Probabilistic Engineering Mechanics 26 (2011) 405–412

Contents lists available at ScienceDirect

Probabilistic Engineering Mechanics

journal homepage: www.elsevier.com/locate/probengmech

Modified Metropolis–Hastings algorithm with delayed rejection

K.M. Zuev*, L.S. Katafygiotis

Department of Civil Engineering, HKUST, Hong Kong, China

ARTICLE INFO

Article history:
Received 17 March 2010
Received in revised form
30 September 2010
Accepted 30 November 2010
Available online 15 December 2010

Keywords:
Reliability
Metropolis–Hastings algorithm
Markov chains
Markov chain Monte Carlo
Variance reduction techniques
Subset simulation
Dynamic analysis

ABSTRACT

The development of an efficient MCMC strategy for sampling from complex distributions is a difficult task that needs to be solved for calculating the small failure probabilities encountered in the high-dimensional reliability analysis of engineering systems. Usually different variations of the Metropolis–Hastings algorithm (MH) are used. However, the standard MH algorithm does not generally work in high dimensions, since it leads to very frequent repeated samples. In order to overcome this deficiency one can use the Modified Metropolis–Hastings algorithm (MMH) proposed in Au and Beck (2001) [1]. Another variation of the MH algorithm, called the Metropolis–Hastings algorithm with delayed rejection (MHDR) has been proposed by Tierney and Mira (1999) [7]. The key idea behind the MHDR algorithm is to reduce the correlation between states of the Markov chain. In this paper we combine the ideas of MMH and MHDR and propose a novel modification of the MH algorithm, called the Modified Metropolis–Hastings algorithm with delayed rejection (MMHDR). The efficiency of the new algorithm is demonstrated with a numerical example where MMHDR is used together with Subset simulation for computing small failure probabilities in high dimensions.

1. Introduction

One of the most important and challenging problems in reliability engineering is to compute the failure probability given by the following expression:

$$p_F = \int_{\mathbb{R}^N} I_F(x)\pi(x)dx. \quad (1)$$

Here $\pi(\cdot)$ is the joint probability density function (PDF) of a random vector $x \in \mathbb{R}^N$, which represents the uncertain parameters of the problem; $F \subset \mathbb{R}^N$ is the failure domain that is usually defined as $F = \{x \in \mathbb{R}^N | G(x) < 0\}$, where G is called the limit-state function; I_F is the indicator function of F, i.e. $I_F(x) = 1$ if $x \in F$ and $I_F(x) = 0$ if $x \notin F$. Throughout this work we assume that the parameter space $\mathbb{R}^N$ is high-dimensional and the failure probability p_F is very small.

Each advanced stochastic simulation algorithm for the computation of small failure probabilities (1) encountered in the reliability analysis of engineering systems consists of two main steps. 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 [1] and Adaptive Linked Importance Sampling [2] we sample from the important sampling density π_{is}, family of conditional distributions $\pi(\cdot|F)$ and family of intermediate distributions π_α respectively. In many cases where there is no explicit expression for $\tilde{\pi}$, Markov chain Monte Carlo (MCMC) is employed to generate the required samples. Usually different variations of the Metropolis–Hastings (MH) algorithm are used.

The main objective of this paper is to develop a novel effective MCMC algorithm for sampling from high-dimensional conditional distributions.

Our starting point is the standard MH algorithm [3,4]. It has been shown by Au and Beck [1] that the standard MH algorithm generally does not work in high dimensions, since it leads to extremely frequent repeated samples. A geometric understanding of why this is true is given in [5], see also [6]. In order to overcome this deficiency of the MH algorithm one can use the Modified Metropolis–Hastings algorithm (MMH) proposed in [1] for sampling from high-dimensional distributions. It should be emphasized that the MMH algorithm is suitable only for sampling from very specific distributions, namely, from the conditional distributions $\pi(\cdot|F)$, where the unconditional PDF π can be factorized into a product of easily sampled 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 the Metropolis–Hastings algorithm with delayed rejection (MHDR) has been proposed in [7]. 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

* Corresponding author.
E-mail addresses: zuev@ust.hk, konstantin.zuev@mail.ru (K.M. Zuev), lambros@ust.hk (L.S. Katafygiotis).

0266-8920/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.probengmech.2010.11.008

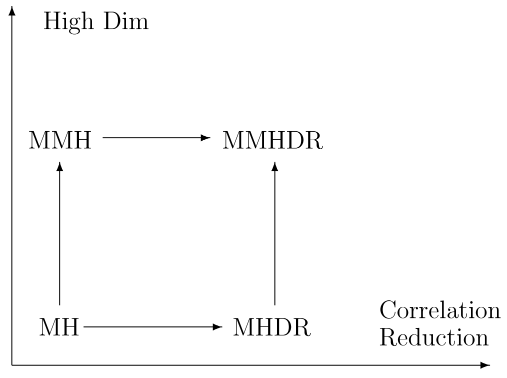


Fig. 1. Modifications of the standard Metropolis–Hastings algorithm.

generated from the proposal distribution is frequently 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 is 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 paper we combine the ideas of both the MMH and MHDR algorithms. As a result we obtain an efficient algorithm, called the 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. 1.

2. Modifications of the Metropolis–Hastings algorithm

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

2.1. 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 for the specific case of the Boltzmann distribution, and W. Keith Hastings, who generalized it in 1970.

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; let x_0 be the current state of the Markov chain; and let $S(\cdot|x_0)$, called proposal PDF, be an N-dimensional PDF depended on x_0. Then the MH update $x_0 \to x_1$ of the Markov chain works as follows:

(1) Simulate ξ according to $S(\cdot|x_0)$,
(2) Compute the acceptance probability

$$a(x_0, \xi) = \min\left\{1, \frac{\pi(\xi)S(x_0|\xi)}{\pi(x_0)S(\xi|x_0)}I_F(\xi)\right\}. \tag{2}$$

(3) Accept or reject ξ by setting

$$x_1 = \begin{cases} \xi, & \text{with prob. } a(x_0, \xi); \\ x_0, & \text{with prob. } 1 - a(x_0, \xi). \end{cases} \tag{3}$$

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

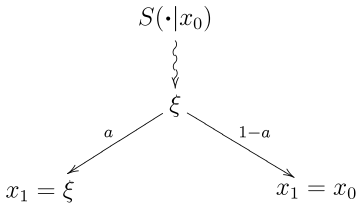


Fig. 2. Standard Metropolis–Hastings algorithm.

$$x_0 \sim \pi(\cdot|F) \Rightarrow x_1 \sim \pi(\cdot|F). \tag{4}$$

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 [3]. The MH update is schematically shown in Fig. 2.

2.2. Metropolis–Hastings algorithm with delayed rejection

Rejecting the candidate state ξ with probability $1 - a(x_0, \xi)$ in (3) 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, it is desirable to reduce the number of rejected candidate states, in order to improve the standard MH algorithm.

The Metropolis–Hastings algorithm with delayed rejection (MHDR), proposed in [7], allows us to achieve this goal: when a reject decision in (3) 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 a conditional distribution $\pi(\cdot|F)$ works as follows:

(1) Simulate ξ_1 according to $S_1(\cdot|x_0)$,
(2) Compute the acceptance probability

$$a_1(x_0, \xi_1) = \min\left\{1, \frac{\pi(\xi_1)S_1(x_0|\xi_1)}{\pi(x_0)S_1(\xi_1|x_0)}I_F(\xi_1)\right\}. \tag{5}$$

(3) Accept or reject ξ_1 by setting

$$x_1 = \begin{cases} \xi_1, \text{ go to step (7)} & \text{with prob. } a_1(x_0, \xi_1); \\ \text{go to step (4)}, & \text{with prob. } 1 - a_1(x_0, \xi_1). \end{cases} \tag{6}$$

(4) Simulate ξ_2 according to $S_2(\cdot|x_0, \xi_1)$.
(5) Compute the acceptance probability

$$a_2(x_0, \xi_1, \xi_2) = \min\left\{1, \frac{\pi(\xi_2)S_1(\xi_1|\xi_2)S_2(x_0|\xi_2, \xi_1)(1 - a_1(\xi_2, \xi_1))}{\pi(x_0)S_1(\xi_1|x_0)S_2(\xi_2|x_0, \xi_1)(1 - a_1(x_0, \xi_1))}I_F(\xi_2)\right\}. \tag{7}$$

(6) Accept or reject ξ_2 by setting

$$x_1 = \begin{cases} \xi_2, & \text{with prob. } a_2(x_0, \xi_1, \xi_2); \\ x_0, & \text{with prob. } 1 - a_2(x_0, \xi_1, \xi_2). \end{cases} \tag{8}$$

(7) End.

One possible way in which this scheme can be used is described in [7]: the first stage proposal PDF is set to be an independent proposal, $S_1(\xi_1|x_0) = f(\xi_1)$, where $f(\cdot)$ is thought to be a good approximation of the target PDF, and the second stage proposal PDF is defined as a random walk $S_2(\xi_2|x_0, \xi_1) = g(\xi_2 - x_0)$. If f is indeed a good approximation, then the first candidates will rarely

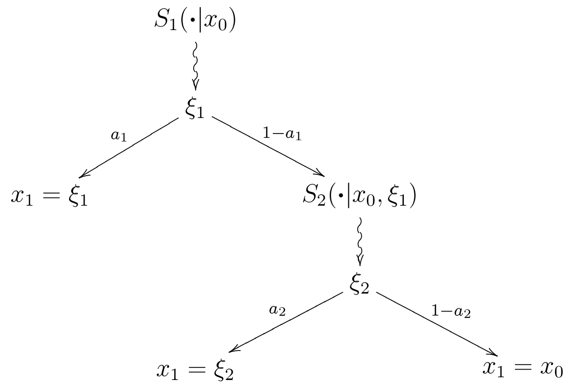


Fig. 3. Metropolis–Hastings algorithm with delayed rejection.

be rejected and the random walk will rarely be used. However, if f is not a good approximation then the random walk defined by S_2 provides protection against the potentially very poor behavior of independence chains.

An interesting feature of the MHDR 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 the previously rejected candidate does not destroy the Markovian property of the sampler. Thus 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. 3.

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

2.3. 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 a repeated sample, $x_1 = x_0$ [1,5]. Clearly, the MHDR update has the same problem. Thus, a Markov chain of practically meaningful length constructed in high dimensions having applied either the 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 algorithm a sequence of one-dimensional proposals $S^j(\cdot|x_0^j)$, $j=1,\dots,N$, is used. Namely, each coordinate ξ^j of the candidate state is generated separately using a one-dimensional proposal distribution $S^j(\cdot|x_0^j)$ depending 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 $\xi=(\xi^1,\dots,\xi^N)$:
For each $j=1,\dots,N$
(1a) Simulate $\hat{\xi}^j$ according to $S^j(\cdot|x_0^j)$,
(1b) Compute the acceptance probability

$$a^j(x_0^j,\hat{\xi}^j)=\min\left\{1,\frac{\pi_j(\hat{\xi}^j)S^j(x_0^j|\hat{\xi}^j)}{\pi_j(x_0^j)S^j(\hat{\xi}^j|x_0^j)}\right\}. \quad (9)$$

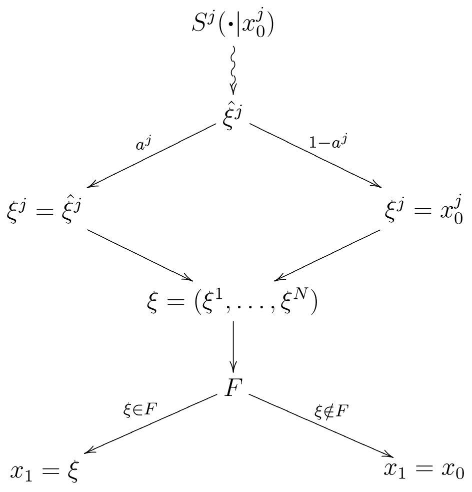


Fig. 4. Modified Metropolis–Hastings algorithm.

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

$$\xi^j=\begin{cases}\hat{\xi}^j, & \text{with prob. } a^j(x_0^j,\hat{\xi}^j);\\ x_0^j, & \text{with prob. } 1-a^j(x_0^j,\hat{\xi}^j).\end{cases} \quad (10)$$

(2) Accept or reject ξ by setting

$$x_1=\begin{cases}\xi, & \text{if } \xi\in F;\\ x_0, & \text{if } \xi\notin F.\end{cases} \quad (11)$$

It can be easily seen that the MMH algorithm overcomes the deficiency of the standard MH algorithm by producing distinct Markov chain states, rather than repeated samples [1,5]. The MMH update is schematically shown in Fig. 4.

2.4. Modified Metropolis–Hastings algorithm with delayed rejection

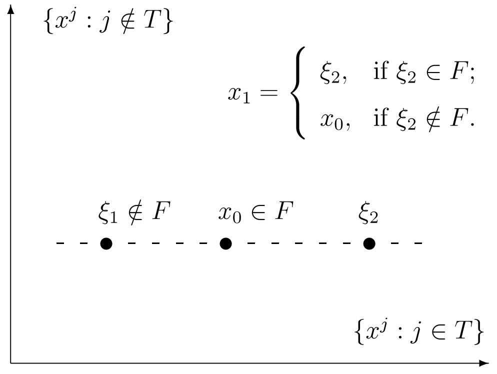
In this paper we propose a new MCMC method, called the Modified Metropolis–Hastings algorithm with delayed rejection (MMHDR), which combines the ideas of both MMH and MHDR algorithms.

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: $I=T\cup\overline{T}$, 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 having unchanged corresponding coordinates.

In the MMH algorithm once the candidate is generated we need to check whether it belongs to the failure domain or not. If it does, we accept the candidate as a new state of the Markov chain, and if it does not, we reject the candidate and get a repeated sample. In the proposed MMHDR algorithm when a reject decision in (11) 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 already been transformed at the first stage already. Schematically this is shown in Fig. 5.

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

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



$\{x^j : j \notin T\}$
$x_1 = \begin{cases} \xi_2, & \text{if } \xi_2 \in F; \\ x_0, & \text{if } \xi_2 \notin F. \end{cases}$
$\xi_1 \notin F$
$x_0 \in F$
ξ_2
$\{x^j : j \in T\}$

Fig. 5. MMHDR update at the second stage.

(1) Generate a candidate state $\xi_1 = (\xi_1^1, \dots, \xi_1^N)$:
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

$$a_1^j(x_0^j, \hat{\xi}_1^j) = \min\left\{1, \frac{\pi_j(\hat{\xi}_1^j)S_1^j(x_0^j|\hat{\xi}_1^j)}{\pi_j(x_0^j)S_1^j(\hat{\xi}_1^j|x_0^j)}\right\}. \quad (12)$$

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

$$\xi_1^j = \begin{cases} \hat{\xi}_1^j, j \in T & \text{with prob. } a_1^j(x_0^j, \hat{\xi}_1^j); \\ x_0^j, j \in \overline{T} & \text{with prob. } 1 - a_1^j(x_0^j, \hat{\xi}_1^j). \end{cases} \quad (13)$$

(2) Accept or reject ξ_1 by setting

$$x_1 = \begin{cases} \xi_1, \text{go to step (5)} & \text{if } \xi \in F; \\ \text{go to step (3)} & \text{if } \xi \notin F. \end{cases} \quad (14)$$

(3) Generate a new candidate state $\xi_2 = (\xi_2^1, \dots, \xi_2^N)$:
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

$$a_2^j(x_0^j, \xi_1^j, \hat{\xi}_2^j) = \min\left\{1, \frac{\pi_j(\hat{\xi}_2^j)S_1^j(\xi_1^j|\hat{\xi}_2^j)S_2^j(x_0^j|\hat{\xi}_2^j, \xi_1^j)a_1^j(\hat{\xi}_2^j, \xi_1^j)}{\pi_j(x_0^j)S_1^j(\xi_1^j|x_0^j)S_2^j(\hat{\xi}_2^j|x_0^j, \xi_1^j)a_1^j(x_0^j, \xi_1^j)}\right\}. \quad (15)$$

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

$$\xi_2^j = \begin{cases} \hat{\xi}_2^j, & \text{with prob. } a_2^j(x_0^j, \xi_1^j, \hat{\xi}_2^j); \\ x_0^j, & \text{with prob. } 1 - a_2^j(x_0^j, \xi_1^j, \hat{\xi}_2^j). \end{cases} \quad (16)$$

(4) Accept or reject ξ_2 by setting

$$x_1 = \begin{cases} \xi_2, & \text{if } \xi_2 \in F; \\ x_0, & \text{if } \xi_2 \notin F. \end{cases} \quad (17)$$

(5) End.

The MMHDR update is schematically shown in the Fig. 6. 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 the distribution $\pi(\cdot|F)$ invariant. The reader is referred to the Appendix for the proof.

The MMHDR algorithm preserves an interesting feature of the MHDR algorithm. Namely, the one-dimensional proposal distributions at the second stage are allowed to depend on the corresponding coordinates of the previously rejected candidate. The usage of information from the previously rejected candidate can potentially help us to generate a better candidate in the second stage that hopefully can be accepted as the new state of the Markov chain. However, to utilize the information about the rejected candidate and construct a more efficient proposal PDF is not a trivial task and it is left for future research. Nevertheless, MMHDR with any choice of S_1^j and S_2^j certainly reduces the overall probability of remaining in the current state if compared with the MMH algorithm and, therefore, leads to an improved sampler. This improvement is achieved at the expense of an additional computational cost. Thus, whether the MMHDR algorithm is useful for solving reliability problems depends on whether the gained reduction in variance compensates for the additional required computational effort.

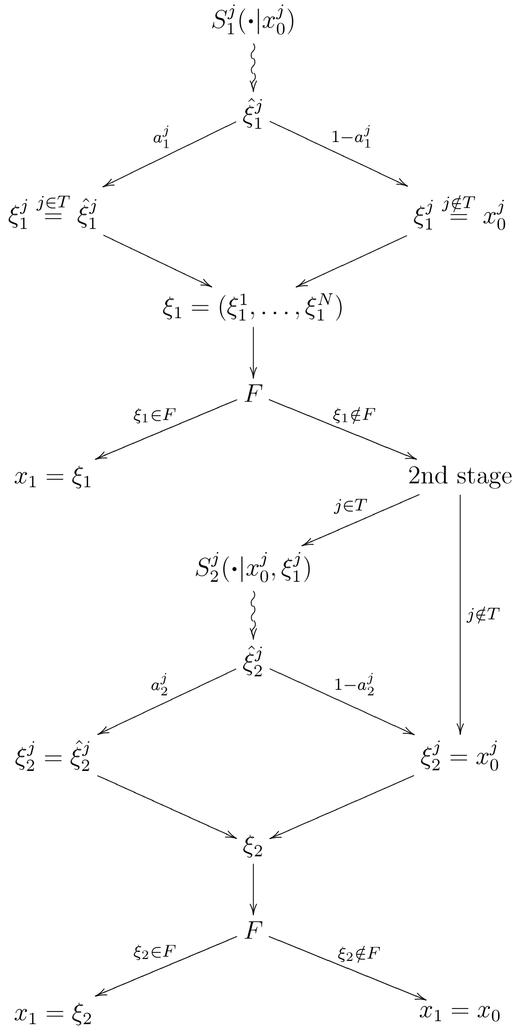


Fig. 6. Modified Metropolis–Hastings algorithms with delayed rejection.

3. Examples

To demonstrate the advantage of the MMHDR algorithm over the MMH algorithm we apply Subset simulation (SS) [1] with both MMHDR and MMH algorithms for evaluating the small failure probabilities of high-dimensional failure domains of different geometry.

3.1. *Linear failure domain*

At first, we consider a linear failure domain. Let $N = 1000$ be the dimension of the linear problem and $p_F = 10^{-5}$ be the failure probability. The failure domain F is defined as

$$F = \{x \in \mathbb{R}^N : \langle x, e\rangle \geq \beta\}, \quad (18)$$

where $e \in \mathbb{S}^N$ is a random unit vector, uniformly distributed on the unit sphere $\mathbb{S}^N$, 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.

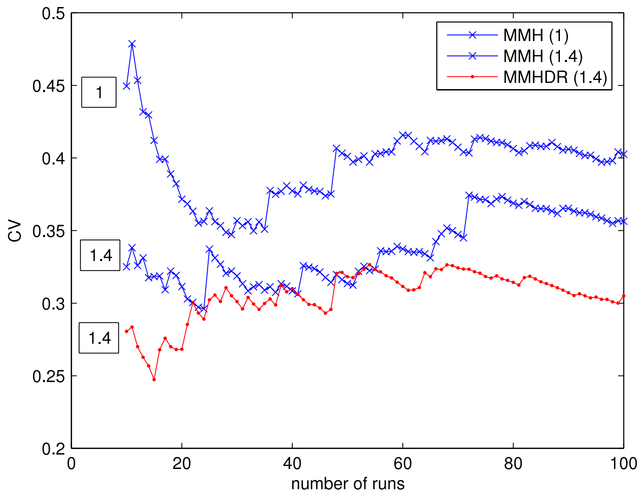


Fig. 7. The CV of the estimates obtained by Subset simulation with MMH and MMHDR algorithms.

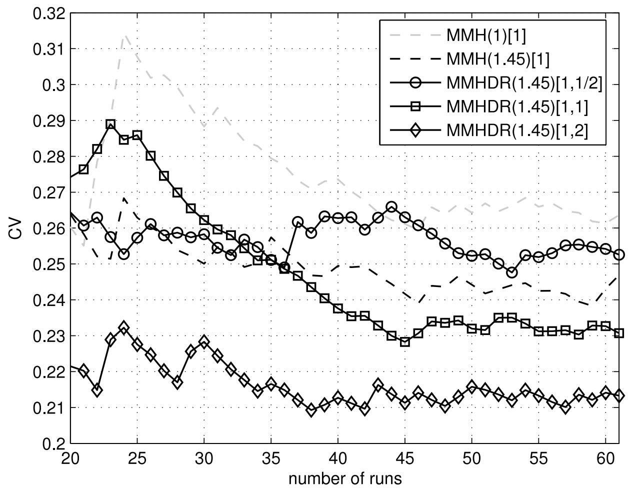


Fig. 8. The CV of the estimates obtained by Subset simulation with MMH and MMHDR algorithms.

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:

$$S^j(\cdot|x_0^j) = S_1^j(\cdot|x_0^j) = S_2^j(\cdot|x_0^j, \xi_1^j) = \mathcal{N}_{x_0^j,1}(\cdot). \tag{19}$$

Here, when the MMHDR algorithm is used, we do not change the second stage proposal distributions to include information from the rejected candidate ξ_1. How to generate a better candidate ξ_2 at the second stage including such information is in need of additional research. In this example we just want to check which one of the following two strategies is more effective: to have more Markov chains 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 the SS method against the number of runs is given in Fig. 7. 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 runs). Finally, the curve MMH(1.4) corresponds to the SS with MMH algorithm where for each intermediate subset $n = 1400$ 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 reduction in CV achieved with MMH(1.4) compared to MMH(1) is about 11% only. Simulation shows that to achieve the same CV as MMHDR(1.4), the total computational effort of MMH should be approximately 1.7. Thus, by employing MMHDR rather than MMH in linear problem, we save about 17% of the total computational effort.

3.2. Nonlinear failure domain of parabolic shape

Next, we consider a paraboloid in N-dimensional space defined as follows:

$$P: \quad x_1 = a\sum_{i=2}^{N} x_i^2 - b. \tag{20}$$

Define the failure domain as the interior of this paraboloid:

$$F = \left\{ x \in \mathbb{R}^N : x_1 > a\sum_{i=2}^{N} x_i^2 - b \right\}, \tag{21}$$

where $a = 0.025$, $b = 20.27$ and $N = 1000$. The probability of this parabolic failure domain calculated using standard Monte Carlo simulation (10^5 samples) is equal to $p_F = 7.8 \times 10^{-4}$ with CV $\delta = 0.11$.

The CVs of the failure probability estimates obtained by the SS method, using MMH and MMHDR, against the number of runs are given in Fig. 8. Here, similar to the previous example, we use the following notation:

- MMH(λ)[σ^2] denotes SS with the MMH algorithm, where λ is the total computational cost of this method ($\lambda = 1$ corresponds to the total cost when $n = 1000$ samples are used for each intermediate subset), and σ^2 is the variance of the proposal distribution, $S^j(\cdot|x_0^j) = \mathcal{N}_{x_0^j,\sigma^2}(\cdot)$;
- MMHDR(λ)[σ_1^2, σ_2^2] denotes SS with the MMHDR algorithm, where λ is the total computational cost of this method, and σ_1^2 and σ_2^2 are the variances of the proposal distributions in the first and in the second stage, respectively, i.e., $S_1^j(\cdot|x_0^j) = \mathcal{N}_{x_0^j,\sigma_1^2}(\cdot)$, $S_2^j(\cdot|x_0^j) = \mathcal{N}_{x_0^j,\sigma_2^2}(\cdot)$.

The total computational cost of MMHDR(1.45)[1, σ_2^2] is 45% higher than MMH(1)[1] and the maximum reduction in CV achieved (when $\sigma_2^2 = 2$) is about 20% (based on 61 runs). The total computational cost of MMH(1.45)(1) is the same as for MMHDR(1.45)[1, σ_2^2], however, the reduction in CV achieved is about 6% only. To achieve the same CV as MMHDR(1.4)[1, 2], the total computational effort of MMH should be approximately 1.61. Thus, by employing MMHDR rather than MMH in this parabolic problem, we save about 10% of the total computational effort. It is important to mention that the parabolic failure domain (21) provides an example of when the change of the second stage proposal distribution by increasing its variance is useful. Interestingly, this numerical result is consistent with a general observation about optimal scaling of the proposal PDFs made in [8]. Namely, it was proved that the optimal standard deviation of the N-variate Gaussian proposal PDF is approximately $\sigma \approx 2.4/\sqrt{N}$ and observed that "if one cannot be optimal, it seems better to use too high a value of σ than too low". In the considered numerical example an increased variance of the second stage proposal PDFs improves the performance of the MMHDR algorithm.

So, in the considered numerical examples, SS with the MMHDR algorithm outperforms SS with the MMH algorithm. In other words, "quality" (fewer Markov chains with less correlated states) defeats "quantity" (more Markov chains with more correlated states).

4. Conclusions

In this paper a novel modification of the MH algorithm, called the Modified Metropolis–Hastings algorithm with delayed rejection (MMHDR), is proposed. Based on two well-known sampling techniques: the Modified Metropolis–Hastings algorithm [1] and the Metropolis–Hastings algorithm with delayed rejection [7], the new algorithm is designed specially for sampling from high dimensional conditional distributions. The efficiency of the MMHDR algorithm is demonstrated with a numerical example where MMHDR is used together with Subset simulation for computing small failure probabilities in high dimensions.

Acknowledgements

This research has been supported by the Hong Kong Research Grants Council under grant 614008. This support is gratefully acknowledged. We are also grateful to the referees whose comments were very helpful in the revision of this paper.

Appendix

In this Appendix we prove that the 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 the updating process prescribed in the MMHDR algorithm, starting 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 some of the necessary 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

$$P(X_{n+1} \in A|X_n = x, X_j, j < n) = P(X_{n+1} \in A|X_n = x) \equiv K_n(x, A), \tag{22}$$

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 depend 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:

$$K(x, dy) = k(x, y)dy + r(x)\delta_x(dy). \tag{23}$$

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 the point mass at x (Dirac measure):

$$\delta_x(A) = \begin{cases} 1, & \text{if } x \in A; \\ 0, & \text{if } x \notin A. \end{cases} \tag{24}$$

Thus, the transition kernel (23) 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 $r(x)$. The transition kernel (23) is schematically shown in Fig. 9.

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

$$\pi(dx) = \pi(x)dx. \tag{25}$$

For simplicity, π will be used to denote both distribution and density. The probability distribution π is called invariant distribution

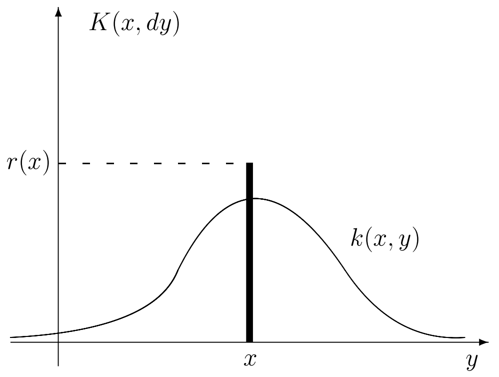


Fig. 9. Transition kernel with both continuous and discrete components.

for a transition kernel K if

$$\pi(dy) = \int_{x \in F} \pi(x)K(x, dy)dx. \tag{26}$$

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:

$$\pi(dx)K(x, dy) = \pi(dy)K(y, dx). \tag{27}$$

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), starting 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. Even if it has, it is hard to tell for sure.

Now let K denote the transition 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 $\pi(\cdot|F)$:*

$$\pi(dx_0|F)K(x_0, dx_1) = \pi(dx_1|F)K(x_1, dx_0). \tag{28}$$

Proof. By 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 (28) 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 the 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:

$$\begin{aligned} &\text{1st stage: } F \to \mathbb{R}^N, \quad x_0 \mapsto \xi_1, \\ &\text{2nd stage: } F \times \bar{F} \to F, \quad (x_0, \xi_1) \mapsto x_1. \end{aligned} \tag{29}$$

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

$$K(x_0, dx_1) = K_1(x_0, dx_1) + \int_{\xi \in \bar{F}} K_1(x_0, d\xi)K_2(x_0, \xi, dx_1). \tag{30}$$

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

$$\pi(dx_0)K_1(x_0, dx_1) = \pi(dx_1)K_1(x_1, dx_0). \quad (31)$$

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:

$$K_1(x_0, dx_1) = \prod_{j=1}^{N} K_1^j(x_0^j, dx_1^j), \quad (32)$$

where K_1^j is the transition kernel for the jth coordinate of x_0 at the first stage. Therefore, (31) can be equivalently rewritten in the coordinates as follows:

$$\prod_{j=1}^{N} \pi_j(dx_0^j)K_1^j(x_0^j, dx_1^j) = \prod_{j=1}^{N} \pi_j(dx_1^j)K_1^j(x_1^j, dx_0^j). \quad (33)$$

To prove (33) it is sufficient to show that for any $j = 1, \ldots, N$

$$\pi_j(dx_0^j)K_1^j(x_0^j, dx_1^j) = \pi_j(dx_1^j)K_1^j(x_1^j, dx_0^j). \quad (34)$$

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

$$K_1^j(x_0^j, dx_1^j) = k_1^j(x_0^j, x_1^j)dx_1^j + r_1^j(x_0^j)\delta_{x_0^j}(dx_1^j), \quad (35)$$

where

$$k_1^j(x_0^j, x_1^j) = S_1^j(x_1^j|x_0^j)a_1^j(x_0^j, x_1^j), \quad (36)$$

and a_1^j is given by (12), namely

$$a_1^j(x_0^j, x_1^j) = \min\left\{1, \frac{\pi_j(x_1^j)S_1^j(x_0^j|x_1^j)}{\pi_j(x_0^j)S_1^j(x_1^j|x_0^j)}\right\}. \quad (37)$$

Assume that $x_0^j \neq x_1^j$, since otherwise (34) 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:

$$\begin{aligned}
\pi_j(dx_0^j)K_1^j(x_0^j, dx_1^j) &= \pi_j(x_0^j)k_1^j(x_0^j, x_1^j)dx_0^j dx_1^j \\
&= \pi_j(x_0^j)S_1^j(x_1^j|x_0^j)\min\left\{1, \frac{\pi_j(x_1^j)S_1^j(x_0^j|x_1^j)}{\pi_j(x_0^j)S_1^j(x_1^j|x_0^j)}\right\} dx_0^j dx_1^j \\
&= \pi_j(x_1^j)S_1^j(x_0^j|x_1^j)\min\left\{1, \frac{\pi_j(x_0^j)S_1^j(x_1^j|x_0^j)}{\pi_j(x_1^j)S_1^j(x_0^j|x_1^j)}\right\} dx_0^j dx_1^j \\
&= \pi_j(x_1^j)k_1^j(x_1^j, x_0^j)dx_0^j dx_1^j = \pi_j(dx_1^j)K_1^j(x_1^j, dx_0^j). \quad (38)
\end{aligned}$$

So, Lemma 1 is proved. □

Remark 1. In essence, the proof of Lemma 1 repeats the one given in [1] for the Modified Metropolis–Hastings algorithm.

Thus, keeping in mind (30) and Lemma 1, it remains to show that for any $x_0, x_1 \in F$ and $x_0 \neq x_1$

$$\begin{aligned}
&\pi(dx_0)\int_{\xi\in\bar{F}} K_1(x_0, d\xi)K_2(x_0, \xi, dx_1) \\
&= \pi(dx_1)\int_{\xi\in\bar{F}} K_1(x_1, d\xi)K_2(x_1, \xi, dx_0). \quad (39)
\end{aligned}$$

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:

$$K_2(x_0, \xi, x_1) = \prod_{j=1}^{N} K_2^j(x_0^j, \xi^j, dx_1^j), \quad (40)$$

where K_2^j is the transition kernel for the jth coordinate of x_0 at the second stage. Therefore, (39) can be equivalently rewritten in terms of coordinates as follows:

$$\begin{aligned}
&\int_{\xi\in\bar{F}} \prod_{j=1}^{N} \pi_j(dx_0^j)K_1^j(x_0^j, d\xi^j)K_2^j(x_0^j, \xi^j, dx_1^j) \\
&= \int_{\xi\in\bar{F}} \prod_{j=1}^{N} \pi_j(dx_1^j)K_1^j(x_1^j, d\xi^j)K_2^j(x_1^j, \xi^j, dx_0^j). \quad (41)
\end{aligned}$$

To satisfy the condition (41) it is sufficient to show that for any $j = 1, \ldots, N$ the following holds:

$$\begin{aligned}
&\pi_j(dx_0^j)K_1^j(x_0^j, d\xi^j)K_2^j(x_0^j, \xi^j, dx_1^j) \\
&= \pi_j(dx_1^j)K_1^j(x_1^j, d\xi^j)K_2^j(x_1^j, \xi^j, dx_0^j). \quad (42)
\end{aligned}$$

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

$$K_2^j(x_0^j, \xi^j, dx_1^j) = \begin{cases} \delta_{x_0^j}(dx_1^j), & \text{if } \xi^j = x_0^j; \\ k_2^j(x_0^j, \xi^j, x_1^j)dx_1^j + r_2^j(x_0^j, \xi^j)\delta_{x_0^j}(dx_1^j), \\ \quad \text{if } \xi^j \neq x_0^j, \end{cases} \quad (43)$$

where

$$k_2^j(x_0^j, \xi^j, x_1^j) = S_2^j(x_1^j|x_0^j, \xi^j)a_2^j(x_0^j, \xi^j, x_1^j), \quad (44)$$

and a_2^j is given by (15), namely

$$\begin{aligned}
&a_2^j(x_0^j, \xi^j, x_1^j) \\
&= \min\left\{1, \frac{\pi_j(x_1^j)S_1^j(\xi^j|x_1^j)S_2^j(x_0^j|x_1^j, \xi^j)a_1^j(x_1^j, \xi^j)}{\pi_j(x_0^j)S_1^j(\xi^j|x_0^j)S_2^j(x_1^j|x_0^j, \xi^j)a_1^j(x_0^j, \xi^j)}\right\}. \quad (45)
\end{aligned}$$

Assume that $x_0^j \neq x_1^j$, since otherwise condition (42) 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).

1. $\xi^j \neq x_0^j, \xi^j \neq x_1^j$.
   In this case we have:

$$\begin{aligned}
&\pi_j(dx_0^j)K_1^j(x_0^j, d\xi^j)K_2^j(x_0^j, \xi^j, dx_1^j) \\
&= \pi_j(x_0^j)k_1^j(x_0^j, \xi^j)k_2^j(x_0^j, \xi^j, x_1^j)dx_0^j d\xi^j dx_1^j \\
&= \pi_j(x_0^j)S_1^j(\xi^j|x_0^j)a_1^j(x_0^j, \xi^j)S_2^j(x_1^j|x_0^j, \xi^j) \\
&\quad \times \min\left\{1, \frac{\pi_j(x_1^j)S_1^j(\xi^j|x_1^j)S_2^j(x_0^j|x_1^j, \xi^j)a_1^j(x_1^j, \xi^j)}{\pi_j(x_0^j)S_1^j(\xi^j|x_0^j)S_2^j(x_1^j|x_0^j, \xi^j)a_1^j(x_0^j, \xi^j)}\right\} \\
&\quad \times dx_0^j d\xi^j dx_1^j \\
&= \pi_j(x_1^j)S_1^j(\xi^j|x_1^j)a_1^j(x_1^j, \xi^j)S_2^j(x_0^j|x_1^j, \xi^j) \\
&\quad \times \min\left\{1, \frac{\pi_j(x_0^j)S_1^j(\xi^j|x_0^j)S_2^j(x_1^j|x_0^j, \xi^j)a_1^j(x_0^j, \xi^j)}{\pi_j(x_1^j)S_1^j(\xi^j|x_1^j)S_2^j(x_0^j|x_1^j, \xi^j)a_1^j(x_1^j, \xi^j)}\right\} \\
&\quad \times dx_0^j d\xi^j dx_1^j \\
&= \pi_j(x_1^j)k_1^j(x_1^j, \xi^j)k_2^j(x_1^j, \xi^j, x_0^j)dx_0^j d\xi^j dx_1^j \\
&= \pi_j(dx_1^j)K_1^j(x_1^j, d\xi^j)K_2^j(x_1^j, \xi^j, dx_0^j). \quad (46)
\end{aligned}$$

   So, in this case (42) is fulfilled.
2. $\xi^j = x_0^j$.
   In this case the left-hand side of (42) 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 (42) when $\xi^j = x_0^j$:

$$\pi_j(dx_1^j)K_1^j(x_1^j, dx_0^j)K_2^j(x_1^j, x_0^j, dx_0^j) = \pi_j(dx_1^j)k_1^j(x_1^j, x_0^j) \times \left(k_2^j(x_1^j, x_0^j, x_0^j)dx_0^j + r_2^j(x_1^j, x_0^j)\delta_{x_1^j}(dx_0^j)\right) dx_0^j$$
$$= \pi_j(dx_1^j)k_1^j(x_1^j, x_0^j)k_2^j(x_1^j, x_0^j, x_0^j)dx_0^j dx_0^j = 0, \quad (47)$$

since $dx_0^j dx_0^j = 0$, which is a standard result of the measure theory. So, the right-hand side of (42) is also zero. Basically, when $\xi^j = x_0^j$ the left-hand side of (42) is zero by definition of the MMHDR algorithm: the 2nd stage 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 infinitesimally small.

3. $\xi^j = x_1^j$.
   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 the MMHDR algorithm is used together with Subset simulation the starting state of the Markov chain is already distributed according to $\pi(\cdot|F)$. This 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 (28).

References

[1] Au SK, Beck JL. Estimation of small failure probabilities in high dimensions by subset simulation. Probabilistic Engineering Mechanics 2001;16(4):263–77.
[2] Katafygiotis LS, Zuev KM. Estimation of small failure probabilities in high dimensions by Adaptive Linked Importance Sampling, COMPDYN 2007, Rethymno Crete Greece; 13–16 June 2007.
[3] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculations by fast computing machines. Journal of Chemical Physics 1953;21:1087–92.
[4] Hastings WK. Monte Carlo sampling methods using Markov chains and their applications. Biometrika 1970;57:97–109.
[5] Katafygiotis LS, Zuev KM. Geometric insight into the challenges of solving high-dimensional reliability problems. Probabilistic Engineering Mechanics 2008; 23:208–18.
[6] Schuëller GI, Pradlwarter HJ, Koutsourelakis PS. A critical appraisal of reliability estimation procedures for high dimensions. Probabilistic Engineering Mechanics 2004;19:463–74.
[7] Tierney L, Mira A. Some adaptive Monte Carlo methods for Bayesian inference. Statistics in Medicine 1999;18:2507–15.
[8] Gelman A, Roberts GO, Gilks WR. Efficient metropolis jumping rules. Bayesian Statistics 1996;5:599–607.