Advanced stochastic simulation methods for solving high-dimensional reliability problems

by

Konstantin Zuev

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Civil Engineering

January 2009, Hong Kong

Authorization

Konstantin Zuev

Konstantin Zuev

6 January, 2009

Advanced stochastic simulation methods for solving
high-dimensional reliability problems

by

Konstantin Zuev

This is to certify that I have examined the above PhD thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

[illegible]

Professor Lambros S. Katafygiotis (Thesis Supervisor)

[illegible]

Professor Moe M. S. Cheung (Department Head)

Department of Civil and Environmental Engineering
6 January, 2009

Acknowledgements

My colleague Jia (Jayson) Wang greatly helped with the experimental sections.

Special thanks go to my thesis supervisor Professor Lambros Katafygiotis for his advice, guidance, editing, and helping me to keep things in perspective.

Also, I want to thank my thesis supervision committee Professor Chih-Chen Chang and Professor Jun Shang Kuang from the Department of Civil and Environmental Engineering, Professor Li Qiu from the Department of Electronic and Computer Engineering, and Professor Jianye Ching from the National University of Taiwan.

Last but not least, I want to thank my Mother and my future wife Liza Bradulina for their support, which cannot be overestimated.

Contents

List of Figures

List of Tables

Advanced stochastic simulation methods for solving high-dimensional reliability problems

by Konstantin Zuev

Abstract

This work is dedicated to the exploration of commonly used and development of new advanced stochastic simulation algorithms for solving high-dimensional reliability problems.

Firstly, adopting a geometric point of view we highlight and explain a range of results concerning the performance of several reliability methods. Namely, we discuss Importance Sampling in high dimensions and provide a geometric understanding as to why Importance Sampling does generally "not work" in high dimensions. We furthermore challenge the significance of "design point" when dealing with strongly nonlinear problems. We conclude by showing that for the general high-dimensional nonlinear reliability problems the selection of an appropriate Importance Sampling density (ISD) is practically impossible. Also, we provide a geometric explanation as to why the standard Metropolis-Hastings (MH) algorithm does "not work" in high-dimensions.

Next, we develop two useful modifications of the well-known reliability methods. The first is Adaptive Linked Importance Sampling (ALIS), which generalizes Subset Simulation (SS) and in some cases can offer drastic improvements over SS. The second is Modified Metropolis-Hastings algorithm with Delayed Rejection (MMHDR) which is a novel modification of the MH algorithm, designed specially for sampling from conditional high-dimensional distributions.

Finally, we propose a novel advanced stochastic simulation algorithm called Horseracing Simulation (HRS). The idea behind HS is the following. Although the reliability problem itself is high-dimensional, the limit-state function maps this high-dimensional parameter space into a one-dimensional real line. This mapping transforms a high-dimensional random parameter vector, which represents the input load, into a random variable with unknown distribution, which represents the structure response. It turns out, that the corresponding cumulative distribution function (CDF) of this random variable of interest can be accurately approximated by empirical CDFs constructed from specially designed samples. The accuracy and efficiency of the new method is demonstrated with a real-life wind engineering example.

Chapter 1

Introduction

"Where shall I begin, please your Majesty ?"
"Begin at the beginning," the King said gravely,
"and go on till you come to the end: then stop."

Alice's Adventures in Wonderland, Lewis Carroll

In reliability engineering our task is to calculate the reliability or equivalently the probability of failure of a given structure under uncertain loading conditions. The mathematical models of the uncertain input load x and the structure response $f(x)$ are random vector $x \in \mathbb{R}^N$ with joint probability density function (PDF) π_0 and function $f : \mathbb{R}^N \to \mathbb{R}_+$ correspondingly. For example, if our structure is a tall building, the stochastic input may represent wind velocities along the building height and the response may represent the maximum roof displacement or the maximum interstory drift (absolute value) under the given wind load.

Define the failure domain $F \subset \mathbb{R}^N$ as the set of inputs that lead to the exceedance of some prescribed critical threshold $b \in \mathbb{R}_+$:

$$F = \{x \in \mathbb{R}^N | f(x) > b\} \tag{1.1}$$

In the above example the critical threshold b represents the maximum permissible roof displacement or maximum permissible interstory drift and the failure domain F represents the set of all wind loads that lead to the collapse or to some other damage of the tall building.

So, the structural reliability problem is to compute the probability of failure, that is given by the following expression:

$$p_F = P(x \in F) = \int\limits_F \pi_0(x)dx = \int\limits_{\mathbb{R}^N} I_F(x)\pi_0(x)dx = E_{\pi_0}[I_F] \tag{1.2}$$

where I_F is the indicator function ($= 1$ if $x \in F$, $= 0$ otherwise) and E_{π_0} denotes expectation with respect to the distribution π_0.

Throughout this work we assume we are dealing with probability integrals (1.2) under the following context:

i. The computation of probability integral (1.2) is extremely challenging for real-world structures and can be done only in approximate ways. A well established methodology (see, for example, [10]) consists of introducing a one-to-one transformation Υ between the physical space of variables x and the standard Gaussian space of variables y and then computing the probability of failure as $p_F = \int_{\Upsilon(F)} \mathcal{N}(y)dy$, where $\mathcal{N}$ denotes the standard Gaussian joint PDF and $\Upsilon(F)$ is the image of the failure domain in the standard Gaussian space, $\Upsilon(F) = \{y \in \mathbb{R}^N | f(\Omega^{-1}(y)) > b\}$. Abusing the notation, *we shall assume that the PDF π_0 is the N-dimensional standard Gaussian distribution*, in particular we can evaluate $\pi_0(x)$ for any given x and we can generate random samples from π_0 efficiently.

ii. *The relationship between x and $I_F(x)$ is not explicitly known.* Although for any x we can check whether it is a failure point or not, i.e. calculate the value $I_F(x)$ for a given x, we cannot obtain other information (such as explicit formula, gradient, and so on).

iii. *The computational effort for evaluating $I_F(x)$ for each value of x is assumed to be significant so that it is essential to minimize the number of such function evaluations.* In the context of the tall building example the last two statements mean the following. Since the structure is too complex, we cannot predict whether a given wind load will cause a damage or not. The only one thing we can do is to check the latest by performing an experiment (for instance, a wind tunnel test). However, such an experiment is very expensive, so we want to reduce the number of experiments.

iv. *The probability of failure p_F is assumed to be very small.* In other words, the structure is assumed to be design properly, so that the failure is a very rare event. In our examples we shall consider $p_F \sim 10^{-3} - 10^{-6}$.

v. *The parameter space $\mathbb{R}^N$ is assumed to be high-dimensional.* As we have already mentioned before, for real-world structures the reliability problem has no exact analytical solution and one has to use a computer in order to find an approximate value of the failure probability. Any continuous problem must be discretized before it can be treated computationally, that is why the input load x is modeled as a random vector in $\mathbb{R}^N$. The bigger N the more precisely this discrete model describes the continuous input. So, the assumption of high dimensionality of parameter space comes from the aspiration to use a good discrete model of a real continuous problem. Also computational algorithms that are applicable in high dimensions are generally more robust. In our examples we shall consider $N \sim 10^3$.

He probably said to himself,
"Must stop or I shall be getting silly."
That is why there are only ten commandments.

Mrs. Patrick Campbell on Moses

The first idea that comes to mind is to use straightforward numerical integration for estimation probability integral (1.2), where the integral is approximated by a linear combination of values of the integrand. It is well known that in this method the number of function evaluations needed for a certain degree of accuracy increases very fast as the dimension of integration increases (for example, see [21]). So, due to the above assumptions (iii) and (v), numerical integration cannot be efficiently used for solving reliability problems.

Among all procedures developed for estimation of p_F, a prominent position is held by stochastic simulation methods. The expression of p_F as a mathematical expectation (1.2) renders standard Monte Carlo method [26] directly applicable, where p_F is estimated as a sample average of I_F over independent and identically distributed samples of x drawn from the PDF π_0 :

$$\hat{p}_{mc} = \frac{1}{n}\sum_{k=1}^{n} I_F(x^{(k)}), \qquad x^{(k)} \sim \pi_0 \tag{1.3}$$

This estimate is unbiased and the coefficient of variation (CV), serving as a measure of the statistical error, is

$$\delta_{mc} = \sqrt{\frac{(1-p_F)}{np_F}} \tag{1.4}$$

Although standard Monte Carlo is independent of the dimension N of the parameter space, it is inefficient in estimating small probabilities because it requires a large number of samples ($\sim 1/p_F$) to achieve an acceptable level of accuracy. For example, if $p_F = 10^{-4}$ and we want to achieve an accuracy of $\delta_{mc} = 10\%$ we need approximately 10^6 samples. Therefore, due to the earlier assumptions (iii) and (iv), standard Monte Carlo becomes computationally prohibitive for our problems of interest involving small failure probabilities.

The main goal of present research is to investigate well-known and to develop new advanced stochastic simulation methods for solving reliability problems under the real-life conditions (i)-(v).

This Thesis is organized as follows. In the Chapter 2 the widely used stochastic simulation algorithms are discussed from the geometric point of view. We highlight the difficulties associated with these methods when dealing with high-dimensional problems. Next, in Chapters 3 and 4 we introduce useful modifications of the well-known reliability methods: Adaptive Linked Importance Sampling (Chapter 3) and Modified Metropolis-Hastings algorithm with Delayed Rejection (Chapter 4). Finally, in Chapter 5 we propose a completely novel advanced stochastic simulation algorithm, called Horseracing Simulation, and demonstrate its accuracy and efficiency with real-life example.

Chapter 2

A Geometric Perspective

All my physics is nothing but geometry.

Rene Descartes

In this chapter we adopt a geometric perspective to highlight the challenges associated with solving high-dimensional reliability problems. Adopting a geometric point of view we highlight and explain a range of results concerning the performance of several well-known reliability methods. This chapter is based on the papers [17, 18].

Taking into account the term (i), let us start the discussion with the investigation of geometric properties of the N-dimensional standard Gaussian space and the distribution of samples in such a space.

2.1 Geometry of high-dimensional Gaussian space

Let $x = (x_1, \ldots, x_N)$ be a random vector in $\mathbb{R}^N$ where each of its components follows the standard Gaussian distribution:

$$x_i \sim \mathcal{N}(0,1), \;\; i = 1, \ldots, N. \tag{2.1}$$

By definition the square of the Euclidean norm of x is distributed according to the chi-square distribution with N degrees of freedom:

$$R^2 = \sum_{i=1}^{N} x_i^2 \sim \chi_N^2. \tag{2.2}$$

As N tends to infinity, the distribution of R^2 tends to normality (by Central Limit Theorem). In particular, it becomes more and more symmetric. However, the tendency is slow: the

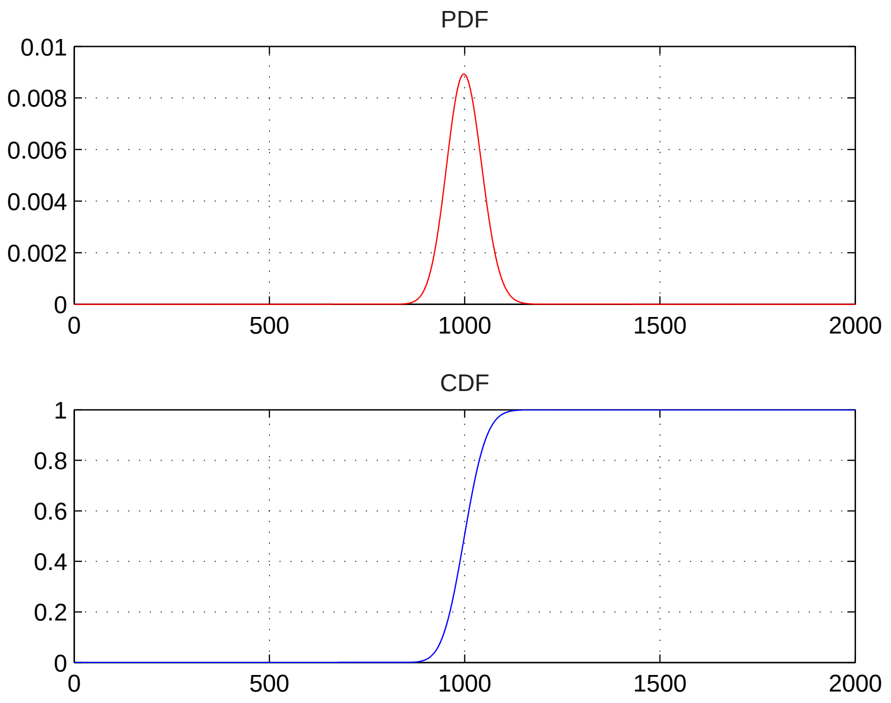


Figure 2.1: PDF and CDF of chi-square distribution with $N = 10^3$ degrees of freedom.

skewness is $\sqrt{8/N}$. The probability density function (PDF) and cumulative distribution function (CDF) of R^2 for $N = 10^3$ are plotted in Fig. 2.1. It can be shown (see, for example, [4], [11]) that $\sqrt{2R^2}$ is approximately normally distributed with mean $\sqrt{2N-1}$ and unit variance. Hence the norm of the random vector $\|x\| = R$ is also approximately a Gaussian random variable:

$$R \overset{\text{app}}{\sim} \mathcal{N}\left(\sqrt{N-1/2}, 1/2\right) \approx \mathcal{N}\left(\sqrt{N}, 1/2\right), \tag{2.3}$$

when $N \to \infty$. This means that the huge part of probability mass in the N-dimensional standard Gaussian space belongs in a spherical ring, so called Important Ring,

$$\sqrt{N} - r < R < \sqrt{N} + r, \tag{2.4}$$

where r depends on the amount of probability mass that we want to contain inside the Important Ring. For example, if $N = 10^3$ and $r = 3.46$ the probability of the corresponding Important Ring $28.16 < R < 35.08$ is more than $1 - 10^{-6}$. *Thus, any sample $x \in \mathbb{R}^N$ distributed according to the high-dimensional standard Gaussian distribution will lie with extremely large probability in the Important Ring.*

Now let us fix one particular direction (here direction means a ray passing through zero), say $e = (1, 0, \ldots, 0)$, and explore the distribution of the angle $\alpha = \widehat{xe}$ between this direction and a random vector x. By definition

$$f_\alpha(\alpha_0)d\alpha = P(\alpha_0 \leq \alpha \leq \alpha_0 + d\alpha) = P(\alpha_0 \leq \widehat{xe} \leq \alpha_0 + d\alpha), \tag{2.5}$$

where f_α is the PDF of α. Since the Gaussian space is isotropic (there are no preferable directions) and any point that lies along a particular ray forms the same angle with e, we can simplify the problem by considering the $(N-1)$-dimensional sphere of unit radius $\mathbb{S}_1^{N-1}$. Note that herein a sphere is defined, as usually in geometry, as the set of all points located at distance equal to R from a given fixed point corresponding to the center of the sphere. Thus, in a three dimensional space, according to our definition, a sphere is a two-dimensional surface, while the interior of the sphere, comprised of all points at distance smaller than R from the center, is a three-dimensional manifold. Clearly, on the sphere $\mathbb{S}_1^{N-1}$ all points are uniformly distributed. Therefore, $f_\alpha(\alpha_0)$ is proportional to the geometric volume of part of this sphere:

$$f_\alpha(\alpha_0) \sim Vol(\Omega_{\alpha_0}), \tag{2.6}$$

$$\Omega_{\alpha_0} = \{x \in \mathbb{S}_1^{N-1} : \widehat{xe} = \alpha_0\}. \tag{2.7}$$

If $\langle \cdot, \cdot \rangle$ denotes the standard scalar product in $\mathbb{R}^N$ then $\langle x, e \rangle = x_1$. On the other hand $\langle x, e \rangle = \|x\| \|e\| \cos \widehat{xe}$. Therefore

$$\Omega_{\alpha_0} = \{x \in \mathbb{S}_1^{N-1} : x_1 = \cos \alpha_0\}. \tag{2.8}$$

One can rewrite (2.8) as the intersection of the hypersphere $\mathbb{S}_1^{N-1}$ with the hyperplane

$$\pi_{\alpha_0}^{N-1} = \{x \in \mathbb{R}^N : x_1 = \cos \alpha_0\}. \tag{2.9}$$

Thus,

$$\Omega_{\alpha_0} = \mathbb{S}_1^{N-1} \cap \pi_{\alpha_0}^{N-1}. \tag{2.10}$$

This intersection can be easily evaluated.

$$\begin{cases} x_1^2 + \ldots + x_N^2 &= 1, \\ x_1 &= \cos \alpha_0. \end{cases} \quad \Leftrightarrow \tag{2.11}$$

$$x_2^2 + \ldots + x_N^2 = \sin^2 \alpha_0. \tag{2.12}$$

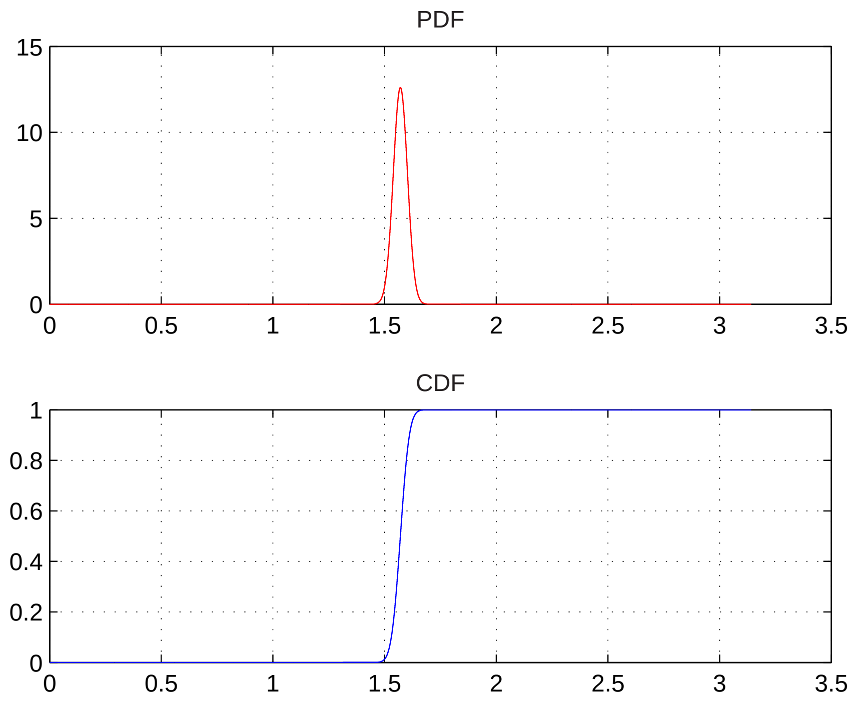


Figure 2.2: PDF and CDF of angle α.

Thus, the region in which we are interested is a $(N-2)$-dimensional sphere of radius $\sin\alpha_0$:

$$\Omega_{\alpha_0} = \mathbb{S}^{N-2}_{\sin\alpha_0}. \tag{2.13}$$

It is well known that the volume of a k-dimensional sphere is proportional to the radius in the power k. So we have:

$$f_\alpha(\alpha_0) \sim Vol(\Omega_{\alpha_0}) \sim \sin^{N-2}\alpha_0. \tag{2.14}$$

Finally, we can obtain that the PDF and the CDF of α (Fig. 2.2) are correspondingly equal to

$$f_\alpha(\alpha) = \frac{\sin^{N-2}\alpha}{\int_0^\pi \sin^{N-2}\alpha d\alpha}, \tag{2.15}$$

$$F_\alpha(\alpha) = \frac{\int_0^\alpha \sin^{N-2}\alpha d\alpha}{\int_0^\pi \sin^{N-2}\alpha d\alpha}. \tag{2.16}$$

From this result and by plotting these distributions for large N it follows that *if we fix a particular direction e then a sample $x \in \mathbb{R}^N$ that is distributed according to the high-dimensional standard Gaussian distribution will be with high probability almost perpendicular to e.* Although we proved this result for the specific direction $e = (1, 0, \dots, 0)$ this also holds for arbitrary e since the Gaussian space is isotropic.

This also can be argued in a more intuitive way. If x is such a sample then

$$\cot^2 \alpha = \frac{x_1^2}{x_2^2 + \ldots + x_N^2}. \tag{2.17}$$

Since x_i are independent and identically distributed random variables we have that expectation:

$$E[\cot^2 \alpha] = \frac{1}{N-1} \to 0 \ \text{ as } \ N \to \infty. \tag{2.18}$$

Therefore,

$$\alpha \to \pi/2 \ \text{ as } \ N \to \infty. \tag{2.19}$$

2.2 Importance Sampling

Importance Sampling is a fundamental technique in stochastic simulation that tries to reduce the CV of the Monte Carlo estimate. In statistical physics literature this procedure is also called "simple importance sampling" and "free energy perturbation". The basic idea of Importance Sampling is to generate more samples in the important region of the failure domain, i.e., in the region of the failure domain that contains most of the probability mass and, therefore, contributes mostly to the integral (1.2). Roughly speaking standard Monte Carlo does not work because the vast majority of terms in the sum (1.3) are zero and only very few are equal to one. Using Importance Sampling we want instead of estimating p_F as the average of a vast majority of 0's and, occasionally some (if any) 1's, to calculate it as the average of less zeros and many more nonzero small numbers, each being ideally of the order of p_F.

Specifically, let π_{is} be any PDF on the parameter space $\mathbb{R}^N$ such that its support (domain where π_{is} is not zero) contains the intersection of failure domain with the support of π_0:

$$\text{supp } \pi_{is} \supset F \cap \text{supp } \pi_0. \tag{2.20}$$

Then we can rewrite the probability integral (1.2) as follows:

$$p_F = \int\limits_{\mathbb{R}^N} I_F(x)\pi_0(x)dx = \int\limits_{\mathbb{R}^N} I_F(x)\frac{\pi_0(x)}{\pi_{is}(x)}\pi_{is}(x)dx = E_{\pi_{is}}\left[I_F \frac{\pi_0}{\pi_{is}}\right]. \tag{2.21}$$

Suppose that we are able to generate random samples from π_{is}, called importance sampling density (ISD), and compute the value of $\pi_{is}(x)$ easily for any given x. Then similarly to (1.3) we have:

$$\hat{p}_{is} = \frac{1}{n}\sum_{k=1}^{n} I_F(x^{(k)})\frac{\pi_0(x^{(k)})}{\pi_{is}(x^{(k)})}, \qquad x^{(k)} \sim \pi_{is}. \tag{2.22}$$

Note that the standard Monte Carlo method is a special case of Importance Sampling when $\pi_{is} = \pi_0$. The estimate $\hat{p}_{is}$ in (2.22) has the same statistical properties as $\hat{p}_{mc}$ in (1.3), i.e., it is unbiased and it converges to p_F with probability 1. Choosing π_{is} in some appropriate way we hope to be able to reduce the CV.

The most important task in applying Importance Sampling is the construction of the ISD π_{is}. If it is "good" then we can get great improvement in efficiency.

What does it mean: "good" ISD ? Let us consider an example. Suppose we know that a certain vector ξ belongs to the failure domain $\xi \in F$. It is natural to assume that in the neighborhood of ξ there are more points from F. Thus, we can consider π_{is} to be a Gaussian PDF centered at ξ:

$$\pi_{is}(x|\xi) = \mathcal{N}_{\xi,1}(x) = \frac{1}{(\sqrt{2\pi})^N}\exp\left(-\frac{\|x-\xi\|^2}{2}\right). \tag{2.23}$$

As discussed in the previous section, the main part of probability mass is concentrated inside the Important Ring. This means that we can restrict the sample space and consider only the part of failure domain that belongs to the Important Ring, since the contribution of the remaining part of the failure domain to the probability p_F is comparatively very small. So we can consider that ξ belongs in the Important Ring. Again, from the previous section we know that the angle between the vectors ξ and $(y-\xi)$, where y is a random vector drawn from π_{is}, is approximately equal to $\pi/2$. Moreover y will lie in the Important Ring corresponding to π_{is}. This is schematically shown in Fig. 2.3.

It follows that in high dimensions we will have:

$$\|\xi\| \approx \sqrt{N}, \;\; \|y-\xi\| \approx \sqrt{N}. \tag{2.24}$$

Therefore

$$\|y\| \approx \sqrt{2N}. \tag{2.25}$$

Now the estimator (2.22) gives:

$$\hat{p}_{is} = \frac{1}{n}\sum_{k=1}^{n} I_F(y^{(k)})\frac{\pi_0(y^{(k)})}{\pi_{is}(y^{(k)}|\xi)}. \tag{2.26}$$

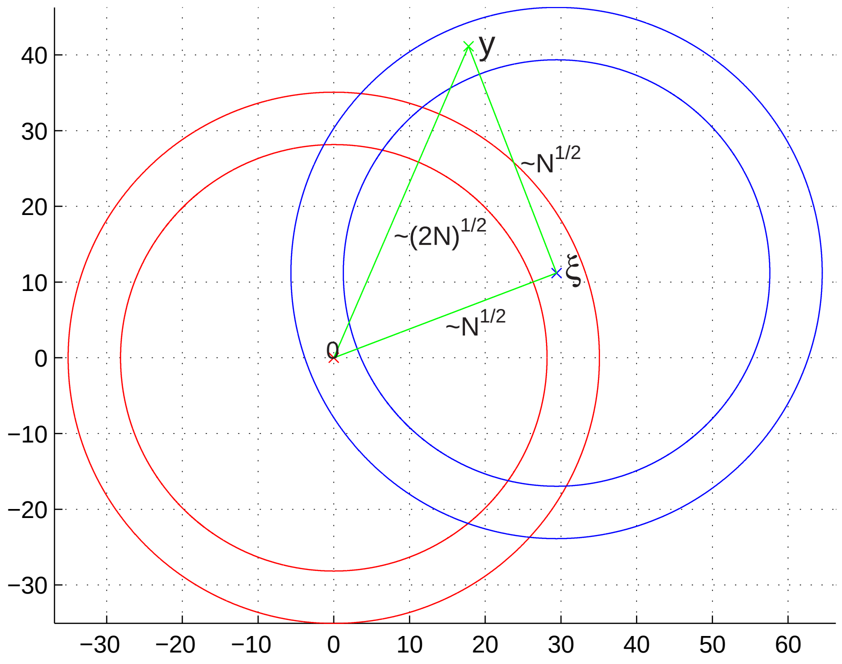


Figure 2.3: Vectors $\xi, y \in \mathbb{R}^N$ are drawn from the standard Gaussian and Gaussian centered at ξ distributions, respectively, $N = 1000$.

Since each $y^{(k)}$ is drawn from the Gaussian distribution centered at ξ and, therefore, satisfies (2.25), it follows from (2.24), (2.25) that the ratio

$$\frac{\pi_0(y^{(k)})}{\pi_{is}(y^{(k)}|\xi)} \approx e^{-N/2}, \tag{2.27}$$

which is extremely small in high dimensions. *Thus, Importance Sampling leads to underestimation of* p_F. Here we provided a geometrical explanation as to why this happens; specifically, we showed that the simulation of samples distributed according to the chosen importance sampling density is very unlikely to yield samples that lie within the Important Ring centered at zero. Note that the important region of any failure domain should be a subset of this important ring, as the probability volume outside the important ring can be considered to be negligible. Therefore, we can conclude that the chosen importance sampling density fails to generate samples in the important region of the failure domain rendering Importance Sampling inapplicable, severely underestimating the true failure probability.

The above results are next confirmed using simulations. A linear failure domain with reliability index $\beta = 3$ is considered and Importance sampling is applied using a Gaussian ISD with unit variance centered at one of the failure points, i.e., π_{is} is given by (2.23).

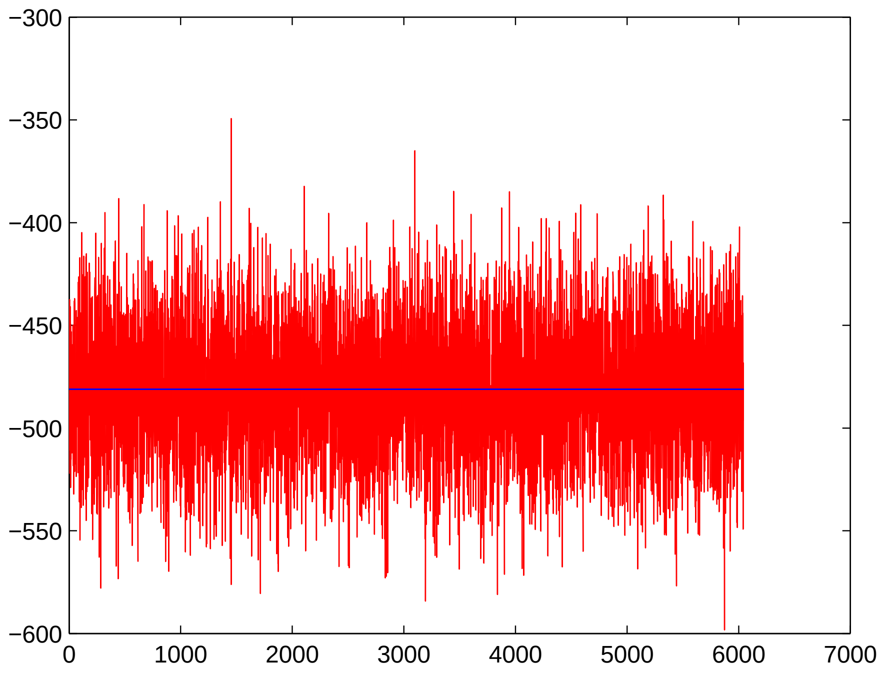


Figure 2.4: The ratio $\frac{\pi_0(y^{(k)})}{\pi_{is}(y^{(k)}|\xi)}$ plotted in log scale against the sample number, for $n = 10^4$ samples and for $N = 1000$.

Fig. 2.4 shows for a single run the ratio $\frac{\pi_0(y^{(k)})}{\pi_{is}(y^{(k)}|\xi)}$ plotted in log scale against the sample number, for $n = 10^4$ samples and for $N = 1000$. Note that this figure shows only the ratio for those samples corresponding to failure points, in this case for 6042 points. The horizontal line shows the mean value of the log of these values which is calculate to be equal -481. This is consistent with the rough estimate -500 given by (2.27). Fig. 2.5 shows the failure probability estimate p_{is} in log scale, as a function of the dimension N. Here, $n = 10^4$ samples were used for each run; the plotted value of p_{is} corresponds to the average value of 50 runs. Confirming the earlier discussion, the failure probability is found to be severely underestimated, the underestimation becoming worse as the dimension N increases. Note that p_{is} is found to be of order not quite as low as $\exp(-N/2)$. The reason is that the ratios $\frac{\pi_0(y^{(k)})}{\pi_{is}(y^{(k)}|\xi)}$ are, when plotted in linear scale, varying tremendously, so that the order of p_{is} as calculated by (2.26) is governed by the largest of these terms. For example, in the case of the run corresponding to Fig. 2.4 the maximum term was of the order of $\exp(-350)$. This explains why the dependency on N is not quite as bad as $\exp(-N/2)$.

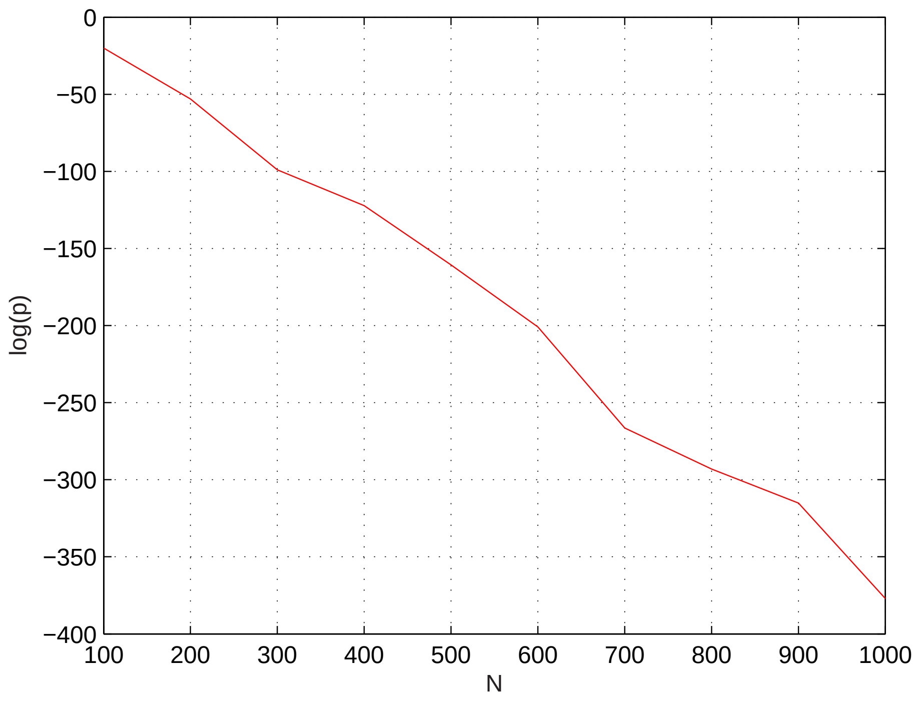


Figure 2.5: Failure probability estimate $\hat{p}_{is}$ plotted in log scale, as a function of the dimension N.

2.3 Design points and nonlinear problems

The next question we discuss is the significance of design points when dealing with strongly nonlinear problems.

First, returning to the definition of the failure domain (1.1), define the limit-state function (LSF):

$$G(x) = b - f(x), \tag{2.28}$$

so that failure domain F is defined as the subset of $\mathbb{R}^N$ where G is negative. Basically, the calculation of the probability of failure p_F in (1.2) is the evaluation of the total probability volume corresponding to the failure domain $F \subset \mathbb{R}^N$ defined by $G(x) < 0$.

The design point is defined as the point x^* on the limit-state surface $\{x : G(x) = 0\}$ that is nearest to the origin when the random variables are assumed to have been transformed to the standard Gaussian space (see (i)). Due to the rotational symmetry of the standard Gaussian distribution, the design point is the most likely single realization of the random variables that gives rise to the failure event, i.e., it is the failure point with the largest probability density. The norm of the design point, i.e., its distance from the origin, is referred to as the reliability index β. Note that in the case of high-dimensional reliability problems it is always true that $\beta << R = \sqrt{N}$. This leads to the important, although often misunderstood, point that *the design point does not belong in the important ring and, therefore, according to our earlier discussions, it is practically impossible during simulations to obtain a sample in its neighborhood.* Thus, although the design point is the point with the maximum likelihood, a sample in its vicinity will practically never be realized through simulations. The reason for this is that the geometric volume describing the vicinity of the design point is extremely small in high dimensions, making the probability mass associated with the neighborhood of the design point negligible. In other words, the relatively smaller probability density corresponding to the failure points located in the important ring is overcompensated by the vast number of these failure points so that if one tries to simulate a failure realization it is almost certain that he will obtain a sample belonging in the important ring rather than in the vicinity of the design point.

It is considered to be that design points play very important role in solving reliability problems. Without doubt this is true in the linear case. Consider a linear reliability problem with LSF expressed in terms of the standard Gaussian vector x as follows:

$$G(x) = a^T x + b, \tag{2.29}$$

where $a \in \mathbb{R}^N$ and b are fixed coefficients. The design point x^* is then the point on the plane $G(x) = 0$ that is located closest to the origin and can be easily calculated in terms of a and b as follows:

$$x^* = -\frac{b}{\|a\|^2} a, \tag{2.30}$$

which reliability index is given by:

$$\beta = \|x^*\| = \frac{b}{\|a\|}. \tag{2.31}$$

It is well known that the failure probability corresponding to this linear failure domain is given in terms of β by the expression:

$$p_F = P(x : G(x) < 0) = 1 - \Phi(\beta), \tag{2.32}$$

where Φ denotes the CDF of the standard Gaussian variable. So, the failure probability in the linear case is completely defined by the design point. In the case where the failure domain is almost linear the first order reliability method (FORM) based on (2.32) for the given design point provides a good approximation of p_F.

In the case where the LSF can be approximated by a second order polynomial function, the design point still plays a very important role and is the basis of the second order reliability method (SORM). However in nonlinear case the significance of the design point(s) is not as clear and is in need of research.

In the next section we consider a nonlinear dynamic problem to show that the design point does not provide sufficient information to describe the complex geometry of the corresponding nonlinear failure domain.

2.3.1 Duffing oscillator subjected to white noise

This nonlinear elastic system is taken from [20]. Consider the Duffing oscillator defined by

$$m\ddot{z}(t) + c\dot{z}(t) + k[z(t) + \gamma z(t)^3] = f(t), \tag{2.33}$$

with $m = 1000$ kg, $c = 200\pi$ Ns/m, $k = 1000(2\pi)^2$ N/m, and $\gamma = 1$ m^{-2}, and assume the input is white-noise with intensity $S_0 = 10^6$ N^{2}s/rad. Then in the discrete form $f(t)$ is a vector of pulses $f = [f_1, \dots, f_N]^T = \sigma x$, where $\sigma = \sqrt{2\pi S_0/\Delta t}$ and x is a standard Gaussian random vector. As described in [8], most events of interest in random vibration analysis can be represented in terms of the instantaneous failure event

$$E_\tau = \{z(\tau) > z_0\}, \tag{2.34}$$

i.e., the event that the response at a specified time τ exceeds a specified threshold z_0. The corresponding failure domain is:

$$F_\tau = \{x : z(\tau) > z_0\}, \tag{2.35}$$

We consider the specific time instance $\tau = 12$s, three different thresholds $z_0 = K\sigma_0$, where $K = 3, 4, 5$ and $\sigma_0^2 = \pi S_0/ck$ is the stationary response variance for the linear case ($\gamma = 0$). We use $\Delta\tau = 0.01s$ so that the dimension $\dim x = N = \tau/\Delta\tau + 1 = 1201$.

If the threshold is too large, then the probability $p_F = P(F_\tau)$ is small and the intersection of F_τ with a random 2-dimensional plane is the empty set. Let x^* and x^*_L denote the design points for the nonlinear and for the linear ($\gamma = 0$) problems correspondingly. Then, the 2-dimensional plane $\mathcal{P}(x^*, x^*_L)$ formed by x^* and x^*_L contains failure points for sure. Note, that

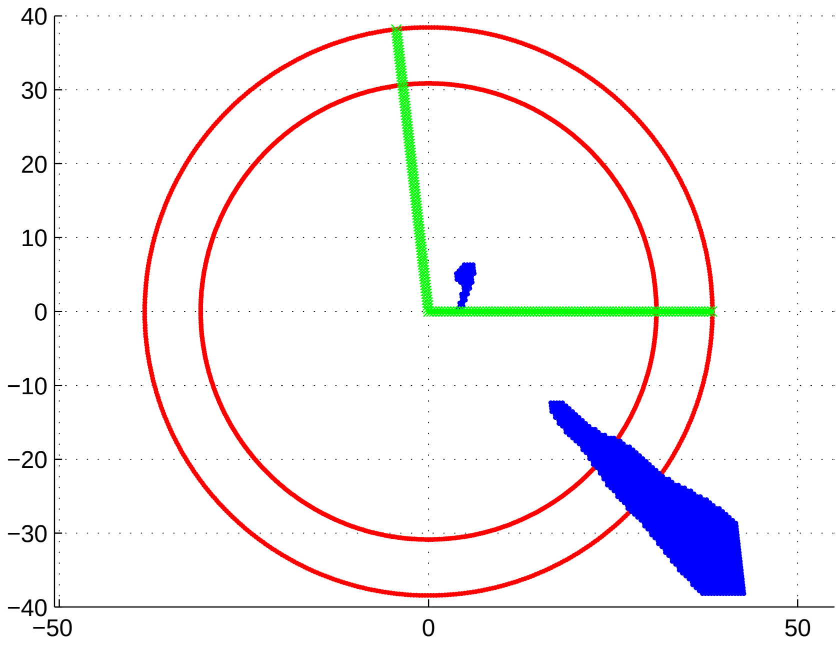


Figure 2.6: Intersection of failure domain F_τ with the 2-dimensional plane $\mathcal{P}(x^*, x_L^*)$ for $K = 3$.

x^* can be found using a mirror-image excitation, [20]. In Figures 2.6, 2.7, 2.8 the intersections $F_\tau \cap \mathcal{P}(x^*, x_L^*)$ for $K = 3, 4, 5$ are shown. The concentric circles are the intersection of the Important Ring with this plane and the rays shown are the directions of x^* (the ray in the horizontal direction) and x_L^*. The reliability index for this nonlinear problem is $\beta = 3.76$, with corresponding failure probability $1 - \Phi(\beta) = 8.5 \cdot 10^{-5}$. Note, that the true failure probability is equal to 0.0092 (Monte Carlo with 10^4 samples).

Let f_s^* denote an excitation along the design point direction $f^* = \sigma x^*$, i.e.

$$f_s^* = sf^* = s\sigma x^*, \tag{2.36}$$

where s is a scaling factor allowing to realize scaled versions of the design excitation. The corresponding response $z(\tau)$ at a specified time τ is shown in Fig. 2.9 plotted against the norm of f_s^*. When $s = 1$ we have an excitation that correspond to the design point itself and the response reaches the threshold z_0. Then there is a small period of exceedance of the threshold and after that the response begins to decrease so that around the Important Ring ($s \sim \sqrt{N} = 34.64$) we have a safe region.

These figures show that *the design point itself as well as the direction of the design point*

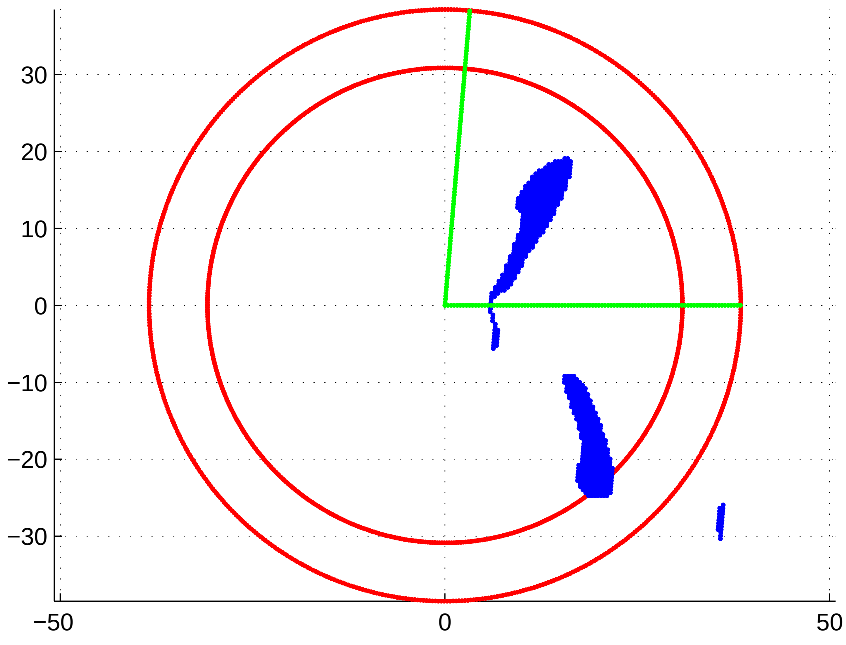


Figure 2.7: Intersection of failure domain F_τ with the 2-dimensional plane $\mathcal{P}(x^*, x_L^*)$ for $K = 4$.

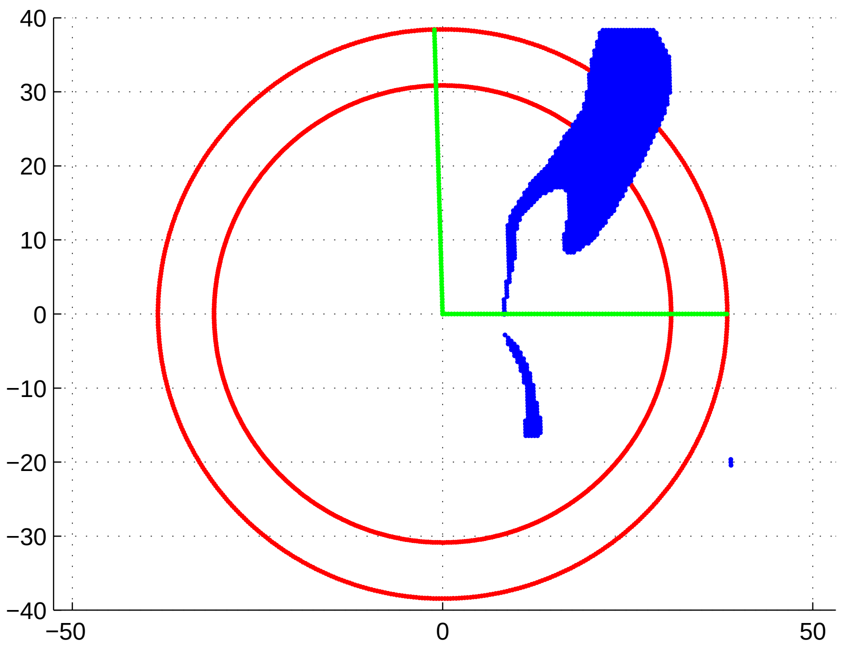


Figure 2.8: Intersection of failure domain F_τ with the 2-dimensional plane $\mathcal{P}(x^*, x_L^*)$ for $K = 5$.

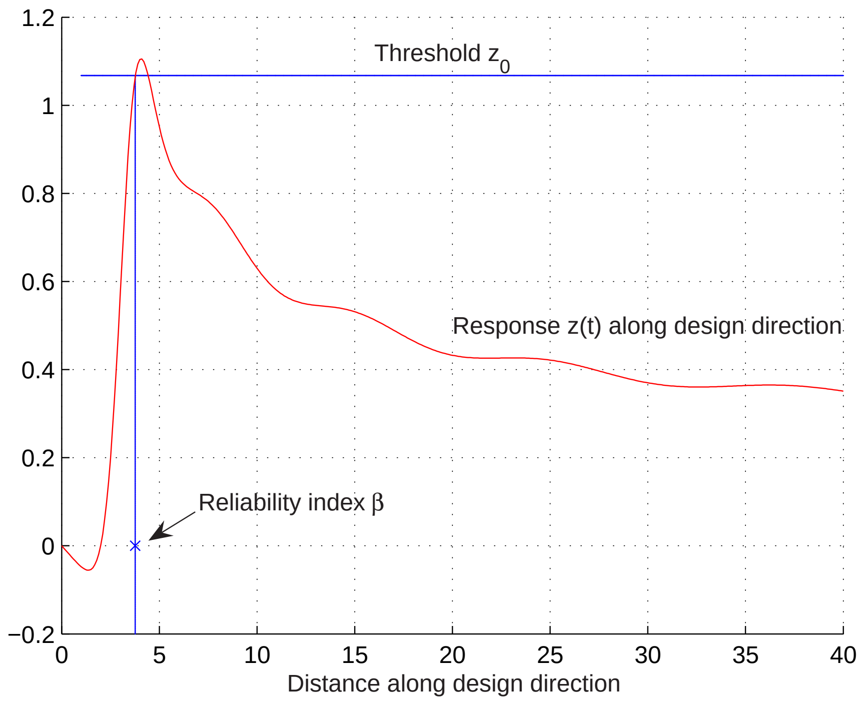


Figure 2.9: Response of Duffing oscillator along design point direction.

can be of no consequence when searching the main parts of the failure domain (intersections of failure domain with the Important Ring).

2.3.2 Parabolic failure domain

The design points are usually used in conjunction with Importance Sampling for calculating the failure probability. For the first passage problem the failure domain F corresponding to a time duration T can be represent as a union of elementary failure regions F_i, defined similarly to (2.35) as exceedance at time $\tau_i = i\Delta\tau$:

$$F_i = \{x : z(\tau_i) > z_0\}, \ \ i = 0, \dots, N = \frac{T}{\Delta\tau}. \tag{2.37}$$

When the design points x_i^* are known, the ISD can be constructed as a weighted sum of Gaussian PDFs centered at the design points, i.e.:

$$\pi_{is}(x) = \sum_{i=1}^{N} w_i \mathcal{N}_{x_i^*,1}(x) = \sum_{i=1}^{N} w_i \mathcal{N}_{0,1}(x - x_i^*). \tag{2.38}$$

Our goal is to demonstrate that when the failure domain is strongly nonlinear Importance Sampling with ISD (2.38) can be feeble.

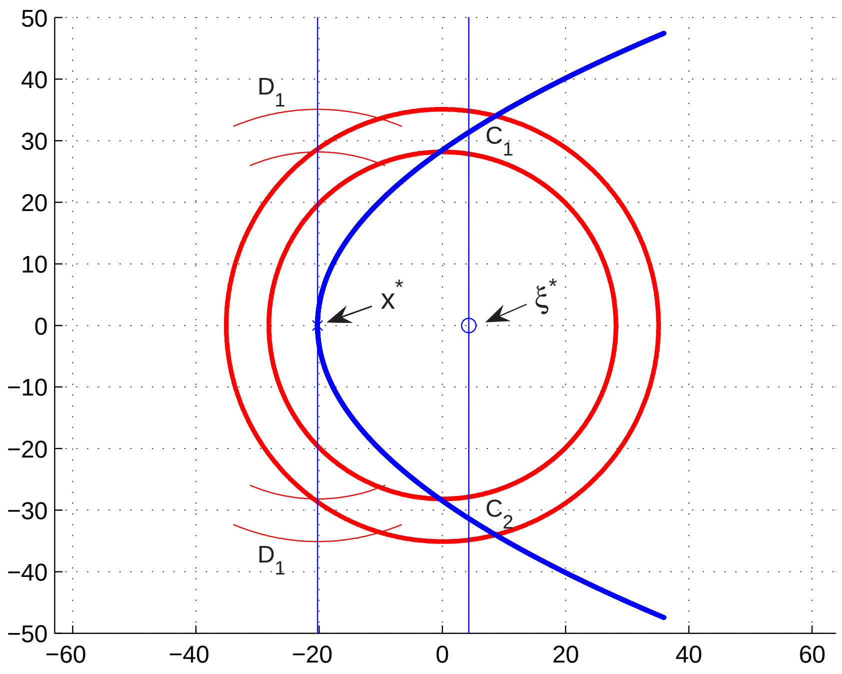


Figure 2.10: Failure domain of parabolic shape.

We consider a paraboloid in N-dimensional space defined as follows:

$$Par: \quad x_1 = a\sum_{i=2}^{N} x_i^2 - b, \tag{2.39}$$

and define failure domain as the interior of this paraboloid:

$$F = \left\{ x \in \mathbb{R}^N : x_1 > a\sum_{i=2}^{N} x_i^2 - b \right\}. \tag{2.40}$$

The intersection of this high-dimensional failure domain with an arbitrary plane containing the x_1 direction is shown in Fig. 2.10. In this example $a = 0.025$, $b = 20.27$ and $N = 1000$. The probability of this parabolic failure domain calculated using standard MC simulation (10^4 samples and 100 runs) is equal to $p_F = 0.00074$ with CV $\delta = 0.41$.

However if we will use Importance Sampling method with Gaussian ISD centered at the design point $x^* = (-b, 0, \dots, 0)$ we will get underestimation. For 10^4 samples and 20 runs we got $p_F^{is} = 0$. This happens because, as was explained in section 2.2, all samples generated from the Gaussian density centered at x^* will lie in the Important Ring centered at x^* and will be almost perpendicular to the fixed x_1-direction. In the 2-dimensional picture of Fig.2.10 these regions are denoted as D_1 and D_2 which clearly do not correspond to failure regions.

Thus, all samples generated by such an ISD yield zero indicator function and, therefore, the resulting Importance Sampling estimate of the failure probability turns out to be zero.

Suppose now that we know the shape of the failure domain. Then from the previous discussion a good choice of ISD would be a Gaussian PDF $\mathcal{N}(\xi^*, \sigma^2)$ centered at ξ^*, where ξ^* is a point along the x_1-direction, such that

$$Par \cap \mathcal{P}^{\perp}_{\xi^*} = Par \cap \mathbb{S}^{N-1}_{\sqrt{N}}, \tag{2.41}$$

where $\mathcal{P}^{\perp}_{\xi^*}$ is a hyperplane passing through ξ^* and normal to the x_1 direction, $\mathbb{S}^{N-1}_{\sqrt{N}}$ is a hypersphere of radius $\sqrt{N}$ (middle hypersphere in the Important Ring). In Fig. 2.10 regions C_1 and C_2 belong to the intersection described by (2.41). The value of ξ^* can be easily calculated as:

$$\xi^* = \frac{\sqrt{4a^2N - 4ab + 1} - 1}{2a}. \tag{2.42}$$

The variance σ^2 of this Gaussian ISD has to be such that

$$E\|x - \xi^*\|^2 = \|\xi^* C_1\|^2, \ \ x \sim \mathcal{N}(\xi^*, \sigma^2). \tag{2.43}$$

Since

$$E\|x - \xi^*\|^2 = N\sigma^2, \ \ \|\xi^* C_1\|^2 = N - (\xi^*)^2, \tag{2.44}$$

we finally obtain:

$$\sigma^2 = \frac{N - (\xi^*)^2}{N}. \tag{2.45}$$

Now if we will use the described ISD then almost all samples will lie in the important region of the failure domain, that is, in the intersection of the failure domain and the important ring and as a result the estimate of the failure probability obtained will be unbiased and accurate. The result obtained from simulations is $p_F = 0.00076$ with CV $\delta = 0.83$ (10^4 samples and 100 runs), showing that the obtained failure probability estimate is unbiased. However, the coefficient of variation is larger than that obtained from Monte Carlo.

Note that finding ξ^* assumes knowledge not only of the axis of the paraboloid but also of its curvature. Furthermore, in non-toy problems the geometry of the important region of the failure domain is by far not as simple as the $N-2$ dimensional sphere described by (2.41). As has become evident by now, in order for Importance Sampling to be successful its ISD must be chosen such that its corresponding Important Ring contains the important region of the failure domain; the latter is a subset of the Important Ring corresponding to the standard

Gaussian distribution. Thus, the question of finding an appropriate ISD is analogous to finding a sphere (corresponding to the ISD), such that its intersection with a fixed sphere (corresponding to the standard Gaussian density) is a given set of fixed, generally complex, geometry. Clearly selecting such a sphere is impossible, unless the specified intersection is a sphere in itself (of one dimension lower) as in the paraboloid example considered earlier.

The above point is made again more formally as follows. Let F denote the failure domain and S the Important Ring corresponding to the standard Gaussian. Then, the Important Region of the failure domain is $\widetilde{F} = F \cap S$. Based on the earlier discussions: $p_{\widetilde{F}} = p_F$. Let now S_{is} denote the Important Ring corresponding to the ISD. A necessary condition for Importance Sampling to work is that it must satisfy $\widetilde{F} \subset S_{is}$. Therefore, since $\widetilde{F} \subset S$ also the following must hold: $\widetilde{F} \subset (S_{is} \cap S)$. The intersection of two N-dimensional spherical rings has very specific geometry. In particular, if we idealize the spherical rings by spheres, this intersection is a sphere itself of one dimension less. In the general case where $\widetilde{F}$ is of arbitrary geometry, the necessary condition $\widetilde{F} \subset (S_{is} \cap S)$ can therefore, not be satisfied, unless $S \equiv S_{is}$ which reduces to standard Monte Carlo.

Therefore, we conclude that *Importance Sampling is not applicable in general nonlinear high-dimensional reliability problems of practical interest.*

2.4 Subset Simulation

In the previous section we saw that getting information about the failure domain is quite difficult. Another approach for evaluating failure probability is Subset Simulation introduced by Au and Beck in [1].

The main idea of this method is as follows. Given the original failure domain F let $\mathbb{R}^N = F_0 \supset F_1 \supset \ldots \supset F_n = F$ be a filtration, in other words a sequence of failure events so that $F_k = \cap_{i=0}^{k} F_i$. Using the definition of conditional probability it can be shown that,

$$p_F = P(F_1) \prod_{i=1}^{n-1} P(F_{i+1}|F_i). \tag{2.46}$$

The main observation is that, even if p_F is small, by choosing n and $F_i, i = 1, \ldots, n-1$ appropriately, the conditional probabilities can be made large enough for efficient evaluation by simulation.

As it was discussed in introduction, in engineering applications the failure event usually can be expressed in terms of exceedance of some demand-capacity ratio and, therefore, the probability of failure can be written in the form $p_F = P(f(x) > b)$, where f is the response function and b is a critical threshold. The sequence of intermediate failure events $\{F_1, \ldots, F_{n-1}\}$ can then be chosen as $F_i = \{f(x) > b_i\}$ for some intermediate thresholds $b_1 < \ldots < b_n = b$. During Subset Simulation, $b_1, \ldots, b_{n-1}$ are adaptively chosen such that all probabilities $P(F_1), P(F_2|F_1), \ldots, P(F_n|F_{n-1})$ are equal to, say, $p_0 = 0.1$.

Let us briefly recall how Subset Simulation works. We start by simulating n samples $\{x_0^{(k)}, k = 1, \ldots, n\}$ from π_0 by standard Monte Carlo simulation. Perform n system analysis to obtain the corresponding response values $\{f(x_0^{(k)}), k = 1, \ldots, n\}$. Then, the first intermediate threshold b_1 is adaptively chosen as the $((1-p_0)n-1)$-th value in the ascending list of response values, so that the sample estimate for $P(F_1) = P(f(x) > b_1)$ is equal to p_0. There are np_0 samples among $\{x_0^{(k)}, k = 1, \ldots, n\}$ whose response f is greater than b_1, and hence lie in F_1. These samples are distributed as $\pi_0(\cdot|F_1)$ and provide "seeds" for simulating additional samples. Starting from each of these samples Markov chain Monte Carlo[1] simulation (MCMC) is used to obtain an additional $(1-p_0)n$ samples, making up a total of n conditional samples $\{x_1^{(k)}, k = 1, \ldots, n\}$ distributed according to $\pi_0(\cdot|F_1)$. The intermediate threshold b_2 is then adaptively chosen as the $((1-p_0)n-1)$-th value in the ascending list of $\{f(x_1^{(k)}), k = 1, \ldots, n\}$, and it defines the next intermediate failure event $F_2 = \{f(x) > b_2\}$. The sample estimate for $P(F_2|F_1)$ is automatically equal to p_0. Repeating this process, one can generate conditional samples for higher conditional levels until the target failure probability level has been reached.

Original Metropolis algorithm

The Metropolis algorithm belongs to the class of very powerful techniques, called Markov chain Monte Carlo simulations, for simulating samples according to an arbitrary distribution. In these methods samples are simulated as the states of a Markov chain which has the target distribution as its invariant distribution.

The significance of the Metropolis algorithm in Subset Simulation is that it allows to construct a Markov chain with $\pi_0(\cdot|F_i)$ as its stationary distribution. Therefore, we can

[1]A brief review of Markov chains theory and Markov chain Monte Carlo simulation is given in the Appendix.

use this algorithm for simulating new samples starting from "seeds" that were obtained in the previous step of Subset Simulation. Even if the current sample is not distributed as $\pi_0(\cdot|F_i)$, the limiting distribution property of Markov chain guarantees that the distribution of simulated samples will tend to $\pi_0(\cdot|F_i)$ as the number of Markov steps increases.

Given a current sample $x^{(1)}$ (a "seed" point) the original Metropolis algorithm works as follows. Let $S(\cdot|x)$, called proposal PDF, be a N-dimensional PDF centered at x with symmetry property $S(\xi|x) = S(x|\xi)$. Generate a sequence of samples $\{x^{(1)}, x^{(2)}, \ldots\}$ starting from a given sample $x^{(1)}$ by computing $x^{(k+1)}$ from $x^{(k)}$ as follows:

1) *Generate candidate state $\tilde{x}$.*
   Simulate ξ according to $S(\cdot|x^{(k)})$, compute the acceptance ratio $a(x^{(k)}, \xi) = \frac{\pi_0(\xi)}{\pi_0(x^{(k)})}$, set $\tilde{x} = \xi$ with probability $\min\{1, a(x^{(k)}, \xi)\}$ and set $\tilde{x} = x^{(k)}$ with the remaining probability.

2) *Accept/Reject $\tilde{x}$.*
   If $\tilde{x} \in F_i$ accept it as a next step, i.e. $x^{(k+1)} = \tilde{x}$; otherwise reject it and take the current sample as a next state of the Markov chain, i.e., $x^{(k+1)} = x^{(k)}$.

One can show that such updates leave $\pi_0(\cdot|F_i)$ invariant, and hence the chain will eventually converge to $\pi_0(\cdot|F_i)$ as its equilibrium distribution.

Au and Beck in [1] realized that *the original Metropolis algorithm does not work in high dimensions.* The geometric reason of this inapplicability is that the same effect as that explained by Fig. 2.3 arises, where the symbols ξ and y in this Figure now represent $x^{(k)}$ and ξ, respectively. Therefore, for the reasons explained previously, in each step of the original Metropolis algorithm the ratio $a(x^{(k)}, \xi) = \pi_0(\xi)/\pi_0(x^{(k)})$ will be extremely small, broadly speaking of the order of $\exp(-N/2)$. Therefore, with extremely high probability one obtains repeated samples. Thus, a chain of practically meaningful length may consist of as few as a single sample. This renders Subset Simulation practically inapplicable.

Modified Metropolis algorithm

The modified Metropolis algorithm (Au and Beck, [1]) differs from the original Metropolis algorithm in the way the candidate state $\tilde{x}$ is generated. Rather than using a N-dimensional proposal to directly obtain the candidate state $\tilde{x}$, in the modified algorithm a sequence of one-dimensional proposals $S_j(\cdot|x_j^{(k)}), j = 1, \ldots, N$ is used. Specifically, each component of $\tilde{x}_j$

of the candidate state $\tilde{x}$ is generated separately using a one-dimensional proposal centered at $x_j^{(k)}$. Thus, being at state $x^{(k)}$ the candidate $\tilde{x}$ for the next state x^{k+1} of the Markov chain is generated as follows:

1$'$) For each component $j = 1, \ldots, N$:
Simulate ξ_j from $S_j(\cdot|x_j^{(k)})$, compute the acceptance ratio $a_j(x_j^{(k)}, \xi_j) = \frac{\pi_{0,j}(\xi_j)}{\pi_{0,j}(x_j^{(k)})}$, set $\tilde{x}_j = \xi_j$ with probability $\min\{1, a_j(x_j^{(k)}, \xi_j)\}$ and set $\tilde{x}_j = x_j^{(k)}$ with the remaining probability.

Once the candidate state $\tilde{x}$ has been generated according to the above procedure, the second step of the standard Metropolis algorithms is performed involving the acceptance/rejection of $\tilde{x}$ as the next state of the Markov chain. Thus, if $\tilde{x}$ is a found to be a failure point it is accepted as the next state x^{k+1}; otherwise, $x^{(k+1)} = x^{(k)}$, i.e., one obtains a repeated sample.

It can be easily seen that *the modified Metropolis algorithm overcomes the deficiency of standard Metropolis algorithm by producing distinct candidate states, rather than repeated samples.* Specifically, for large N it is highly unlikely that the candidate state $\tilde{x}$ is equal to the current state $x^{(k)}$, because this would mean that all N components ξ_j were rejected as candidate state components, which is highly unlikely.

However, it can be shown that the above modified algorithm is not perfect, in the sense that it does not guarantee for a chain starting within the important ring that it will remain within this ring at all times. Specifically, we will show that there is a speculative chance that such a chain may exit the important ring for a while. This is demonstrated by the following example. Ignore for a moment the acceptance/rejection step, in other words assume that the entire space $\mathbb{R}^N$ is a failure domain. Consider a Markov chain starting from a random vector $\hat{x}$ drawn from the N-dimensional standard Gaussian distribution and governed by the modified Metropolis algorithm with proposal PDF $S_j(\cdot|x_j) = \mathcal{N}(x_j, 1)$. In Fig. 2.11 the evolution of the Euclidean norm (length) of the Markov chain state is shown. The horizontal lines show the inner and outer radii of the Important Ring. This Figure demonstrates that all states of the generated chain belong in the Important Ring.

Now consider the same Markov chain but starting not from a random point within the important ring but from a point along one of the coordinate axes. For example, consider as starting point the point $\hat{x} = (\sqrt{N}, 0, \ldots, 0)$ which clearly also belongs in the important ring. The evolution of the Euclidean norm of the states of such a Markov chain is shown in the

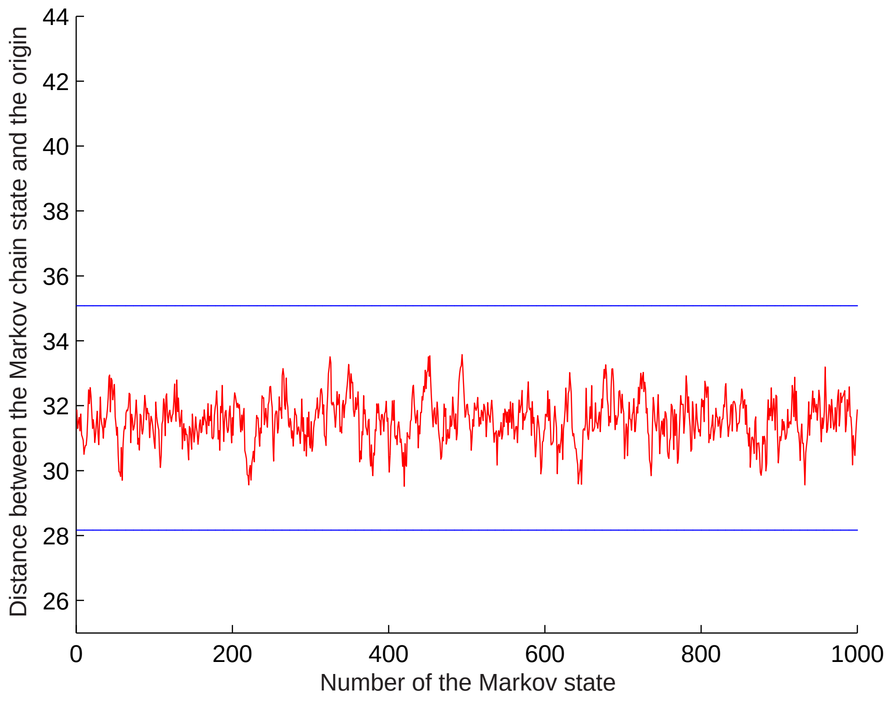


Figure 2.11: Markov chain using MMA starting from random Gaussian vector, $N = 10^3$.

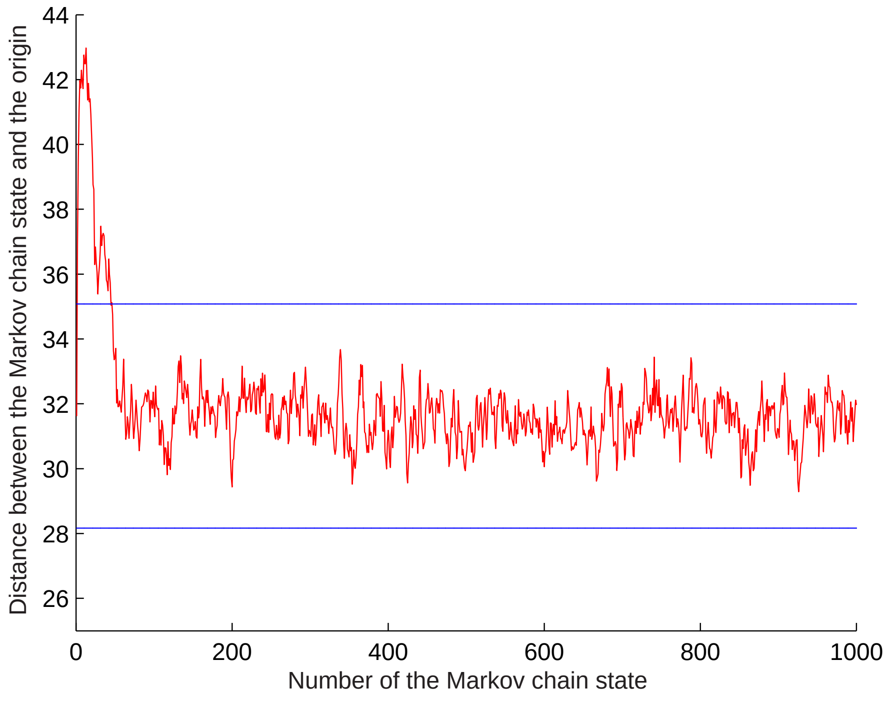


Figure 2.12: Markov chain using MMA starting from $\hat{x} = (\sqrt{N}, 0, \ldots, 0)$, $N = 10^3$.

Fig. 2.12. As can be seen from this figure the Markov chain jumps out of the Important Ring and only after approximately 50 steps returns to it. Thus, using a chain of small length, as is done in practical applications, will result into misleading conclusions. This is because such chain has anything but reached its underlying stationary distribution.

This happens because the modified Metropolis algorithm assumes some structure while the standard Gaussian space is isotropic and has no preferred directions. However, in the modified Metropolis algorithm each coordinate axis direction is a special direction. The closer the initial state of a Markov chain is to one of the x_j-directions the worse the modified Metropolis algorithm works. Of course the probability to obtain an initial state closely aligned to one of the coordinates is very small.

To demonstrate the effect of the above behavior when calculating reliability estimates consider a linear problem with LSF (2.29) with corresponding failure probability (2.32). Let us apply Subset Simulation with modified Metropolis algorithm for evaluating the failure probabilities of two "equivalent" linear failure problems. The design point of the first problem is chosen as $x_1^* = \beta\frac{x}{\|x\|}$, where x is drawn from N-dimensional standard Gaussian distribution, while that of the second problem is along one of the "preferred" directions, namely, $x_2^* = (\beta, 0, \ldots, 0)$. These failure domains have equal probabilities, since they have the same reliability index β. In Fig. 2.13 the CV for corresponding estimators for different values of n (number of samples in each intermediate Subset Simulation level) are shown. Here $\beta = 3$, $N = 10^3$ and 50 runs of the algorithm are used. Clearly in the second case the modified Metropolis algorithm works worse. This is due to the existence of preferred directions and the earlier discussed deficiency of Markov Chains generated from points closely aligned with these directions.

In the implementation of Subset Simulation the choice of proposal distribution is very important since it governs the efficiency in generating the samples by MCMC. Let us again consider a Markov chain simulated by the modified Metropolis algorithm when the entire space is a failure domain, i.e., when there is no acceptance/rejection step. For each component we use a Gaussian proposal as before but instead of using unit variance we consider it as a parameter:

$$S_j(\cdot|x_j) = \mathcal{N}(x_j, \sigma^2), \;\; j = 1, \ldots, N. \tag{2.47}$$

The question we consider is how σ affects the correlation between samples. The expected

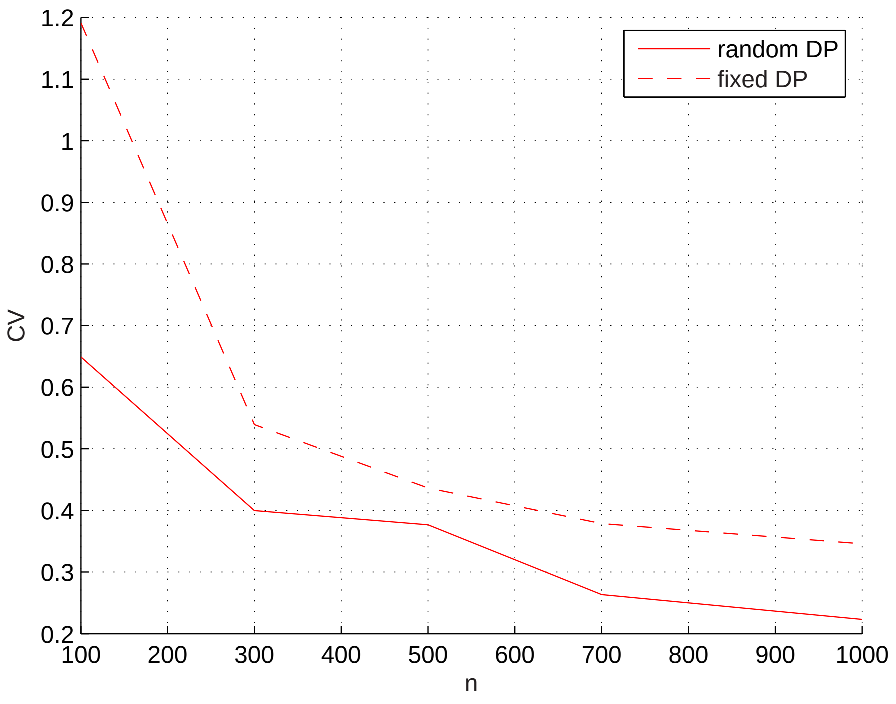


Figure 2.13: CV of p_F estimator for linear failure domain with random and fixed design points.

value of the angle α between two consecutive samples is a good parameter for measuring the correlation of the Markov chain states as it provides a measure of the separation of samples that is independent of the dimension N while alternative measures, such as the Euclidean distance between samples, depend on N. The larger the angle, the more independent are the samples and, therefore, the more ergodic the chain. Clearly, standard Monte Carlo provides the most uncorrelated samples satisfying:

$$E[\alpha] = \pi/2. \tag{2.48}$$

For the described one-parameter family of Markov chains the expected value of α is shown in Fig. 2.14. As we can see the "optimal" standard deviation is somewhere between 2 and 3.

Of course in real engineering applications the failure domain is only a small subset of the parameter space and, therefore, adopting too large standard deviation in the proposal PDF may lead to too many rejections in the second step of the modified Metropolis algorithm yielding a highly correlated chain.

Spherical Subset Simulation (S^3)

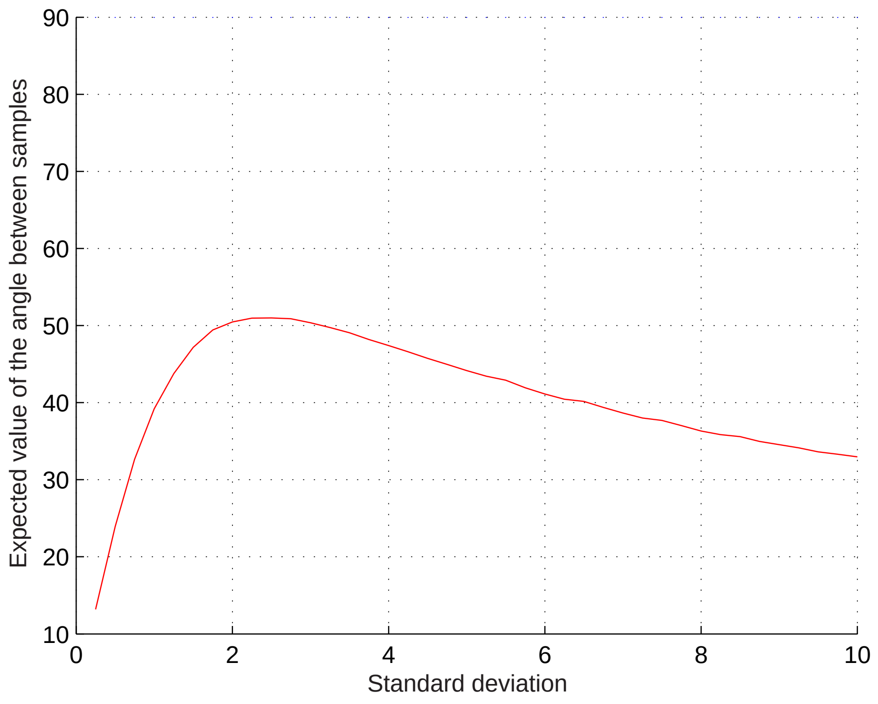


Figure 2.14: Expected value of angle between samples plotted against standard deviation of proposal, $N = 10^3$.

A new MCMC algorithm was proposed by Katafygiotis and Cheung in [15]. Given a sample $x = Ru$, where $u = x/\|x\| \in \mathbb{S}^{N-1}$ is a vector of unit length and $R = \|x\|$, this algorithm consists of two steps. The first step involves sampling an intermediate failure point $x' = Ru'$, having the same length R as the current sample x. That is, in the first step we simulate a new failure point lying on the sphere of radius R. Clearly, all failure points on this sphere are uniformly distributed as they have equal corresponding PDF values. Therefore, the selection of one such failure point is performed by first selecting a random plane $\mathcal{P}(u, \tilde{u})$ containing the direction u and a uniformly distributed random direction $\tilde{u}$ realized as $\tilde{u} = y/\|y\|$ where y is distributed according to the standard Gaussian. Next, we consider the circle C of radius R defined as the intersection of the sphere of radius R and the random plane $\mathcal{P}(u, \tilde{u})$, i.e.,

$$C = \left\{ x : x = R\frac{(u\cos\phi + \tilde{u}\sin\phi)}{\|u\cos\phi + \tilde{u}\sin\phi\|} \right\}, \tag{2.49}$$

where $\phi \in [0, 2\pi]$. Finally, we simulate uniformly distributed points on this circle, by drawing ϕ uniformly from the the interval $[0, 2\pi]$ until a failure point Ru' is reached.

The second step involves radial sampling along the direction u' to obtain R'. The distinct

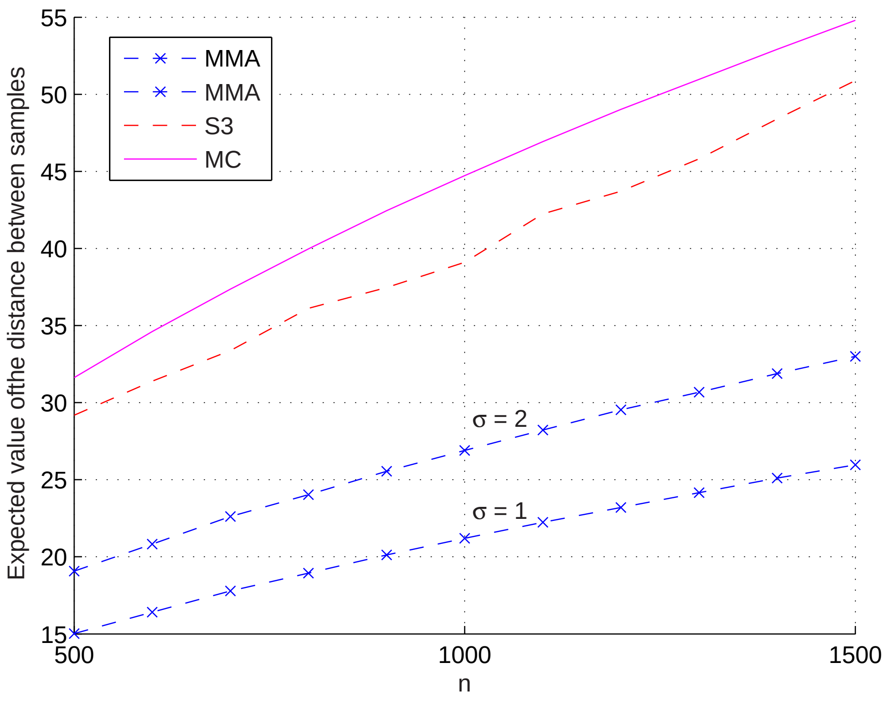


Figure 2.15: Expected value of angular distance between samples plotted against dimension N for MMA, S^3 and MC.

sample $x'' = R'u'$ is then the next state of the chain, i.e. it is accepted with probability one. For more details, refer to [15]. From a geometric point of view the advantage of this method is that it is consistent with the nature of the Gaussian space, namely there are no preferred directions as in the modified Metropolis algorithm. If we assume that the entire domain corresponds to failure it is clear that the expectation of the angle between samples generated by the S^3 method is equal to $\pi/2$ as in standard Monte Carlo.

Next, we compare the angular distances between failure samples generated by standard Monte Carlo, modified Metropolis algorithm (MMA) with $\sigma = 1, 2$ and the above described S^3 MCMC algorithm in the case of a linear failure domain, where the Markov chain is initiated at one of the failure points obtained by Monte Carlo simulations. The results are presented in Fig. 2.15. We can see that in terms of the angular distances between samples, the S^3 method is much closer to Monte Carlo simulation and, therefore, one can argue that it provides a better quality Markov chain.

Finally Fig. 2.16 shows the CV of the failure probability estimates obtained using Subset Simulation employing two alternative Markov chain algorithms: a) the S^3 MCmC algorithm

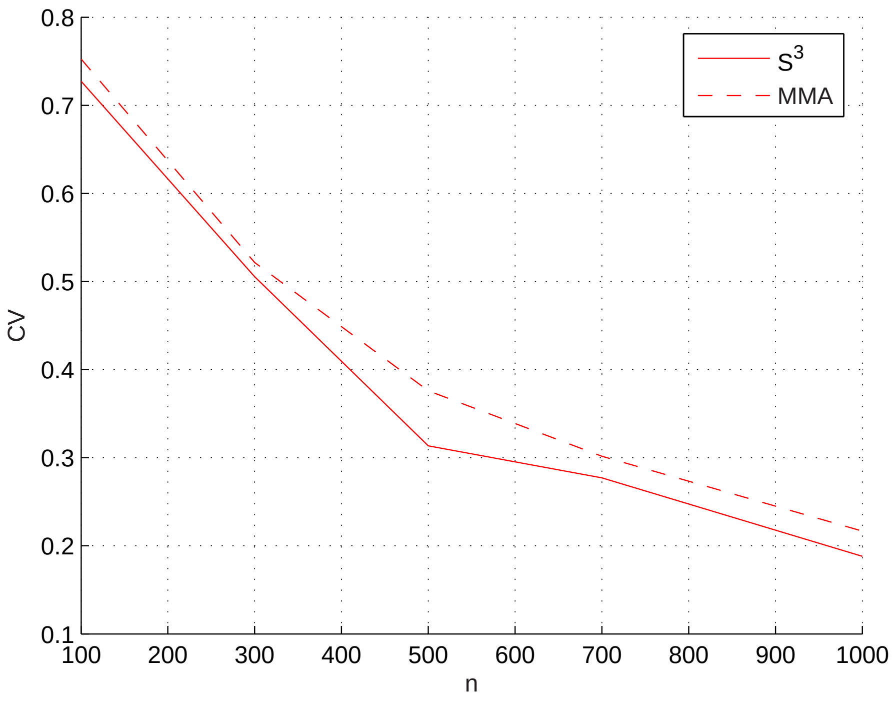


Figure 2.16: The CV of p_F estimator for linear failure domain using S^3 and MMA.

described in this section and b) MMA ($\sigma = 1$). Here reliability index $\beta = 3$ and 50 runs of each algorithms were used. As can be seen, the S^3 algorithm provides better estimates due to the higher quality (more uncorrelated) chain it produces.

2.5 Summary

In this chapter a geometric perspective has been adopted to provide insight into the computational difficulties arising in high-dimensional reliability problems. Several important results are explained following this perspective. An explanation is provided as to why Importance Sampling using a fixed importance sampling density is inapplicable in moderate to strongly nonlinear high-dimensional reliability problems. The reason why the standard Metropolis Algorithm is inapplicable in high dimensions and is bound to produce repeated samples is also given. Potential deficiencies of the modified Metropolis algorithm are revealed and an alternative algorithm that overcomes these deficiencies is presented.

In summary, this chapter provides a deeper understanding of the geometric features of high-dimensional reliability problem. This understanding is invaluable in developing novel efficient algorithms for treating such problems.

Chapter 3

Adaptive Linked Importance Sampling

The theoretical understanding of the world,
which is the aim of philosophy,
is not a matter of great practical importance to animals,
or to savages, or even to most civilised men.

Bertrand Russell

In this chapter we propose a novel advanced simulation approach, called Adaptive Linked Importance Sampling (ALIS), for estimating small failure probabilities encountered in high-dimensional reliability analysis of engineering systems.

It was shown by Au and Beck in [2] that Importance Sampling does generally not work in high dimensions. A geometric understanding of why this is true when one uses a fixed importance sampling density (ISD) was given in chapter 2 (see also [17]). The basic idea of ALIS is instead of using a fixed ISD, as done in standard Importance Sampling, to use a family of intermediate distributions that will converge to the target optimal ISD corresponding to the conditional probability given the failure event. We show that Subset Simulation, which was introduced in [1] and discussed in the section 2.4, does correspond to a special case of ALIS, where the intermediate ISDs are chosen to correspond to the conditional distributions given adaptively chosen intermediate nested failure events. However, the general formulation of ALIS allows for a much richer choice of intermediate ISDs. As the concept of subsets is not a central feature of ALIS, the failure probability is not any longer expressed as a product of conditional failure probabilities as in the case of Subset Simulation. Starting with a one-dimensional example we demonstrate that ALIS can offer drastic improvements over Subset Simulation. In particular, we show that the choice of intermediate ISDs prescribed by

Subset Simulation is far from optimal. The generalization to high-dimensions is discussed. The accuracy and efficiency of the method is demonstrated with numerical examples. This chapter is based on the paper [19].

3.1 Proposed methodology

We proceed from the classical Importance Sampling that was described in the section 2.2. For convenience, from now on we shall use π_1 to denote ISD instead of π_{is}.

It is easy to see that the theoretically optimal choice of π_1 is the conditional PDF:

$$\pi_1^{opt}(x) = \pi_0(x|F) = \frac{\pi_0(x)I_F(x)}{p_F}. \tag{3.1}$$

For this ISD the estimate (2.22) is $p_{is} = p_F$ for any number of samples n (even for $n = 1$) with resulting CV $\delta_{is} = 0$. We can conclude that the problem of estimating the failure probability is equivalent to estimating the normalizing constant for the following "optimal" non-normalized density function:

$$p_1^{opt}(x) = \pi_0(x)I_F(x). \tag{3.2}$$

Now we will reformulate the problem of estimating the failure probability in that of estimating the ratio of the normalizing constants of two distributions.

3.1.1 Reformulation of the problem

Consider two PDFs π_0 and π_1 on the same space $\mathbb{R}^N$:

$$\pi_0(x) = \frac{p_0(x)}{Z_0}, \quad \pi_1(x) = \frac{p_1(x)}{Z_1}, \tag{3.3}$$

where Z_0 and Z_1 are the corresponding normalizing constants ensuring the total probability is equal to one. Suppose that we are not able to directly compute π_0 and π_1 since we do not know the normalizing constants Z_0 and Z_1, but p_0 and p_1 are known pointwise. Our goal is to find a Monte Carlo estimate for the ratio

$$r = \frac{Z_1}{Z_0}. \tag{3.4}$$

Assuming this problem can be tackled, the estimation of failure probability easily follows by appropriately choosing p_0 and p_1:

$$\begin{aligned} p_0(x) &= \pi_0(x), & Z_0 &= 1, \\ p_1(x) &= \pi_0(x) I_F(x), & Z_1 &= p_F, \end{aligned} \quad \Rightarrow \; r = p_F. \tag{3.5}$$

This observation links the problem of estimating failure probabilities to two other extensively researched problems that can also be considered as problems of estimating the ratio of normalizing constants:

- Finding the free energy of a physical system (Statistical Physics),
- Finding the marginal likelihood of a Bayesian statistical model (Bayesian Statistics).

In this chapter we will address the problem of reliability estimation as a special case of calculating the ratio r of normalizing constants of two distributions.

Using (3.3), (3.4) it is follows that

$$E_{\pi_0}\left[\frac{p_1}{p_0}\right] = \int\limits_{\mathbb{R}^N} \frac{p_1(x)}{p_0(x)} \pi_0(x) dx = \int\limits_{\mathbb{R}^N} \frac{Z_1}{Z_0} \pi_1(x) dx = \frac{Z_1}{Z_0}. \tag{3.6}$$

This means that we can estimate r by

$$\hat{r}_{mc(sis)} = \frac{1}{n} \sum_{k=1}^{n} \frac{p_1(x^{(k)})}{p_0(x^{(k)})}, \qquad x^{(k)} \sim \pi_0. \tag{3.7}$$

In the context of failure probabilities, i.e., using (3.5), one can easily recognize that (3.7) becomes exactly the expression for estimating p_F using standard Monte Carlo. In statistical physics (3.7) is also known as the Simple Importance Sampling (SIS) estimate. Now we turn to the idea of intermediate distributions.

3.1.2 Intermediate distributions

If π_0 and π_1 are not close enough, the estimate (3.7) will be improper: the variance of this estimate will be very large. In such a situation, it may be possible to obtain a good estimate by introducing artificial intermediate distributions (AIDs).

Let us define a sequence $p_{\alpha_0}, \ldots, p_{\alpha_m}$ of non-normalized (or unnormalized) density functions (UDF) using $0 = \alpha_0 < \alpha_1 < \ldots < \alpha_{m-1} < \alpha_m = 1$, so that the first and last functions in the sequence are p_0 and p_1 with the intermediate UDF's interpolating between them. Let

$Z_\alpha = \int p_\alpha(x)dx$ and $\pi_{\alpha_0}, \dots, \pi_{\alpha_m}$ denote the corresponding sequence of normalizing constants and PDFs, respectively. Then we can represent the ratio $r = Z_1/Z_0$ as follows:

$$\frac{Z_1}{Z_0} = \frac{Z_{\alpha_1}}{Z_0}\frac{Z_{\alpha_2}}{Z_{\alpha_1}} \dots \frac{Z_{\alpha_{m-1}}}{Z_{\alpha_{m-2}}}\frac{Z_1}{Z_{\alpha_{m-1}}}. \tag{3.8}$$

Now, if π_{α_j} and $\pi_{\alpha_{j+1}}$ are sufficiently close and we can sample from π_{α_j}, we can accurately estimate each factor $Z_{\alpha_{j+1}}/Z_{\alpha_j}$ using simple importance sampling (3.7), and from these estimates obtain an estimate for r:

$$\hat{r}_{gis} = \prod_{j=0}^{m-1}\left(\frac{1}{n}\sum_{k=1}^{n}\frac{p_{\alpha_{j+1}}(x_j^{(k)})}{p_{\alpha_j}(x_j^{(k)})}\right), \; x_j^{(k)} \sim \pi_{\alpha_j}. \tag{3.9}$$

We will refer to (3.9) as Generalized Importance Sampling (GIS).

From (3.9) it follows that Subset Simulation (section 2.4) is exactly Generalized Importance Sampling with specific intermediate distributions:

$$\pi_{\alpha_j}(x) = \pi_0(x|F_{\alpha_j}), \quad j = 0, \dots, m. \tag{3.10}$$

Intuition suggest that it is rather unlikely for the family of conditional intermediate distributions (3.10) with sharp boundaries to be optimal for estimating failure probability. Indeed, our numerical examples will show that this family of AIDs is far from optimal and, therefore, there is room to further improve efficiency beyond that offered by Subset Simulation.

Here we propose two smooth families of AIDs designed for reliability problems:

$$p_\alpha^I(x) = \pi_0(x)\min\{e^{-\alpha G(x)}, 1\}, \tag{3.11}$$

$$p_\alpha^{II}(x) = \frac{\pi_0(x)}{1 + e^{\alpha G(x)}}, \tag{3.12}$$

where G is a LSF. It is easy to check, that for both distributions we have:

$$\lim_{\alpha\to+\infty} p_\alpha(x) = p_0(x)I_F(x) = p_\infty(x). \tag{3.13}$$

Although before it was assumed that $\alpha \in [0, 1]$, in (3.11), (3.12) the α belongs in the ray $\alpha \in [0, +\infty]$. Obviously this is not of fundamental importance. We will refer to (3.11), (3.12) as to the distributions of the first and the second types correspondingly. Note, that for the first type distributions $p_0 = \pi_0$ and $Z_0 = 1$ as in (3.5), but for the second ones we have $p_0 = \pi_0/2$ and therefore $Z_0 = 0.5$. Again this can be easily incorporated in the expression for the estimate of p_F. Specifically, as can be easily derived from (3.4) and (3.5), one simply has to multiply the estimate for r by Z_0, i.e., $p_F = 0.5r$.

The intermediate distributions (3.10) for Subset Simulation and (3.11), (3.12) for ALIS are shown in Figures 3.1, 3.2, and 3.3, respectively.

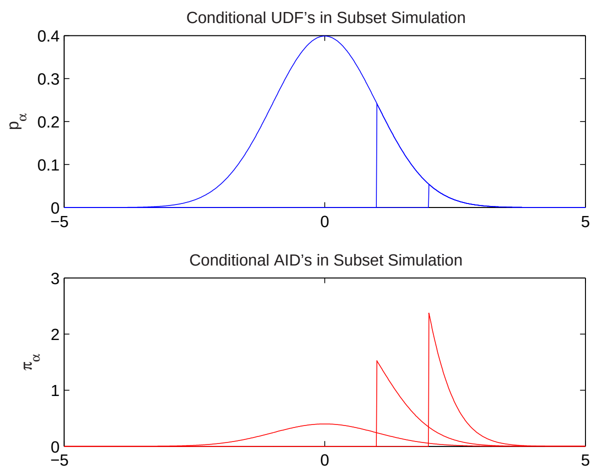


Figure 3.1: Intermediate distributions in Subset Simulation.

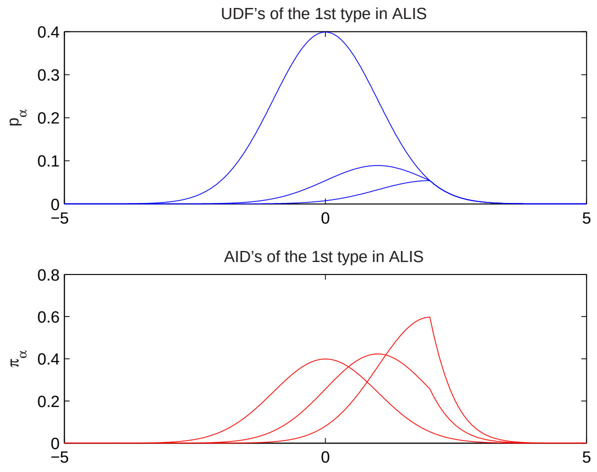


Figure 3.2: Intermediate distributions in ALIS, type I.

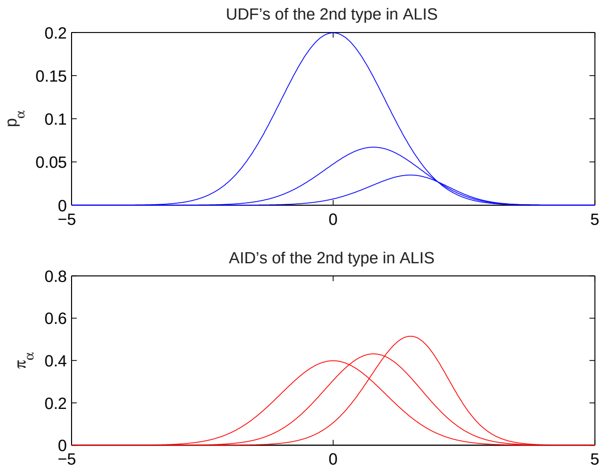


Figure 3.3: Intermediate distributions in ALIS, type II.

3.1.3 Bridging

There is a potential problem one needs to be aware of when using intermediate distributions and that is the following. If the distributions $\{\pi_\alpha\}$ are not nested, one can not hope to estimate $Z_{\alpha_{j+1}}/Z_{\alpha_j}$ by sampling just from π_{α_j}. Indeed, this can be easily demonstrated with the following example involving uniform distributions.

Let $p_{\alpha_j} = I_{[0,4]}$ and $p_{\alpha_{j+1}} = I_{[3,5]}$, so that $Z_{\alpha_j} = 4$, $Z_{\alpha_{j+1}} = 2$ and $Z_{\alpha_{j+1}}/Z_{\alpha_j} = 1/2$. Suppose we have n points drawn uniformly from $[0,4]$, i.e. from π_{α_j}. The fraction of these that lie in $[3,5]$, i.e. the simple importance sampling estimate (3.7), will obviously not give an estimate of $Z_{\alpha_{j+1}}/Z_{\alpha_j}$ since it converges to $1/4$ when $n \to \infty$ rather than $1/2$. Instead it will be an estimate of Z_*/Z_{α_j}, where Z_* is the length of $[3,4] = [0,4] \cap [3,5]$. Suppose now that we also have n samples drawn uniformly from $[3,5]$, i.e., from $\pi_{\alpha_{j+1}}$. The fraction of these that lie in $[0,4]$ will be an estimate of $Z_*/Z_{\alpha_{j+1}}$. Taking the ratio of these two estimates $(Z_*/Z_{\alpha_j})/(Z_*/Z_{\alpha_{j+1}})$ gives an estimate of $Z_{\alpha_{j+1}}/Z_{\alpha_j}$.

This idea can be generalized and non-nested distributions can be worked up by "bridging". In this method we replace the simple importance sampling estimate for $Z_{\alpha_{j+1}}/Z_{\alpha_j}$ by a ratio of estimates for Z_*/Z_{α_j} and $Z_*/Z_{\alpha_{j+1}}$, where Z_* is the normalizing constant for a "bridge distribution", $\pi_*(x) = p_*(x)/Z_*$, which is chosen so that it is overlapped by both π_{α_j} and $\pi_{\alpha_{j+1}}$. Using simple importance sampling estimates for Z_*/Z_{α_j} and $Z_*/Z_{\alpha_{j+1}}$, we can obtain the estimate for $Z_{\alpha_{j+1}}/Z_{\alpha_j}$:

$$\frac{Z_{\alpha_{j+1}}}{Z_{\alpha_j}} = \frac{E_{\pi_{\alpha_j}}\left[\frac{p_*}{p_{\alpha_j}}\right]}{E_{\pi_{\alpha_{j+1}}}\left[\frac{p_*}{p_{\alpha_{j+1}}}\right]} \approx \frac{\frac{1}{n}\sum_{k=1}^{n} \frac{p_*(x_j^{(k)})}{p_{\alpha_j}(x_j^{(k)})}}{\frac{1}{n}\sum_{k=1}^{n} \frac{p_*(x_{j+1}^{(k)})}{p_{\alpha_{j+1}}(x_{j+1}^{(k)})}}, \qquad \begin{array}{l} x_j^{(k)} \sim \pi_{\alpha_j}, \\ x_{j+1}^{(k)} \sim \pi_{\alpha_{j+1}}. \end{array} \tag{3.14}$$

This technique was introduced by Bennett in [3], who called it the "acceptance ratio" method. This method was later rediscovered by Meng and Wong in [24], who called it "bridge sampling". Lu, Singh and Kofke in [22] give a recent review.

One natural choice for the bridge distribution is the "geometric" bridge:

$$p_*^{geo}(x) = \sqrt{p_{\alpha_j}(x) p_{\alpha_{j+1}}(x)}, \tag{3.15}$$

that is in some sense an average between π_{α_j} and $\pi_{\alpha_{j+1}}$. It was shown in [3], that an asymptotically optimal choice of bridge distribution is

$$p_*^{opt}(x) = \frac{p_{\alpha_j}(x) p_{\alpha_{j+1}}(x)}{\frac{n_j}{n_{j+1}} \frac{Z_{\alpha_{j+1}}}{Z_{\alpha_j}} p_{\alpha_j}(x) + p_{\alpha_{j+1}}(x)}. \tag{3.16}$$

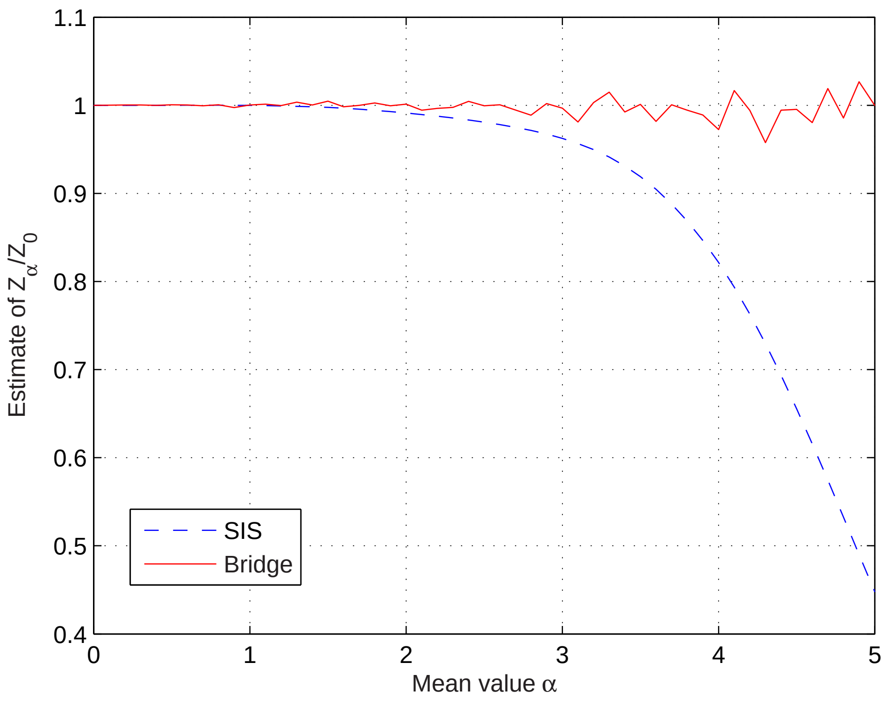


Figure 3.4: Estimate of ratio of normalizing constants for Gaussian distributions using SIS and geometrical bridge.

In general we cannot use this bridge distribution in practice, since we do not know $Z_{\alpha_{j+1}}/Z_{\alpha_j}$ beforehand. But we can use a preliminary guess of $Z_{\alpha_{j+1}}/Z_{\alpha_j}$ to define a bridge distribution (3.16) and then iterate until convergence is observed.

Next we discuss the meaning of "nested" for non-uniform distributions. Again let us start with an easy example. Let $x_1, \ldots, x_n$ be samples following the standard one-dimensional Gaussian distribution. It is easy to check that for $n = 10^4$ the probability that at least one sample lies outside the interval $[-6, 6]$ is smaller than 10^{-4}. Although, strictly speaking, the support of the standard Gaussian distribution is the entire real line (theoretical support), in practice the support may be regarded as just a segment (practical support). So if π_{α_j} and $\pi_{\alpha_{j+1}}$ are two Gaussian distributions with different mean values we cannot hope to estimate $Z_{\alpha_{j+1}}/Z_{\alpha_j}$ by sampling just from π_{α_j}. The reason is the same as in the case of uniform distributions: their practical supports are not nested. As an example, the dependence between the estimate of Z_α/Z_0, where $\pi_\alpha = \mathcal{N}(\alpha, 1)$ is a Gaussian distribution centered at α and having unit variance is plotted against α in Fig. 3.4. Two methods are used: a) SIS according to (3.7) and b) using geometrical bridge according to (3.14). Clearly the SIS estimate deteriorates as α increases, i.e., as the practical supports of the two distributions become further apart. On the contrary, the results using bridge sampling remain relatively unaffected. Note that the Important Ring that was introduced in the section 2.1 is the practical support of the high-dimensional standard Gaussian distribution.

So, in order to apply AIDs for estimating the of ratio of normalizing constants we have to use the concept of "bridge", especially if these AIDs are not nested as is generally the case. Thus, for each couple of adjacent intermediate distributions p_{α_j} and $p_{\alpha_{j+1}}$ we have to construct a bridge p_{j*j+1} using, for example, (3.15) or (3.16) and then for each multiplier in (3.8) use the bridged approximation (3.14). Note, that in Subset Simulation the conditional intermediate distributions (3.10) are nested and, therefore, bridging makes no sense.

3.1.4 Sampling

As we can see from the previous sections we need some procedure to sample independently from an intermediate PDF π_α. Although, by the assumption (i), we can always do this directly for π_0 it can not be done for other distributions with $\alpha \neq 0$. Recall that we can only evaluate the target distribution π_α up to a normalizing constant Z_α, i.e. $\pi_\alpha = p_\alpha / Z_\alpha$. Here we assume that the non-negative function p_α is known pointwise.

A practical framework for sampling under the above context is that provided by discrete time Markov chains underlying the Markov chain Monte Carlo (MCMC) methods. The main idea of all MCMC methods is the following: although it is difficult to efficiently generate independent samples according to the target PDF, it is possible, using a specially designed Markov chain, to efficiently generate dependent samples that are at least asymptotically (as the number of Markov steps increases) distributed as the target PDF. For more details on the MCMC, please, refer to the Appendix A.

So, for each α we define a Markov chain whose invariant distribution is π_α. Simulating such chain for some time, and discarding the early portion, provides dependent points sampled approximately according to π_α. It is importance to mention that usually it is difficult to tell whether the chain reaches equilibrium in a reasonable time or not. Furthermore, even if it has reached equilibrium, it is hard to find out for sure. This is a common drawback of all MCMC methods.

3.2 ALIS

We have described all the components for the ALIS and now we are ready to give a summary of this algorithm.

Adaptive Linked Importance Sampling

```
I. INITIALIZATION
    • Set $j = 0$, $\alpha_0 = 0$.
    • For $k = 1, \dots, n$ sample $x_0^{(k)} \sim \pi_{\alpha_0} = \pi_0$.
    • Evaluate the number $n_F$ of failure samples $\tilde{x}_0^{(k)} \in F \cap \{x_0^{(k)}\}$.
While $\frac{n_F}{n} < T_0$ (for some threshold $T_0$)
II. CONSTRUCTION THE NEXT AID
    • Set $j = j + 1$.
    • Find $\alpha_j$ using equation $\frac{1}{n} \sum_{k=1}^{n} \frac{p_{\alpha_j}(x_{j-1}^{(k)})}{p_{\alpha_{j-1}}(x_{j-1}^{(k)})} = T_j$, for some threshold $T_j$.
III. CONSTRUCTION THE BRIDGE AND LINK STATE
    • Define the bridge $p_{j-1*j}$ between $\pi_{\alpha_{j-1}}$ and $\pi_{\alpha_j}$ using (3.15) or (3.16).
    • For $k = 1, \dots, n$ evaluate $w^{(k)} = \frac{p_{j-1*j}(x_{j-1}^{(k)})}{p_{\alpha_{j-1}}(x_{j-1}^{(k)})}$ and normalize these weights
      $\tilde{w}^{(k)} = \frac{w^{(k)}}{\sum_{k=1}^{n} w^{(k)}}$.
    • Set the link state $x_{j-1*j}$ equals to $x_{j-1}^{(k)}$ with probability $\tilde{w}^{(k)}$.
IV. SAMPLING
    • Set $x_j^{(1)} = x_{j-1*j}$.
    • Starting from $x_j^{(1)}$ sample $x_j^{(k)} \sim \pi_{\alpha_j}$ for $k = 2, \dots, n$ using MCMC.
    • Evaluate the estimate $\widehat{\frac{Z_{\alpha_j}}{Z_{\alpha_{j-1}}}}$ of $\frac{Z_{\alpha_j}}{Z_{\alpha_{j-1}}}$ using (3.14).
    • Evaluate the number $n_F$ of failure samples $\tilde{x}_j^{(k)} \in F \cap \{x_j^{(k)}\}$.
End While
V. ESTIMATE OF THE LAST FACTOR
```

- Define the bridge $p_{j*\infty}$ between π_{α_j} and $\pi_\infty = \pi_0(\cdot|F)$ using (3.15) or (3.16).
- For $k = 1, \ldots, n_F$ evaluate $w_\infty^{(k)} = \frac{p_{j*\infty}(\tilde{x}_j^{(k)})}{p_{\alpha_j}(\tilde{x}_j^{(k)})}$ and normalize these weights $\tilde{w}_\infty^{(k)} = \frac{w^{(k)}}{\sum_{k=1}^n w^{(k)}}$.
- Set the link state $x_{j*\infty}$ equals to $\tilde{x}_j^{(k)}$ with probability $\tilde{w}_\infty^{(k)}$.
- Starting from $x_\infty^{(1)} = x_{j*\infty}$ sample $x_\infty^{(k)} \sim \pi_\infty$ for $k = 2, \ldots, n$ using MCMC.
- Evaluate the estimate $\widehat{\frac{Z_\infty}{Z_{\alpha_j}}}$ of $\frac{Z_\infty}{Z_{\alpha_j}}$ using (3.14).

VI. Estimate of r

- According to (3.8), estimate r as follows :

$$\hat{r}_{alis} = \prod_{i=0}^{j-1} \widehat{\frac{Z_{\alpha_{i+1}}}{Z_{\alpha_i}}} \cdot \widehat{\frac{Z_\infty}{Z_{\alpha_j}}}. \tag{3.17}$$

Estimate $\hat{p}_{alis}$ is obtained from $\hat{r}_{alis}$ by multiplication by Z_0. For the first type of AIDs $Z_0 = 1$, but for the second one $Z_0 = 0.5$.

We can use several runs of ALIS and average them in order to obtain the final estimate of r, since it can be shown that estimate (3.17) is unbiased, see [27] for details. Here we just want to mention that the crucial aspect of ALIS is that when moving from distribution $\pi_{\alpha_{j-1}}$ to π_{α_j}, a link state, x_{j-1*j}, is randomly selected from the samples $x_{j-1}^{(1)}, \ldots, x_{j-1}^{(n)}$ that are drawn from $\pi_{\alpha_{j-1}}$. We can consider the link state as a sample associated with $\pi_{\alpha_{j-1}}$ as well as that associated with π_{α_j}.

Let us now discuss the details of this procedure. In the step (II) having UDF $p_{\alpha_{j-1}}$ we construct the next intermediate distribution p_{α_j} adaptively using SIS (3.7). Namely, we try to find α_j such that the SIS estimate of $Z_{\alpha_j}/Z_{\alpha_{j-1}}$ equals to some threshold T_j. Later in (IV) we improve this estimate using bridging. Thus, $\widehat{\frac{Z_{\alpha_j}}{Z_{\alpha_{j-1}}}} \approx T_j$ is refined estimate after fixing α_j.

The step (V) highly depends on the type of AIDs. If the intermediate distributions are of the first type, than it can be easily shown that both geometrical and optimal bridge from p_α to p_∞ is equal to

$$p_{\alpha*\infty}^I(x) = p_\infty(x). \tag{3.18}$$

And therefore the estimate of the last factor in this case :

$$\widehat{\frac{Z_\infty}{Z_{\alpha_j}}} = \frac{1}{n}\sum_{k=1}^n \frac{p_\infty(x_j^{(k)})}{p_{\alpha_j}(x_j^{(k)})} = \frac{1}{n}\sum_{k=1}^n \frac{I_F(x_j^{(k)})}{\min\{e^{-\alpha G(x_j^{(k)})}, 1\}} = \frac{n_F}{n} \tag{3.19}$$

So if we use the AIDs of the first type the step (V) in ALIS (in particular, sampling from π_∞) is no longer needed. The estimate of the last factor is given by (3.19).

The case of the second type AIDs is more complicated. The optimal bridge from p_{α_j} to p_∞ has the following form :

$$p^{II,opt}_{\alpha_j * \infty}(x) = \frac{\pi_0(x) I_F(x)}{1 + T_j + e^{\alpha_j G(x)}} \tag{3.20}$$

According to (3.14), the estimate of the last factor in this case :

$$\widehat{\frac{Z_\infty}{Z_{\alpha_j}}} = \frac{1}{n} \sum_{k=1}^{n_F} \frac{1 + e^{\alpha_j G(\tilde{x}_j^{(k)})}}{1 + T_j + e^{\alpha_j G(\tilde{x}_j^{(k)})}} \Bigg/ \frac{1}{n} \sum_{k=1}^{n} \frac{1}{1 + T_j + e^{\alpha_j G(x_\infty^{(k)})}} \tag{3.21}$$

The similar formula can be obtained when geometrical bridge is used. The sampling from π_∞ can be done by modified Metropolis algorithm (section 4.1.3).

Suppose now that during the run of this procedure m AIDs were used : $\pi_0, \pi_{\alpha_1}, \ldots, \pi_{\alpha_m}, \pi_{\alpha_\infty}$. And from each of these distributions n samples were drawn. If we have used the AIDs of the first type, than the total amount of samples is $n(m+1)$. Instead, if the second type AIDs were used, than the total amount of samples is $n(m+2)$, since we need to sample from π_{α_∞} too. So, in terms of computational effort (iii), the AIDs of the first type are more efficient.

3.3 Examples

The ALIS algorithm is applied to calculate the failure probability in two examples. In these examples we use intermediate distributions of both types (3.11), (3.12) and optimal bridge (3.16). The value T_j of all thresholds is chosen to be 0.1, which is found to yield good efficiency. ALIS will be compared with Subset Simulation and standard Monte Carlo in terms of the coefficient of variation (CV) of the failure probability estimates.

3.3.1 One-dimensional ray

Consider the simplest possible one-dimensional example. Let the failure domain F be a ray on the real line:

$$F = \{x \in \mathbb{R} | x > b\}, \tag{3.22}$$

for some threshold b, where x is a standard Gaussian random variable, $x \sim \mathcal{N}(0,1)$. Obviously, the failure probability in this case is

$$p_F = 1 - \Phi(b), \tag{3.23}$$

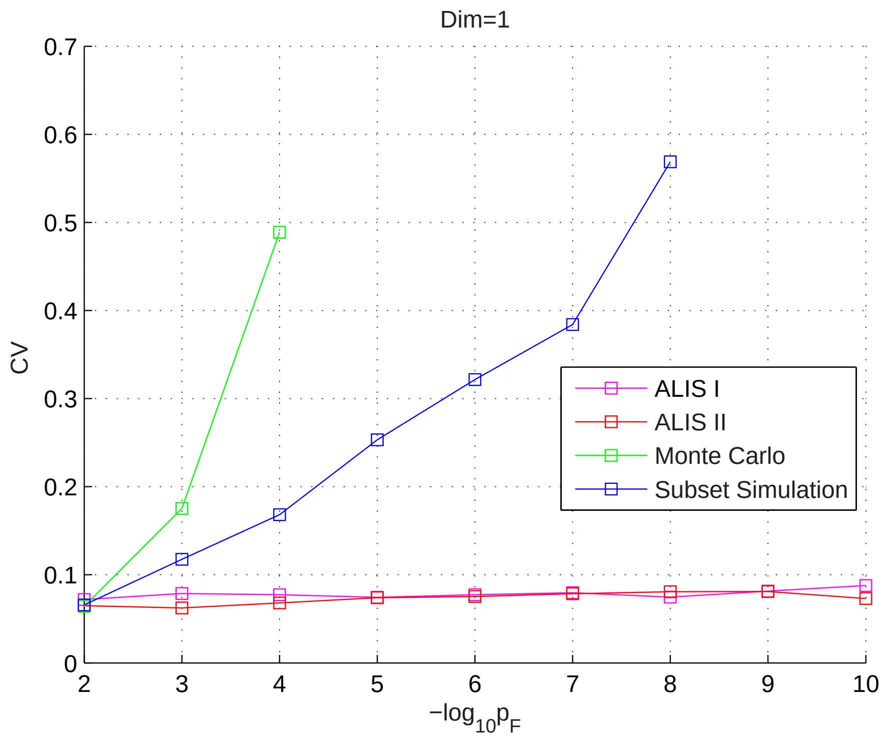


Figure 3.5: The CV of estimates obtained by ALIS, Subset Simulation and Monte Carlo in one-dimensional problem.

where Φ is the CDF of the standard Gaussian distribution. We apply the described simulation methods for estimating the following values of failure probability: $p_F = 10^{-k}$, $k = 2, \ldots, 10$. The corresponding thresholds are:

$$b_k = \Phi^{-1}(1 - 10^{-k}). \tag{3.24}$$

The CV of the estimates obtained by ALIS, Subset Simulation and Monte Carlo are given in Fig. 3.5 using 100 runs. For each intermediate distribution $n = 10^4$ samples were drawn for ALIS and Subset Simulation. The comparison between the different methods is made for the same total computational effort. It can be clearly seen that ALIS clearly outperforms Subset Simulation. Furthermore, the computational effort required by ALIS in order to achieve a certain level of accuracy is found to increase only linearly with the order of target failure probability while this is not the case for Subset Simulation.

3.3.2 High-dimensional sphere

The next example is high-dimensional. Let $x \in \mathbb{R}^N$ be a standard Gaussian random vector. As discussed in section 2.1 and [17], the vast majority of the probability mass in the N-dimensional standard Gaussian space belongs in an Important Ring (= practical support),

$$\sqrt{N} - r < R < \sqrt{N} + r. \tag{3.25}$$

Thus, a sample $x \in \mathbb{R}^N$ distributed according to this high-dimensional standard Gaussian distribution will lie with extremely large probability in this Important Ring.

Consider in $\mathbb{R}^N$ the interior of a cone with axis a and angle φ:

$$\mathbb{C}_{a,\varphi} = \{x \in \mathbb{R}^N | \widehat{x, a} < \varphi\}, \tag{3.26}$$

and define the failure domain to be the intersection of $\mathbb{C}_{a,\varphi}$ and the hypersphere $\mathbb{S}^{N-1}_{\sqrt{N}}$ with radius $\sqrt{N}$ and center in the origin (i.e., the middle hypersphere in the Important Ring):

$$F = \mathbb{C}_{a,\varphi} \cap \mathbb{S}^{N-1}_{\sqrt{N}}. \tag{3.27}$$

Instead of a Gaussian distribution in $\mathbb{R}^N$, we consider here a uniform distribution on the hypersphere $\mathbb{S}^{N-1}_{\sqrt{N}}$ with failure domain defined by (3.27). Unlike in the one-dimensional example, where an input load x with sufficiently large "energy" causes failure, in this example all inputs have the same energy. In this case safety or failure depends on the "direction" of the input: the closer the direction of x to the cone axis a, the more unsafe the system is. In this sense this example quite differs from the previous one.

We generate a candidate state in Metropolis update for sampling from π_α in the following way. If the current sate of the chain is $x \in \mathbb{S}^{N-1}\sqrt{N}$, we first generate a sample $y \sim \mathcal{N}_{(x,1)}$ from the multidimensional Gaussian distribution centered in x. The candidate state is then the projection of y on the sphere $\mathbb{S}^{N-1}_{\sqrt{N}}$, $\tilde{x} = \text{proj}_{\mathbb{S}} y$. Obviously, the corresponding proposal PDF is symmetric, $S(\tilde{x}|x) = S(x|\tilde{x})$, and the acceptance ratio $a(x, \tilde{x}) = p_\alpha(\tilde{x})/p_\alpha(x)$. Since everything is taking place on a sphere, the probability to obtain a repeated sample during these updates is not close to 1, and the corresponding Markov chain is able to explore the entire parameter space.

Choosing φ we can vary the failure probability. We consider the following values of failure probability: $p_F = 10^{-k}$, $k = 2, \ldots, 7$. The CV of the failure probability estimates obtained by

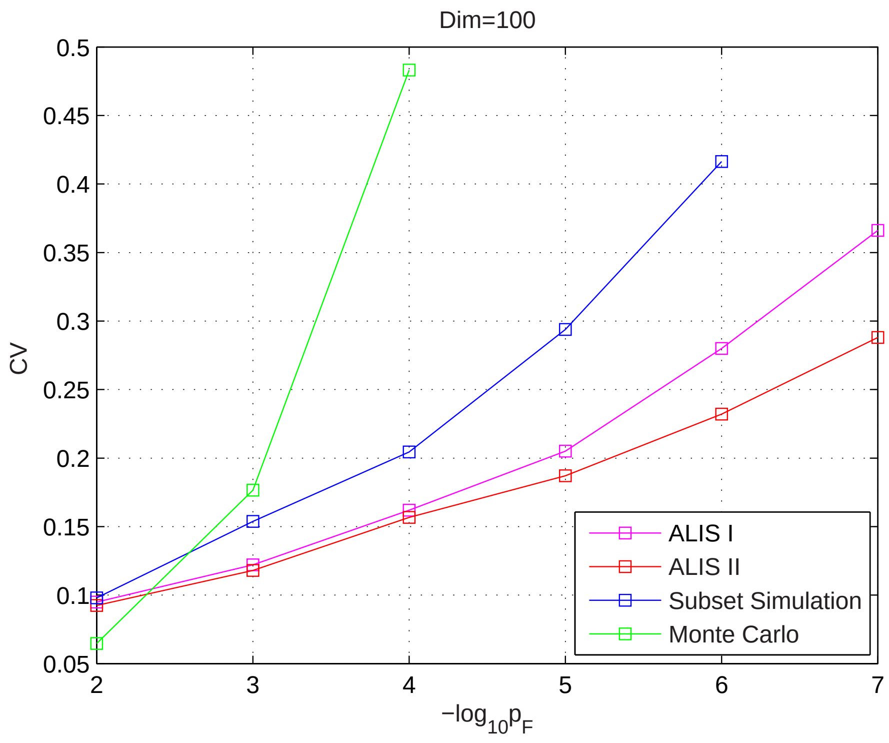


Figure 3.6: The CV of estimates obtained by ALIS, Subset Simulation and Monte Carlo in 100-dimensional problem.

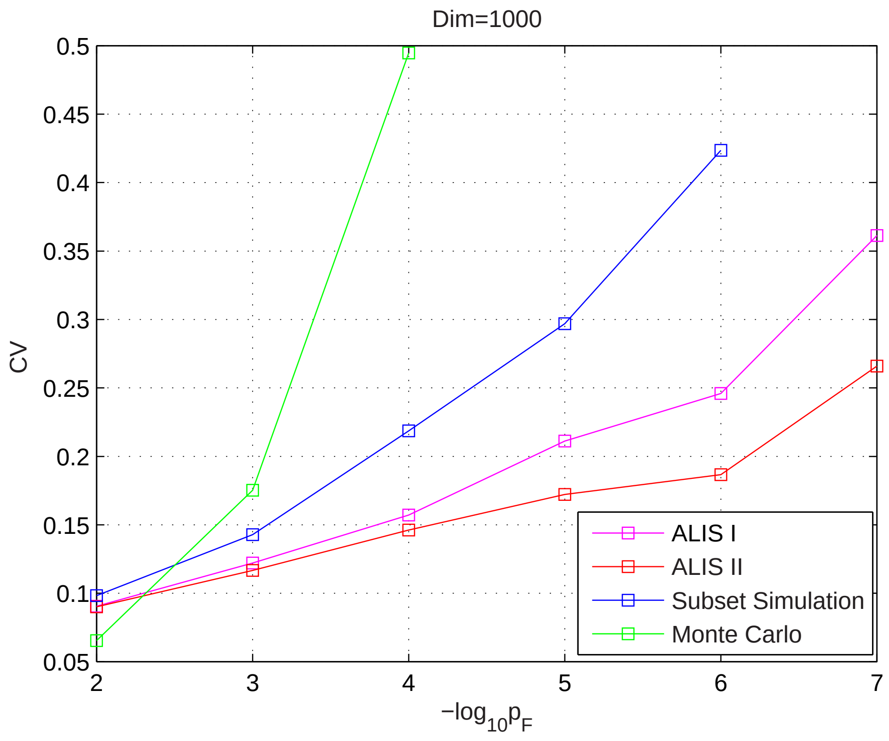


Figure 3.7: The CV of estimates obtained by ALIS, Subset Simulation and Monte Carlo in 1000-dimensional problem.

ALIS, Subset Simulation and Monte Carlo are given in Fig. 3.6 and Fig. 3.7 for dimensions $N = 100$ and $N = 1000$ respectively, using 100 runs each time. For each intermediate distribution $n = 10^4$ samples were drawn. The comparison between the different methods is made for the same total computational effort. It can be clearly seen that ALIS clearly again outperforms Subset Simulation although the computational effort does not quite increase linearly with the order of failure probability in this case.

Chapter 4

Modified Metropolis-Hastings Algorithm with Delayed Rejection

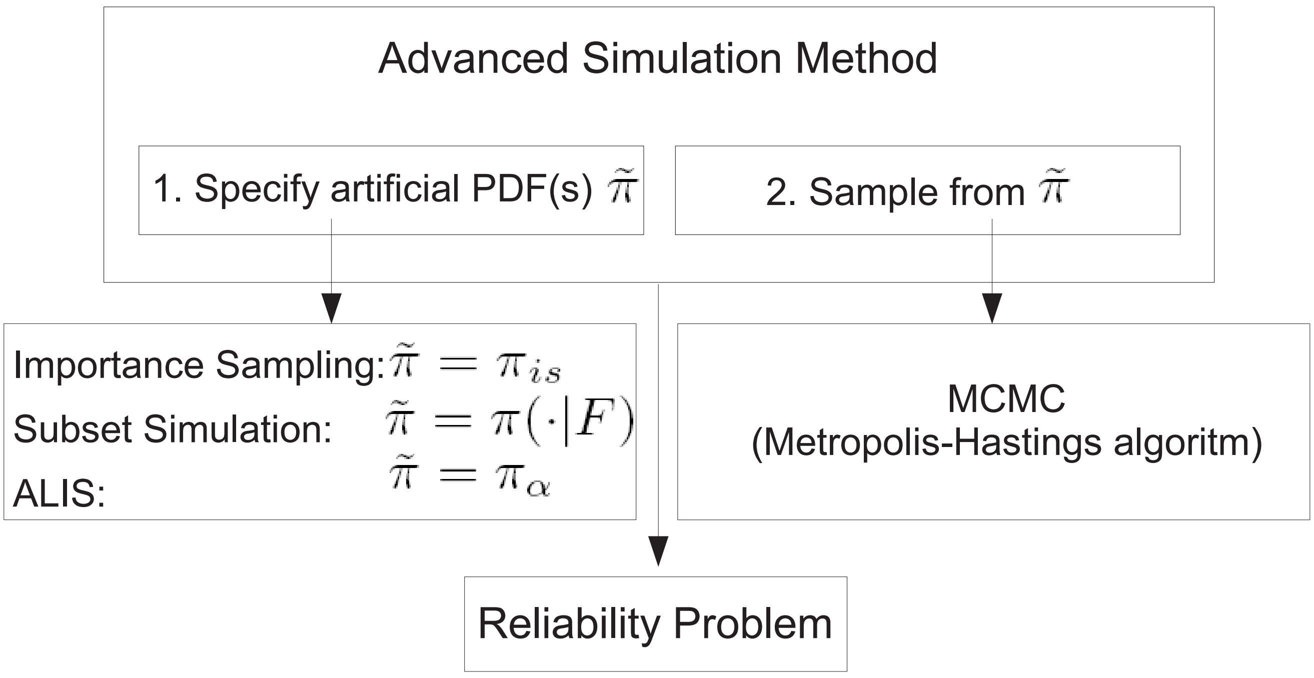
It is a bad plan that admits of no modification.

Publilius Syrus

Each advanced stochastic simulation algorithm for computation of small failure probabilities (1.2) encountered in reliability analysis of engineering systems consists of two main steps. At first, we need to specify some artificial PDF(s) $\tilde{\pi}$ from which we are going to sample during the run of the algorithm. For example, in Importance Sampling, Subset Simulation and Adaptive Linked Importance Sampling we sample from the important sampling density π_{is}, family of conditional distributions $\pi(\cdot|F)$ and family of intermediate distributions π_α respectively. The second step is the development of the Markov chain Monte Carlo (MCMC) strategy for sampling from the above specified artificial PDF(s). Usually different variations of the Metropolis–Hastings (MH) algorithm are used. Schematically the structure of advanced simulation algorithm is shown in Fig. 4.1.

The main objective of this Chapter is to develop a novel effective MCMC algorithm for sampling from complex high-dimensional distributions. This Chapter is based on paper [34].

Our starting point is the standard Metropilis-Hastings algorithm [25, 14]. It has been shown by Au and Beck in [1] the standard MH algorithm does generally not work in high dimensions, since it leads to extremely frequent repeated samples. A geometric understanding of why this is true was given in section 2.4, see also [17, 18, 29]. In order to overcome this deficiency of the MH algorithm one can use the modified Metropolis–Hastings algorithm



Advanced Simulation Method
1. Specify artificial PDF(s) $\tilde{\pi}$
2. Sample from $\tilde{\pi}$
Importance Sampling: $\tilde{\pi} = \pi_{is}$
Subset Simulation: $\tilde{\pi} = \pi(\cdot|F)$
ALIS: $\tilde{\pi} = \pi_{\alpha}$
MCMC
(Metropolis-Hastings algoritm)
Reliability Problem

Figure 4.1: Structure of a general advanced simulation method.

(MMH) proposed in [1] for sampling from high-dimensional distributions. It should be emphasized that MMH algorithm is suitable only for sampling from very specific distributions, namely, from the conditional distributions $\pi(\cdot|F)$, where unconditional PDF π can be factorized into a product of easy to sample from one-dimensional distributions. To the best of our knowledge, at the moment there does not exist any efficient algorithm for sampling from an arbitrary high-dimensional distribution.

Another variation of the MH algorithm, called Metropolis–Hastings algorithm with delayed rejection (MHDR) has been proposed by Tierney and Mira in [33]. The key idea behind the MHDR algorithm is that when a Markov chain remains in the same state for some time, the estimate obtained by averaging along the chain path becomes less efficient. For the MH algorithm this happens when a candidate generated from the proposal distribution is rejected. Therefore, we can improve the MH algorithm by reducing the number of rejected candidates. A way to achieve this goal is the following: whenever a candidate is rejected, instead of taking the current state of a Markov chain as its new state, as the case in the standard MH algorithm, we propose a new candidate. Of course, the acceptance probability of the new candidate has to be adjusted in order to keep the distribution invariant.

To address high-dimensional reliability problems, in this Chapter we combine the ideas of

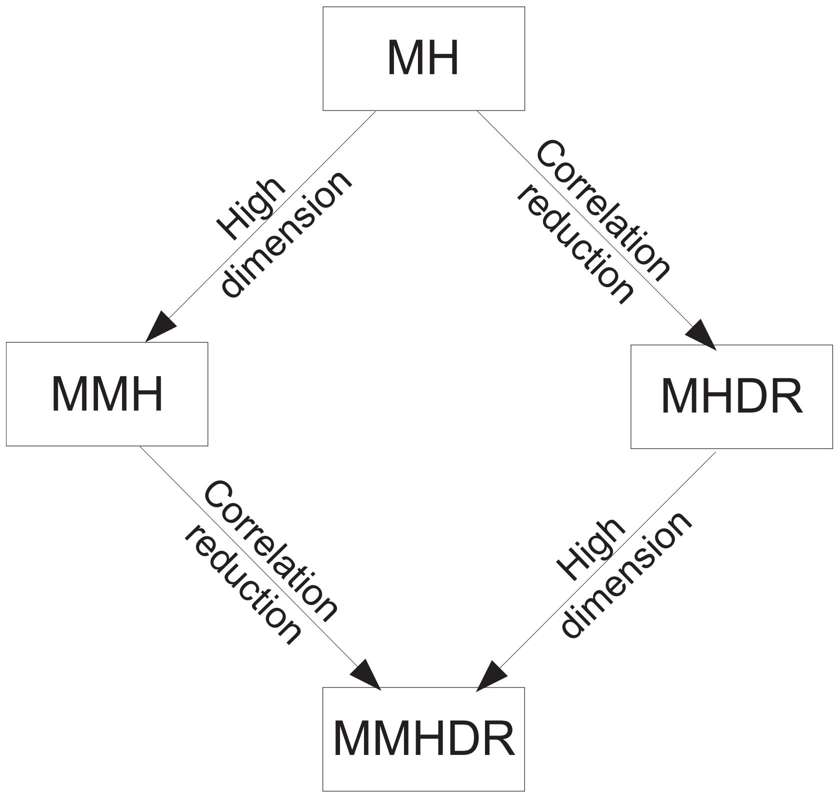


Figure 4.2: Modifications of the standard Metropolis–Hastings algorithm.

both the MMH and MHDR algorithms. As a result we obtain an efficient algorithm, called Modified Metropolis–Hastings algorithm with delayed rejection (MMHDR), for sampling from high-dimensional conditional distributions.

Different variations of the standard MH algorithm are schematically shown in Fig. 4.2.

4.1 Modifications of the Metropolis–Hastings algorithm

Throughout this Chapter all the variations of the MH algorithm are discussed in the context of high-dimensional conditional distributions.

4.1.1 Standard Metropolis–Hastings algorithm

The MH algorithm is the most common MCMC method for sampling from a probability distribution that is difficult to sample from directly. The algorithm is named after Nicholas Metropolis, who proposed it in 1953, [25] for the specific case of the Boltzmann distribution, and W. Keith Hastings, who generalized it in 1970, [14].

$$\begin{tikzcd}
& S(\cdot|x_0) \arrow[d, squiggly] & \\
& \xi \arrow[dl, "a"'] \arrow[dr, "1\text{-}a"] & \\
x_1 = \xi & & x_1 = x_0
\end{tikzcd}$$

Figure 4.3: Standard Metropolis–Hastings algorithm.

In this method samples are simulated as the states of a Markov chain, which has the target distribution, i.e., the distribution we want to sample from, as its equilibrium distribution. Let the target distribution be $\pi(\cdot|F) = \pi(\cdot)I_F(\cdot)/Z$, where $Z = P(F)$ is a normalizing constant; x_0 be the current state of the Markov chain; and $S(\cdot|x_0)$, called proposal PDF, be an N-dimensional PDF depended on x_0. Then the MH update $x_0 \rightarrow x_1$ of the Markov chain works as follows:

1) Simulate ξ according to $S(\cdot|x_0)$,

2) Compute the acceptance probability

$$a(x_0, \xi) = \min\left\{1, \frac{\pi(\xi)S(x_0|\xi)}{\pi(x_0)S(\xi|x_0)} I_F(\xi)\right\}, \tag{4.1}$$

3) Accept or reject ξ by setting

$$x_1 = \begin{cases} \xi, & \text{with prob. } a(x_0, \xi); \\ x_0, & \text{with prob. } 1 - a(x_0, \xi). \end{cases} \tag{4.2}$$

One can show that such update leaves $\pi(\cdot|F)$ invariant, i.e. if x_0 is distributed according to $\pi(\cdot|F)$, then so is x_1:

$$x_0 \sim \pi(\cdot|F) \Rightarrow x_1 \sim \pi(\cdot|F). \tag{4.3}$$

Hence the chain will eventually converge to $\pi(\cdot|F)$ as its equilibrium distribution. Note, that the MH algorithm does not require information about the normalizing constant Z. Assuming a symmetric proposal distribution, i.e. $S(x|y) = S(y|x)$, one obtains the original Metropolis algorithm, [25]. The MH update is schematically shown in the Fig. 4.3.

4.1.2 Metropolis–Hastings algorithm with delayed rejection

Rejecting the candidate state ξ with probability $1 - a(x_0, \xi)$ in (4.2) is necessary for keeping the target distribution $\pi(\cdot|F)$ invariant under the MH update. However, remaining in the current state x_0 for some time affects the quality of the corresponding Markov chain by increasing the autocorrelation between its states and, therefore, it reduces the efficiency of any simulation method that uses the standard MH algorithm. Thus, reducing the number of rejected candidate states will improve the standard MH algorithm.

Metropolis-Hastings algorithm with delayed rejection (MHDR), proposed in Tierney and Mira in [33], allows to achieve this goal: when a reject decision in (4.2) is taken, instead of getting a repeated sample, we generate a second candidate state using a different proposal distribution and accept or reject it based on a suitably computed probability. So, the Markov chain update $x_0 \to x_1$ in the MHDR algorithm when dealing with conditional distribution $\pi(\cdot|F)$ works as follows:

1) Simulate ξ_1 according to $S_1(\cdot|x_0)$,

2) Compute the acceptance probability

$$a_1(x_0, \xi_1) = \min\left\{1, \frac{\pi(\xi_1)S_1(x_0|\xi_1)}{\pi(x_0)S_1(\xi_1|x_0)} I_F(\xi_1)\right\}, \tag{4.4}$$

3) Accept or reject ξ_1 by setting

$$x_1 = \begin{cases} \xi_1, & \text{with prob. } a_1(x_0, \xi_1); \\ \text{go to step 4)}, & \text{with prob. } 1 - a_1(x_0, \xi_1). \end{cases} \tag{4.5}$$

4) Simulate ξ_2 according to $S_2(\cdot|x_0, \xi_1)$,

5) Compute the acceptance probability

$$a_2(x_0, \xi_1, \xi_2) = \min\left\{1, \frac{\pi(\xi_2)S_1(\xi_1|\xi_2)S_2(x_0|\xi_2, \xi_1)(1 - a_1(\xi_2, \xi_1))}{\pi(x_0)S_1(\xi_1|x_0)S_2(\xi_2|x_0, \xi_1)(1 - a_1(x_0, \xi_1))} I_F(\xi_2)\right\}, \tag{4.6}$$

6) Accept or reject ξ_2 by setting

$$x_1 = \begin{cases} \xi_2, & \text{with prob. } a_2(x_0, \xi_1, \xi_2); \\ x_0, & \text{with prob. } 1 - a_2(x_0, \xi_1, \xi_2). \end{cases} \tag{4.7}$$

$$\begin{tikzcd}
& S_1(\cdot|x_0) \arrow[d, squiggly] & & & \\
& \xi_1 \arrow[dl, "a_1"'] \arrow[dr, "1-a_1"] & & & \\
x_1 = \xi_1 & & S_2(\cdot|x_0, \xi_1) \arrow[d, squiggly] & & \\
& & \xi_2 \arrow[dl, "a_2"'] \arrow[dr, "1-a_2"] & & \\
& x_1 = \xi_2 & & x_1 = x_0 &
\end{tikzcd}$$

Figure 4.4: Metropolis–Hastings algorithm with delayed rejection.

An interesting feature of this algorithm is that the proposal distribution S_2 at the second stage is allowed to depend on the rejected candidate ξ_1 as well as on the current state x_0 of the chain. Allowing the proposal PDF S_2 to use information about previously rejected candidate does not destroy the Markovian property of the sampler. So all the asymptotic Markov chain theory used for the standard MH algorithm can be used for the MHDR method as well. The MHDR update is schematically shown in Fig. 4.4.

Whether MHDR algorithm is useful depends on whether the reduction in variance achieved compensates for the additional computational cost.

4.1.3 Modified Metropolis–Hastings algorithm

The standard MH algorithm does not generally work in high dimensions meaning that with extremely high probability the update of the Markov chain leads to the repeated sample, $x_1 = x_0$, see [1, 17, 18, 29]. Clearly, the MHDR update has the same problem. Thus, a Markov chain of practically meaningful length constructed in high dimensions having applied either MH or MHDR algorithm may consist of as few as a single sample. This renders simulation methods, such as Subset simulation, practically inapplicable.

The modified Metropolis–Hastings algorithm (MMH) was developed in [1] especially for sampling from high-dimensional conditional distributions. The MMH algorithm differs from the standard MH algorithm in the way the candidate state ξ is generated. Instead of using an N-dimensional proposal PDF S to directly obtain the candidate state ξ, in the MMH

$$\begin{tikzcd}
\cdots & S^j(\cdot|x_0^j) \arrow[d, squiggly] & \cdots \\
& \hat{\xi}^j \arrow[dl, "a^j"'] \arrow[dr, "1\text{-}a^j"] & \\
(\xi^1,\ldots,\xi^j=\hat{\xi}^j & {} \arrow[d] & \xi^j=x_0^j,\ldots,\xi^N) \\
& F \arrow[dl, "\text{yes}"'] \arrow[dr, "\text{no}"] & \\
x_1=\xi & & x_1=x_0
\end{tikzcd}$$

Figure 4.5: Modified Metropolis–Hastings algorithm.

algorithm a sequence of one-dimensional proposals $S^j(\cdot|x_0^j), j = 1 \ldots N$ is used. Namely, each coordinate ξ^j of the candidate state is generated separately using a one-dimensional proposal distribution $S^j(\cdot|x_0^j)$ depended on the jth coordinate x_0^j of the current state. Finally, we check whether the generated candidate belongs to the failure domain or not. Thus, the MMH update of the Markov chain works as follows:

1) Generate candidate state ξ:

   For each $j = 1 \ldots N$

   1a) Simulate $\hat{\xi}^j$ according to $S^j(\cdot|x_0^j)$,

   1b) Compute the acceptance probability

$$a^j(x_0^j, \hat{\xi}^j) = \min\left\{1, \frac{\pi_j(\hat{\xi}^j)S^j(x_0^j|\hat{\xi}^j)}{\pi_j(x_0^j)S^j(\hat{\xi}^j|x_0^j)}\right\}, \tag{4.8}$$

   1c) Accept or reject $\hat{\xi}^j$ by setting

$$\xi^j = \begin{cases} \hat{\xi}^j, & \text{with prob. } a^j(x_0^j, \hat{\xi}^j); \\ x_0^j, & \text{with prob. } 1 - a^j(x_0^j, \hat{\xi}^j). \end{cases} \tag{4.9}$$

2) Accept or reject ξ by setting

$$x_1 = \begin{cases} \xi, & \text{if } \xi \in F; \\ x_0, & \text{if } \xi \notin F. \end{cases} \tag{4.10}$$

It can be easily seen that the MMH algorithm overcomes the deficiency of standard MH algorithm by producing distinct Markov chain states, rather than repeated samples [1, 18]. The MMH update is schematically shown in Fig. 4.5.

4.1.4 Modified Metropolis–Hastings algorithm with delayed rejection

Here we propose a new MCMC method, called modified Metropolis–Hastings algorithm with delayed rejection (MMHDR), which combines the ideas of both MMH and MHDR algorithms.

Let $\xi_1 = (\xi_1^1, \dots, \xi_1^N)$ be a candidate state generated during the MMH update. Divide the set of all indexes $I = \{1, \dots, N\}$ into two disjoint subsets: $I = T \sqcup \overline{T}$, where $T = \{j \in I : \xi_1^j = \hat{\xi}_1^j\}$ and $\overline{T} = \{j \in I : \xi_1^j = x_0^j\}$. So, T is a set of all indexes such that the corresponding coordinates of the current state x_0 were really transformed and $\overline{T}$ is a set of all the remaining indexes.

Following the MMH algorithm after the candidate is generated we need to check whether it belongs to the failure domain or not. In the former case we accept the candidate as a new state of the Markov chain, in the latter case we reject the candidate and get a repeated sample. In the MMHDR algorithm when a reject decision in (4.10) is taken, instead of getting a repeated sample, we generate a second candidate state ξ_2 using a different one-dimensional proposal distribution S_2^j for each $j \in T$ and take $\xi_2^j = x_0^j$ for all $j \in \overline{T}$. In other words, at the second stage we try to update only those coordinates of the current state x_0 that have been transformed at the first stage already. Schematically this is shown in Fig. 4.6.

Finally, when the second candidate is generated, we check whether it belongs to the failure domain, in which case we obtain a really new state of the Markov chain, or not, in which case we still get a repeated sample.

So, the MMHDR update $x_0 \to x_1$ of the Markov chain works as follows:

1) Generate candidate state ξ_1:
   For each $j = 1 \dots N$
   1a) Simulate $\hat{\xi}_1^j$ according to $S_1^j(\cdot|x_0^j)$,

1b) Compute the acceptance probability

$$a_1^j(x_0^j, \hat{\xi}_1^j) = \min\left\{1, \frac{\pi_j(\hat{\xi}_1^j)S_1^j(x_0^j|\hat{\xi}_1^j)}{\pi_j(x_0^j)S_1^j(\hat{\xi}_1^j|x_0^j)}\right\}, \tag{4.11}$$

1c) Accept or reject $\hat{\xi}_1^j$ by setting

$$\xi_1^j = \begin{cases} \hat{\xi}_1^j, j \in T & \text{with prob. } a_1^j(x_0^j, \hat{\xi}_1^j); \\ x_0^j, j \in \overline{T} & \text{with prob. } 1 - a_1^j(x_0^j, \hat{\xi}_1^j). \end{cases} \tag{4.12}$$

2) Accept or reject ξ_1 by setting

$$x_1 = \begin{cases} \xi_1, & \text{if } \xi \in F; \\ \text{go to step 3)} & \text{if } \xi \notin F. \end{cases} \tag{4.13}$$

3) Generate candidate state ξ_2:

For each $j = 1 \dots N$

if $j \in \overline{T}$, set $\xi_2^j = x_0^j$,

if $j \in T$

3a) Simulate $\hat{\xi}_2^j$ according to $S_2^j(\cdot|x_0^j, \xi_1^j)$,

3b) Compute the acceptance probability

$$a_2^j(x_0^j, \xi_1^j, \hat{\xi}_2^j) = \min\left\{1, \frac{\pi_j(\hat{\xi}_2^j)S_1^j(\xi_1^j|\hat{\xi}_2^j)S_2^j(x_0^j|\hat{\xi}_2^j, \xi_1^j)a_1^j(\hat{\xi}_2^j, \xi_1^j)}{\pi_j(x_0^j)S_1^j(\xi_1^j|x_0^j)S_2^j(\hat{\xi}_2|x_0^j, \xi_1^j)a_1^j(x_0^j, \xi_1^j)}\right\}, \tag{4.14}$$

3c) Accept or reject $\hat{\xi}_2^j$ by setting

$$\xi_2^j = \begin{cases} \hat{\xi}_2^j, & \text{with prob. } a_2^j(x_0^j, \xi_1^j, \hat{\xi}_2^j); \\ x_0^j, & \text{with prob. } 1 - a_2^j(x_0^j, \xi_1^j, \hat{\xi}_2^j). \end{cases} \tag{4.15}$$

4) Accept or reject ξ_2 by setting

$$x_1 = \begin{cases} \xi_2, & \text{if } \xi_2 \in F; \\ x_0, & \text{if } \xi_2 \notin F. \end{cases} \tag{4.16}$$

The MMHDR update is schematically shown in the Fig. 4.7. It can be shown that if x_0 is distributed according to $\pi(\cdot|F)$, then so is x_1, i.e. the MMHDR update leaves distribution $\pi(\cdot|F)$ invariant. The reader is referred to the Appendix B for the proof.

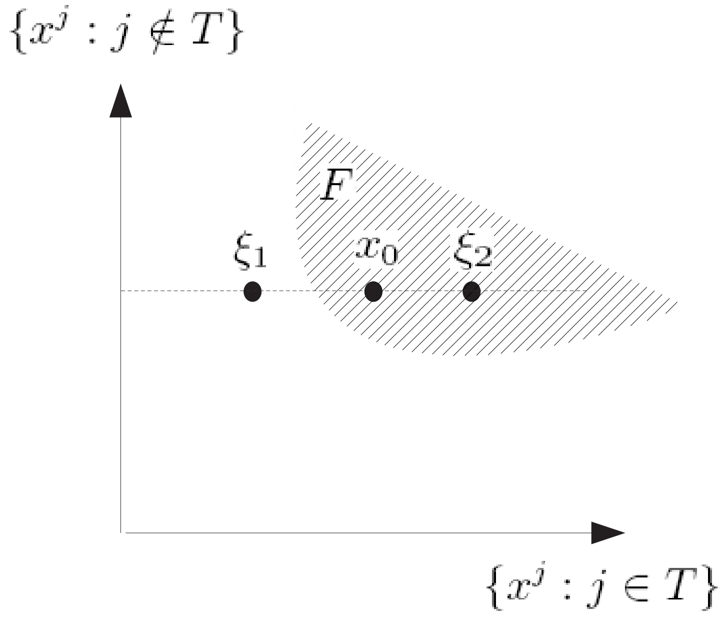


Figure 4.6: MMHDR update at the second stage.

$$\begin{tikzcd}
 & \cdots & S_1^j(\cdot|x_0^j) \arrow[d, squiggly] & \cdots & \\
 & & \hat{\xi}_1^j \arrow[dl, "a_1^j"'] \arrow[dr, "1\text{-}a_1^j"] & & \\
(\xi_1^1, \ldots, & \underset{j \in T}{\xi_1^j = \hat{\xi}_1^j} & {} \arrow[d] & \underset{j \notin T}{\xi_1^j = x_0^j} & , \ldots, \xi_1^N) \\
x_1 = \xi_1 & & F \arrow[ll, "\text{yes}"'] \arrow[r, "\text{no}"] & {} \arrow[d] & \\
\cdots & & & {} \arrow[dl, "j \in T"'] \arrow[dr, "j \notin T"] & \cdots \\
 & & S_2^j(\cdot|x_0^j, \xi_1^j) \arrow[d, squiggly] & & \xi_2^j = x_0^j \\
 & & \hat{\xi}_2^j \arrow[dl, "a_2^j"'] \arrow[dr, "1\text{-}a_2^j"] & & \\
(\xi_2^1, \ldots, & \xi_2^j = \hat{\xi}_2^j & {} \arrow[d] & \xi_2^j = x_0^j & , \ldots, \xi_2^N) \\
x_1 = \xi_2 & & F \arrow[ll, "\text{yes}"'] \arrow[rr, "\text{no}"] & & x_1 = x_0
\end{tikzcd}$$

Figure 4.7: Modified Metropolis–Hastings algorithms with delayed rejection.

The MMHDR algorithm preserves an attractive feature of the MHDR algorithm. Namely, one-dimensional proposal distributions at the second stage are allowed to depend on the corresponding coordinates of the first rejected candidate. The usage of information on the previously rejected candidate potentially can help us to generate a better candidate at the second stage that will be accepted as a new state of the Markov chain. This certainly reduces the overall probability of remaining in the current state if compare with the MMH algorithm, and therefore leads to an improved sampler. However, this improvement is achieved at the expense of an additional computational cost. Whether the MMHDR algorithm is useful for solving reliability problems depends on whether the gained reduction in variance compensates for the additional computational effort.

4.2 Example

To demonstrate the advantage of the MMHDR algorithm over the MMH algorithm we apply Subset Simulation (SS), with both MMHDR and MMH algorithms for evaluating the small failure probability of linear problem in high dimensions.

Let $N = 1000$ be the dimension of the linear problem, $p_F = 10^{-5}$ be the failure probability and the failure domain F is defined as

$$F = \{x \in \mathbb{R}^N : \langle x, e \rangle \geq \beta\}, \tag{4.17}$$

where $e \in \mathbb{R}^N$ is a random unit vector drawn from uniform distribution and $\beta = \Phi^{-1}(1-p_F) = 4.265$ is the reliability index. Here Φ denotes the CDF of the standard normal distribution. Note, that $x^* = e\beta$ is the design point of the failure domain F.

All one-dimensional proposal distributions in both MMH and MMHDR algorithms are set to be normal distributions with unit variance and centered at the corresponding coordinates of the current state:

$$S^j(\cdot|x_0^j) = S_1^j(\cdot|x_0^j) = S_2^j(\cdot|x_0^j, \xi_1^j) = \mathcal{N}_{x_0^j,1}(\cdot) \tag{4.18}$$

Here, when MMHDR algorithm is used, we do not change the second stage proposal distributions. How to generate a better candidate at the second stage based on the information on the first rejected candidate is in need of additional research. In this example we want just to check which one of the following two strategies is more effective: to have more Markov chains

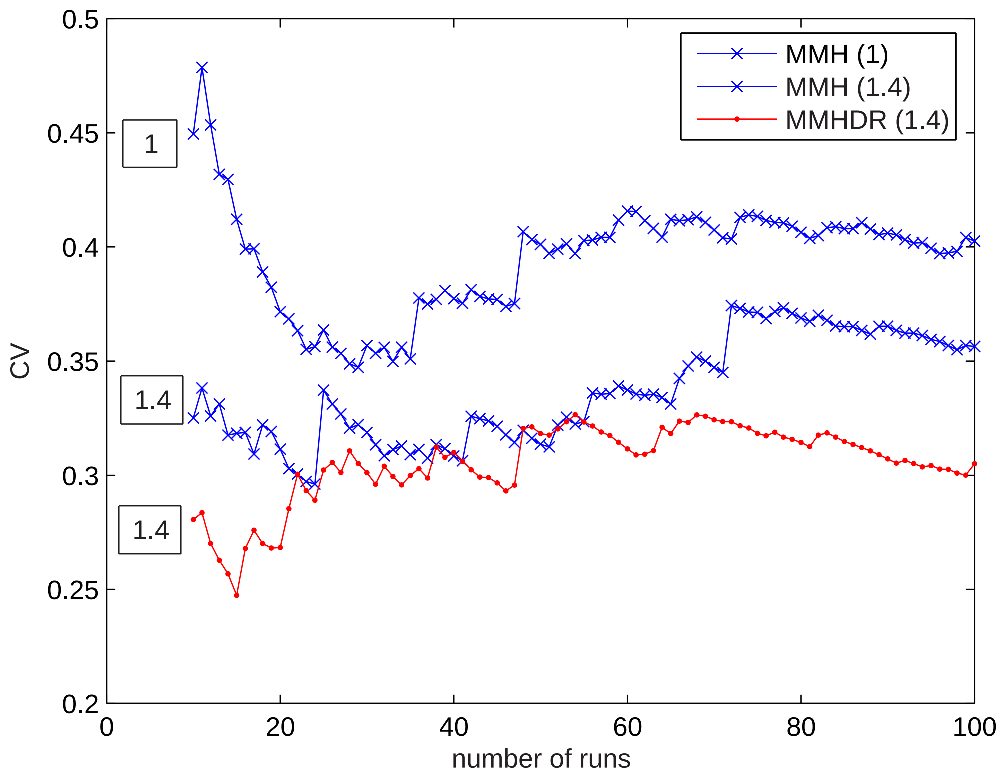


Figure 4.8: The CV of the estimates obtained by Subset simulation with MMH and MMHDR algorithms.

with more correlated states (MMH) or to have fewer Markov chains with less correlated states (MMHDR).

The coefficient of variation (CV) of the failure probability estimates obtained by SS method against the number of runs are given in Fig. 4.8. The curve denoted as MMH(1) corresponds to the SS with MMH algorithm where for each intermediate subset $n = 1000$ samples are used. We refer to the total computational cost of this method, i.e., the mean of the total number of samples used, as 1. The curve denoted as MMHDR(1.4) corresponds to the SS with MMHDR algorithm where $n = 1000$ of MMHDR updates are performed per each intermediate subset. It turns out that the total computational cost of MMHDR(1.4) is 40% higher than MMH(1) and the reduction in CV achieved is about 25% (based on 100 of runs). Finally, the curve MMH(1.4) corresponds to the SS with MMH algorithm where for each intermediate subset $n = 1450$ samples are used. The total computational cost of MMH(1.4) is the same as for MMHDR(1.4), i.e. 40% higher than for MMH(1). However, the the reduction in CV achieved with MMH(1.4) compared to MMH(1) is about 11% only.

So, in this example SS with MMHDR algorithm clearly outperforms SS with MMH algorithm. In other words, “quality” (fewer Markov chains with less correlated states) defeats “quantity” (more Markov chains with more correlated states).

Chapter 5

Horseracing Simulation

A horse is dangerous at both ends
and uncomfortable in the middle.

Ian Fleming

In this Chapter we propose a novel advanced stochastic simulation algorithm for solving high-dimensional reliability problems, called Horseracing Simulation. The main idea behind Horseracing Simulation is as follows. Although the reliability problem itself is high-dimensional, the limit-state function maps this high-dimensional parameter space into a one-dimensional real line. This mapping transforms a high-dimensional random parameter vector, which represents the input load, into a random variable with unknown distribution, which represents the structure response. It turns out, that the corresponding unknown cumulative distribution function (CDF) of this random variable can be accurately approximated by the empirical CDFs constructed from specially designed samples.

5.1 Basic idea of Horseracing Simulation

Let us start with the discussion of the following auxiliary problem. Let z be a continuous random variable with PDF f and CDF F, which are unknown. Suppose, that we can draw samples from the distribution f. Our goal is, trying to use as few samples as possible, to approximate F in some neighbourhood of a given point $z^* \in \mathbb{R}$. If the point z^* is not very far from the median $\tilde{z}$, then we can just draw Monte Carlo samples from f and use the empirical CDF $F^{(0)}$, constructed based on these samples, as an approximation of F, see Fig. 5.1. However, if the probability $p = 1 - F(z^*)$ is very small $p \ll 1$, then the Monte Carlo method

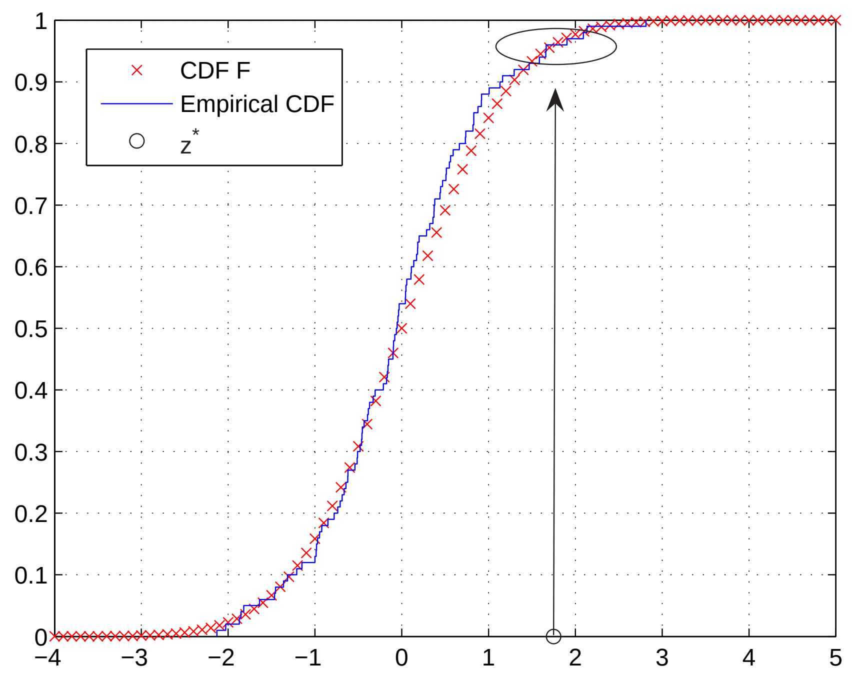


Figure 5.1: Approximation of the standard Gaussian CDF by empirical CDF, constructed based on Monte Carlo samples.

will require a lot of samples in order to get some information about F in the neighbourhood of z^*. Therefore, since it is essential to minimize the number of samples, the direct Monte Carlo method is not applicable.

Assume now, that we can propagate our Monte Carlo samples towards the important region (neighbourhood of z^*). Namely, for any sample $z^{(0)} \sim f(z)$ we are able to draw samples from the conditional distribution $f(z|z \geq z^{(0)})$, for any sample $z^{(1)} \sim f(z|z \geq z^{(0)})$ we are able to draw samples from the conditional distribution $f(z|z \geq z^{(1)})$, etc. It can be proven (see Appendix B), that the kth random variable $z^{(k)}$, defined by this process, has PDF

$$f_k(z) = \frac{(-1)^k}{k!} f(z) \left[\log(1 - F(z))\right]^k . \tag{5.1}$$

It is well-known from the importance sampling theory (e.g. see [28]), that if $x_1, \ldots, x_n$ are independently drawn from a trial distribution h and the weight $w_i / \sum_{i=1}^{n} w_i$, where $w_i = f(x_i)/h(x_i)$, is assigned to each x_i, then as $n \to \infty$ this approach produces a sample that is approximately distributed according to f. In standard terminology, a trial distribution h

and weights w_i and $w_i/\sum_{i=1}^n w_i$ are called “the importance distribution”, “the importance weight” and “normalized importance weight” correspondingly.

So, if $z_1^{(k)}, \dots, z_n^{(k)}$ are independently distributed according to the distribution f_k, then the weighted samples $(z_1^{(k)}, w_1^{(k)}), \dots, (z_n^{(k)}, w_n^{(k)})$, where

$$w_i^{(k)} \propto \frac{f(z_i^{(k)})}{f_k(z_i^{(k)})} \propto \frac{1}{\left[\log(1-F(z_i^{(k)}))\right]^k}, \tag{5.2}$$

are approximately distributed according to f. Therefore, we can construct the empirical CDF $F^{(k)}$ based on $\{(z_i^{(k)}, w_i^{(k)})\}_{i=1}^n$ and use it for updating of the empirical CDF $F^{(0)}$, which was constructed based on Monte Carlo samples $\{z_i^{(0)}\}_{i=1}^n$. Note, that the importance weights in (5.2) depend explicitly only on the CDF F that we want to approximate and do not depend on the unknown PDF f.

The above discussion suggests the following scheme of an algorithm (which is a prototype of the Horseracing Simulation algorithm) for the approximation of F.

Horseracing Simulation Scheme

```
I.   Sample $z_1^{(0)}, \dots, z_n^{(0)}$ from $f_0 = f$,
     Set $k = 0$.

II.  Construct the empirical CDF $F^{(k)}$ based on $\{z_i^{(k)}\}_{i=1}^n$.

          While the stopping criterion $C(z^*)$ is not fulfilled do:

III. Sample $z_i^{(k+1)}$ from $f_0(z|z \geq z_i^{(k)})$ for each $i = 1 \dots n$.

IV.  Construct the empirical CDF $G^{(k+1)}$ based on $\{(z_i^{(k+1)}, w_i^{(k+1)})\}_{i=1}^n$.

V.   Update CDF $F^{(k)}$ to $F^{(k+1)}$, $(F^{(k)}, G^{(k+1)}) \rightsquigarrow F^{(k+1)}$,
     Set k=k+1.
```

Of course the steps of this algorithm should be specified and the the stopping criterion $C(z^*)$ should be properly chosen. Then we can naturally expect that in some neighbourhood of z^*

$$F^{(k)} \approx F. \tag{5.3}$$

Before we explain how this scheme can help to solve the reliability problem, let us strike some life into notation and explain the name origin for this algorithm. One can think of

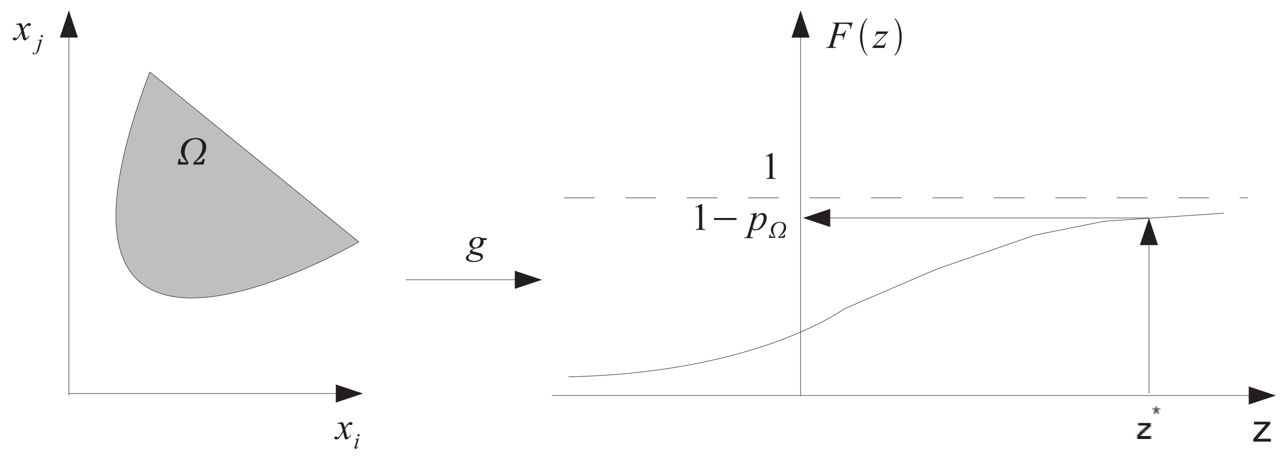


Figure 5.2: Dimension reduction induced by the limit-state function.

$z_1, \ldots, z_n$ as about horses participating in a race, where $z_i^{(k)}$ denotes the position of the ith horse at time instant k. The race is over when the finishing rule given by $C(z^*)$ is fulfilled, for example, when one of the horses ($z_i^{(k)}$) reaches the finish line ($z_i^{(k)} \geq z^*$).

Let us now relate the Horserace Simulation scheme with the reliability problem. Recall, that the structural reliability problem is to compute the probability of failure, that is given by the following expression:

$$p_\Omega = P(x \in \Omega) = \int\limits_\Omega \pi_0(x)dx, \tag{5.4}$$

where $x \in \mathbb{R}^N$ is a random vector with joint PDF π_0, which represents the uncertain input load or other uncertain model parameters, and $\Omega \subset \mathbb{R}^N$ is the failure domain[1], i.e. the set of inputs that lead to the exceedance of some prescribed critical threshold $z^* \in \mathbb{R}_+$:

$$\Omega = \{x \in \mathbb{R}^N |\, g(x) > z^*\}. \tag{5.5}$$

Here $g : \mathbb{R}^N \to \mathbb{R}_+$ is the limit-state function, which maps the high-dimensional parameter space into a one-dimensional real line. This mapping transforms the high-dimensional random parameter vector x into a random variable $z = g(x)$, which represents the structure response. This is schematically shown in Fig. 5.2.

Let f and F be PDF and CDF of z respectively, then the probability of failure in (5.4)

[1]In the previous Chapters the symbol F was used to denote a failure domain. In this Chapter we shall use Ω instead, since F is the standard notation for a CDF, which will be used frequently.

can be rewritten as follows:

$$p_\Omega = \int_{z^*}^{\infty} f(z)dz = 1 - F(z^*). \tag{5.6}$$

If the limit-state function g is continuously differentiable, then

$$f(z) = \int_{\{x:\, z=g(x)\}} \frac{\pi_0(x)}{\|\nabla g(x)\|}\, dV, \tag{5.7}$$

where $\nabla g(x) = (\partial g/\partial x_1, \dots, \partial g/\partial x_n)$ is the gradient of the limit-state function, integration is taking over the $(N-1)$-dimensional surface $\{x :\ z = g(x)\}$ and the differential volume dV must be replaced by a parametrization of this surface for a particular calculation. Although for any given x we can calculate the value $g(x)$, we cannot obtain any other information such as explicit formula, gradient, and so on (see condition (ii) in Chapter 1). As a consequence, neither f nor F is known. Hence, the limit-state function allows us to shift the difficulty of the reliability problem from geometry (in (5.4) we have to calculate the high-dimensional integral over a complex domain that is only known implicitly) to probability ((5.6) is just a one-dimensional integral, yet of an unknown function), see Fig. 5.2.

Thus, in order to use (5.6) for failure probability estimation one should find an approximation of the CDF F of the random variable z. In the rest part of this Chapter we shall show how the Horseracing Simulation scheme can be successfully used for this purpose.

5.2 Implementation issues

In this section we discuss the details of the proposed Horseracing Simulation scheme in the context of the reliability problem.

5.2.1 Sampling

According to step I of the scheme, we have to sample from the distribution $f_0 = f$. In other words, we have to define the initial positions of the horses participating in a race. Although the distribution f_0 is unknown, it is very simple to get a sample from it. Namely, Monte Carlo samples $x_1^{(0)}, \dots, x_n^{(n)} \sim \pi_0$, being transformed by the limit-state function, will automatically provide independent samples from f_0:

$$z_1^{(0)} = g(x_1^{(0)}), \dots, z_n^{(0)} = g(x_n^{(n)}) \sim f_0. \tag{5.8}$$

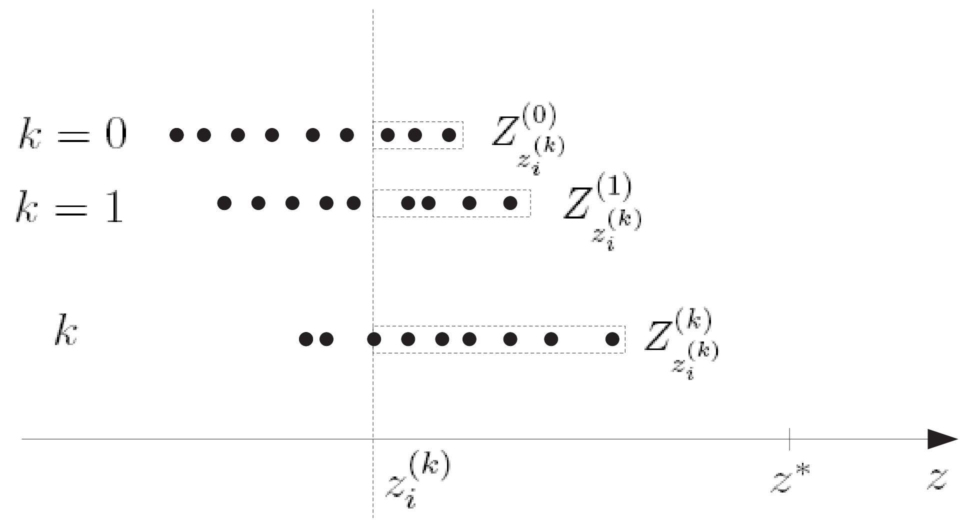


Figure 5.3: Conditionally distributed samples.

Based on these samples F^0 is constructed, according to the step II.

Next, according to step III of the Horseracing Simulation scheme, we have to sample from $f_0(z|z \geq z_i^{(k)})$, where $z_i^{(k)} \sim f_k$, i.e., we need to find the position of the ith horse at time $k+1$. The main idea is the same as in step 1: to sample in the high-dimensional parameter space and then apply transformation, generated by the limit-state function.

Let $x_i^{(k)}$ be one of the previously generated samples, that corresponds to $z_i^{(k)}$, i.e., $g(x_i^{(k)}) = z_i^{(k)}$. Define the subset $\Omega_{z_i^{(k)}} \subset \mathbb{R}^N$ as follows:

$$\Omega_{z_i^{(k)}} = \{x \in \mathbb{R}^N \mid g(x) \geq z_i^{(k)}\}. \tag{5.9}$$

Note, that $x_i^{(k)}$ belongs to the boundary of $\Omega_{z_i^{(k)}}$, i.e. $x_i^{(k)} \in \partial\Omega_{z_i^{(k)}}$. It is clear, that if x is sampled from the conditional distribution $\pi_0(x|x \in \Omega_{z_i^{(k)}})$, then $z = g(x)$ is automatically distributed according to $f_0(z|z \geq z_i^{(k)})$. So, the problem of sampling from $f_0(z|z \geq z_i^{(k)})$ reduces to the sampling from $\pi_0(x|x \in \Omega_{z_i^{(k)}})$. It turns out, that the latter task can be done by using the modified Metropolis-Hastings (MMH) algorithm.

Let $\mathbb{R}_{z_i^{(k)}}$ denote the half-line in front of $z_i^{(k)}$,

$$\mathbb{R}_{z_i^{(k)}} = g(\Omega_{z_i^{(k)}}) = \{z \in \mathbb{R} \mid z \geq z_i^{(k)}\}, \tag{5.10}$$

and $Z^{(t)}_{z_i^{(k)}}$ denote the set of all horse positions at time t, which are in front of $z_i^{(k)}$:

$$Z^{(t)}_{z_i^{(k)}} = \{z_j^{(t)}\}_{j=1}^n \cap \mathbb{R}_{z_i^{(k)}} = \{z_j^{(t)} \mid z_j^{(t)} \geq z_i^{(k)}, j = 1, \dots, n\}, \;\; t = 1, \dots, k. \tag{5.11}$$

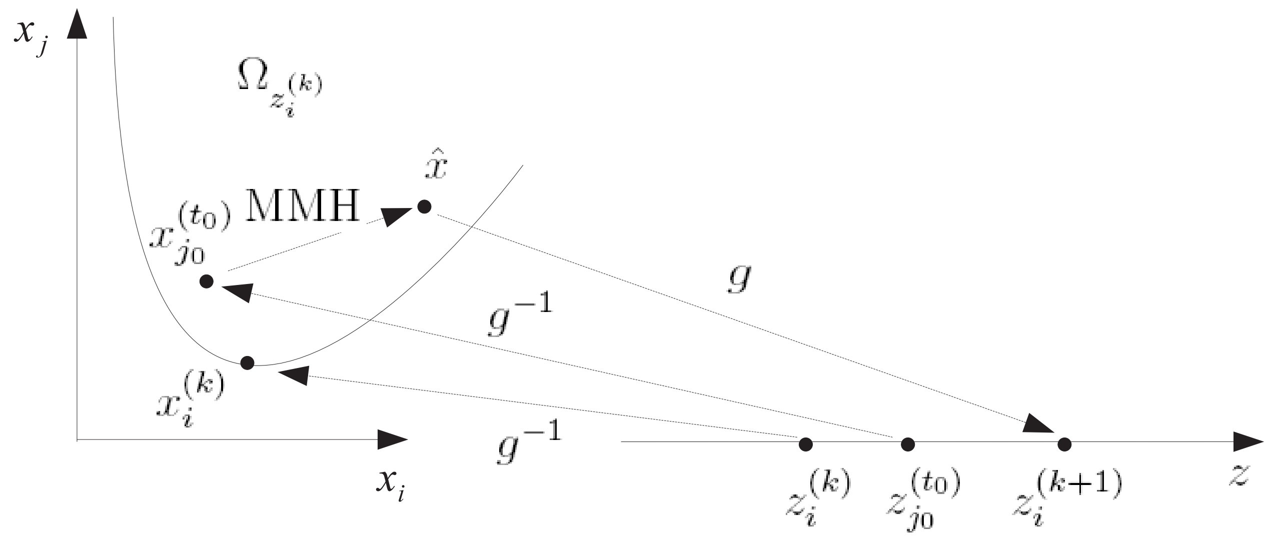


Figure 5.4: Sampling algorithm.

All samples from $Z^{(t)}_{z_i^{(k)}}$ are distributed according to the conditional distribution $f_t(z|z \geq z_i^{(k)})$. Therefore, the weighted samples $\{(z_j^{(t)}, w_j^{(t)}) \mid z_j^{(t)} \in Z^{(t)}_{z_i^{(k)}}\}$, where

$$w_j^{(t)} \propto \frac{f_0(z_j^{(t)})}{f_t(z_j^{(t)})} \propto \frac{1}{\left[\log(1 - F(z_j^{(t)}))\right]^t}, \tag{5.12}$$

are approximately distributed according to the conditional distribution $f_0(z|z \geq z_i^{(k)})$. Note, that instead of the unknown CDF F in (5.12) we can use its approximation $F^{(k)}$, obtained in step V (or in step II) of the scheme. So, at time k we have k sets of weighted samples approximately drawn from the conditional distribution $f_0(z|z \geq z_i^{(k)})$. These sets are schematically shown in Fig. 5.3.

The following algorithm can be used for sampling from $f_0(z|z \geq z_i^{(k)})$:

Sampling Algorithm

i. Select $Z^{(t_0)}_{z_i^{(k)}}$ from $Z^{(t)}_{z_i^{(k)}}$, $t = 1, \ldots, k$ according to their sample sizes $\chi^{(t)}_{z_i^{(k)}} = \#(Z^{(t)}_{z_i^{(k)}})$.

ii. Select $z_{j_0}^{(t_0)}$ from $Z^{(t_0)}_{z_i^{(k)}}$ according to the weights given by (5.12).

iii. Take previously generated sample $x_{j_0}^{(t_0)}$, that corresponds to $z_{j_0}^{(t_0)}$, i.e. $g(x_{j_0}^{(t_0)}) = z_{j_0}^{(t_0)}$, and perform the MMH update $x_{j_0}^{(t_0)} \to \hat{x}$ with invariant distribution $\pi_0(x|x \in \Omega_{z_i^{(k)}})$.

iv. Set $z_i^{(k+1)} = g(\hat{x})$.

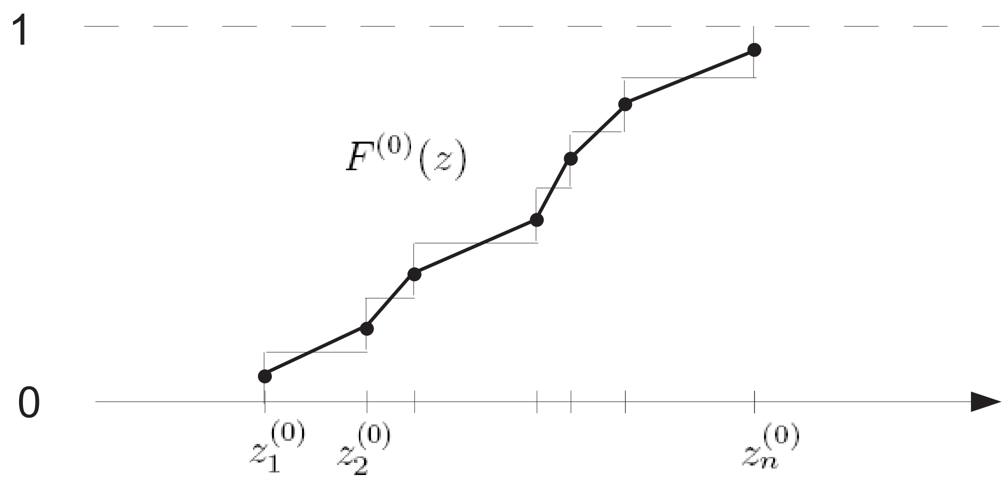


Figure 5.5: The zeroth approximation.

The way we select $z_{j_0}^{(t_0)}$ guarantees that it is distributed according to $f_0(z|z > z_i^{(k)})$. Therefore, the corresponding sample $x_{j_0}^{(t_0)}$ has conditional distribution $\pi_0(x|x \in \Omega_{z_i^{(k)}})$. Since the MMH update preserves the invariant distribution, $\hat{x}$ has distribution $\pi_0(x|x \in \Omega_{z_i^{(k)}})$ as well. From the latter in turn, it follows, that $z_i^{(k+1)} = g(\hat{x})$ is distributed according to $f_0(z|z \geq z_i^{(k)})$. The proposed sampling algorithm is schematically shown in Fig. 5.4.

5.2.2 Construction of the empirical CDF and its updating

In step II of the Horseracing Simulation scheme, we have to construct a zeroth approximation $F^{(0)}$ of the CDF of interest F, based on the Monte Carlo samples $z_1^{(0)}, \ldots, z_n^{(0)} \sim f_0$. For this purpose we use the following piecewise linear approximation:

$$F^{(0)}(z) = \frac{z}{n\left(z_{i+1}^{(0)} - z_i^{(0)}\right)} + \frac{(2i-1)z_{i+1}^{(0)} - (2i+1)z_i^{(0)}}{2n\left(z_{i+1}^{(0)} - z_i^{(0)}\right)}, \text{ for } z \in [z_i^{(0)}, z_{i+1}^{(0)}], \tag{5.13}$$

where $i = 1, \ldots, n-1$. Samples $z_1^{(0)}, \ldots, z_n^{(0)}$ are assumed to be ordered in (5.13), so that $z_1^{(0)} < z_2^{(0)}, \ldots, < z_n^{(0)}$. The zeroth approximation $F^{(0)}$ is shown in Fig. 5.5. Note, that $F^{(0)}$ is not defined for $z \in (-\infty, z_1^{(0)}) \cup (z_n^{(0)}, \infty)$.

Next, in step IV we have to construct the empirical CDF $G^{(k+1)}$ based on the weighted samples $\{(z_i^{(k+1)}, w_i^{(k+1)})\}_{i=1}^n$. If the weights $\{w_i^{(k+1)}\}_{i=1}^n$ are given, then we can define the empirical CDF $G^{(k+1)}$ in a similar way as the zeroth approximation:

$$G^{(k+1)}(z) = \frac{w_i^{(k+1)} + w_{i+1}^{(k+1)}}{2\left(z_{i+1}^{(k+1)} - z_i^{(k+1)}\right)} z + \frac{\left(2\sum_{j=1}^{i} w_j^{(k+1)} - w_i^{(k+1)}\right) z_{i+1}^{(k+1)} - \left(2\sum_{j=1}^{i} w_j^{(k+1)} + w_{i+1}^{(k+1)}\right) z_i^{(k+1)}}{2\left(z_{i+1}^{(k+1)} - z_i^{(k+1)}\right)}, \tag{5.14}$$

for $z \in [z_i^{(k+1)}, z_{i+1}^{(k+1)}]$ and $i = 1, \dots, n-1$. Samples $z_1^{(k+1)}, \dots, z_n^{(k+1)}$ are assumed to be ordered in (5.14), so that $z_1^{(k+1)} < z_2^{(k+1)}, \dots, < z_n^{(k+1)}$. Note, that if all weights are equal, i.e. $w_i^{(k+1)} = 1/n$, then (5.14) becomes (5.13).

In order to use (5.14), the weights $w_1^{(k+1)}, \dots, w_n^{(k+1)}$ should be calculated. Since the samples $z_1^{(k+1)}, \dots, z_n^{(k+1)}$ are approximately distributed according to f_{k+1}, their weights are given by (5.12), where $t = k+1$ and F is replaced by its approximation $F^{(k)}$, obtained in step V (or in step II). However, the approximation $F^{(k)}$, which was constructed using samples $\{(z_1^{(t)}, \dots, z_n^{(t)})\}_{t=1}^{k}$, is defined only for $z \in [\min_i z_i^{(0)}, \max_i z_i^{(k)}]$. So, in order to calculate the weights of the "fastest" horses, i.e. of $z_i^{(k+1)}$ such that $z_i^{(k+1)} > \max_i z_i^{(k)}$, we have to interpolate $F^{(k)}$ on the interval $[\max_i z_i^{(k)}, \max_i z_i^{(k+1)}]$. Thus, the weights $w_1^{(k+1)}, \dots, w_n^{(k+1)}$ are defined as follows:

$$w_i^{(k+1)} \propto \begin{cases} \frac{1}{\left[\log(1-F^{(k)}(z_i^{(k+1)}))\right]^{k+1}}, & \text{if } z_i^{(k+1)} \leq \max_i z_i^{(k)}; \\ \frac{1}{\left[\log(1-F_{int}^{(k)}(z_i^{(k+1)}))\right]^{k+1}}, & \text{if } z_i^{(k+1)} > \max_i z_i^{(k)}, \end{cases} \tag{5.15}$$

where $F_{int}^{(k)}$ is the interpolation of the CDF $F^{(k)}$. According to the notation introduced in the sampling algorithm (section 5.2.1), $\chi_{\max_i z_i^{(k)}}^{(k+1)}$ is the number of samples $\{z_i^{(k+1)}\}_{i=1}^{n}$, that are larger than $\max_i z_i^{(k)}$. Let $z_{j_0}^{(k+1)}$ be the smallest from such samples, $z_{j_0}^{(k+1)} = \min_i\{z \in Z_{\max_i z_i^{(k)}}^{(k+1)}\}$. Then, the interpolation of the CDF $F^{(k)}$ is defined as the piecewise linear function shown in Fig. 5.6. Note, that $\chi_{\max_i z_i^{(k)}}^{(k+1)} = n - j_0 + 1$.

Finally, in step V we need to update the CDF $F^{(k)}$, using new information provided by $G^{(k+1)}$, and construct a new approximation $F^{(k+1)}$ of the CDF F. Suppose for convenience, that samples $z_1^{(t)}, \dots, z_n^{(t)}$ are ordered for each $t = 1, \dots, k+1$:

$$\begin{aligned} & z_1^{(0)} < z_2^{(0)}, \dots, < z_n^{(0)}, \\ & z_1^{(1)} < z_2^{(1)}, \dots, < z_n^{(1)}, \\ & \dots \\ & z_1^{(k+1)} < z_2^{(k+1)}, \dots, < z_n^{(k+1)}. \end{aligned} \tag{5.16}$$

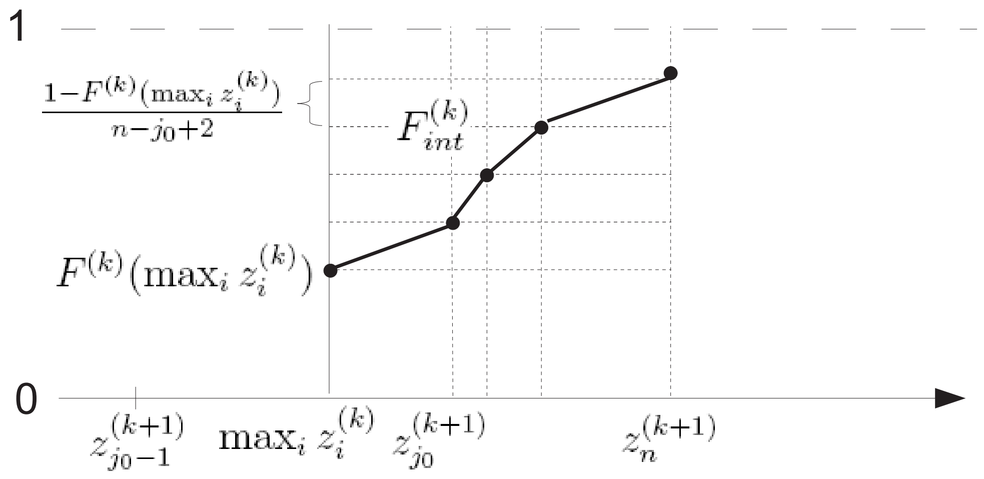


Figure 5.6: The interpolation of $F^{(k)}$.

The CDF $G^{(k+1)}$ is not defined on the interval $[z_1^{(0)}, z_1^{(k+1)})$. This means, that at time $t = k+1$ we do not get any new information about CDF F on this interval relative to the information available at time $t = k$, so $F^{(k+1)}(z) = F^{(k)}(z)$ for $z \in [z_1^{(0)}, z_1^{(k+1)})$. On the interval $[z_1^{(k+1)}, z_n^{(0)}]$, the approximation $F^{(k)}$ is constructed using $k+1$ sets of samples $\{z_i^{(0)}\}_{i=1}^n, \dots, \{z_i^{(k)}\}_{i=1}^n$, while the approximation $G^{(k+1)}$ is based only on one set $\{z_i^{(k+1)}\}_{i=1}^n$. Therefore, it is natural to define a new approximation $F^{(k+1)}(z) = ((k+1)F^{(k)}(z) + G^{(k+1)}(z))/(k+2)$, for $z \in [z_1^{(k+1)}, z_n^{(0)}]$. Using this line of reasoning, we define the new approximation $F^{(k+1)}$ as follows:

$$F^{(k+1)}(z) = \begin{cases} F^{(k)}(z), & \text{for } z \in [z_1^{(0)}, z_1^{(k+1)}); \\ \frac{(k+1)F^{(k)}(z)+G^{(k+1)}(z)}{k+2}, & \text{for } z \in [z_1^{(k+1)}, z_n^{(0)}]; \\ \frac{kF^{(k)}(z)+G^{(k+1)}(z)}{k+1}, & \text{for } z \in (z_n^{(0)}, z_n^{(1)}]; \\ \dots, & \dots \\ \frac{F^{(k)}(z)+G^{(k+1)}(z)}{2}, & \text{for } z \in (z_n^{(k-1)}, z_n^{(k)}]; \\ G^{(k+1)}(z), & \text{for } z \in (z_n^{(k)}, z_n^{(k+1)}]. \end{cases} \tag{5.17}$$

However, since some of the weights in (5.15) are calculated using the interpolation function $F_{int}^{(k)}$, the CDF $G^{(k+1)}$ approximates not exactly F. More precisely, $G^{(k+1)}$ can be decomposed as follows:

$$G^{(k+1)} = G_F^{(k+1)} + G_{int}^{(k+1)}, \tag{5.18}$$

where $G_F^{(k+1)}$ is an approximation of F and $G_{int}^{(k+1)}$ is a perturbation due to the slightly incorrect weights. Therefore, $F^{(k+1)}$, given by (5.17), also approximates not exactly F due to the perturbation term $G_{int}^{(k+1)}$. Note, that the norm $||G_{int}^{(k+1)}||$ is a decreasing function of $\chi^{(k+1)}_{\max_i z_i^{(k)}}$: the less number of samples with weights calculated with $F_{int}^{(k)}$ in (5.15), the less the norm $||G_{int}^{(k+1)}||$. So, if the number of such samples is relatively small, which is the case in applications, we can assume, that the norm $||G_{int}^{(k+1)}||$ is small enough. In order to completely eliminate the influence of $G_{int}^{(k+1)}$, we propose the following iterative updating algorithm:

UPDATING ALGORITHM

```
i. Set $s = 1$, $\varepsilon = 1$,
   Define $H_s$ according to (5.17).

While the error $\varepsilon > \varepsilon_0$ do:

   ii. Recalculate the weights $\{w_i^{(k+1)}\}_{i=1}^n$ using (5.15) with $H_s$ instead of $F^{(k)}$ and $F_{int}^{(k)}$.
   iii. Recalculate $G^{(k+1)}$ using (5.14) with new weights.
   iv. Define $H_{s+1}$ according to (5.17).
   v. Recalculate the error:
      $\varepsilon = \max_{z \in \{z_i^{(t)}\}_{i=1,t=1}^{n,k+1}} \left\{ \left| \frac{H_{s+1}(z) - H_s(z)}{1 - H_s(z)} \right| \right\}$   (5.19)
      Set $s = s + 1$.

End while

vi. Set $F^{(k+1)} = H_s$.
```

The error ε, defined in (5.19), describes the relative change in two successive iterations H_s and H_{s+1}. Note, that ε is more sensitive to the changes in the important region, where H_s is close to 1. When this error is smaller than some prescribed threshold ε_0 (in the further example the value $\varepsilon_0 = 0.01$ is used) we take the last H_s as the new approximation $F^{(k+1)}$ of the CDF F.

5.2.3 Stopping criterion

The stopping criterion $C(z^*)$ plays a very important role in the Horsracing Simulation algorithm. As it was already mentioned in section 5.1, one of the possible choices for $C(z^*)$ is the following rule: the race is over when at least one of the horses ($z_i^{(k)}$) reaches the finish line ($z_i^{(k)} \geq z^*$). The main advantage of this rule is that it allows to obtain the estimate of the failure probability $p_\Omega = 1 - F(z^*)$ with the minimum possible computation effort (as soon as we reach the threshold z^* we stop the algorithm). However, this rule has a serious drawback: the estimate may be very inaccurate. Indeed, if, for instance, $p_\Omega = 0.01$ and we use $n = 100$ samples, then in average 1 out of 100 Monte Carlo samples $z_1^{(0)}, \dots, z_n^{(0)}$ will be a failure sample. In this case, the estimate for the failure probability will have coefficient of variation $\delta = \sqrt{(1 - p_\Omega)/np_\Omega} \approx 1$.

Another natural candidate for the stopping criterion is the following rule: the race is over when $r\%$ of horses reach z^*. The problem with this criterion is the it is difficult to find an optimal ratio r, since it is a trade-off between the accuracy of the estimate and the computational effort. As a matter of fact, the situation is even more complicated, since due to (5.19), the approximation $F^{(k)}$ of the CDF F tends to be more accurate in the region of the largest values of $F^{(k)}$. So, if we go too far beyond z^*, the estimate of the failure probability may degenerate.

In the real-life example considered in this Chapter, we use the following stopping criterion: the race is over when 10 horses reach z^*. This rule preserves the advantage of the first discussed criterion (relatively small computation effort) and at the same time guarantees, that if the Horseracing Simulation reduces to the Monte Carlo algorithm, then the corresponding CV of the failure probability estimate will be approximately 30%. Indeed, if n samples are used and 10 of them turn out to be failure samples, then the CV is $\delta = \sqrt{(1 - p_\Omega)/np_\Omega} \approx 1/\sqrt{10} \approx 0.32$.

The Fig. 5.7 summarizes the Horserace Simulation Scheme and the discussed implementation issues into the Horserace Simulation Algorithm.

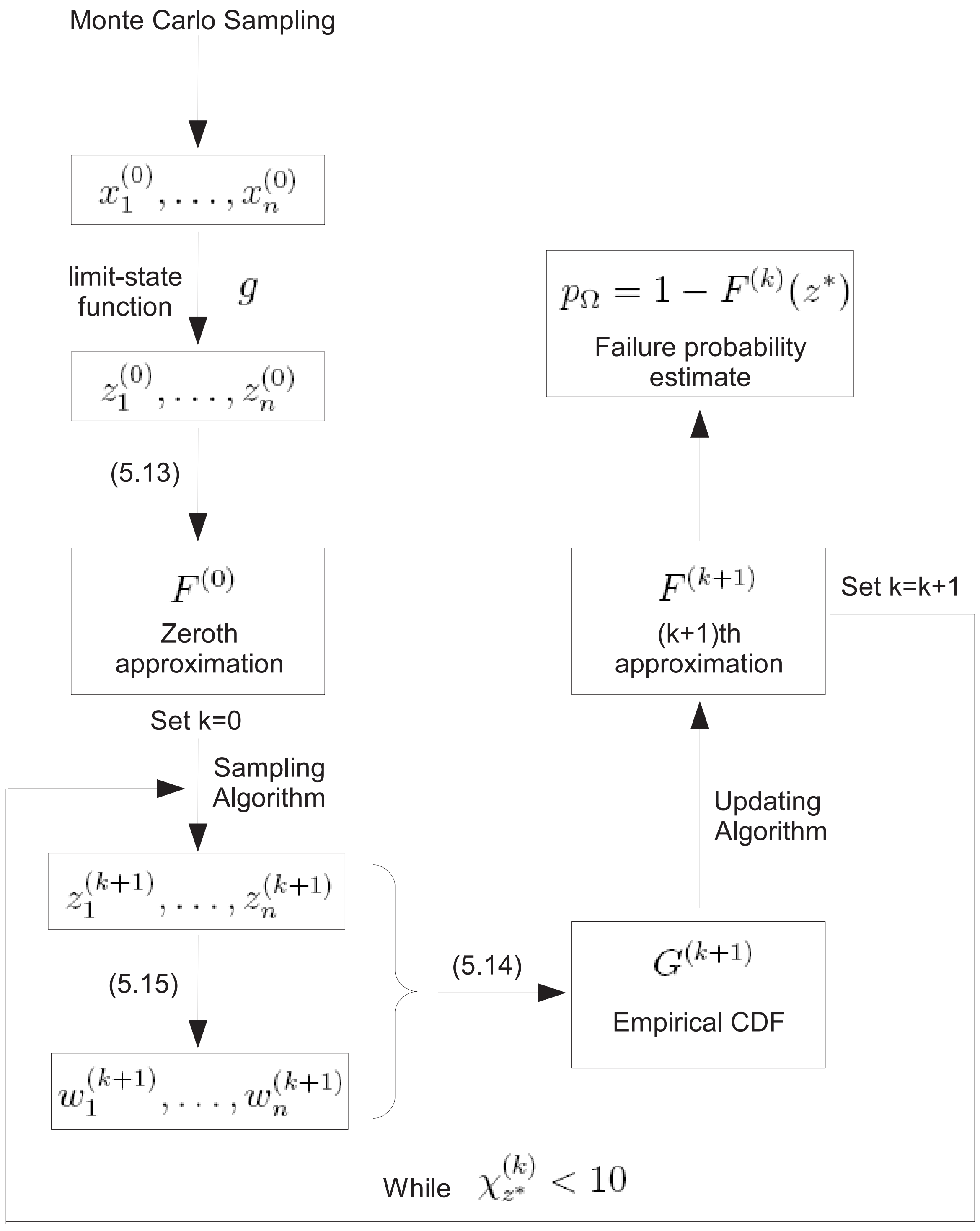

Monte Carlo Sampling
$x_1^{(0)}, \dots, x_n^{(0)}$
limit-state function
g
$z_1^{(0)}, \dots, z_n^{(0)}$
(5.13)
$F^{(0)}$
Zeroth approximation
Set k=0
Sampling Algorithm
$z_1^{(k+1)}, \dots, z_n^{(k+1)}$
(5.15)
$w_1^{(k+1)}, \dots, w_n^{(k+1)}$
(5.14)
$G^{(k+1)}$
Empirical CDF
Updating Algorithm
$F^{(k+1)}$
(k+1)th approximation
Set k=k+1
$p_\Omega = 1 - F^{(k)}(z^*)$
Failure probability estimate
While $\chi_{z^*}^{(k)} < 10$

Figure 5.7: Horeracing Simulation Algorithm.

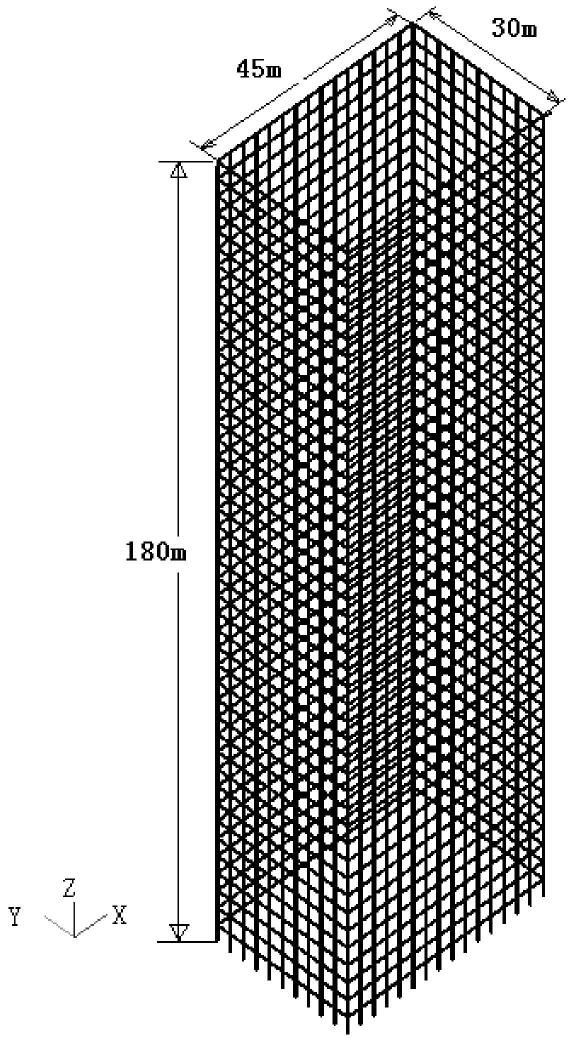


Figure 5.8: CAARC standard tall building model.

5.3 Example

In this section we demonstrate the efficiency and accuracy of the Horseracing Simulation algorithm with a real-life example which is taken from [16].

5.3.1 CAARC standard tall building model

We consider an along-wind excited steel building (Fig. 5.8), which has the same geometric shape as the Commonwealth Advisory Aeronautical Research Council (CAARC) standard tall building model [23]. A 45-story, 10-bay by 15-bay rectangular tubular framework is used to model this building. With story height of 4 m and bay width of 3 m, the building has a total height of 180 m and a rectangular floor with dimension 30 m by 45 m. Each floor

Floor zone	Column members	Beam members
$1 \sim 9$F	W14X550	W30X357
$10 \sim 18$F	W14X500	W30X326
$19 \sim 27$F	W14X370	W30X292
$28 \sim 36$F	W14X257	W30X261
$37 \sim 45$F	W14X159	W30X221

Table 5.1: Design of column members and beam members.

Excitation	Acting height (m)	Acting area (m^2)
$U_1(t)$	24	45×45
$U_2(t)$	68	45×45
$U_3(t)$	112	33.75×45
$U_4(t)$	136	22.5×45
$U_5(t)$	156	22.5×45
$U_6(t)$	176	11.25×45

Table 5.2: Acting heights and acting areas of 6 excitation forces in the discretization scheme.

is assumed to be rigid and has lumped swaying mass of 6.75×10^5 kg and rotational mass moment of inertia of 1.645×10^8 kg.m^2 at the geometric center of the floor. The members of beams and columns have standard AISC steel sections, and the details of the design are presented in Table 5.1. With the above configurations, the established building model has the following first three modal frequencies: 0.197 Hz, 0.251 Hz and 0.422 Hz.

5.3.2 Wind excitation

The along-wind excitation in the Y-direction of the building is considered. In our example the excitation field is discretized using $N_u = 6$ excitation forces $U_1(t), \ldots, U_{N_u}(t)$. The acting heights and acting areas for this discretization scheme are shown in Table 5.2, and the discretized excitation field is schematically shown in Fig. 5.9.

At a given point located at height h_j from the ground, the wind velocity is

$$V_j(t) = \bar{V}_j + v_j(t), \tag{5.20}$$

where $\bar{V}_j$ is the mean wind speed and $v_j(t)$ is the fluctuating component of the wind velocity.

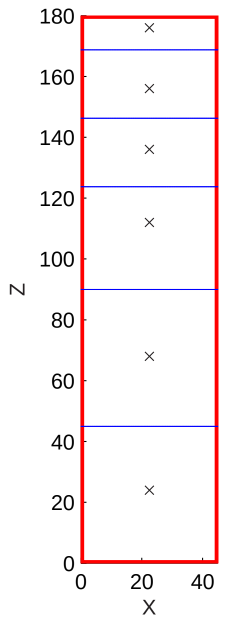


Figure 5.9: Discretized excitation filed.

According to the Hong Kong wind code, the mean wind speed $\bar{V}_j$ (m/s) is given by the power law [32]:

$$\bar{V}_j = 41\left(\frac{h_j}{180}\right)^{0.25}, \quad j = 1, \ldots, N_u. \tag{5.21}$$

The generation of the fluctuating components is carried out by simulation of an N_u-variate zero-mean stationary stochastic vector process $v(t) = [v_1(t), \ldots, v_{N_u}]^T$ using the spectral representation method [30, 31, 7, 13]. In this method, the stochastic vector process is simulated using its cross-power spectral density matrix

$$S^0(\omega) = \begin{pmatrix} S^0_{11}(\omega) & \ldots & S^0_{1N_u}(\omega) \\ \vdots & & \vdots \\ S^0_{N_u1}(\omega) & \ldots & S^0_{N_uN_u}(\omega) \end{pmatrix}. \tag{5.22}$$

We refer the reader to Appendix C for details.

The cross spectral density matrix $S^0(\omega)$ is modeled by formulas proposed by Davenport in [5, 6]. Namely, the power spectral density function $S^0_{jj}(\omega)$ of $v_j(t)$, $j = 1, \ldots, N_u$, is given by

$$S^0_{jj}(\omega) = \frac{\bar{V}_j^2 K^2}{\left(\ln\frac{h_j}{h_0}\right)^2} \frac{8\pi a(\omega)^2}{\omega(1 + a(\omega)^2)^{4/3}}, \tag{5.23}$$

$$a(\omega) = \frac{600\omega}{\pi\bar{V}_{10}}, \tag{5.24}$$

where ω (rad/s) is the frequency, $K = 0.4$ is Von Karman's constant, $h_0 = 0.05$ m is the roughness length, and $\bar{V}_{10} = 19.9$ m/s is the mean wind velocity at height of 10 m. The cross-power spectral density function $S^0_{jk}(\omega)$ of $v_j(t)$ and $v_k(t)$ is given by

$$S^0_{jk}(\omega) = \sqrt{S^0_{jj}(\omega)S^0_{kk}(\omega)}\gamma_{jk}(\omega), \quad j,k = 1,\dots,N_u,\ j \neq k, \tag{5.25}$$

$$\gamma_{jk}(\omega) = \exp\left(-\frac{\omega}{2\pi}\frac{C_h|h_j - h_k|}{0.5(\bar{V}_j + \bar{V}_k)}\right), \tag{5.26}$$

where $\gamma_{jk}(\omega)$ is the coherence function between $v_j(t)$ and $v_k(t)$, and C_h is a constant that can be set equal to 10 for structural design purposes [32].

To perform the generation of the wind velocity fluctuations according to (6.34), the cutoff frequency is taken as $\omega_c = 0.8\pi$ rad/s, so that the ratio r_c of the neglected power spectrum content over the total content, defined in (6.32), is less than 10% for all components $S^0_{jk}(\omega)$, $j,k = 1,\dots,N_u$. The frequency step is set equal to $\triangle\omega = \pi/900$, therefore, the period $T_v = 4\pi/\triangle\omega$ of the fluctuating wind velocity components $v(t)$ is 3600 s.

The wind excitation forces $U_j(t)$, $j = 1,\dots,N_u$ can be expressed as follows:

$$U_j(t) = \frac{1}{2}\rho A_j V_j(t)^2 = \frac{1}{2}\rho A_j(\bar{V}_j + v_j(t))^2, \tag{5.27}$$

where ρ is the air density, taken to be 1.2 kg/m^3, and A_j is the area upon which the discretized force $U_j(t)$ is assumed to act (see Table 5.2).

5.3.3 Geometric description of the failure domain

From the above chosen parameters and (6.34), it follows that the number of standard gaussian random variables involved in the simulation of wind excitation is

$$N = 2 \times N_u \times N_\omega = 2 \times N_u \times \omega_c/\triangle\omega = 8640. \tag{5.28}$$

In other words, the failure domain Ω is a subset of a high-dimensional parameter space $\Omega \subset \mathbb{R}^N$, where $N = 8640$.

In this example we assume that the displacement response $Y(t)$ at the top floor of the building is of interest. The relationship between the response $Y(t)$ and the excitation forces U_j, $j = 1,\dots,N_u$ is given by

$$Y(t) = \sum_{j=1}^{N_u} \int_0^\infty q_j(t,\tau)U_j(\tau)\,d\tau, \tag{5.29}$$

where $q_j(t,\tau)$ is the response function for $Y(t)$ at time t due to a unite impulse excitation for U_j at time τ. We assume that the system starts with zero initial conditions, is time-invariant, i.e., $q_j(t,\tau) = q_j(t-\tau)$, and is causal, i.e., $q_j(t,\tau) \equiv 0$ for $t < \tau$, so that (5.29) can be rewritten as follows:

$$Y(t) = \sum_{j=1}^{N_u} \int_0^t q_j(t-\tau) U_j(\tau)\, d\tau. \tag{5.30}$$

The required impulse response functions $q_1(t), \dots, q_{N_u}(t)$ are obtained through N_u dynamic analyses of the established finite element model of the building using the software SAP 2000.

Summarizing the above discussion, the simulation scheme is shown in Fig. 5.10.

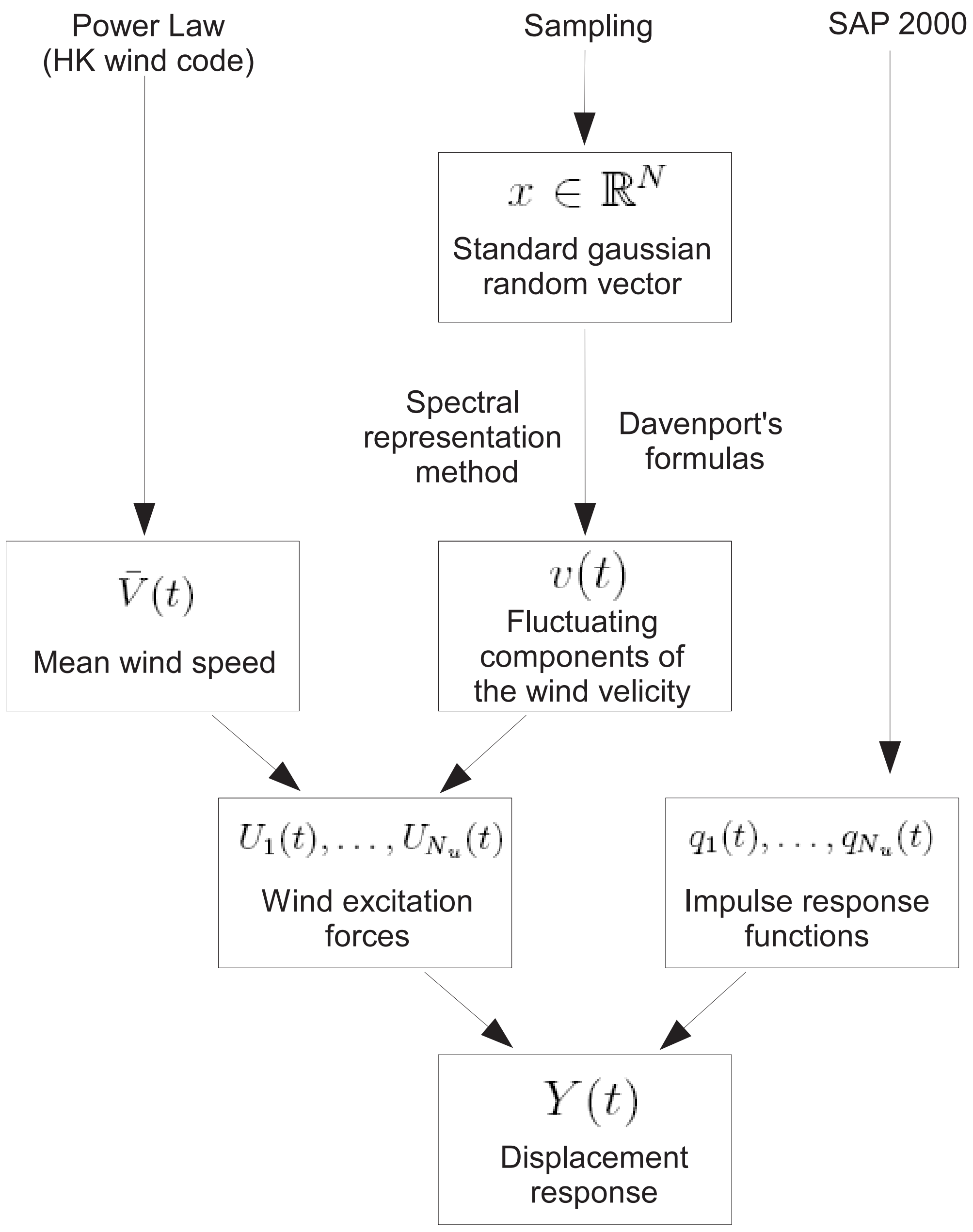
The failure event is defined as the response $Y(t)$ exceeding in magnitude a specified threshold z^* within one hour, i.e., the assumed duration time is $T = 3600$ s. This duration time is conventionally used in wind engineering, for consistence with the duration of actual strong winds. Thus, in the discrete time formulation, where the sampling time interval is chosen to be $\triangle t = 0.01$ s and the number of time instants is $N_t = T/\triangle t = 3.6 \cdot 10^5$, the failure domain $\Omega \subset \mathbb{R}^N$ is defined as follows:

$$\Omega = \bigcup_{i=1}^{N_t} \left\{ x \in \mathbb{R}^N \;:\; |Y(i)| > z^* \right\}. \tag{5.31}$$

So, in the space of standard normal random variables, the failure domain Ω is a union of $2N_t$ elementary failure domains: $\left\{x \in \mathbb{R}^N \;:\; Y(i) > z^*\right\}$ and $\left\{x \in \mathbb{R}^N \;:\; Y(i) < -z^*\right\}$, for $i = 1, \dots, N_t$. The limit-state function is given by

$$g(x) = \max\{|Y(i)|,\ i = 1, \dots, N_t\}. \tag{5.32}$$

For each sample $x \in \mathbb{R}^N$ a dynamic analysis is required in order to evaluate the corresponding value $g(x)$ of the limit-state function. We refer to the total number of such dynamic analysis (or, equivalently, to the total number of limit-state function evaluations) used in a run of an algorithm as the total computation effort of the algorithm.



Power Law (HK wind code)
Sampling
SAP 2000
$x \in \mathbb{R}^N$
Standard gaussian random vector
Spectral representation method
Davenport's formulas
$\bar{V}(t)$
Mean wind speed
$v(t)$
Fluctuating components of the wind velicity
$U_1(t), \ldots, U_{N_u}(t)$
Wind excitation forces
$q_1(t), \ldots, q_{N_u}(t)$
Impulse response functions
$Y(t)$
Displacement response

Figure 5.10: Simulation scheme.

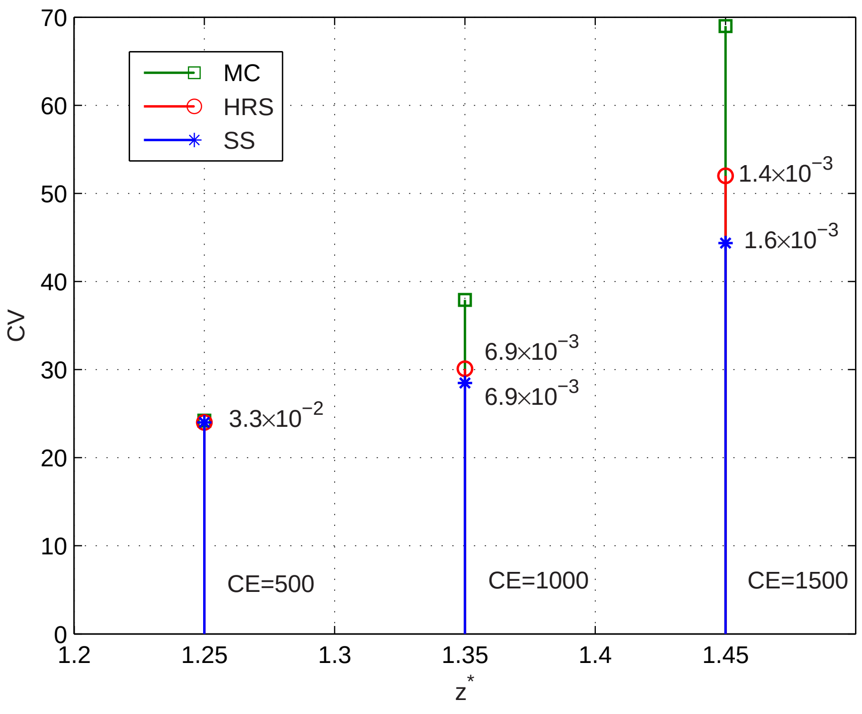


Figure 5.11: The CV of estimates obtained by Horseracing Simulation and Monte Carlo.

5.3.4 Simulation results

The failure events with thresholds $z_1^* = 1.25$ m, $z_2^* = 1.35$ m, and $z_3^* = 1.45$ m are considered in the simulation. The corresponding Monte Carlo (MC) estimates of the failure probabilities are found to be $p_{\Omega_1} = 3.3\times10^{-2}$ (with CV $\delta_1 = 5.4\%$), $p_{\Omega_2} = 6.8\times10^{-3}$ (with CV $\delta_2 = 12.1\%$), and $p_{\Omega_3} = 1.5 \times 10^{-3}$ (with CV $\delta_3 = 25.8\%$). For each failure event $n_{MC} = 10^4$ samples were used.

The Horseracing Simulation algorithm (HRS) is applied with $n = 500$ initial samples. The total computational efforts (CE) required by the algorithm are $\text{CE}_1 = 500$, $\text{CE}_2 = 1000$, and $\text{CE}_3 = 1500$ for p_{Ω_1}, p_{Ω_2}, and p_{Ω_3} respectively. In order to get approximately the same CE, Subset Simulation is applied with $n = 500$, $n = 360$, and $n = 540$ initial samples respectively. The obtained mean values of the failure probability estimates and their CVs based on 40 runs of these algorithms are shown in the Fig. 5.11 in comparison with the CVs of the Monte Carlo estimates (with the same CE). In the first case, when $z_1^* = 1.25$ m, both HRS and SS reduces to MC. The latter is expected, since the average number n_Ω of failure samples out of n is $n_\Omega = np_\Omega = 500 \times 3.3 \times 10^{-2} = 16.5 > 10$. In the second case ($z_2^* = 1.35$) as well

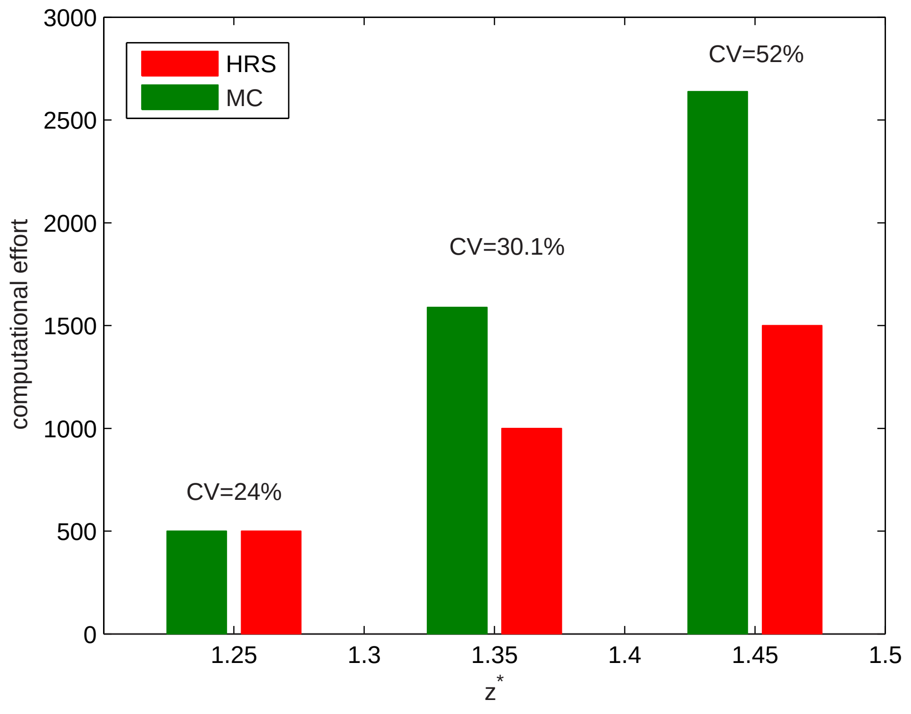


Figure 5.12: Computational effort required by Horseracing Simulation and Monte Carlo for the same CV.

as in the third case ($z_3^* = 1.45$) the HRS outperforms MC. The reductions in CV achieved are $(\delta_2^{MC} - \delta_2^{HRS})/\delta_2^{MC} = 20.6\%$ and $(\delta_3^{MC} - \delta_3^{HRS})/\delta_3^{MC} = 24.6\%$ correspondingly. However, in the two last cases SS outperforms the presented version of HRS. Since the Horseracing Simulation algorithm is a new developed reliability method, it contains a lot of rooms for improvements. So, the future research will be dedicated to the fine-tuning of HRS, which may result into superiority over Subset Simulation algorithm.

In Fig. 5.12 the comparison between HRS and MC is made in terms of computational effort. It is well known, that the number of MC samples requited to achieve the CV δ, when trying to compute failure probability p_Ω, is given by

$$n = \frac{1 - p_\Omega}{p_\Omega \delta^2}. \tag{5.33}$$

Using (5.33), we calculate the CE required by MC in order to obtain the CV achieved by HRS. The results are shown in Fig. 5.12. In the first case the CEs are the same as expected. In the second and third cases MC requires 1.58 and 1.76 times more CE.

So, the Horserace Simulation algorithm clearly outperforms the standard Monte Carlo Simulation for the considered real-life example.

Chapter 6

Conclusion

I guess I should warn you, if I turn out to be particularly clear, you've probably misunderstood what I've said.

From a 1988 speech at the Economic Club of New York by Alan Greenspan, Chairman of Federal Reserve

In this Chapter we briefly summarize the main results obtained in the present work.

The Thesis is dedicated to the exploration and development of advanced stochastic simulation algorithms for solving high-dimensional reliability problems.

Chapter 2 can be considered as a "critics" of the well-known and widely used reliability methods. The critics is based on deep understanding of the geometric features of high-dimensional reliability problem. We highlight the difficulties associated with these methods. Namely,

- The design point itself as well as the direction of the design point can be of no consequence when searching the main parts of the failure domain, which are the intersections of failure domain with the Important Ring.
- Importance Sampling is not applicable in general nonlinear high-dimensional reliability problems of practical interest.
- A geometric explanation as to why the standard Metropolis-Hastings algorithm does "not work" in high-dimensions is given.

In Chapters 3 and 4 we develop two useful modifications of the well-known algorithms:

- Adaptive Linked Importance Sampling (ALIS) generalizes Subset Simulation (SS) and in some cases, as it was shown with academic examples, can offer drastic improvements over SS.

- Modified Metropolis-Hastings algorithm with Delayed Rejection (MMHDR) is a novel modification of the Metropolis-Hastings algorithm, designed specially for sampling from conditional high-dimensional distributions.

Finally, in Chapter 5:

- A new advanced stochastic simulation algorithm, called Horseracing Simulation (HRS), is proposed for solving high-dimensional reliability problems. The accuracy and efficiency of the new method is demonstrated with a real-life wind engineering example.

Bibliography and References

[1] Au S.K., Beck J.L., (2001), *Estimation of small failure probabilities in high dimensions by subset simulation*, Probab. Engineering Mechanics; 16(4): 263-277. 1.

[2] Au S.K., Beck J.L., (2003), *Importance Sampling in high dimensions*, Structural Safety, 25(2): 139-163.

[3] Bennett C.H., (1976), *Efficient estimation of free energy differences from Monte Carlo data* , J. of Computational Physics, vol. 22, pp. 245-268.

[4] Canal L., (2005), *A normal approximation for the chi-square distribution*, Computational Statistics & Data Analysis, 48, 803-808.

[5] Davenport A.G., (1961), *The spectrum of horizontal gustiness near the ground in high winds*, J. R. Meteorol. Soc., 87, 194-211.

[6] Davenport A.G., (1968), *The dependence of wind load upon meteorological parameters*, Proceedings of the International Research Seminar on Wind Effects on Buildings and Structures, Toronto.

[7] Deodatis G., (1996), *Simulation of ergodic multivariate stochastic processes*, J. Eng. Mech., 122(8), 778-787.

[8] Der Kiureghian A., Li C-C., (1996), *Nonlinear random vibration analysis through optimization*, Proceedings of the seventh IFIP WG 7.5 Conference on optimization of structural systems, Boulder, Colorado, 197-206.

[9] Di Paola M., Gullo I., (2001) *Digital generation of multivariate wind field processes*, Probab. Eng. Mech., 16, 1-10.

[10] Ditlevsen O., Madsen H.O., (1996), *Structural Reliability Methods*, New York: Wiley.

[11] Fisher R.A., (1922), *On the interpretation of χ^2 from contingency tables and calculations of P*, J.R. Statist. Soc. A 85, 87-94.

[12] Geman S, Geman D, (1984), *Stochastic Relaxation, Gibbs Distribution and the Bayesian Restoration of Images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 6, 721-741.

[13] Grigoriu M., (2002), *Stochastic Calculus: Applications in Science and Engineering*, Birkhauser Boston, New York.

[14] Hastings W.K., (1970), *Monte Carlo sampling methods using Markov chains and their applications*, Biometrika, 57:97-109.

[15] Katafygiotis L.S., Cheung S.H., (2007), *Application of Spherical Subset Simulation Method and Auxiliary Domain Method on a Benchmark Reliability Study*, Structural Safety, 29(3), 194-207.

[16] Katafygiotis L.S., Wang J., *Reliability Analysis of Wind-excited Structures using Domain Decomposition Method and Line Sampling*, Submitted to the International Journal of Structural Engineering and Mechanics.

[17] Katafygiotis L.S., Zuev K.M., *Geometric insight into the challenges of solving high-dimensional reliability problems*, Proceedings Fifth Computational Stochastic Mechanics Conference, Rhodos Greece, 21-23 June 2006.

[18] Katafygiotis L.S., Zuev K.M., (2008), *Geometric insight into the challenges of solving high-dimensional reliability problems*, Probabilistic Engineering Mechanics; 23, 208–218.

[19] Katafygiotis L.S., Zuev K.M., *Estimation of small failure probabilities in high dimensions by Adaptive Linked Importance Sampling*, Proceedings Computational Methods in Structural Dynamics and Earthquake Engineering Conference, Rethymno Crete Greece, 13-16 June 2007.

[20] Koo H., Der Kiureghian A., Fujimura K., (2005), *Design-point excitation for non-linear random vibrations*, Probabilistic Engineering Mechanics, 20(2), 136-147.

[21] Krylov V.I., (1962), *Approximate Calculation of Integrals*, New York: Macmillan.

[22] Lu N., Singh J.K., Kofke D.A., (2003), *Appropriate methods to combine forward and reverse free-energy perturbation averages*, Journal of Chemical Physics, vol. 118, pp. 2977-2984.

[23] Melbourne W.H., (1980), *Comparison of measurements on the CAARC standard tall building model in simulated model wind flows*, J. Wind. Eng. Ind. Aerodyn., 6, 73-88.

[24] Meng X.-L., Wong H.W., (1996), *Simulating ratios of normalizing constants via a simple identity: A theoretical exploration*, Statistica Sinica, vol. 6, pp. 831-860.

[25] Metropolis N. et al., (1953), *Equation of State Calculations by Fast Computing Machines*, J. Chemical Physics, vol. 21, pp. 1087–1092.

[26] Metropolis N., Ulam S., (1949), *The Monte Carlo method*, J. Amer. statistical assoc., vol. 44, N. 247, pp. 335-341.

[27] Neal R.M., (2005), *Estimating ratios of normalizing constants using Linked Importance Sampling*, Technical Report No. 0511, Dept. of Statistics, University of Toronto.

[28] Rubin D.B., (1987), *A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when the fraction of missing information is modest: the SIR algorithm*, Journla of the American Statistical Association 82: 543–546.

[29] Schuëller G.I., Pradlwarter H.J. and Koutsourelakis P.S. (2004), *A critical appraisal of reliability estimation procedures for high dimensions*, Probabilistic Engineering Mechanics; 19, 463–474.

[30] Shinozuka M., Jan C.M., (1972), *Digital simulation of random processes and its applications*, J. Sound Vib., 25(1), 111-128.

[31] Shinozuka M., Deodatis G., (1991), *Simulation of stochastic processes by spectral representation*, Appl. Mech. Rev., 44(4), 191-204.

[32] Simiu E., Scanlan R.H., (1986), *Wind Effects on Structures*, John Wiley & Sons, Inc., New York.

[33] Tierney L, Mira A, (1999), *Some adaptive Monte Carlo methods for Bayesian inference*, Statist. Med. 18, 2507-2515.

[34] Zuev K.M., Katafygiotis L.S., (2008), *Modified Metropolis-Hastings algorithm with delayed rejection*, Asian-Pacific Symposium on Structural Reliability and its Applications, Hong Kong, 2008.

Appendix A: Markov chains

The only good Monte Carlo is a dead Monte Carlo

Trotter & Tukey

In this appendix we prove that conditional distribution $\pi(\cdot|F)$ is an equilibrium distribution for any Markov chain generated by the MMHDR algorithm, described in section 2.4. In other words, if we generate a Markov chain $X_0, X_1, \ldots$ using MMGDR update, started from essentially any $X_0 \in F$, then for large n the distribution of X_n will be approximately $\pi(\cdot|F)$. We start with recalling of the needed definitions and facts from the theory of Markov chains.

A Markov chain on a state space $F \subset \mathbb{R}^N$ is a sequence of random vectors $\{X_n, n \geq 0\}$ such that

$$P(X_{n+1} \in A|X_n = x, X_j, j < n) = P(X_{n+1} \in A|X_n = x) \equiv K_n(x, A), \tag{6.1}$$

for all $A \subset F$ and $x \in F$. The probability measure $K_n(x, \cdot)$ is called the transition kernel. Typically, we assume that the transition kernel does not dependent on the time n, $K_n = K$. In this case the corresponding Markov chain is called time-homogeneous.

Usually the transition kernel in Markov chain simulations has both continuous and discrete components and can be expressed as follows:

$$K(x, dy) = k(x, y)dy + r(x)\delta_x(dy). \tag{6.2}$$

Here $k : F \times F \to \mathbb{R}_+$ with $k(x, x) = 0$ describes the continuous part of the transition kernel, $r(x) = 1 - \int_F k(x, y)dy$, and δ_x denotes point mass at x (Dirac measure):

$$\delta_x(A) = \begin{cases} 1, & \text{if } x \in A; \\ 0, & \text{if } x \notin A. \end{cases} \tag{6.3}$$

Thus, transition kernel (6.2) specifies that transitions of the Markov chain from x to y occur according to $k(x, y)$ and the Markov chain remains at x with probability $r(x)$. Transition kernel (6.2) is schematically shown in the Fig. 6.1.

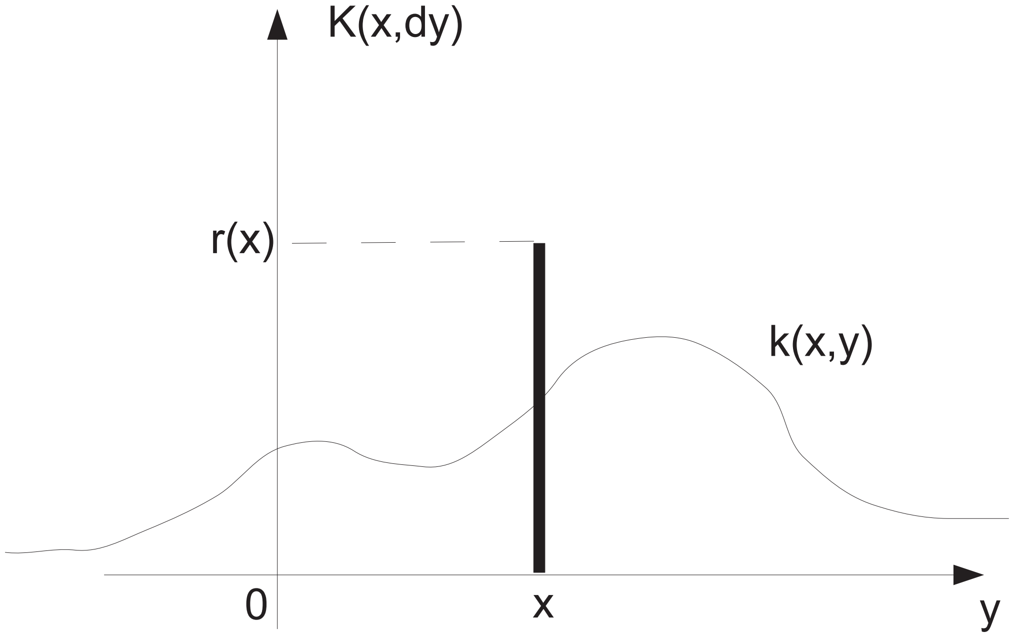


Figure 6.1: Transitional kernel with both continuous and discrete components.

Let π be a probability distribution on F. Assume that π has a density with respect to the Lebesgue measure:

$$\pi(dx) = \pi(x)dx. \tag{6.4}$$

For simplicity, π will be used to denote both distribution and density. The probability distribution π is called invariant distribution for a transition kernel K if

$$\pi(dy) = \int\limits_{x \in F} \pi(x)K(x, dy)dx. \tag{6.5}$$

It is easy to check, that a sufficient condition for π to be the invariant distribution for K is to satisfy the so-called reversibility condition:

$$\pi(dx)K(x, dy) = \pi(dy)K(y, dx). \tag{6.6}$$

The central result of the Markov chain theory is the following. Let K be a transition kernel with invariant distribution π. In addition, assume that the transition kernel K satisfies certain ergodic conditions (it is irreducible and aperiodic). Then the invariant distribution π is the equilibrium distribution of the corresponding Markov chain: if we run the Markov chain for a long time (burn-in period), started from anywhere in the state space, then for large n the distribution of X_n will be approximately π. The required burn-in period heavily depends on the choice of the transition kernel K and on π itself. It also should be mentioned, that in practical application it is very difficult to check whether the Markov chain has reached its invariant distribution or not. Even if it has, it is hard to tell for sure.

Now let K denote the transitional kernel of the Markov chain generated by the MMHDR algorithm.

Theorem 1. *The transition kernel K of the MMHDR update satisfies the reversibility condition with respect to the conditional distribution $\pi(\cdot|F)$:*

$$\pi(dx_0|F)K(x_0, dx_1) = \pi(dx_1|F)K(x_1, dx_0). \tag{6.7}$$

Proof. By the definition of the MMHDR algorithm all the Markov chain samples lie in F, therefore, it is sufficient to consider the transition only between states in F. So, without loss of generality, we assume that both x_0 and x_1 belong to F, $x_0, x_1 \in F$. In addition, we assume that $x_0 \neq x_1$, since otherwise (6.7) is trivial.

The MMHDR update is naturally divided into two stages. At the first stage (Steps 1 and 2), being in the current state $x_0 \in F$, we generate a candidate state ξ_1, which can either belong to the failure domain F or not. At the second stage (Steps 3 and 4), still being in the current state x_0 and having rejected candidate $\xi_1 \in \bar{F} = \mathbb{R}^N \setminus F$, we generate the second candidate state ξ_2, and take ξ_2 or x_0 as the next state x_1 of the Markov chain depending on whether ξ_2 belongs to the failure domain or not:

$$\begin{aligned} &\text{1st stage} : F \to \mathbb{R}^N, \ \ x_0 \mapsto \xi_1, \\ &\text{2nd stage} : F \times \bar{F} \to F, \ \ (x_0, \xi_1) \mapsto x_1. \end{aligned} \tag{6.8}$$

Denote the transition kernels of the first and second stages by K_1 and K_2, respectively. Then the transition kernel of the MMHDR update can be written as follows:

$$K(x_0, dx_1) = K_1(x_0, dx_1) + \int\limits_{\xi \in \bar{F}} K_1(x_0, d\xi)K_2(x_0, \xi, dx_1). \tag{6.9}$$

Lemma 1. *If $x_0, x_1 \in F$, then*

$$\pi(dx_0)K_1(x_0, dx_1) = \pi(dx_1)K_1(x_1, dx_0). \tag{6.10}$$

Proof. According to Step 1, the transition of individual coordinates of x_0, when the first candidate state is generated, are independent. So the transition kernel K_1 can be expressed as a product of the coordinate transition kernels:

$$K_1(x_0, dx_1) = \prod_{j=1}^{N} K_1^j(x_0^j, dx_1^j), \tag{6.11}$$

where K_1^j is the transition kernel for the jth coordinate of x_0 at the first stage. Therefore, (6.10) can be equivalently rewritten in the coordinates as follows:

$$\prod_{j=1}^{N} \pi_j(dx_0^j) K_1^j(x_0^j, dx_1^j) = \prod_{j=1}^{N} \pi_j(dx_1^j) K_1^j(x_1^j, dx_0^j). \tag{6.12}$$

To prove (6.12) it is sufficient to show that for any $j = 1 \ldots, N$

$$\pi_j(dx_0^j) K_1^j(x_0^j, dx_1^j) = \pi_j(dx_1^j) K_1^j(x_1^j, dx_0^j). \tag{6.13}$$

According to Step 1, the transition kernel K_1^j for x_0^j at the first stage can be written as follows:

$$K_1^j(x_0^j, dx_1^j) = k_1^j(x_0^j, x_1^j) dx_1^j + r_1^j(x_0^j) \delta_{x_0^j}(dx_1^j), \tag{6.14}$$

where

$$k_1^j(x_0^j, x_1^j) = S_1^j(x_1^j | x_0^j) a_1^j(x_0^j, x_1^j), \tag{6.15}$$

and a_1^j is given by (4.11), namely

$$a_1^j(x_0^j, x_1^j) = \min\left\{1, \frac{\pi_j(x_1^j) S_1^j(x_0^j | x_1^j)}{\pi_j(x_0^j) S_1^j(x_1^j | x_0^j)}\right\}. \tag{6.16}$$

Assume that $x_0^j \neq x_1^j$, since otherwise (6.13) is trivial. Then, using the identity $b \min\{1, a/b\} = a \min\{1, b/a\}$, which is valid for any two positive numbers a and b, we have:

$$\begin{aligned}
\pi_j(dx_0^j) K_1^j(x_0^j, dx_1^j) &= \pi_j(x_0^j) k_1^j(x_0^j, x_1^j) dx_0^j dx_1^j \\
&= \pi_j(x_0^j) S_1^j(x_1^j | x_0^j) \min\left\{1, \frac{\pi_j(x_1^j) S_1^j(x_0^j | x_1^j)}{\pi_j(x_0^j) S_1^j(x_1^j | x_0^j)}\right\} dx_0^j dx_1^j \\
&= \pi_j(x_1^j) S_1^j(x_0^j | x_1^j) \min\left\{1, \frac{\pi_j(x_0^j) S_1^j(x_1^j | x_0^j)}{\pi_j(x_1^j) S_1^j(x_0^j | x_1^j)}\right\} dx_0^j dx_1^j \\
&= \pi_j(x_1^j) k_1^j(x_1^j, x_0^j) dx_0^j dx_1^j = \pi_j(dx_1^j) K_1^j(x_1^j, dx_0^j).
\end{aligned} \tag{6.17}$$

So, Lemma 1 is proved.

Remark 1. *In essence, the proof of Lemma 1 repeats the one given in Au and Beck (2001) for the Modified Metropolis algorithm.*

Thus, keeping in mind (6.9) and Lemma 1, it remains to show that for any $x_0, x_1 \in F$ and $x_0 \neq x_1$

$$\pi(dx_0) \int\limits_{\xi \in \bar{F}} K_1(x_0, d\xi) K_2(x_0, \xi, dx_1) = \pi(dx_1) \int\limits_{\xi \in \bar{F}} K_1(x_1, d\xi) K_2(x_1, \xi, dx_0). \tag{6.18}$$

According to Step 3, the transition of individual coordinates of x_0, when the second candidate state is generated, are independent. So the transition kernel K_2, as well as K_1, can be expressed as a product of the coordinate transition kernels:

$$K_2(x_0, \xi, x_1) = \prod_{j=1}^{N} K_2^j(x_0^j, \xi^j, dx_1^j), \tag{6.19}$$

where K_2^j is the transition kernel for the jth coordinate of x_0 at the second stage. Therefore, (6.18) can be equivalently rewritten in the coordinates as follows:

$$\int\limits_{\xi \in \bar{F}} \prod_{j=1}^{N} \pi_j(dx_0^j) K_1^j(x_0^j, d\xi^j) K_2^j(x_0^j, \xi^j, dx_1^j) = \int\limits_{\xi \in \bar{F}} \prod_{j=1}^{N} \pi_j(dx_1^j) K_1^j(x_1^j, d\xi^j) K_2^j(x_1^j, \xi^j, dx_0^j). \tag{6.20}$$

To satisfy the condition (6.20) it is sufficient to show that for any $j = 1 \ldots, N$ the following holds:

$$\pi_j(dx_0^j) K_1^j(x_0^j, d\xi^j) K_2^j(x_0^j, \xi^j, dx_1^j) = \pi_j(dx_1^j) K_1^j(x_1^j, d\xi^j) K_2^j(x_1^j, \xi^j, dx_0^j). \tag{6.21}$$

According to Step 3, the transition kernel K_2^j for x_0^j at the second stage can be written as follows:

$$K_2^j(x_0^j, \xi^j, dx_1^j) = \begin{cases} \delta_{x_0^j}(dx_1^j), & \text{if } \xi^j = x_0^j; \\ k_2^j(x_0^j, \xi^j, x_1^j) dx_1^j + r_2^j(x_0^j, \xi^j) \delta_{x_0^j}(dx_1^j), & \text{if } \xi^j \neq x_0^j, \end{cases} \tag{6.22}$$

where

$$k_2^j(x_0^j, \xi^j, x_1^j) = S_2^j(x_1^j | x_0^j, \xi^j) a_2^j(x_0^j, \xi^j, x_1^j), \tag{6.23}$$

and a_2^j is given by (4.14), namely

$$a_2^j(x_0^j, \xi^j, x_1^j) = \min\left\{1, \frac{\pi_j(x_1^j) S_1^j(\xi^j | x_1^j) S_2^j(x_0^j | x_1^j, \xi^j) a_1^j(x_1^j, \xi^j)}{\pi_j(x_0^j) S_1^j(\xi^j | x_0^j) S_2^j(x_1^j | x_0^j, \xi^j) a_1^j(x_0^j, \xi^j)}\right\}. \tag{6.24}$$

Assume that $x_0^j \neq x_1^j$, since otherwise condition (6.21) is trivial. Consider the following three cases separately: $\xi^j \neq x_0^j$ and $\xi^j \neq x_1^j$ (1st case), $\xi^j = x_0^j$ (2nd case) and $\xi^j = x_1^j$ (3d case).

i. $\xi^j \neq x_0^j$, $\xi^j \neq x_1^j$.

Then we have:

$$\begin{aligned}
&\pi_j(dx_0^j)K_1^j(x_0^j, d\xi^j)K_2^j(x_0^j, \xi^j, dx_1^j) \\
&= \pi_j(x_0^j)k_1^j(x_0^j, \xi^j)k_2^j(x_0^j, \xi^j, x_1^j)dx_0^j d\xi^j dx_1^j \\
&= \pi_j(x_0^j)S_1^j(\xi^j|x_0^j)a_1^j(x_0^j, \xi^j)S_2^j(x_1^j|x_0^j, \xi^j)\times \\
&\min\left\{1, \frac{\pi_j(x_1^j)S_1^j(\xi^j|x_1^j)S_2^j(x_0^j|x_1^j, \xi^j)a_1^j(x_1^j, \xi^j)}{\pi_j(x_0^j)S_1^j(\xi^j|x_0^j)S_2^j(x_1^j|x_0^j, \xi^j)a_1^j(x_0^j, \xi^j)}\right\} dx_0^j d\xi^j dx_1^j \\
&= \pi_j(x_1^j)S_1^j(\xi^j|x_1^j)a_1^j(x_1^j, \xi^j)S_2^j(x_0^j|x_1^j, \xi^j)\times \\
&\min\left\{1, \frac{\pi_j(x_0^j)S_1^j(\xi^j|x_0^j)S_2^j(x_1^j|x_0^j, \xi^j)a_1^j(x_0^j, \xi^j)}{\pi_j(x_1^j)S_1^j(\xi^j|x_1^j)S_2^j(x_0^j|x_1^j, \xi^j)a_1^j(x_1^j, \xi^j)}\right\} dx_0^j d\xi^j dx_1^j \\
&= \pi_j(x_1^j)k_1^j(x_1^j, \xi^j)k_2^j(x_1^j, \xi^j, x_0^j)dx_0^j d\xi^j dx_1^j \\
&= \pi_j(dx_1^j)K_1^j(x_1^j, d\xi^j)K_2^j(x_1^j, \xi^j, dx_0^j).
\end{aligned} \tag{6.25}$$

So, in this case (6.21) is fulfilled.

ii. $\xi^j = x_0^j$.

In this case the left-hand side of (6.21) is zero, since $K_2^j(x_0^j, x_0^j, dx_1^j) = \delta_{x_0^j}(dx_1^j) = 0$. The last equality holds, because we have assumed that $x_0^j \neq x_1^j$. Let us now analyze the right-hand side of (6.21) when $\xi^j = x_0^j$:

$$\begin{aligned}
&\pi_j(dx_1^j)K_1^j(x_1^j, dx_0^j)K_2^j(x_1^j, x_0^j, dx_0^j) \\
&= \pi_j(dx_1^j)k_1^j(x_1^j, x_0^j)\left(k_2^j(x_1^j, x_0^j, x_0^j)dx_0^j + r_2^j(x_1^j, x_0^j)\delta_{x_1^j}(dx_0^j)\right)dx_0^j \\
&= \pi_j(dx_1^j)k_1^j(x_1^j, x_0^j)k_2^j(x_1^j, x_0^j, x_0^j)dx_0^j dx_0^j = 0,
\end{aligned} \tag{6.26}$$

since $dx_0^j dx_0^j = 0$, which is a standard result of the measure theory. So, the right-hand side of (6.21) is also zero. Basically, when $\xi^j = x_0^j$ the left-hand side of (6.21) is zero by definition of the MMHDR algorithm: in the 2nd stage the MMHDR update transforms only those coordinates of the current state, which have been already transformed during the 1st stage. At the same time, the right-hand side is zero, because the "probability" to transform any coordinate of the current state to the same value twice (during both 1st and 2nd stages) is infinitessimally small.

iii. $\xi^j = x_1^j$.

This case can be considered in exactly the same way as the 2nd one where x_0^j and x_1^j are replaced by each other.

Thus, Theorem 1 is proved.

Corollary 1. *The conditional distribution $\pi(\cdot|F)$ is invariant for the kernel K and, therefore, any Markov chain generated by the MMHDR algorithm will eventually converge to $\pi(\cdot|F)$ as its equilibrium distribution.*

Remark 2. *When MMHDR algorithm is used together with Subset simulation the starting state of the Markov chain is already distributed according to $\pi(\cdot|F)$. It means that there is no burn-in period in this case.*

Remark 3. *Following the proof of Theorem 1, it can be shown that keeping fixed those coordinates of the current state, that have not been transformed at the first stage of the MMHDR update, is essential for satisfying the reversibility condition (6.7).*

Appendix B: f_k-distribution

Let z_0 be a random variable with PDF f_0 and CDF F_0, and let $\xi \sim f_0$ be its realization. Define z_1 to be a new random variable with conditional distribution $f_0(z|z \geq \xi)$. In general, if $\xi \sim f_k$ is a realization of the random variable z_k, define z_{k+1} to be a new random variable with conditional distribution $f_0(z|z \geq \xi)$. This procedure defines a Markov chain $z_0, z_1, \ldots$, which, as a matter of fact, is completely defined by the distribution of the random variable z_0.

Theorem 2. *The PDF f_k of the random variable z_k is*

$$f_k(z) = \frac{(-1)^k}{k!} f_0(z) \left[\log(1 - F_0(z))\right]^k \tag{6.27}$$

Proof. We prove this theorem by induction on k. For $k = 0$ the statement of the theorem is obvious. Suppose that (6.27) holds for k. Then for $k+1$ we have:

$$\begin{aligned}
f_{k+1}(z) &= \int_{-\infty}^{\infty} f_0(z|z \geq \xi) f_k(\xi) d\xi \\
&= \frac{(-1)^k}{k!} \int_{-\infty}^{\infty} \frac{f_0(z) I_F\{z \geq \xi\}}{1 - F_0(\xi)} f_0(\xi) \left[\log(1 - F_0(\xi))\right]^k d\xi \\
&= \frac{(-1)^k}{k!} f_0(z) \int_{-\infty}^{z} \frac{\left[\log(1 - F_0(\xi))\right]^k dF_0(\xi)}{1 - F_0(\xi)} \qquad (6.28) \\
&= \frac{(-1)^{k+1}}{k!} f_0(z) \int_{-\infty}^{z} \frac{\left[\log(1 - F_0(\xi))\right]^k d(1 - F_0(\xi))}{1 - F_0(\xi)} \\
&= \frac{(-1)^{k+1}}{(k+1)!} f_0(z) \left[\log(1 - F_0(z))\right]^{k+1}.
\end{aligned}$$

Which proves the theorem. □

Appendix C: Spectral representation method

According to the spectral representation method, the stochastic vector process is simulated as a superposition of harmonic waves with random phases or random amplitudes.

Let $S^0(\omega)$ be the cross-power spectral density matrix of the stochastic vector process $v(t) = [v_1(t), \ldots, v_N(t)]^T$,

$$S^0(\omega) = \begin{pmatrix} S^0_{11}(\omega) & \ldots & S^0_{1N}(\omega) \\ \vdots & & \vdots \\ S^0_{N1}(\omega) & \ldots & S^0_{NN}(\omega) \end{pmatrix}, \tag{6.29}$$

which is assumed to be real due to the negligibility of the quadrature spectrum [32], symmetric and positive definite [9] for each frequency ω. Then, according to Cholesky decomposition, the matrix $S^0(\omega)$ can be factorized as follows:

$$S^0(\omega) = H(\omega)H(\omega)^T, \tag{6.30}$$

where $H(\omega)$ is a lower triangular real matrix (so called "the Cholesky triangle" of $S^0(\omega)$). Let $H_j(\omega)$, $j = 1, \ldots, N$ denote the j-th column vector of $H(\omega)$, so that

$$H(\omega) = [H_1(\omega), \ldots, H_N(\omega)]. \tag{6.31}$$

Let ω_c be the cutoff frequency above which all components $S^0_{jk}(\omega)$, $j, k = 1, \ldots, N$ are insignificant for practical purposes. More precisely, if $r_{c,jk}$ denotes the ratio of the neglected power spectrum content over the total content, i.e.

$$r_{c,jk} = \frac{\int\limits_{\omega_c}^{\infty} S^0_{jk}(\omega)\, d\omega}{\int\limits_{0}^{\infty} S^0_{jk}(\omega)\, d\omega}, \tag{6.32}$$

then ω_c is defined such that all $r_{c,jk}$ are smaller than some predefined threshold. Divide the interval $[0, \omega_c]$ into N_ω equal segments, each having length $\triangle\omega = \omega_c/N_\omega$, and define a sequence of frequencies as follows:

$$\omega_i = \frac{(2i-1)\triangle\omega}{2}, \;\; i = 1, \ldots, N_\omega. \tag{6.33}$$

Then, the stochastic vector process $v(t)$ is simulated according to the following formula [13]:

$$v(t) = \sqrt{\triangle\omega} \sum_{i=1}^{N_\omega} \sum_{j=1}^{N} H_j(\omega_i)(x_{ij}^1 \cos(\omega_i t) + x_{ij}^2 \sin(\omega_i t)), \tag{6.34}$$

where x_{ij}^1 and x_{ij}^2, $i = 1 \ldots, N_\omega, j = 1, \ldots, N$ are independent standard gaussian variables.