

Douglas da Silva Luiz

*Avaliação do método "Importance
Sampling" para simulação da taxa de
erro de bit em sistemas de comunicação*

Florianópolis

2018

Douglas da Silva Luiz

Avaliação do método "*Importance Sampling*" para simulação da taxa de erro de *bit* em sistemas de comunicação

**Trabalho de Conclusão
de Curso submetido à
Universidade Federal
de Santa Catarina,
como requisito neces-
sário para obtenção do
grau de bacharel em
Engenharia Eletrônica**

Orientador: Prof. Fernando Rangel de
Sousa, Dr.

Florianópolis, Julho de 2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Luiz, Douglas da Silva
Avaliação do método "Importance Sampling" para simulação
da taxa de erro de bit em sistemas de comunicação / Douglas
da Silva Luiz ; orientador, Fernando Rangel de Sousa,
2018.
73 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro Tecnológico,
Graduação em Engenharia Eletrônica, Florianópolis, 2018.

Inclui referências.

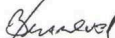
1. Engenharia Eletrônica. 2. Taxa de erro de bit. 3.
Importance Sampling. 4. Monte Carlo. I. Sousa, Fernando
Rangel de . II. Universidade Federal de Santa Catarina.
Graduação em Engenharia Eletrônica. III. Título.

Douglas da Silva Luiz

Avaliação do método “Importance Sampling” para simulação da taxa de erro de bit em sistemas de comunicação

Este Trabalho foi julgado adequado para obtenção do Título de Bacharel em Engenharia Eletrônica e aprovado em sua forma final pela Banca Examinadora

Florianópolis, 09 de Julho de 2018.



Prof. Jefferson Luiz Brum Marques, Dr.
Coordenador do Curso de Engenharia Eletrônica

Banca Examinadora:



Prof. Fernando Rangel de Sousa, Dr.
Orientador (a)
Universidade Federal de Santa Catarina



Prof. Carlos Aurélio Faria da Rocha, Dr.
Universidade Federal de Santa Catarina



Eng. Rodrigo Eduardo Rottava, MSc.
Chipus Microeletrônica

Dedico este trabalho à meus familiares, amigos, professores e a universidade. Obrigado principalmente a meus pais, por me apoiar em todos estes momentos, tanto difíceis quanto bons pelos quais passei.

Agradeço também ao meu orientador, professor Fernando, pela oportunidade de desenvolver este trabalho ao seu lado, este que sempre esteve bem disposto e empolgado no desenvolvimento do projeto, incentivando-me sempre a buscar o melhor de mim.

Resumo

Nesta era da informação, onde praticamente todos os dispositivos utilizados estão conectados de forma digital, é indispensável que a comunicação entre dispositivos seja feita de maneira rápida, eficiente e principalmente com baixa taxa de erro. Para avaliar o desempenho do sistema, geralmente analisa-se a taxa de erro de *bit* (*Bit Error Rate - BER*). Se o método utilizado para a simulação não for otimizado, o tempo de simulação do sistema pode exceder o limite do aceitável. Neste trabalho é estudado o *Importance Sampling* (*IS*), um método que permite a aceleração da análise da *BER* em relação ao método tradicional *Monte Carlo* (*MC*). Apresentam-se resultados de comparação do método avaliado com o método *MC*, e conclui-se que o método avaliado possui desempenho superior.

Palavras-chave: Taxa de Erro de Bit, *Bit Error Rate*, *BER*, *Importance Sampling*, *Monte Carlo*.

Abstract

In this information age, where practically all the devices used are connected in a digital way, it is essential that the communication between devices is done quickly, efficiently and especially with low error rate. To evaluate the system performance, the bit error rate(BER) is usually analyzed. If the method used for the simulation is not optimized, the system simulation time may exceed the acceptable limit. In this work we study Importance Sampling(IS), a method that allows the acceleration of the BER analysis in comparison to the traditional Monte Carlo(MC) method. Results of comparison of the method evaluated with the MC method are presented, and it is concluded that the evaluated method has superior performance.

Keywords: *Bit Error Rate, BER, Importance Sampling, Monte Carlo.*

Lista de ilustrações

Figura 1 – Distribuição Normal	30
Figura 2 – PDF Original f , PDF f_a obtida com Scaling e parâmetro $a = 3.78$ e PDF ótima f_{opt}	35
Figura 3 – Ganho em redução de amostras com Scaling	36
Figura 4 – Gráfico do número necessário de amos- tras K_{IS} pelo método IS em função da probabilidade de erro utilizando <i>Scaling</i> e <i>Translation</i>	38
Figura 5 – Algoritmo Importance Sampling Adap- tativo	41
Figura 6 – Exemplo	44
Figura 7 – Diagrama de Blocos de um <i>BERT</i> .	48
Figura 8 – Diagrama de Blocos do Sistema . . .	49
Figura 9 – Linear Feedback Shift Register de Fibonacci	50
Figura 10 – Linear Feedback Shift Register de Galois	50
Figura 11 – Simulação 1	56
Figura 12 – Simulação 2	57
Figura 13 – Simulação 3	58

Figura 14 – Simulação 4	59
Figura 15 – Simulação 5	60
Figura 16 – Simulação 6	61
Figura 17 – Simulação 7	62

Lista de tabelas

Tabela 1 – Referente a Simulação 1	56
Tabela 2 – Referente a Simulação 2	57
Tabela 3 – Referente a Simulação 3	58
Tabela 4 – Referente a Simulação 4	59
Tabela 5 – Referente a Simulação 5	60
Tabela 6 – Referente a Simulação 6	61
Tabela 7 – Referente a Simulação 7	62
Tabela 8 – Resultados da segunda etapa de si- mulações	63
Tabela 9 – Resultados da segunda etapa de si- mulações	63

Lista de abreviaturas e siglas

BER	<i>Bit Error Rate</i>
IS	<i>Importance Sampling</i>
MC	<i>Monte Carlo</i>
SNR	<i>Signal-to-Noise Ratio</i>
IoT	<i>Internet of Things</i>
LRF	<i>Radio Frequency Laboratory</i>
DUT	<i>Device Under Test</i>
SDR	<i>Software-Defined Radio</i>
PDF	<i>Probability Density Function</i>
CDF	<i>Cumulative Density Function</i>
MLE	<i>Maximum Likelihood Estimator</i>
BERT	<i>Bit Error Rate Tester</i>
LFSR	<i>Linear Feedback Shift Register</i>
FPGA	<i>Field Programmable Gate Array</i>
LRF	<i>Laboratório de Radiofrequência</i>

Lista de símbolos

X	Variável aleatória sobre análise
t	Limiar para a ocorrência de erros
p_t	Probabilidade da variável aleatória X ser maior do que o limiar t
$f(x)$	<i>PDF</i> da variável aleatória X
$F(x)$	<i>CDF</i> da variável aleatória X
K	Número total de amostras
k_t	Número de amostras acima do limiar t
$f_*(x)$	<i>Biasing Probability Density Function</i>
$W(x)$	Função de ponderação(<i>Weight</i>)
μ	Média de uma distribuição do tipo Normal
σ	Desvio padrão de uma distribuição do tipo Normal
a	Parâmetro de dimensionamento do método de <i>Scaling</i>

c Deslocamento a ser feito na média da *PDF* original para a obtenção da nova *PDF*

c_{opt} Deslocamento a ser feito na média da *PDF* original para a obtenção da nova *PDF* que maximiza os ganhos via *IS*

Sumário

1	INTRODUÇÃO	19
1.1	Objetivo	21
1.1.1	Objetivo Geral	21
1.1.2	Objetivos específicos	22
1.2	Motivação	22
1.3	Organização do Trabalho	23
2	FUNDAMENTAÇÃO TEÓRICA	25
2.1	Introdução	25
2.2	Método de <i>Monte Carlo</i>	26
2.2.1	Princípio	26
2.3	<i>Importance Sampling</i>	31
2.3.1	Princípio	31
2.3.2	Escolha da Função de <i>Biasing</i>	33
2.3.2.1	<i>Scaling</i>	34
2.3.2.2	<i>Translation</i>	36
2.4	Trabalhos relacionados	39
3	DESENVOLVIMENTO	45
3.1	Metodologia	46
3.1.1	Gerador de Números Aleatórios	47
3.1.2	Fonte de Ruído Gaussiano	52

3.1.3	Algoritmo de Medição da BER	53
3.1.4	Estimativa da <i>BER</i>	54
4	RESULTADOS	55
4.1	Discussão de resultados	63
5	CONCLUSÃO	67
6	SUGESTÕES PARA TRABALHOS FUTUROS	69
	REFERÊNCIAS	71

1 Introdução

Há algumas décadas, a comunicação entre dispositivos era feita de forma totalmente analógica. Na primeira geração de dispositivos sem fio(1G), as transmissões analógicas eram caracterizadas pela forte atenuação do sinal devido a grandes distâncias, algo que na época era contornado com o envio de sinais com razão sinal-ruído(SNR) muito alta. Nessa época, o custo de um dispositivo para o consumidor final era relativamente caro, fazendo com que apenas pessoas com alto poder aquisitivo tivessem esses aparelhos em casa. Com o desenvolvimento da tecnologia, a comunicação digital surgiu para eliminar vários problemas apresentados na comunicação analógica, através de diferentes técnicas de modulação, uso de códigos corretores de erro, e diferentes técnicas usadas para aumentar a capacidade de canal e amenizar os efeitos de atenuação devido a efeitos como *fading* e *multipath*.

As gerações seguintes de comunicação digital aprimoraram o sistema de forma que hoje, com o 4G, temos a chamada Era da Informação, que trouxe uma grande mudança de paradigma. O uso da comunicação

digital trouxe consigo grandes avanços na tecnologia, que possibilitaram não só uma extrema melhora na qualidade e velocidade do serviço, como também mudaram completamente a comunicação em escala global.

Hoje, com a invenção e popularização dos *smartphones*, a invenção do *Wi-Fi* e o barateamento dos dispositivos devido a grande escala de consumidores, temos o mundo todo conectado com usuários compartilhando informações em tempo real, algo que era impraticável a algumas décadas atrás.

Num futuro próximo, com a popularização da internet das coisas(*IoT*)[1] e junto com ela o 5G[2], a previsão é de que todos os dispositivos estejam conectados e compartilhando dados. Desde *smartphones*, *smart TVs*, *smartwatches*, até carros, dispositivos corporais e residências.

Para garantir que a qualidade da comunicação entre dispositivos esteja acima de um nível pré-especificado e que atenda os requisitos de funcionamento do sistema de comunicação, são utilizadas diferentes métricas de controle de desempenho. Uma métrica quase que universalmente utilizada é a medição da *BER*[3]. Com essa medida, é possível que sejam feitos ajustes no sistema, como o aumento da potência utilizada para transmitir quando a *BER* está alta(ou a diminuição

quando a *BER* está baixa, levando a economia de energia), o uso de diferentes códigos corretores de erro e tipos de modulações dependendo do estado atual da *BER*, e uma série de otimizações que podem ser feitas tanto no transmissor quanto no receptor.

Este trabalho foca em um estudo do desempenho de um método alternativo em relação ao método tradicional para a medição da *BER*, e apresenta uma comparação entre eles através de uma implementação em *software*.

1.1 Objetivo

1.1.1 Objetivo Geral

Este trabalho tem por objetivo avaliar a performance do método *IS* para a medição da *BER* e compará-lo com o método tradicional *MC*, visando estabelecer um estudo inicial que será usado como base para futuramente desenvolver-se um sistema de medição de *BER* para caracterizar a performance de receptores de rádio desenvolvidos no Laboratório de Radiofrequência(LRF).

1.1.2 Objetivos específicos

- Estudo e avaliação de métodos para medição da *BER*;
- Implementação em *software* de um sistema configurado para a avaliação dos métodos estudados;
- Desenvolvimento de um pseudogerador de números aleatórios para a geração de amostras a serem utilizadas como entrada para o sistema;
- Desenvolvimento em *software* de um algoritmo para a medição da *BER*, obedecendo as especificações inicialmente propostas;
- Simulação do sistema analisado para a determinação do desempenho do método proposto;
- Otimização do algoritmo desenvolvido a fim de diminuir o tempo necessário para a análise da *BER*.

1.2 Motivação

Uma das principais motivações desse trabalho é a de fornecer um estudo inicial para o desenvolvimento

de um sistema em Rádio definido por Software (*SDR*) e posteriormente em FPGA (Field-Programmable Gate Array) para caracterizar sistemas de comunicação desenvolvidos no LRF, sem a utilização de fios conectando transmissor e receptor.

A necessidade de desenvolvimento de um sistema surgiu devido aos equipamentos comerciais apresentarem alto custo e não fornecerem flexibilidade em suas configurações. Assim, decidiu-se optar pelo desenvolvimento de um sistema próprio, oferecendo grandes opções de reconfigurabilidade. Este sistema será usado para a realização de testes nos *chips* desenvolvidos no LRF, que necessitam de configurações customizadas.

1.3 Organização do Trabalho

Este trabalho está organizado em 6 capítulos. O capítulo 2 abrange a fundamentação teórica necessária para o desenvolvimento do projeto, junto com a revisão de alguns trabalhos relacionados a área. O capítulo 3 contém a metodologia e desenvolvimento do projeto, e o capítulo 4 apresenta os resultados obtidos, assim como uma análise dos mesmos. O capítulo 5 apresenta uma conclusão final sobre o trabalho e os resultados,

e o capítulo 6 propõe uma direção para uma futura implementação física do algoritmo.

2 Fundamentação Teórica

2.1 Introdução

Devido a complexidade dos sistemas de comunicação ser alta, assim como o número de variáveis que podem influenciar em seu comportamento, a dificuldade em se obter e trabalhar com uma forma analítica da descrição do sistema faz com que simulações sejam preferíveis como métrica para a estimação do desempenho.

Simulações são um meio para emular o comportamento de um sistema em um determinado intervalo de tempo, a fim de inferir suas características com um grau de precisão pré-definido. O grau de confiança de uma simulação está diretamente ligado ao quão próximo da realidade são as equações que descrevem o sistema a ser simulado. O objetivo de uma simulação é encontrar erros, estes que são eventos que ocorrem raramente e podem ser catastróficos para o funcionamento do sistema.

Um dos métodos estatísticos mais utilizados em engenharia para fazer a simulação de um sistema complexo é o método de *Monte Carlo*. Este método utiliza

um modelo discreto dos processos estocásticos encontrados no ambiente para gerar estímulos, usá-los como entrada para o sistema descrito e tentar inferir o comportamento real do sistema através da simulação de um número finito de elementos.

2.2 Método de *Monte Carlo*

Desde a sua concepção[4], o método de *MC* encontrou espaço em diversos campos, como termodinâmica, engenharia de estruturas, mecânica dos fluidos, microeletrônica e principalmente em telecomunicações. Este método baseia-se numa abordagem muito simples: a contagem de erros.

2.2.1 Princípio

Considere para a análise que a transmissão de *bits* sejam eventos, e que um erro seja um evento raro. Então, considere o uso de simulações para estimar a probabilidade p_t do evento $\{X \geq t\}$, onde X é uma variável aleatória com distribuição $F(x)$ e função densidade de probabilidade(*PDF*) $f(x) = F'(x)$, onde o apóstrofo denota a operação de derivação. Será assumido que a *PDF* $f(x)$ existe, e o valor de t é tal que

o evento pode ser considerado raro(ou seja, um erro). Em [5], pode ser visto que o procedimento utilizado em simulações *MC* são os experimentos de Bernoulli, onde uma sequência de tamanho K independente e identicamente distribuída $\{X_i\}_1^K$ é gerada a partir da distribuição F , e o número k_t de amostras que estiverem acima do limiar(*threshold*) t é então contado. A variável aleatória k_t é caracterizada pela distribuição Binomial

$$P(k_t = k) = \binom{K}{k} p_t^k (1 - p_t)^{K-k}, \quad k = 0, 1, \dots, K. \quad (2.1)$$

O estimador de máxima verossimilhança(*MLE*) \hat{p}_t de p_t baseado nas amostras k_t observadas é dado fazendo

$$\frac{\partial P(k_t = k)}{\partial p_t} = 0 \quad (2.2)$$

em 2.1. Assim, temos

$$\begin{aligned} \hat{p}_t &= \frac{k_t}{K} \\ &= \frac{1}{K} \sum_{i=1}^K 1(X_i \geq t) \end{aligned} \quad (2.3)$$

onde

$$1(x \geq t) = \begin{cases} 1, & \text{se } x \geq t \\ 0, & \text{em caso contrário} \end{cases} \quad (2.4)$$

e 2.4 é a função indicadora dos eventos de interesse (erros). Este estimador (\hat{p}_t) é conhecido como estimador MC, é imparcial e tem variância dada por

$$\text{var } \hat{p}_t = \frac{1}{K}(p_t - p_t^2) \approx \frac{p_t}{K} \quad (2.5)$$

para p_t pequeno.

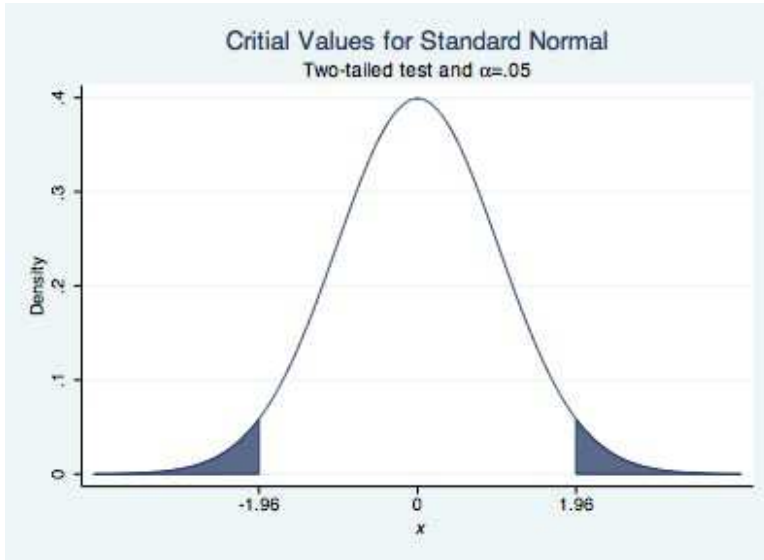
De 2.3 podemos ver que $E\{k_t\} = Kp_t$. Para obter, em média, um número de eventos raros maior que zero, e consequentemente um valor diferente de zero para o estimador \hat{p}_t , é possível provar que é necessário simular um número de amostras K muito maior do que $\frac{1}{p_t}$. A razão de \hat{p}_t poder assumir valores nulos é devido a probabilidade de erro ser muito pequena, e se o número de amostras usadas para estimar essa probabilidade não for suficiente é provável que nenhum erro ocorra. Como exemplo, imagine que se deseje simular e estimar a probabilidade de erro de um sistema, e sabe-se de antemão que o valor da mesma está em torno de 10^{-6} . Ou seja, um erro ocorrerá a cada 10^6 amostras. Agora, imagine

que foi decidido utilizar 10^5 amostras para estimar essa probabilidade. Existe aproximadamente 90% de chance de que nenhum erro aconteça, pois não foi simulado um número de amostras suficiente para que o mesmo ocorra, fazendo com que a probabilidade estimada \hat{p}_t seja zero. Por isso, é necessário uma regra que indique um número inicial de amostras para que a probabilidade estimada seja confiável. Pode ser visto em [1] que para K relativamente grande, e assumindo um intervalo de confiança de 95% (há 95% de chance de a probabilidade encontrada estar dentro do valor encontrado com tolerância de 10%) é necessário que $K > \frac{100}{p_t}$. Então no nosso exemplo, seriam necessários pelo menos 10^8 amostras para que se obtivesse um resultado confiável.

Na prática, para taxas de erros muito pequenas esse método se torna cada vez menos viável, pois as sequências geradas por geradores de números aleatórios precisam ser muito grandes exigindo muito em termos de processamento e memória, estes que aumentam de maneira não linear com a diminuição da taxa de erro. Estes fatores levam a altos tempos de simulação. Existe um limite para o uso do método de *MC* e portanto deve ser procurada uma abordagem alternativa para lidarmos com esse problema.

Podemos perceber analisando a Figura 1, que

Figura 1 – Distribuição Normal



Fonte: Referência [6]

mostra a função densidade de probabilidade de uma distribuição normal qualquer, que a maioria dos pontos tendem a um valor central, que tem sua frequência reduzida conforme o valor da variável vai aumentando. Os eventos de interesse estão na "cauda" da PDF, e o grande problema está em estimar a probabilidade de erro devido a baixa frequência com que esses eventos acontecem. Então, pensando de um jeito diferente: e se pudéssemos modificar a *PDF* de forma que os erros aconteçam com mais frequência?

2.3 Importance Sampling

IS foi um método demonstrado primeiramente em [7] em 1951, baseado numa modificação do método de *MC* tradicional. Com ele é possível reduzir significativamente o número de amostras para estimar uma determinada probabilidade de erro, fazendo com que o mesmo seja essencial em aplicações com *BER* extremamente baixas.

Este método funciona através de uma distorção na *PDF original*, fazendo com que a nova *PDF* tenha uma grande probabilidade de eventos raros acontecerem. Deve haver uma ponderação sobre o resultado, para que o resultado obtido seja correspondente ao do método de *MC*.

2.3.1 Princípio

Imagine uma função densidade de probabilidade alternativa f_* (para X), usualmente chamada de função de "*biasing*". Essa *PDF* permite que eventos de erro aconteçam com maior frequência, fazendo com que sejam necessárias sequências de tamanho K menores dada uma mesma variância em relação ao método de

MC. De [5], introduzindo f_* em p_t temos

$$\begin{aligned} p_t &= E\{1(X \geq t)\} \\ &= \int 1(x \geq t) \frac{f(x)}{f_*(x)} f_*(x) dx \\ &= E_*\{1(X \geq t)W(X)\} \end{aligned} \quad (2.6)$$

onde

$$W(x) \equiv \frac{f(x)}{f_*(x)} \quad (2.7)$$

é a função de ponderação. A notação E_* denota a operação esperança em relação a PDF f_* . Para a equação 2.6 ser válida, é necessário que $f_*(x) > 0$ para todo $x \geq t$ tal que $f(x) > 0$. Esta última igualdade em 2.6 produz o estimador

$$\hat{p}_t = \frac{1}{K} \sum_{i=1}^K 1(X_i \geq t)W(X_i), \quad X_i \sim f_* \quad (2.8)$$

A notação $X \sim f$ indica que a variável aleatória X é retirada da distribuição correspondente a PDF de f . Este é o estimador utilizado em IS, e é imparcial. O procedimento para a simulação é feito através da geração de amostras retiradas da distribuição f_* , onde para

cada amostra que exceda t , o estimador é ponderado pelo peso W e avaliado a cada amostra. O resultado é obtido com a média desses valores após K tentativas.

O diferencial do *IS* está na mudança de PDF, que faz com que o problema encontrado no método de *MC* (insuficiência de eventos raros) seja solucionado. Ao alocar mais "massa" da *PDF* na faixa de eventos raros e tornando os erros frequentes, é possível conseguir um resultado com precisão próxima a do método de *MC*. O desafio na utilização deste método está em achar uma função f_* que minimize o número de amostras necessárias para isso.

2.3.2 Escolha da Função de *Biasing*

Muito do estudo feito em *IS* é focado na escolha de funções de *biasing* f_* que tenham um bom desempenho em simulações. Assim, essa nova função deve ter um aumento da probabilidade de eventos raros na região de interesse. Uma boa forma de encontrar funções de *biasing* que tenham um bom desempenho é através de transformações sobre a *PDF* original f . Nesta seção, serão vistos dois métodos para a obtenção desta função: *Scaling* e *Translation*.

2.3.2.1 *Scaling*

Scaling é um tipo de transformação sobre a *PDF* original que faz com que a *PDF* seja comprimida em amplitude e estendida horizontalmente, fazendo com que uma grande parte da *PDF* original seja alocada na região de eventos de interesse.

A função de *biasing* para o método de *scaling* é dada por

$$f_*(x) = \frac{1}{a} f\left(\frac{x}{a}\right) \quad (2.9)$$

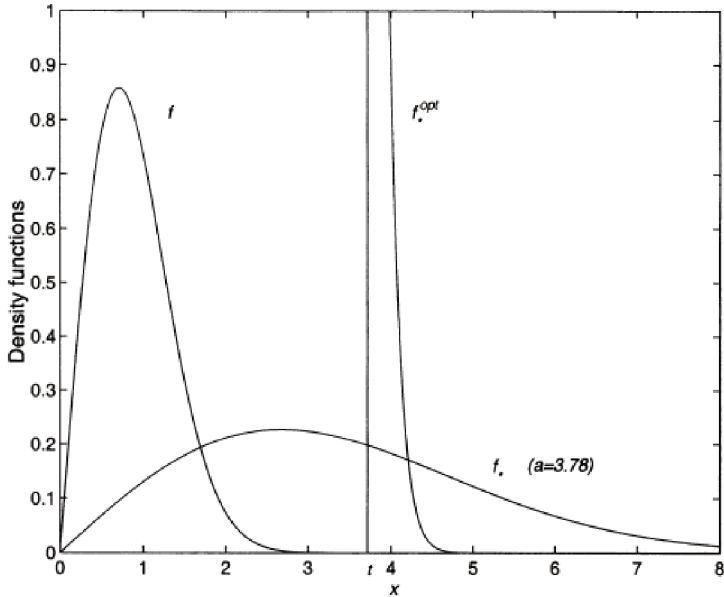
e a função de ponderação por

$$W(x) = a \frac{f(x)}{f(x/a)} \quad (2.10)$$

onde a é o parâmetro de *scaling*.

Na Figura 2 podemos ver que a *PDF* f é redimensionada pelo parâmetro a para a *PDF* f_a para que uma região maior da *PDF* contenha os eventos de interesse. a *PDF* ótima f_{opt} seria a *PDF* ideal, mas como pode ser visto em [5] ela é irrealizável e se sua forma fosse conhecida não haveria a necessidade de se utilizar simulações para a análise da *BER*.

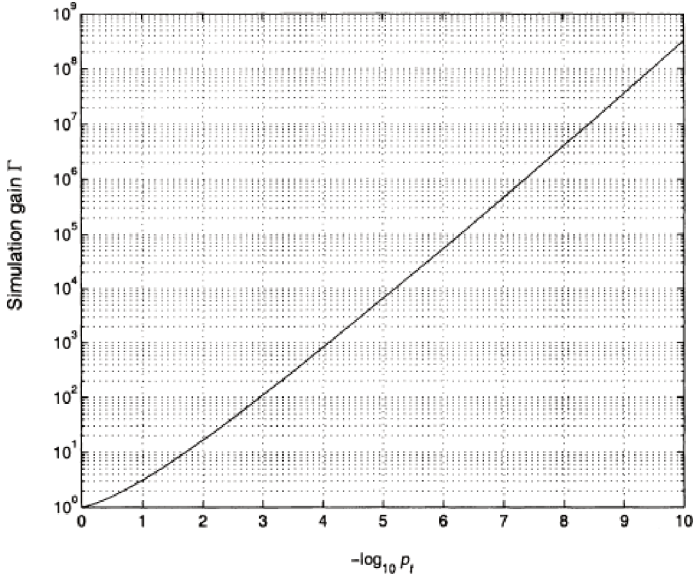
Figura 2 – PDF Original f , PDF f_a obtida com Scaling e parâmetro $a = 3.78$ e PDF ótima f_{opt}



Fonte: Referência [5]

Na figura 3 temos o ganho em número de amostras necessárias em função da probabilidade de erro do sistema. Como podemos perceber, quanto menor a probabilidade de erro maior será o ganho em simulação, chegando a quase 10^5 para uma probabilidade de erro de 10^{-6} .

Figura 3 – Ganho em redução de amostras com Scaling



Fonte: Referência [5]

2.3.2.2 Translation

Translation é um tipo de transformação que faz com que a média da *PDF* seja transladada. Com a escolha dos parâmetros certos, é possível obter uma parte maior da *PDF* na região de interesse. A nova *PDF* é dada por

$$f_*(x) = f(x - c), \quad c > 0 \quad (2.11)$$

e o estimador por

$$\hat{p}_t = \frac{1}{K} \sum_{i=1}^K 1(X_i \geq t) \frac{f(X_i)}{f(X_i - c)}, \quad X_i \sim f_* \quad (2.12)$$

onde

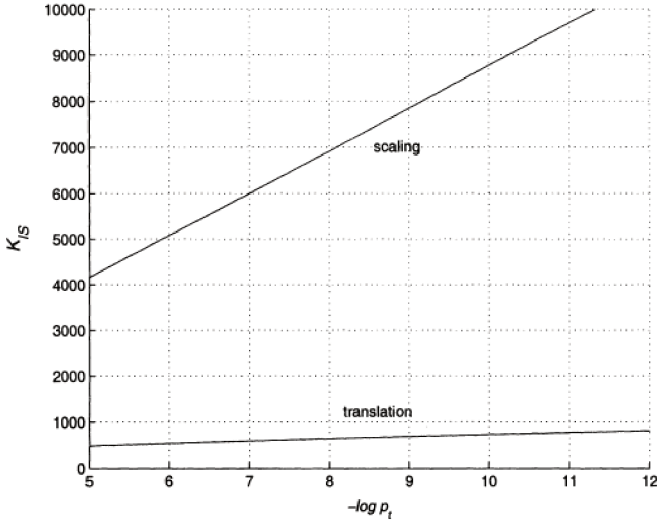
$$W(x) = \frac{f(x)}{f(x - c)} \quad (2.13)$$

e c é o parâmetro de translação.

Na figura 4 temos o número de amostras necessárias com os métodos de *scaling* e *translation* versus a probabilidade de erro. Analisando, vemos que além do método *translation* precisar de um número muito menor de amostras, ele ainda tem uma variação consideravelmente pequena com a probabilidade de erro em relação ao *scaling*. A implementação do método *translation* é mais simples em relação ao de *scaling* por precisar somente que seja feita uma mudança na média da *PDF*, e por esses motivos será optado pela utilização do método *translation* no desenvolvimento deste projeto.

Tendo então o método de *translation* como base, para encontrar o parâmetro c que minimiza o número de amostras necessárias para a simulação da *BER*, é descrito em [5] que se a distribuição for do tipo Normal

Figura 4 – Gráfico do número necessário de amostras K_{IS} pelo método IS em função da probabilidade de erro utilizando *Scaling* e *Translation*



Fonte: Referência [5]

com média μ e desvio padrão σ , o valor de c que otimiza f_* , c_{opt} , é dado por

$$c_{opt} \approx \sqrt{(t - \mu)^2 + \sigma^2} \quad (2.14)$$

Com base na fundamentação vista até agora e nas escolhas feitas para o método de IS a ser utilizado, será desenvolvido no capítulo 3 a parte prática referente ao projeto.

2.4 Trabalhos relacionados

Michel C. Jeruchim faz em [3] uma análise detalhada de diversas técnicas de estimação da *BER* para sistemas de comunicação digital. As técnicas analisadas são *Monte Carlo*, *Importance Sampling*, *Extreme-Value Theory*, *Tail Extrapolation* e *Quasi-Analytical*.

O método de *Monte Carlo* é baseado na simulação do comportamento do sistema, através da análise de suas entradas e saídas e a contagem de erros ocorridos.

O método de *Importance Sampling* parte da proposta do método de *Monte Carlo* e segue um caminho diferente: os erros vistos pelo sistema são eventos que ocorrem raramente, então o método propõe uma mudança na distribuição usada para simular para que um número maior de erros ocorra, fazendo com que seja possível estimar a probabilidade de erro do sistema com um número reduzido de amostras.

O método visto em *Extreme-Value Theory* mostra uma abordagem diferente, onde ao invés de simular amostras a procura de erros, o autor assume que a distribuição analisada pertence a uma família de funções, e o objetivo do autor é a estimação dos parâmetros dessa família de distribuições. Com estes parâmetros, pode-se

estimar a *BER* sem que seja necessário a contagem de erros.

O método *Tail Extrapolation* pertence a um subconjunto do método anterior, e assume que a distribuição analisada pertence a família de exponenciais generalizadas, uma generalização da função gaussiana. Como grande parte do ruído na natureza tem forma gaussiana, o autor sugere que uma função membro dessa família de funções será uma boa aproximação da distribuição real.

Por último, o método *Quasi-Analytical* tem como proposta a aglomeração dos diversos efeitos de ruído em uma fonte equivalente de ruído a entrada do sistema, assumindo que esta tem forma exponencial e que possa ser vista como uma fonte de ruído gaussiano.

Mónica F. Bugallo, et al. propõem em [8] uma forma de *IS* Adaptativo, usado para estimar a distribuição de um sistema desconhecido, através de estimativas dos seus parâmetros.

Ao estudar o método de *IS*, vemos que a parte mais importante do método está na escolha de uma *PDF* que modele melhor o comportamento do problema. Em algumas aplicações, como a localização de objetos em redes de sensores sem fio, *IoT*, e estimação da den-

sidade espectral de potência para a melhora da fala, há uma dificuldade de se estimar a distribuição que modela essas aplicações. Neste trabalho os autores propõem uma forma iterativa de *IS* (*Adaptive Importance Sampling*), onde é proposta uma *PDF* inicial para o sistema analisado e através de sucessivas iterações, é feito um ajuste dos parâmetros dessa *PDF* com o objetivo de deixá-la o mais próxima possível da distribuição real que descreve o sistema.

A Figura 5 mostra o algoritmo proposto:

Figura 5 – Algoritmo Importance Sampling Adaptativo

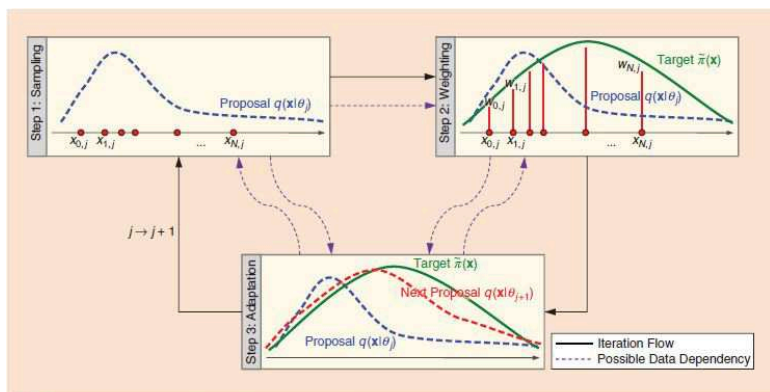


FIGURE 2. A generic flow diagram of the AIS methodology, showing the three steps that must be performed iteratively by any AIS algorithm (sampling, weighting, and adaptation) and the data flow among these steps.

Fonte: Referência [8]

Seja K o número de amostras por distribuição e N o número total de distribuições analisadas, tem-se o algoritmo:

- Primeiro é feita uma amostragem das K amostras de cada uma das N distribuições propostas;
- Depois é calculada a função de ponderação (*weight*) para todas as KN amostras geradas;
- Com os valores gerados pela função ponderação em cada amostra, é feita uma atualização nos parâmetros das *PDFs* e então inicia-se o algoritmo novamente.

A Figura 6 mostra um exemplo do algoritmo usado

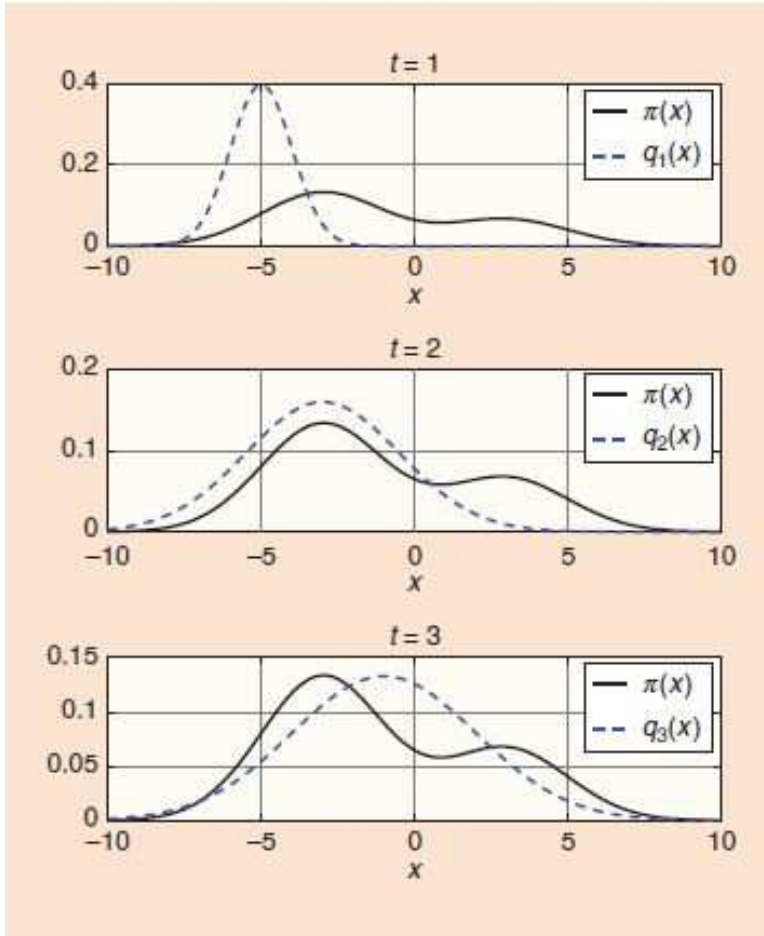
- Nela, os autores propõem como um chute inicial na primeira iteração a *PDF* $q_1(x)$, que não se assemelha muito a *PDF* verdadeira $\pi(x)$;
- Nas duas iterações seguintes são feitos ajustes nos parâmetros, e as *PDFs* encontradas a cada iteração vão se assemelhando cada vez mais a *PDF* alvo.

Este trabalho é muito promissor, visto que em aplicações reais, onde o canal de comunicação está mudando a todo segundo, devido a todas as iterações do ambiente, das pessoas, de outros sistemas de comunicação e do mundo em si, é necessário que haja uma forma

de medida da qualidade da comunicação que possa ser feita em tempo real.

O *Importance Sampling* Adaptativo é uma forma de se fazer isso, e o trabalho desenvolvido nesta monografia tem como objetivo uma futura evolução do algoritmo para uma forma adaptativa.

Figura 6 – Exemplo



Fonte: Referência [8]

3 Desenvolvimento

Com a fundamentação teórica feita no capítulo 2, deu-se início ao desenvolvimento do projeto que seguiu basicamente 5 passos, listados abaixo:

- Revisão da fundamentação teórica;
- Revisão do Estado da Arte;
- Definição do método a ser utilizado;
- Especificação do projeto;
- Desenvolvimento do sistema.

Como pode ser observado, primeiramente foi feita uma revisão da fundamentação teórica seguida de uma revisão do estado da arte. Após a definição do método a ser utilizado, voltou-se atrás e foi feita uma nova revisão da fundamentação teórica. Então, foi feita uma análise e especificação do projeto e só depois desses passos iniciou-se o desenvolvimento do projeto.

3.1 Metodologia

Como método de medição da *BER*, o *IS* é um método que é limitado apenas a simulações. Isso acontece porque para o seu uso, é necessário conhecimento prévio sobre a distribuição f do canal que o sistema avaliado está utilizando, para que se possa então encontrar a distribuição f_* . Porém, o canal muda a todo momento nas aplicações reais, devido a interações que ocorrem constantemente no ambiente, como o passar de carros, pessoas, efeitos como o *fading*, e etc.

Por isso, o foco desse trabalho está em avaliar o método de *IS* no ambiente de simulações, e fixá-lo como base para a procura de um algoritmo alternativo ou uma modificação do algoritmo atual no futuro visando uma aplicação real, pelos motivos citados na seção 1.2.

A proposta inicial do projeto era o desenvolvimento do mesmo em *hardware*, mais especificamente utilizando *FPGA*. Porém optou-se por um desenvolvimento intermediário em *software*. Por isso, a linguagem de programação escolhida para o desenvolvimento do mesmo foi *Python*, uma linguagem muito utilizada atualmente, que contém uma grande gama de ferramentas para o trabalho com processamento digital de sinais.

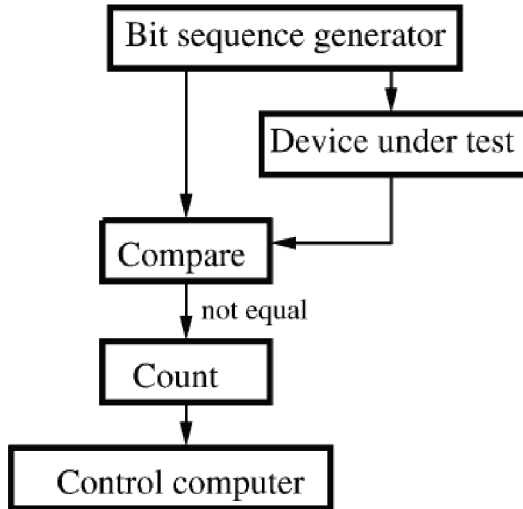
Assim, para o projeto de um *Bit Error Rate Tester* (*BERT*), normalmente modela-se o sistema através de um diagrama de blocos como o referenciado na Figura 7, onde amostras são geradas através de um gerador de sequências de amostras, que então são enviadas para o dispositivo sobre teste (*DUT*). A saída do *DUT* é então comparada com a sequência enviada e os erros encontrados são então contados. O bloco de controle decide o número necessário de amostras para que se possa ter uma medição que represente o comportamento real do sistema avaliado.

Com base na Figura 7, foi desenvolvido o diagrama de blocos mostrado na Figura 8. Este sistema tem como entrada uma soma entre amostras geradas e ruído gaussiano, e apresenta a saída o resultado da medição da *BER*. Abaixo segue uma descrição do funcionamento de cada bloco:

3.1.1 Gerador de Números Aleatórios

O gerador de números aleatórios caracteriza um bloco que produz as amostras que serão utilizadas como estímulos para o sistema. O tamanho da sequência de amostras é configurável.

Como arquitetura para o gerador de números

Figura 7 – Diagrama de Blocos de um *BERT*

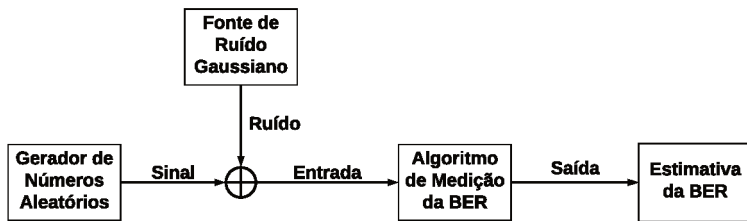
Fonte: Referência [9]

pseudo aleatórios, foi escolhido o *LFSR* (*Linear Feedback Shift Register*), um registrador de deslocamento com realimentação cujo *bit* de entrada é uma função linear do estado anterior.

Uma das condições para que o número de estados seja máximo é de que o polinômio utilizado na realimentação seja primitivo[10]. Uma das características desse tipo de polinômio é que eles estão em sua forma mais simplificada possível.

Entre os tipos de *LFSR* mais usados, estão o

Figura 8 – Diagrama de Blocos do Sistema

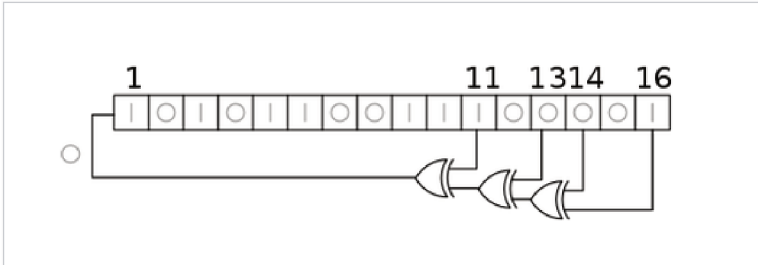


Fonte: Elaborada pelo Autor

LFSR de Fibonacci e o *LFSR* de Galois, que podem ser vistos na figuras 9 e 10

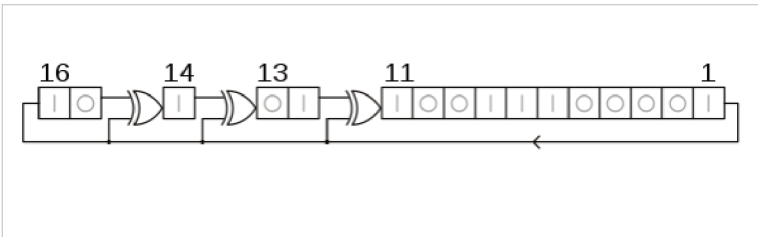
Os *LFSRs* nas figuras usam o polinômio primitivo $x^{16} + x^{14} + x^{13} + x^{11} + 1$, ou seja, usam os *bits* de posição 16, 14, 13, 11 e 1 como *feedback* através de portas *XOR*, fazendo com que a cada iteração um novo número seja gerado, até a duração máxima definida pelo grau do polinômio utilizado. Vale ressaltar que o único estado proibido neste tipo de gerador é o estado zero, pois neste estado o gerador entrará em um *loop* infinito gerando apenas zeros como amostras, já que

Figura 9 – Linear Feedback Shift Register de Fibonacci



Fonte: Referência[11]

Figura 10 – Linear Feedback Shift Register de Galois



Fonte: Referência[11]

uma operação *XOR* entre zeros leva a um zero sempre.

Observando a arquitetura vista na Figura 10 é possível concluir que o *LFSR* de Galois terá um desempenho melhor em *software*, já que suas operações *XOR* podem ser feitas paralelamente, ao contrário do *LFSR* de Fibonacci onde cada porta *XOR* tem como entrada uma outra porta *XOR*, fazendo com que o cálculo seja feito de forma sequencial, levando a um tempo de exe-

ção maior. Assim, o *LFSR* de Galois foi escolhido como base para o gerador.

Abaixo está o trecho do código referente ao gerador de números pseudo-aleatórios, que produz números de 8 *bits* com base no *LFSR* de Galois, com a variável `sig_in` como saída do bloco:

```
#Gerador de Ruído Pseudo-Aleatório
#Declaração de Variáveis e Constantes
#Nº de amostras
num_samples = 1000
#Nº de bits por amostra
bits = 8
#Seed para a LFSR
start_state = 0xA3
#Inicialização da LFSR
lfsr = start_state
#Polinômio primitivo
p_prim = 0xB8
period = 0
counter = 0
sig_in = []

#Inicialização - Estado Inicial
period = 0
lsb = lfsr & 1
lfsr >>= 1
if(lsb == 1):
    lfsr ^= p_prim
```

```

sig_in.append(lfsr)
period= period+1

#Continuação da LFSR em um laço
while (lfsr != 0):
    #Armazena o bit menos significativo
        lsb = lfsr & 1
    #deslocamento a direita dos bits da LFSR
        lfsr >>= 1
        if (lsb == 1):
    #XOR bit-a-bit entre os bits da LFSR e
    #os coeficientes do polinômio gerador
            lfsr ^= p_prim
    #armazena o número gerado em um vetor
        sig_in.append(lfsr)
        period= period+1
    #Fim do laço quando o número requerido de
    #amostras for produzido
        if (period == num_samples):
            break

```

3.1.2 Fonte de Ruído Gaussiano

O bloco fonte de ruído gaussiano tem uma série de funções: primeiro, cria uma *PDF* f com média $\mu = 0$ e desvio $\sigma = 1$. Depois, com esses dois parâmetros (μ e σ) é calculado c_{opt} usando a equação 2.4. Para o projeto, obteve-se $c_{opt} = 1.118$. Substituindo então c_{opt} como média de f criamos então a *PDF* f_* , e é a partir dessa

PDF que são geradas as amostras de ruído para serem somadas com os números gerados no bloco anterior e constituírem a entrada do sistema.

3.1.3 Algoritmo de Medição da BER

Este bloco tem como função fazer o cálculo da *BER* usando o método *IS*. Os parâmetros usados para o cálculo são as sequências de números gerados e da soma dos números gerados com o ruído, as *PDFs* f e f_* e o número de amostras simuladas K .

É realizado então um laço, onde para cada amostra que chega nesse bloco, é feita uma comparação dela com sua versão sem ruído e analisado se houve um erro ou não (se uma amostra é diferente da outra). Em caso afirmativo, pondera-se o erro observado pelo função de ponderação W , soma-se esse valor com o resultado acumulado e então é dada continuação a execução do laço. Após K iterações serem processadas, pondera-se o resultado final pelo número de amostras processadas e então tem-se a estimativa da *BER* para o sistema.

Abaixo segue o trecho de código representando o bloco Algoritmo de Cálculo da *BER*:

```

#Função para o Cálculo da BER via Importance Sampling
#Os parâmetros de entrada são o número de amostras , o
#sinal de entrada x e o sinal de entrada+ruído y, a PDF
#da distribuição original e a PDF gerada através de IS

def error_calc(K, x, y, pdf1, pdf2):
    counter = 0
    for i in range(0, K, 1):
        #Se a amostra na saída for diferente da entrada:
            if(x[k] != y[k]):
                weight = pdf1[k]/pdf2[k]
        #Some o valor da função ponderação na amostra atual
        #e continue com a próxima amostra
            counter = counter + weight
        #Após o laço, o resultado final é dado pelo soma final ponderada
        #pelo número de amostras processadas
            error = float(counter/size)
    return error

```

3.1.4 Estimativa da BER

Neste bloco são apresentados os resultados em forma de gráficos, que mostram a BER estimada em função de E_b/N_0 . Este último foi obtido com uma variação na energia das amostras de ruído, para que se pudesse fazer uma comparação entre os métodos de IS e MC através dessas curvas.

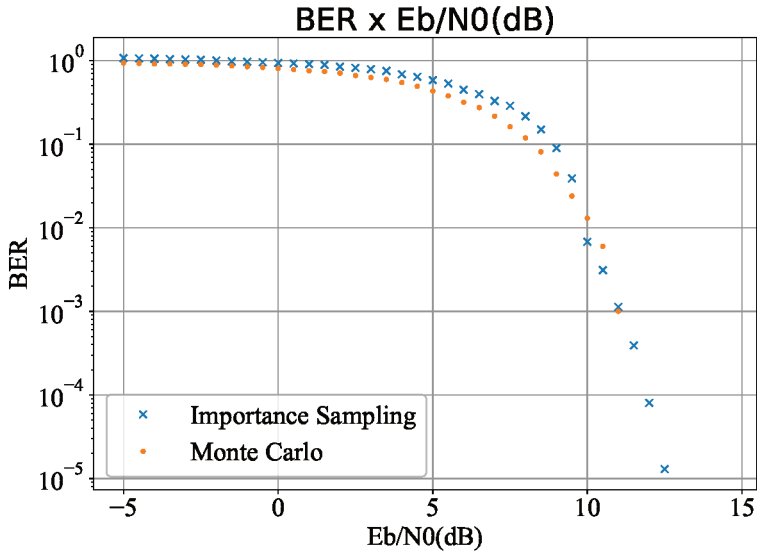
4 Resultados

Para fazer uma análise sobre o desempenho do algoritmo, foram realizadas diversas simulações utilizando os métodos de *IS* e *MC*. Numa primeira etapa, foi feita uma variação no número de amostras em diversas simulações e feitos os *plots* das respectivas curvas de BER x E_b/N_0 .

Para cada simulação realizada, foram analisados o número de amostras utilizadas, o tempo necessário para a execução dos métodos de *IS* e *MC* e a precisão obtida, ou seja, a menor probabilidade de erro que cada algoritmo conseguiu medir.

Os resultados estão apresentados em pares figura-tabela, onde cada figura mostra a relação da *BER* do sistema descrito na Figura 8 com a variação de E_b/N_0 nesse sistema, e cada tabela contém o número de amostras usada por cada método em sua respectiva simulação, assim também como o tempo de execução e precisão obtida por cada algoritmo. Estes resultados são vistos nas figuras a seguir:

Figura 11 – Simulação 1

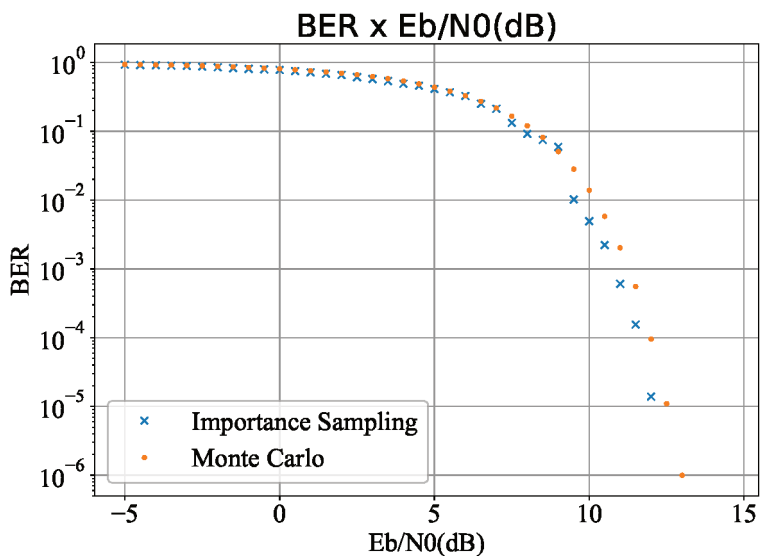


Fonte: Elaborada pelo Autor

Método	Nº de Amostras	Tempo de Execução(seg)	Precisão
Monte Carlo	10^3	1	10^{-3}
Importance Sampling	10^3	1	10^{-5}

Tabela 1 – Referente a Simulação 1

Figura 12 – Simulação 2

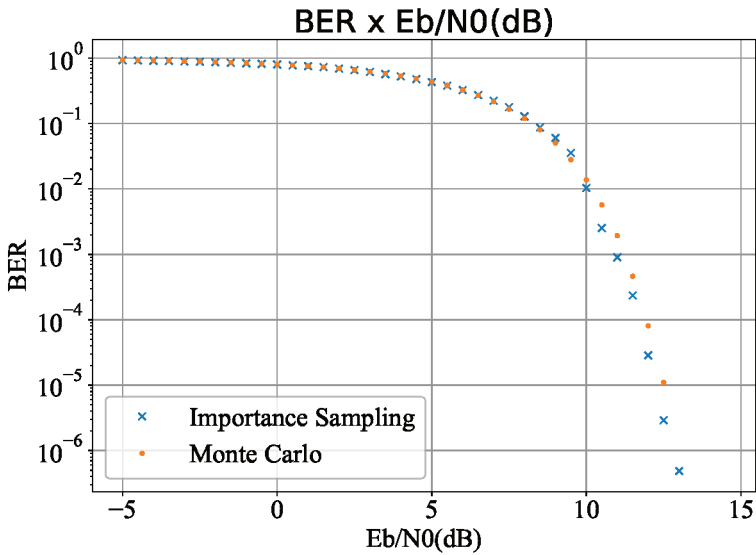


Fonte: Elaborada pelo Autor

Método	Nº de Amostras	Tempo de Execução(seg)	Precisão
Monte Carlo	10^6	57	10^{-6}
Importance Sampling	10^3	1	10^{-5}

Tabela 2 – Referente a Simulação 2

Figura 13 – Simulação 3

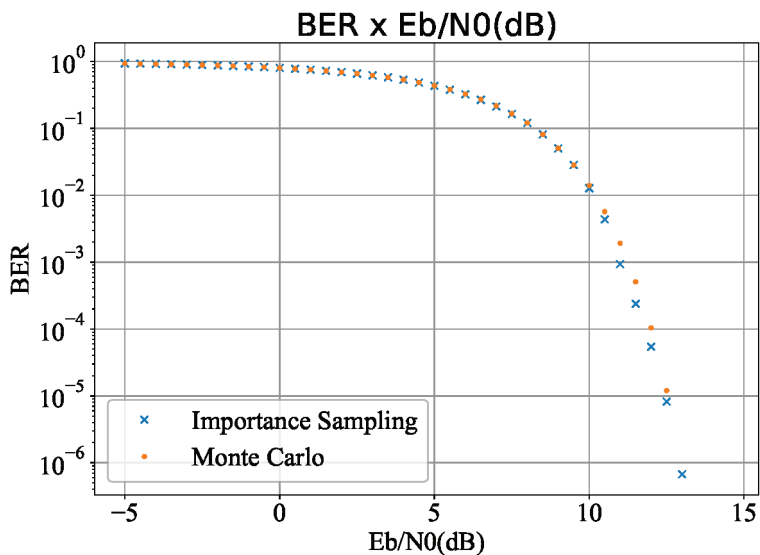


Fonte: Elaborada pelo Autor

Método	Nº de Amostras	Tempo de Execução(seg)	Precisão
Monte Carlo	10^6	57	10^{-5}
Importance Sampling	10^4	1	10^{-6}

Tabela 3 – Referente a Simulação 3

Figura 14 – Simulação 4

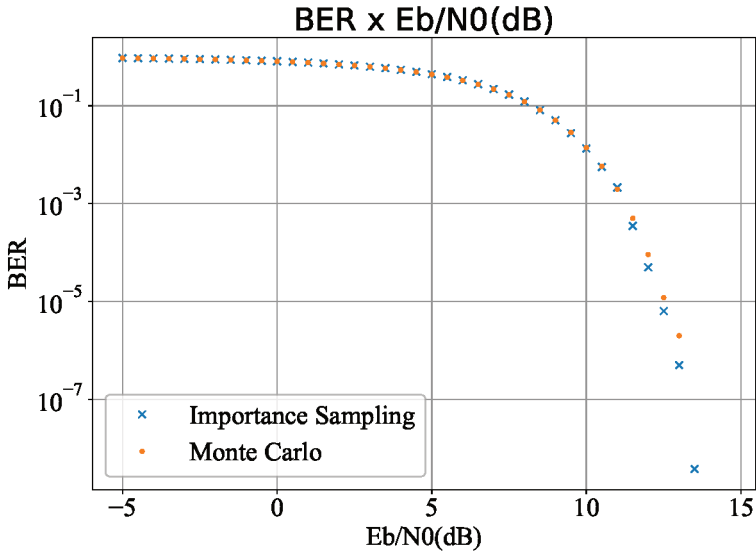


Fonte: Elaborada pelo Autor

Método	Nº de Amostras	Tempo de Execução(seg)	Precisão
Monte Carlo	10^6	57	10^{-5}
Importance Sampling	10^5	7	10^{-6}

Tabela 4 – Referente a Simulação 4

Figura 15 – Simulação 5

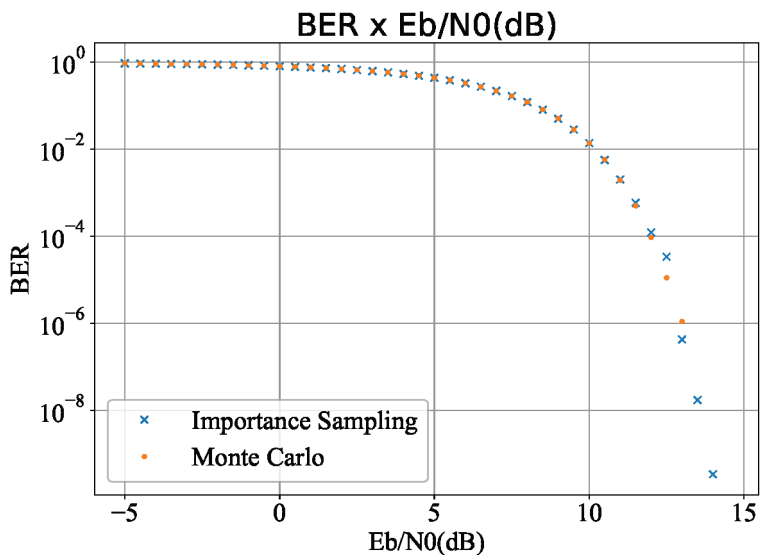


Fonte: Elaborada pelo Autor

Método	Nº de Amostras	Tempo de Execução(seg)	Precisão
Monte Carlo	10^6	57	10^{-6}
Importance Sampling	10^6	78	10^{-7}

Tabela 5 – Referente a Simulação 5

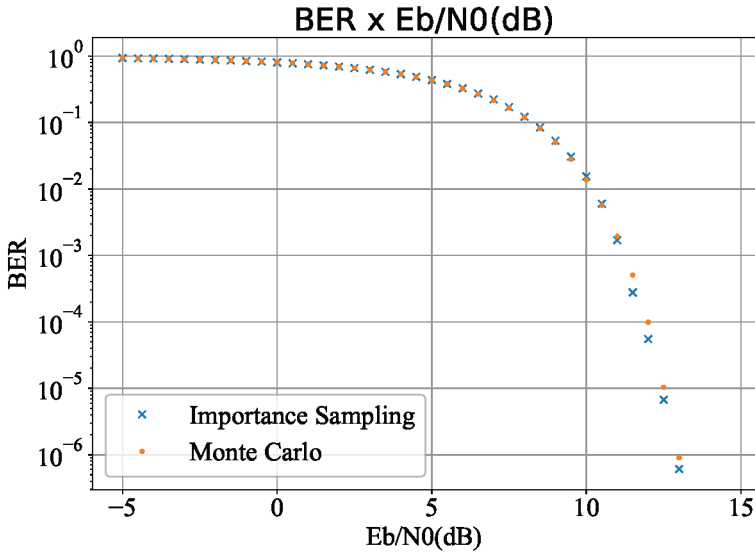
Figura 16 – Simulação 6



Método	Nº de Amostras	Tempo de Execução(seg)	Precisão
Monte Carlo	10^7	570	10^{-6}
Importance Sampling	10^4	788	10^{-8}

Tabela 6 – Referente a Simulação 6

Figura 17 – Simulação 7



Fonte: Elaborada pelo Autor

Método	Nº de Amostras	Tempo de Execução(seg)	Precisão
Monte Carlo	10 ⁷	570	10 ⁻⁶
Importance Sampling	10 ⁵	7	10 ⁻⁶

Tabela 7 – Referente a Simulação 7

Depois, foram feitas novas simulações, em que obteve-se a probabilidade de erro do sistema para uma razão $E_b \setminus N_0$ fixada via MC e então foi simulado o algoritmo de *IS* até que se achasse o menor número de amostras que estimasse a probabilidade de erro do

método de *MC* com até 10% de tolerância. O resultado segue nas tabelas abaixo:

Parâmetros	Monte Carlo	Importance Sampling
Nº de Amostras	10^6	240
Probabilidade de Erro	4.95×10^{-4}	5.05×10^{-4}

Tabela 8 – Resultados da segunda etapa de simulações

Parâmetros	Monte Carlo	Importance Sampling
Nº de Amostras	10^7	3×10^3
Probabilidade de Erro	1.31×10^{-5}	1.386×10^{-5}

Tabela 9 – Resultados da segunda etapa de simulações

4.1 Discussão de resultados

Nas Figuras 11, 12, 13, 14, 15, 16 e 17 é possível ver o desempenho do algoritmo proposto versus o algoritmo de Monte Carlo através de curvas de BER x E_b/N_0 . Nas Figuras 12, 13, 14 e 17 (Simulações 2, 3, 4 e 7) vemos que para um n^o de amostras até três ordens de magnitude menor do que o n^o de amostras pelo método de *MC*, o resultado obtido é muito parecido, mostrando que o resultado através de *IS* converge para o método de *MC* com uma velocidade maior.

Já nas Figuras 11, 15 e 16 (Simulações 1, 5 e 6), com um número de amostras fixado para os dois, a precisão obtida através de IS chega a ser de duas à três ordens de magnitude maior, com um aumento não muito significativo no tempo de execução, dado a precisão que conseguiu ser relatada.

Um dos parâmetros mais afetados, o tempo de execução, tem uma mudança muito impactante quando a precisão precisa ser muito alta. Podemos ver na Figura 17 que para uma precisão de 10^{-6} , o tempo de execução no método de MC foi aproximadamente 81x maior. Devido a restrições de memória no equipamento utilizado, não se pode simular o algoritmo com um n^o de amostras maior que 10^7 , mas pode-se inferir que o ganho em tempo de execução seria ainda maior ao simular um sistema com um número maior de amostras.

Já na segunda rodada de simulações, podemos ver na Tabela 8 que obteve-se a probabilidade de erro desejada com tolerância de 2.02% e uma redução de 4166x no número de amostras necessárias, ou aproximadamente três ordens de grandeza. Na Tabela 9, obteve-se a probabilidade de erro desejada com tolerância de 5.8%, e uma redução no número de amostras de 3333x, ou aproximadamente três ordens de grandeza também.

Estes resultados estão de acordo com os obtidos nas simulações anteriores e mostram que é possível obter resultados confiáveis ao estimar a probabilidade de erro com uma redução de até três ordens magnitude no número de amostras simuladas.

5 Conclusão

Como um método alternativo ao método de *MC*, o *IS* apresenta ganhos tanto em tempo de processamento como memória utilizada. O ganho obtido em números de amostras necessárias para a simulação pelo método avaliado em relação ao método tradicional chegam a três ordens de magnitude, e é alcançada uma redução significativa no tempo de simulação do algoritmo, também em comparação com o método tradicional.

Seja para a implementação em sistemas com baixa capacidade de memória ou em sistemas complexos em que a precisão necessária tem que ser alta, o *Importance Sampling* consegue substituir o método de *Monte Carlo* com eficácia. Pesquisas na área mostram que ainda há muito espaço para novas descobertas nesse assunto, como o *Adaptive Importance Sampling*, que utiliza uma nova abordagem do método para processamento em tempo real.

Este trabalho abre espaço para futuramente trabalhar-se numa melhoria para o método, assim como uma futura implementação física.

6 Sugestões para trabalhos futuros

A seguir, são sugeridas algumas modificações a serem realizadas em trabalhos futuros, com o intuito de aperfeiçoar o algoritmo e testá-lo em um ambiente realista.

- Otimização do algoritmo para o alcance de tempos de simulação ainda menores;
- Modificação do sistema para implementação em *SDR* ou *FPGA* (*Field Programmable Gate Array*);
- Estudo e implementação do *Importance Sampling* Adaptativo, para a implementação de um estimador de BER em tempo real;

Referências

- [1] TAYLOR, S. **10 Predictions for the Future of the Internet of Things**. Cisco Blogs, San Jose, 03/jun. 2015. Disponível em: <<https://blogs.cisco.com/cle/10-predictions-for-the-future-of-the-internet-of-things>>. Acesso em: 05 julho 2018.
- [2] TULLBERG, Hugo et al. The METIS 5G system concept: Meeting the 5G requirements. In: **IEEE Communications Magazine**. Institute of Electrical and Electronics Engineers (IEEE), 2016. p. 132-139.
- [3] JERUCHIM, Michel. Techniques for estimating the bit error rate in the simulation of digital communication systems. **IEEE Journal on selected areas in communications**, v. 2, n. 1, p. 153-170, 1984.
- [4] METROPOLIS, N. Monte Carlo Method. **From Cardinals to Chaos: Reflection on the Life and Legacy of Stanislaw Ulam**, p. 125, 1989.
- [5] SRINIVASAN, Rajan. Importance sampling: **Applications in communications and detection**.

Springer Science & Business Media, 2013.

[6] Psychstatistics. **Stata: Graphing**

Distributions. Disponível em:

<<http://www.psychstatistics.com/2010/11/24/stata-graphing-distributions>>. Acesso em: 10 junho 2018. 6*

[7] KAHN, Herman; HARRIS, Theodore E.

Estimation of particle transmission by random sampling. **National Bureau of Standards applied mathematics series**, v. 12, p. 27-30, 1951.

[8] BUGALLO, Monica F. et al. Adaptive importance sampling: the past, the present, and the future. **IEEE Signal Processing Magazine**, v. 34, n. 4, p. 60-79, 2017.

[9] BARFORD, Lee. Sequential Bayesian bit error rate measurement. **IEEE Transactions on Instrumentation and measurement**, v. 53, n. 4, p. 947-954, 2004.

[10] WOLFRAM. MathWorld. Algebra. Polynomials. **Primitive Polynomial**. Disponível em:

<<http://mathworld.wolfram.com/PrimitivePolynomial.html>>. Acesso em: 05 julho 2018.

[11] WIKIPEDIA, THE FREE ENCYCLOPEDIA.

Linear-feedback shift register. Disponível em: <<https://en.wikipedia.org/wiki/Linear->

feedback_shift_register> Acesso em: 20 maio 2018.

[12] ISLAM, Muhammad et al. Bit-Error-Rate (BER) for modulation technique using Software defined Radio. In: **Electrical Engineering and Informatics, 2009. ICEEI'09. International Conference on.** IEEE, 2009. p. 445-447.

[13] SHANMUGAM, K.; BALABAN, Philip. A modified Monte-Carlo simulation technique for the evaluation of error rate in digital communication systems. **IEEE Transactions on Communications**, v. 28, n. 11, p. 1916-1924, 1980.