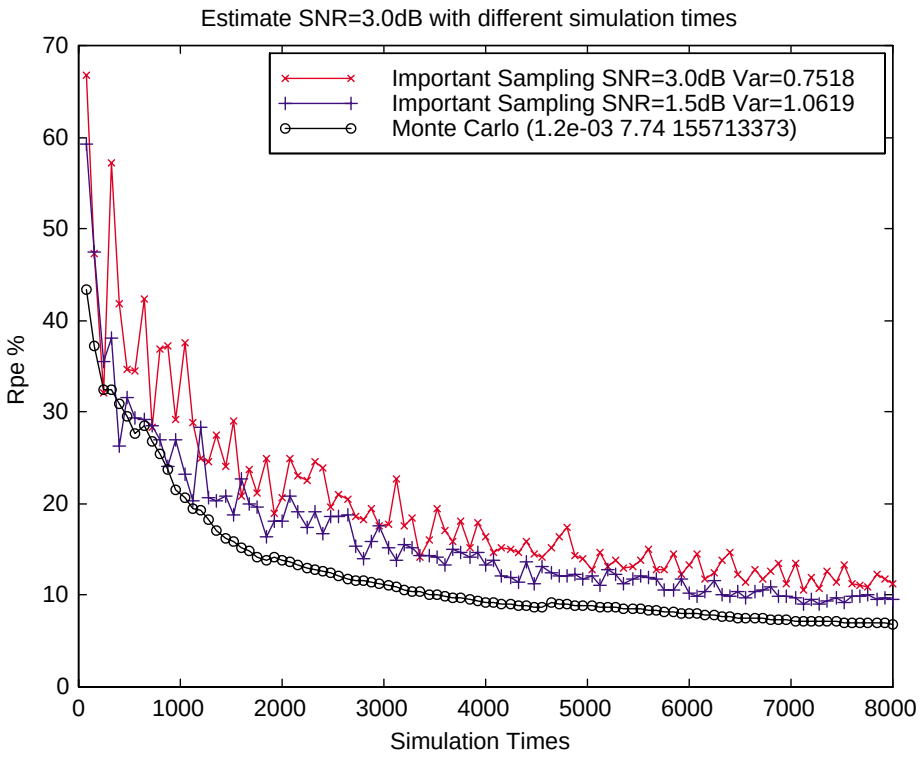

Figure 4.23: The Relative Precision Estimation at SNR=3dB with respect to the "*third non-stationary channels*."

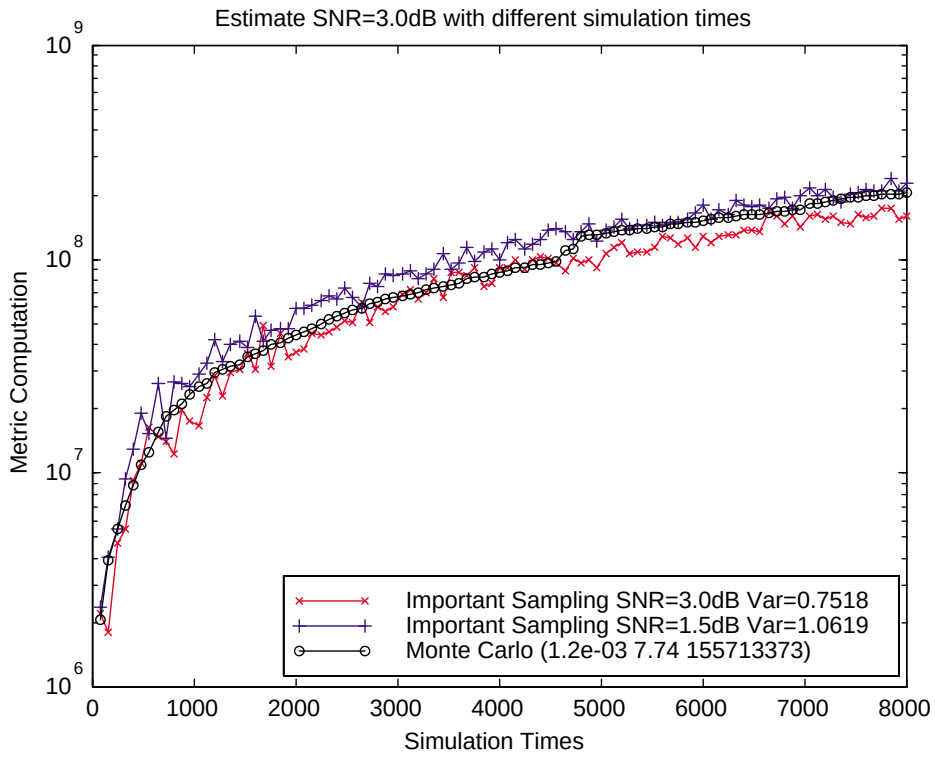

Figure 4.24: The number of metric computation taken at SNR=3dB with respect to the "*third non-stationary channel*."

Monte Carlo Simulation for recording at 300 bit errors									
SNR(dB)	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
Pb	6.5e-02	3.8e-02	2.4e-02	9.8e-03	4.8e-03	2.2e-03	1.2e-03	7.6e-04	4.5e-04
Rpe	17.2	17.2	16.7	14.2	13.4	9.83	7.74	6.87	6.26
Com	222276385	192710434	206555768	215889806	208319093	185857455	155713373	93004594	63063107
Importance Sampling Simulation for recording at the same RPE									
SNR(dB)	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
Pb	1.2e-02	9.3e-03	4.2e-03	3.5e-03	2.8e-03	1.3e-03	1.0e-03	6.2e-04	4.3e-04
Rpe	17.3	17.2	16.3	14.3	13.3	9.76	7.79	6.89	6.26
Com	206742239	190742239	181950128	163045159	89587250	98307554	76220279	50410610	32072173
Gain	0.93011	0.98979	0.88088	0. 75522	0.43005	0.52894	0.48949	0.54202	0.50857

Figure 4.25: The estimation of the bit error probability in terms of the MC estimator and the IS estimator based on the second non-stationary channel at SNR=0∼4dB.

III Estimation of bit error probability in terms of the best biased channel

Based on the previous simulations for SNR=3dB and SNR=8dB, we note that the IS simulator based on the second non-stationary channels can reach the same RPE using the least number of metric computations. We can therefore estimate the bit error probability of the MLSDA-Turbo decoder in terms of such IS estimator.

Figures 4.25 and 4.26 show the IS estimated bit error rate for true SNR=0∼9dB. The MC estimations are also listed for ease of comparisons. The results hint that higher computational gain can be achieved at high SNR; however, there is less or even no computational gain at low SNR.

Figure 4.27 graphically illustrates the estimated bit error probabilities listed in Figures 4.25 and 4.26.

We observed that the estimated bit error by IS simulator is smaller than that by MC simulator. This is due to that the top 8 error events we used in IS estimator is not so

Monte Carlo Simulation for recording at 300 bit errors									
4.5	5.0	5.5	6.0	6.5	7.0	7.5	8.0	8.5	9.0
2.9e-04	1.9e-04	1.1e-04	6.7e-05	3.9e-05	2.4e-05	1.0e-05	5.3e-06	2.2e-06	8.9e-07
6.22	6.03	5.92	5.86	5.84	5.81	5.78	5.80	5.76	5.76
47388940	35587019	33017961	33922090	39719359	49444037	94346705	161673235	358490311	856760117
Importance Sampling Simulation for recording at the same RPE									
4.5	5.0	5.5	6.0	6.5	7.0	7.5	8.0	8.5	9.0
2.8e-04	1.8e-04	1.1e-04	6.8e-05	4.0e-05	2.3e-05	1.1e-05	5.2e-06	2.2e-06	9.7e-07
6.22	6.09	5.96	5.80	5.84	5.81	5.81	5.78	5.77	5.79
18694896	13420903	8055117	5310740	3867645	2886998	2384527	1947871	1933822	1674667
0.39450	0.37713	0.24396	0.15656	0.09737	0.05839	0.02527	0.01205	0.00539	0.00195

Figure 4.26: The estimation of the bit error probability in terms of the MC estimator and the IS estimator based on the second non-stationary channel at SNR=4.5∼9dB.

dominant at SNR=0∼3dB; but the two estimated bit errors still have the same order of magnitude.

Finally, we estimate the P_b of the MLSDA-Turbo decoder by IS estimator for the "unreachable" region by MC simulations due to heavy computational efforts (i.e., SNR$\geq$ 9dB). The results are listed in Figures 4.28 and 4.29, subject to RPE$\approx$6%. The estimation gain in this region is expected to be larger than 500.

Here, we give a performance bound for convolutional codes which is also valid for turbo codes. If the information length for a turbo encoder is L and rate is 1/3, any codeword of this code is with length $N = 3L$. [17]

By using the union bound, the bit error rate(BER) of a finite length convolutional code with maximum-likelihood (ML) decoding over the AWGN channel with SNR of E_b/N_0 can be bounded above by

$$P_b \leq \sum_{i=1}^{2^L} \frac{w_i}{L} Q\left(\sqrt{d_i \frac{2R_c E_b}{N_0}}\right), \tag{4.1}$$

where w_i is the Hamming weight of the i^{th} information sequence, d_i is the total Hamming weight of the i^{th} codeword, and $Q(.)$ is the complimentary error function. Let the number

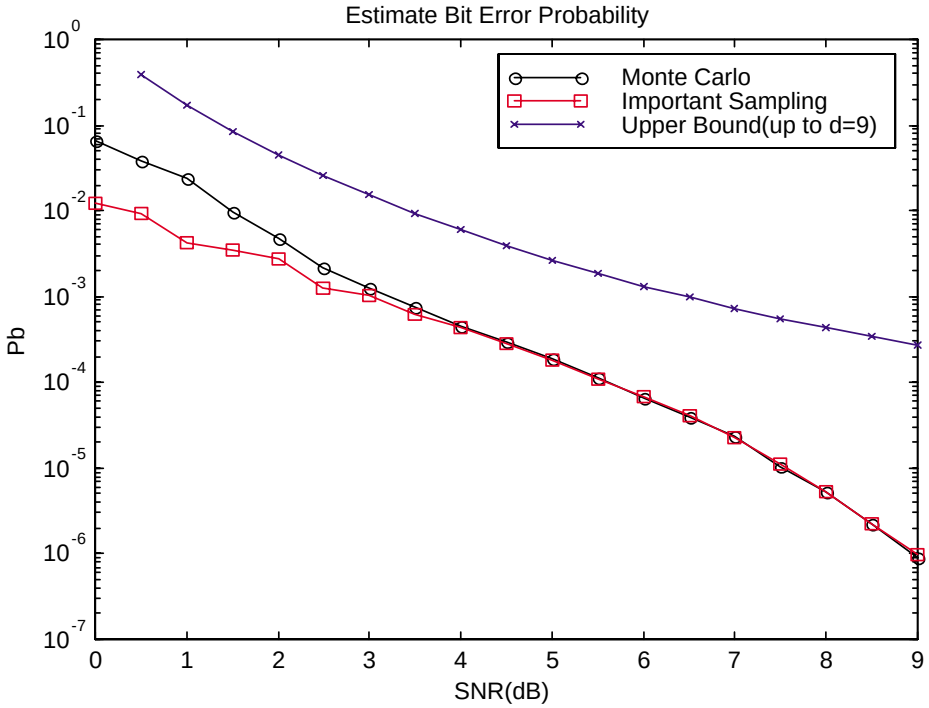

Figure 4.27: The estimated bit error probabilities for MC and IS estimators.

of codewords with Hamming weight d be denoted as H_d, and the total information weight of codewords with Hamming weight d as W_d. Define the *average information weight per codeword* as:

$$\tilde{w}_d = \frac{W_d}{H_d}. \tag{4.2}$$

Then the upper bound given above can be re-written as:

$$P_b \leq \sum_{d=d_{free}}^{3L} \frac{H_d \tilde{w}_d}{L} Q\left(\sqrt{d_i \frac{2R_c E_b}{N_0}}\right). \tag{4.3}$$

The Equation 4.3 can be used to estimate the BER of a turbo code with short length when an ML decoder is applied to.

Importance Sampling Simulation for recording at RPE=6%							
SNR	9.5	10.0	10.5	11.0	11.5	12.0	12.5
Pb	3.5e-07	1.1e-07	3.3e-08	7.9e-09	1.6e-09	2.7e-10	3.8e-11
Rpe	6.10	6.04	6.00	6.03	6.04	6.07	6.06
Com	1397725	1389150	1348576	1317392	1384507	1391025	1402515

Importance Sampling Simulation for recording at RPE=6%							
SNR	13.0	13.5	14.0	14.5	15.0	15.5	16.0
Pb	3.9e-12	3.3e-13	1.9e-14	8.5e-16	2.8e-17	5.3e-19	7.3e-21
Rpe	6.05	6.09	6.10	6.08	6.09	6.05	6.01
Com	1483599	1563825	1610578	1700387	1752832	1805472	1942095

Figure 4.28: The estimated the bit error probability by IS estimator based on the second non-stationary channel subject to RPE=6%.

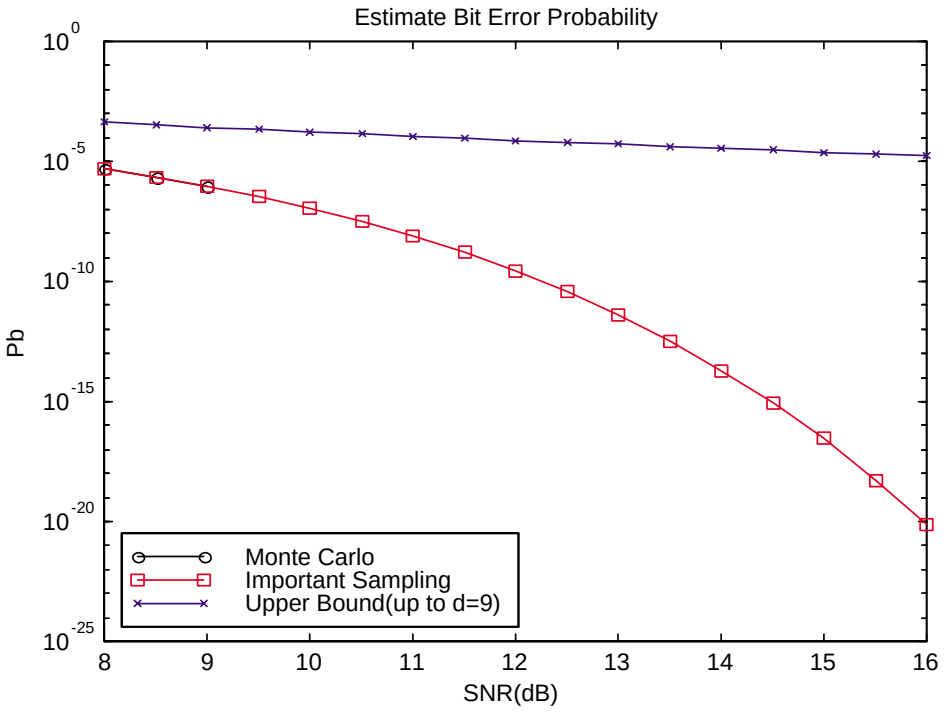

Figure 4.29: The estimated the bit error probability by IS estimator based on the second non-stationary channel subject to RPE=6%.

Chapter 5

Conclusions

In the IS simulations based on the biased stationary channel model, there is no computational gain rendered. In fact, the noisier the channel is, the more metric computations the IS estimator requires.

In the IS simulations based on the biased non-stationary channel model, which is designed based on one of the dominant error events, we found that the second non-stationary model gives us the best IS estimator, among all biased non-stationary channel models we tried. An interpretation to this result is that the second non-stationary channel model will induce more "errors" than the first one during the simulations. When compared the second non-stationary channel model to the third one, although the error events are expected to occur more frequent in the third model, its IS weight function ($w(\cdot,\cdot)$) is apparently smaller, which again yields a less efficient estimate than the second model. These observations imply that a good IS estimator has to properly balance between the increments of the occurrence of error events, and the decrements of the IS weight.

As a consequence, the best *Importance Sampling* channel we found here works well at high SNR, and a little less effective at low SNR. A computational gain of 10-500 can be obtained when SNR=6$\sim$9dB; however, only 0-10 computational gain is rendered for SNR=0$\sim$5.5dB.

In the end, we estimate the P_b of the MLSDA-Turbo decoder for SNR above 9dB, which may not be estimated using the MC technique. As expected, the computational time is reduced to at least 1/500 of the MC computation effort.

Bibliography

[1] K. S. Shanmugarn and P. Balaban, "A modified Monte Carlo simulation technique for the evaluation of error rate in digital communication systems," *IEEE Trans. on Communications*, vol. COM-28, pp. 1916–1924, November 1989.

[2] J. S. Sadowsky, "A new method for Viterbi decoder simulation using importance sampling," *IEEE Trans. on Communications*, vol. 38, no. 9, pp. 1341–1351, September 1990.

[3] K. B. Letaief and K. Muhammad, "An efficient new technique for accurate bit error probability estimation of ZJ decoders," *IEEE Trans. on Communications*, vol. 43, no. 6, pp. 2020–2027, June 1995.

[4] Yunghsiang S. Han and Po-Ning Chen, "Maximum-likelihood soft-decision sequential decoding algorithms for convolutional codes," presented at the recent results session of *the 1998 IEEE International Symposium on Information Theory*, Cambridge, MA, USA, August 1998.

[5] J. M. Hammersley and D. C. Handscomb, "Monte Carlo methods," *New York: Chaoman and Hall*, 1964.

[6] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon Limit error-correcting coding and decoding: Turbo-Codes(1)," *IEEE Int. Conf. on Communications*, vol. 2 pp. 1064–1070, October 1993.

[7] S. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-Codes," *IEEE Trans. on Communications*, vol. 44, pp. 1261–1271, October 1996.

[8] J. G. Proakis, *Digital Communications*. 4th edition, New York: McGraw-Hill, 2000.

[9] A. J. Viterbi and J. K. Omura, *Principles of Digital Communications and Coding*. New York: McGraw-Hill, 1979.

[10] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1983.

[11] P. R. Chevillat and D. J. Costello, Jr., "An analysis of sequential decoding for specific time-invariant convolutional codes," *IEEE Trans. Information Theory*, vol. IT-24, no. 4. pp. 443–451, July 1978.

[12] Hanan Herzberg, "Multilevel turbo coding with short interleavers," *IEEE Journal on selected areas in Communications*, vol. 16, pp. 303–309, February 1998.

[13] L. Ekroot and S. Dolinar, "A* decoding of block codes," *IEEE Trans. on Communications*, vol. 44, pp. 1052–1056, September 1996.

[14] Y. S. Han, "A new treatment of priority-first search maximum-likelihood soft-decision decoding of linear block codes," *IEEE Trans. on Information Theory*, vol. 44, pp. 3091–3096, November 1998.

[15] J. S. Sadowsky, "On the optimality and stability of exponential twisting in Monte Carlo estimation," *IEEE Trans. on Information Theory*, vol. 39, pp. 119–128, January 1993.

[16] P. J. Smith, M. Shafi, and H. Gao "Quick simulation: A review of importance sampling techniques in communications systems," *IEEE Journal on selected Areas in Communications*, vol. 15, no. 4, pp. 597–613, May 1989.

[17] Hsin-Chi Hsu, *A Maximum-Likelihood Decoding Algorithm for Parallel Concatenated Convolutional Codes*. Master Thesis, Dept. of Communications Eng., National Chiao Tung Univ., Taiwan, R.O.C., June 2000.