Adaptive Importance Sampling

CE 653: Structural Reliability and Risk Analysis Course Project
by

Arihant Parsoya (15D260012)

under the guidance of

Prof. Siddhartha Ghosh

Civil Engineering
Indian Institute of Technology, Bombay
Mumbai 400 076

Contents

List of Figures

Chapter 1

Introduction

Numerical techniques to calculate integration have become increasingly popular due to advances in computational capabilities. Monte Carlo is one of the most popular and effective tool used for computing integrals, especially for multivariate problems [1]. However, in many cases, Monte Carlo simulation becomes computationally expensive as a large number of simulations are required to reduce the variance. Several methods have been proposed to reduce the variance and improve the quality of results in less number of simulations such as Importance Sampling[2], Antithetic Variate [1] and Latin Hypercube Sampling [3]. In the late 1980s, Adaptive Importance Sampling was proposed by Christian[7] to reduce statistical error in Importance Sampling and solve other problems associated with importance sampling. Since then, adaptive sampling has been used in several domains such as structural reliability and computer graphics. Several adaptive techniques have been proposed in adaptive sampling after Christian. The purpose of this report is to describe the Adaptive importance sampling algorithm and evaluate its usability for rate event simulation. Iterative Fast Monte Carlo procedure proposed by Christian[7] is described and evaluated.

Chapter 2

Literature Overview

Several numerical techniques used to evaluate integrals such as trapezoidal rule, Simpsons rule and other methods based on polynomials such as Laguerre-Gauss or Gauss-Hermite. These methods however, tend to give poor results when the number of random variables are increased. To deal with these situations, approximate and probabilistic methods such as the Monte Carlo were developed.

Monte Carlo methods use random sampling methods to evaluate the problem and get reasonable estimate of the solution (MacKay [11]). They are used to solve diverse problems such as optimization, integration, statistical physics(Binder[12]) and several other engineering problems. However, the Monte Carlo simulation are often consuming for problems of structural analysis. Several methods such as Importance Sampling[2], Antithetic Variate [1] and Latin Hypercube Sampling [3] have been proposed to make the Monte Carlo more efficient. Monte Carlo methods have been combined with other computation techniques such as Neural Networks[13], Bayes Networks[14] and Markov Chain[15].

Adaptive Monte Carlo simulation is used to improve the quality of ray tracing image in Computer Graphics (Jensen [9]). They are used to reduce aliasing caused due to undersampling in the finally rendered image (Halton[10]). Adaptive sampling is also used in process control[17] and sensor networks[16].

Adaptive importance sampling is used for applications such as system reliability analysis[18], training neural probabilistic models[19], evaluating discrete markov chains[20].

Chapter 3

Adaptive Importance Sampling

3.1 Problem

The problem of the Monte Carlo simulation can be formulated as follows:

$$p_f = \int_{D_f} f_X(x)\mathrm{d}x \tag{3.1}$$

Where p_f is the probability of failure, D_f is the domain of failure where the limit state function $G < 0$ and f_X is the joint probability distribution for the random variables. The Monte Carlo estimate of p_f can be calculated as

$$\overline{p}_\mathrm{f} = E\{I_{D_f}\} = \frac{1}{N}\sum_{i=1}^{N} I_{D_\mathrm{f}}(x_i) \tag{3.2}$$

Where N is the number of samples used in the Monte Carlo simulation. The samples x_i are drawn from the joint probability distribution f_X. I_D is the Indicator function defined as follows

$$I_D(x) = \begin{cases} 1, & G(x) \leq 0 \\ 0, & \text{elsewhere} \end{cases} \tag{3.3}$$

The variance of Monte Carlo estimate from equation (3.2) is

$$\begin{aligned} \operatorname{var}\overline{p}_f &= \tfrac{1}{K}\left(p_f - p_f^2\right) \\ &\approx \tfrac{p_f}{K} \text{ for small } p_f \end{aligned} \tag{3.4}$$

3.2 Importance Sampling

Equation (3.4) can be used to estimate the number of samples needed for the required accuracy of the results. Often in problems of structural reliability, the estimated p_f is extremely small which leads to a large number of samples for the desired accuracy. Importance sampling is a technique which aims to reduce the number of simulation points used in Monte Carlo simulation. This is achieved by using alternative density function g_Y (known as biasing density) to use x_i such that $G(x_i) < 0$ occurs more frequently. The biasing function facilitates more simulation points in the domain of failure. The original integration equation (3.1) can be rewritten as follows

$$p_{\mathrm{f}} = \int_{D_{\mathrm{f}}} \frac{\mathrm{f}_X(y)}{g_Y(y)} g_Y(y)\mathrm{d}x = \int_{D_{\mathrm{f}}} W(y) g_Y(y)\mathrm{d}y \tag{3.5}$$

Where g_Y is called the sampling density and the likelihood ratio W is defined as

$$W(\cdot) \equiv \frac{f_X(\cdot)}{g_Y(\cdot)} \tag{3.6}$$

The new Monte Carlo equation can be rewritten as

$$\overline{p}_{\mathrm{f}} = E\{I_{D_f}(Y)W(y)\} = \frac{1}{N}\sum_{i=1}^{N} I_{D_{\mathrm{f}}}(y_i)\,\frac{f_X(\ y_i)}{g_Y(y_i)} \tag{3.7}$$

Where the samples y_i are drawn from the sampling density function g_Y. The variance for Importance sampling is given by

$$\begin{aligned} \operatorname{var}\overline{p}_f &= \frac{1}{N}\operatorname{var}\{I_{D_{\mathrm{f}}} W(X)\} \\ &= \frac{1}{N}\left[E\left\{I_{D_{\mathrm{f}}}^2 W^2(X)\right\} - p_f^2\right] \end{aligned} \tag{3.8}$$

As observed from the equation (3.8), the variance of $\overline{p}_f$ can be reduced by choosing the right likelihood ratio W. Several techniques have been proposed to choose the likelihood ratio such as scaling (Shanmugam & Balaban [4]), translation (Lu & Yao [5]), exponential twisting (Siegmund [6]) and adaptive importance sampling (Christian [7]).

3.3 Adaptive Importance Sampling

The standard error of estimated probabilities reduces to zero if the sampling density is chosen to be the original density over the probability of failure. It is not practically possible, however, a closer approximation to the original density is found. Adaptive importance sampling techniques use this idea to find better estimates of the likelihood function W to reduce the variance of the simulation results.

Adaptive importance sampling technique adapts the likelihood function W while the simulation is being run. Several techniques have been proposed on how the likelihood function updates after each iteration. Adaptive sampling technique proposed by Christian[7] (also known as Iterative Fast Monte Carlo procedure) is described in the following sections. The algorithm is described for uncorrelated random variables, the same algorithm can be used for correlated random variables after transforming them to uncorrelated space.

An initial simulation is run to find the initial parameters of the sampling density function. Multiple techniques can be used to find the initial parameters of the sampling density function. Two initiation techniques are described in the next section. Any one of them can be used to find the initial parameters of the sampling density function.

3.3.1 Starting Procedure

Sensitivity Analysis

To find the parameters of sampling density function g, a sensitivity analysis is done by simulating each random variable while keeping the remaining random variable at their mean values. The simulation gives us the conditional probability of failure p_{f_i} conditioned on the failure domain. Correspondingly, the Iterative Fast Monte Carlo (IMF) points $\overline{x}_{g_{\mathrm{i}}}$ are found by taking the mean of the points in the domain of failure. The standard deviation of each random variable σ_{g_i} is also found through the simulation. The mean of g is found by taking a weighted mean of all IMF points $\overline{x}_{g_{\mathrm{i}}}$.

$$\overline{x}_g = p_{f_i}\overline{x}_{g_i} / \sum_{i=1}^{n} p_{\mathrm{f}_i} \tag{3.9}$$

The vector $\overline{x}_g$ can also be interpreted as the centre of mass of conditional probabilities p_{f_i}. Obtained $\overline{x}_g$ and standard deviations σ_{g_i} is used to initiate the importance sampling. Sometimes, the obtained vector $\overline{x}_g$ lies entirely in the failure domain which could lead to unconservative estimates of the failure probabilities. In those cases, linear interpolation is done between the $\overline{x}_g$ and the mean vector $\overline{x}$ to find the boundary point of the failure domain. That point is then used as the $\overline{x}_g$.

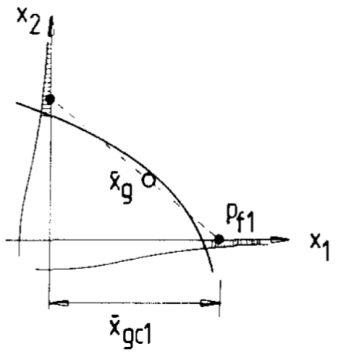

Figure 3.1: Mean vector $\overline{x}_g$ as centre of mass of conditional failure of probabilities p_{f_i} [Adaptive sampling an iterative fast Monte Carlo procedure, Bucher, Christian G. Structural safety 5, no. 2 (1988): 119-126.]

Pure Monte Carlo Simulation

Pure Monte Carlo simulation is run on the entire problem domain to find the points x where $G < 0$. The first and second moments of simulation points are then used to estimate the parameters for g as follows

$$\begin{aligned} E_{\mathrm{g}}[y] &= E_{\mathrm{f}}[y|y \in D_{\mathrm{f}}] \\ E_{\mathrm{g}}\left[yy^{\mathrm{T}}\right] &= E_{\mathrm{f}}\left[yy^{\mathrm{T}}|y \in D_{\mathrm{f}}\right] \end{aligned} \tag{3.10}$$

3.3.2 Algorithm

After the initial starting vector has been found, importance sampling is run multiple times for a given number of iteration. After each run, the sampling density function is updated based on the data obtained on the previous runs. The parameters of the sampling density function are updated to match the first and second moment of the failure points

$$\begin{aligned} E_{\mathrm{g}}[y] &= E_{\mathrm{f}}[y|y \in D_{\mathrm{f}}] \\ E_{\mathrm{g}}\left[yy^{\mathrm{T}}\right] &= E_{\mathrm{f}}\left[yy^{\mathrm{T}}|y \in D_{\mathrm{f}}\right] \end{aligned} \tag{3.11}$$

3.4 Example

The algorithm detailed in section 3.3 was implemented in Python. The code of the program is given in the Appendix. The following examples are solved using both Monte Carlo and Adaptive sampling approaches. Example 1 uses Pure Monte Carlo to find the starting iteration parameters and Example 2 uses sensitivity analysis to find the initial parameters of g.

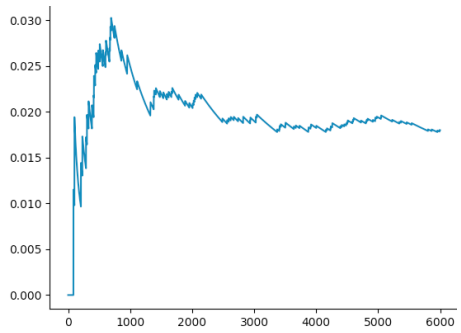

Figure 3.2: Monte Carlo simulation of Example 1

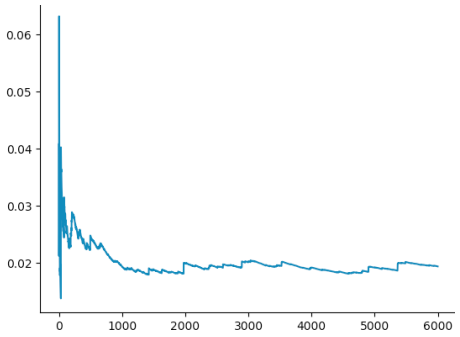

Figure 3.3: Adaptive Importance Sampling simulation of Example 1

3.4.1 Example 1

Consider the limit state function defined as as

$$g = f_y A_{st}\left[d - 0.771\frac{f_y A_{st}}{f_{ck} b}\right] - (w_{\mathrm{DL}} + w_{\mathrm{LL}})\,\frac{L^2}{8} \tag{3.12}$$

RV	Distribution	Mean	SD	Unit
f_y	Normal	470	35	$\mathrm{N/mm^2}$
f_{ck}	Normal	35	4	$\mathrm{N/mm^2}$
A_{st}	Normal	450	25	$\mathrm{mm^2}$
w_{DL}	Normal	25	3	N/mm
w_{LL}	Normal	3	1	N/mm
d	Deterministic	230	-	mm
b	Deterministic	500	-	mm
L	Deterministic	4650	-	mm

Table 3.1: Example 1 Parameters

Whose parameters are defined in table 3.1. Both Monte Carlo and Adaptive Importance Sampling simulation is carried out for a total of 6000 samples. The Adaptive sampling simulation is carried out in 6 sets, each containing 1000 simulations. Pure Monte Carlo was used to estimating the parameters of the sampling density function for the first run.

The table 3.2 and figure (3.3 and 3.2) Compares the result between the results obtained from Adaptive Sampling and Monte Carlo Simulation.

Iteration	Monte Carlo (p_f)	Adaptive Sampling (p_f)
500	0.0260	0.0248
1000	0.0250	0.0194
1500	0.0220	0.0188
2000	0.0205	0.0200
2500	0.0192	0.0192
3000	0.0190	0.0201
3500	0.0182	0.0195
4000	0.0182	0.0192
4500	0.0186	0.0183
5000	0.0194	0.0192
5500	0.0187	0.0201
6000	0.0180	0.0194

Table 3.2: Example 1 results of Monte Carlo and Adaptive Importance Sampling

3.4.2 Example 2

Consider the limit state function defined as

$$G = (4 - x_1)(4 - x_2) \tag{3.13}$$

The variables x_1 and x_2 are normal random variables with mean 0 and standard deviation 2. This limit state function has two "design points". A sensitivity analysis described in section 3.3.1 was used to find the initial point of iteration. Both Monte Carlo and Adaptive Importance Sampling simulation is carried out for a total of 6000 samples. The Adaptive sampling simulation is carried out in 6 sets, each containing 1000 simulations.

Table 3.3 and graphs 3.4 & 3.3 show the results of the simulation. As seen from the results, we can see that adaptive sampling has underestimated the failure probability. The reason for this error is that the adaptive sampling algorithm converged to a local minimum.

Iteration	Monte Carlo (p_f)	Adaptive Sampling (p_f)	$\overline{x}_{g_1}$	σ_{g_1}	$\overline{x}_{g_2}$	σ_{g_2}
500	0.0400	0.0710	2.2739	0.9151	2.5923	0.625
1000	0.0390	0.0408	3.9665	0.9593	2.8091	0.8975
1500	0.0393	0.0395	3.9665	0.9593	2.8091	0.8975
2000	0.0430	0.0440	4.5813	0.7827	2.8608	0.931
2500	0.0424	0.0373	4.5813	0.7827	2.8608	0.931
3000	0.0420	0.0334	4.8626	0.6337	2.7685	0.8187
3500	0.0425	0.0301	4.8626	0.6337	2.7685	0.8187
4000	0.0415	0.0337	4.9749	0.5884	2.7090	0.741
4500	0.0426	0.0310	4.9749	0.5884	2.7090	0.741
5000	0.0420	0.0296	5.0631	0.5434	2.6747	0.6882
5500	0.0430	0.0276	5.0631	0.5434	2.6747	0.6882
6000	0.0440	0.0261	5.0631	0.5434	2.6747	0.6882

Table 3.3: Example 2 results of Monte Carlo and Adaptive Importance Sampling

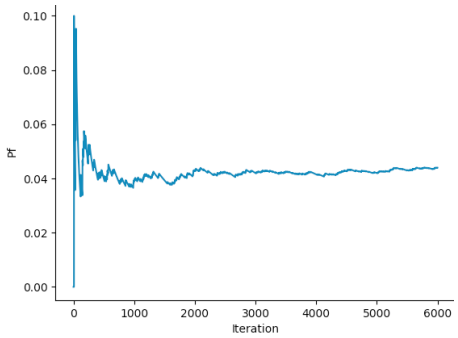

Figure 3.4: Monte Carlo simulation of Example 2

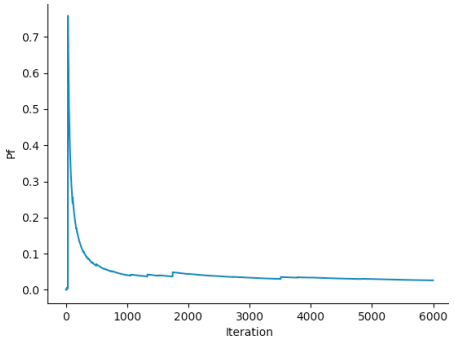

Figure 3.5: Adaptive Importance Sampling simulation of Example 2

Chapter 4

Discussion

From Example 1, it can be observed that the algorithm converges faster than Monte Carlo simulation. However, large number of points were required to be simulated to get good initial estimate of sampling density function. Hence, the algorithm is prone to its initial point of iteration. Also, the number of iterations in each run is required to be large to have a good estimate of sampling density in next run.

As observed from Example 2, the results obtained by the adaptive importance sampling is unconservative. This is due to convergence of the algorithm to a local minimum. Several techniques have been proposed to prevent unconservative estimates.

The algorithm has proven to be very effective over importance sampling because:

- **Initial analysis** such as evaluation of design point is not required. The algorithm automatically finds the domain of failure and samples over D_f without extra analysis such as FORM and SORM.
- **Can handle multiple design points** . Importance sampling cannot handle multiple design points because the sampling density function is unvarying. This makes it impossible for importance sampling to handle multiple design points. Adaptive sampling is capable of handling multiple design points because its sampling density function is not static.
- **Does not require explicit limit state function**

Although this method converges faster than Monte Carlo simulation, the method suffers from several limitations as:

- **Prone to initial iteration point** . The results of the simulation might get altered when the initial sampling point is changed. If the initial sampling point lies in the failure domain, unconservative estimates of the failure probability will have resulted. This is handled by not allowing the sampling point inside the failure domain, instead of restricting it to the limit state boundary.

- **Large number of iteration per run is required** . The algorithm gives better results when a large number iterations are used for each run before updating the sampling density function.

- **Prone to converging at sub optimal point** . As seen in Example 2 of the last section the algorithm converged at a sub optimal point which led to underestimating of the actual probability of failure.

- **Initial sensitivity analysis for estimation of sampling density parameters** does not always work because varying one variable at a time while keeping the remaining at their mean need not fall into the failure domain. In those cases, Pure Monte Carlo can be used to estimate sampling density parameters.

Several other adaptive techniques have been proposed such as Adaptive Stratified Monte Carlo algorithm [8] which is based on dividing the entire domain into smaller sub spaces. The integration is carried out for each sub space and the final result is calculated as the cumulative of integration done in each sub space. The subspace is divided into multiple sub spaces where the value of integrating function is higher (figure 4.1).

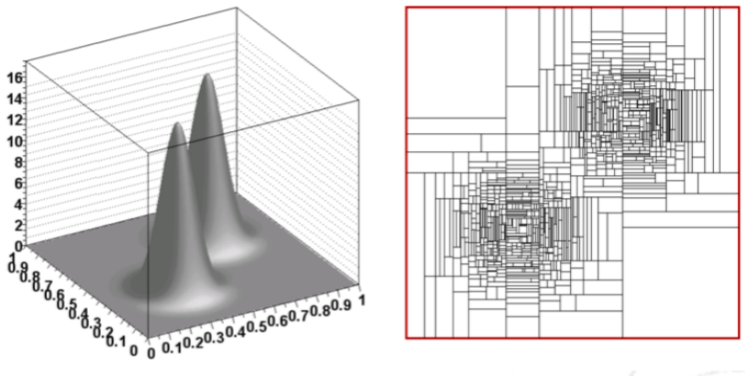

Figure 4.1: Pictorial representation of space stratification for Adaptive Stratified Monte Carlo algorithm from [Adaptive Monte Carlo Integration Methods, Marcin Chrzszcz, University of Zurich]

Chapter 5

Summary

Adaptive importance technique sampling proposed by Christian[7] was studied in detail. The technique adapts the sampling density function while the simulation is being run, this helps in converging to the solution faster than importance sampling approaches. Adaptive importance sampling technique can be useful over importance sampling in cases when the limit state function has multiple potential design points. The technique also reduces the initial computation of finding the design point in importance sampling. While this report deals with adaptive sampling in scope of rare event simulations, variations of this technique are used to solve problems in other academic fields.

Appendix

Code and Documentation

The following is the code for Adaptive Sampling implementation in Python. Three functions are implemented in the code: *PureMonteCarlo*, *SensitivityAnalysis* and *RunImportanceSampling*. The first two functions (*PureMonteCarlo* & *SensitivityAnalysis*) are used to find the initial parameters of sampling density function g. Any one of them can be used.

The function *RunImportanceSampling* runs the simulation after in multiple runs. All global variables of the program are defined at the beginning of the code.

```
from math import *
from sympy import *
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt


# Global Variables
x1, x2= symbols('x1, x2')

Vars = [x1, x2]
fParam = {
#variable: [mean, std]
  x1: [0, 2],
  x2: [0, 2],
}

gParam = { # Sampling Density function
#variable: [mean, std]
  x1: [0, 0],
  x2: [0, 0],
```

```
}

# Limit State Function
G = (4 - x1)*(4 - x2)

Id = []
Pf = []

xFailure = {} # points which lie in domain of failure
for v in Vars:
  xFailure[v] = []

def PureMonteCarlo(iteration):
  print("Running␣Pure␣Monte␣Carlo␣Simulation...")
  xFailure = {} # points which lie in domain of failure
  for v in Vars:
    xFailure[v] = []

  for x in range(0, iteration):
    data = {}
    for v in Vars:
      data[v] = np.random.normal(fParam[v][0], fParam[v][1])

    if G.subs(data) < 0:
      for v in Vars:
        xFailure[v].append(data[v])

  # Update the Mean and Std of Sampling function
  for v in Vars:
    gParam[v][0] = np.mean(xFailure[v])
    gParam[v][1] = np.std(xFailure[v])


def SensitivityAnalysis(iteration):
  print('Sensitivity␣Analysis...')
  Pfi = {}
  xFailure = {} # points which lie in domain of failure

  for v in Vars:
```

```
    Pfi[v] = 0
    xFailure[v] = []

  for v in Vars:
    # Set all the random variables to their mean
    data = {}
    for j in Vars:
      data[j] = fParam[j][0]

    for i in range(0, iteration):
      data[v] = np.random.normal(fParam[v][0], fParam[v][1])
      if G.subs(data)<0:
        xFailure[v].append(data[v])
        Pfi[v] += 1

    Pfi[v] = Pfi[v]/iteration

  NormFactor = 0
  for v in Vars:
    NormFactor += Pfi[v]

  for v in Vars:
    gParam[v][0] = fParam[v][0] + Pfi[v]/NormFactor*(np.mean(
       xFailure[v]) - fParam[v][0])
    gParam[v][1] = np.std(xFailure[v])


def RunImportanceSampling(iteration, run):
  print("Running␣Adaptive␣Importance␣Sampling")
  for r in range(run):
    for x in range(0, iteration):
      data = {}
      for v in Vars:
        data[v] = np.random.normal(gParam[v][0], gParam[v
           ][1])

      if G.subs(data) < 0:
        prob = 1
        for v in Vars:
```

```
            prob *= norm(fParam[v][0], fParam[v][1]).pdf(data[v
               ])
            prob *= 1/norm(gParam[v][0], gParam[v][1]).pdf(data
               [v])
            xFailure[v].append(data[v])
          Id.append(prob)
        else:
          Id.append(0)

        Pf.append(sum(Id)/len(Id))

      # Update the Mean and Std of Sampling function
      for v in Vars:
        gParam[v][0] = np.mean(xFailure[v])
        gParam[v][1] = np.std(xFailure[v])


# Function 'PureMonteCarlo' or 'SensitivityAnalysis' can be
   used to
# find the initial parameters of sampling function g.

# PureMonteCarlo(500)
SensitivityAnalysis(1000)
RunImportanceSampling(1000, 6) # 1000 iterations in 6 runs


print("Probabiliy of Failure is: {}".format(Pf[-1]))
plt.plot(Pf)
plt.xlabel("Iteration")
plt.ylabel("Pf")
plt.title("Adaptive Sampling Simulation")
plt.show()
```

Bibliography

[1] R.Y. Rubinstein *Simulation and the Monte-Carlo Method*. John Wiley Sons, New York, NY, 1981.

[2] A. Harbitz *Efficient and accurate probability of failure calculation by the use of importance sampling technique*. Proc. 4th Int. Conf. on Applications of Statistics and Probability in Soil and Structural Engineering (ICASP 4),Pitagora, Editrice, Bologna, Italy, 1983, pp. 825-886.

[3] R.L. Iman and W.J. Canover *Small sampling sensitivity analysis techniques for computer models with an application to risk assessment*. Commun. Stat., Theory Methods, A9(17) (1980) 1749-1842.

[4] K. S. Shanmugam and P. Balaban *A modified Monte Carlo simulation technique for the evaluation of error rate in digital communication systems*. IEEE Trans. Communications, COM-28, pp. 1916-1924, Nov. 1980.

[5] D. Lu and K. Yao *Improved importance sampling technique for efficient simulation of digital com- munication systems*. IEEE Journal on Selected Areas in Communications, vol. 6, Jan. 1988, pp 67-75.

[6] D. Siegmund *Importance sampling in the Monte Carlo study of sequential tests*. Ann. Statist.,vol. 4, 1976, pp 673-684.

[7] Bucher, Christian G. *Adaptive sampling an iterative fast Monte Carlo procedure*. Structural safety 5, no. 2 (1988): 119-126.

[8] Sayah, Toni *Adaptive stratified monte carlo algorithm for numerical computation of integrals*. Mathematics and Computers in Simulation, Volume 157, March 2019, Pages 143-158

[9] Jensen, Henrik Wann, James Arvo, Phil Dutre, Alexander Keller, Art Owen, Matt Pharr, and Peter Shirley *Monte Carlo ray tracing*. In ACM SIGGRAPH, pp. 27-31. 2003.

[10] Halton, John H. *Monte Carlo Anti-Aliasing*. No. TR88-018. NORTH CAROLINA UNIV AT CHAPEL HILL DEPT OF COMPUTER SCIENCE, 1989.

[11] MacKay, David JC. *Introduction to monte carlo methods*. In Learning in graphical models, pp. 175-204. Springer, Dordrecht, 1998.

[12] Binder, Kurt, Dieter Heermann, Lyle Roelofs, A. John Mallinckrodt, and Susan McKay. *Monte Carlo simulation in statistical physics*. Computers in Physics 7, no. 2 (1993): 156-157.

[13] Papadrakakis, Manolis, Vissarion Papadopoulos, and Nikos D. Lagaros. *Structural reliability analyis of elastic-plastic structures using neural networks and Monte Carlo simulation*. Computer methods in applied mechanics and engineering 136, no. 1-2 (1996): 145-163.

[14] , Dani, and Hedibert F. Lopes. textitMarkov chain Monte Carlo: stochastic simulation for Bayesian inference. Chapman and Hall/CRC, 2006.

[15] Gilks, Walter R., Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman and Hall/CRC, 1995.

[16] Jain, Ankur, and Edward Y. Chang. *Adaptive sampling for sensor networks*. In Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004, pp. 10-16. ACM, 2004.

[17] Runger, George C., and Joseph J. Pignatiello Jr. *Adaptive sampling for process control*. In Proceeedings of the 1st international workshop on Data management for sensor Journal of Quality Technology 23, no. 2 (1991): 135-155.

[18] Bengio, Yoshua, and Jean-Sbastien Sencal. *Adaptive importance sampling to accelerate training of a neural probabilistic language model*. IEEE Transactions on Neural Networks 19, no. 4 (2008): 713-722.

[19] Dey, Animesh, and Sankaran Mahadevan. *Ductile structural system reliability analysis using adaptive importance sampling*. Structural safety 20, no. 2 (1998): 137-154.

[20] Kollman, Craig, Keith Baggerly, Dennis Cox, and Rick Picard. *Adaptive importance sampling on discrete Markov chains*. Annals of Applied Probability (1999): 391-412.