Multicanonical sequential Monte Carlo sampler for uncertainty quantification

Millar, Robert; Li, Hui; Li, Jinglai

DOI:
10.1016/j.ress.2023.109316

Reliability Engineering and System Safety 237 (2023) 109316

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

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

Multicanonical sequential Monte Carlo sampler for uncertainty quantification

Robert Millar *, Hui Li, Jinglai Li

School of Mathematics, University of Birmingham, Birmingham B15 2TT, UK

ARTICLE INFO

Keywords:
Multicanonical Monte Carlo
Sequential Monte Carlo sampler
Rare event simulation
Uncertainty quantification

ABSTRACT

In many real-world engineering systems, the performance or reliability of the system is characterised by a scalar variable. The distribution of this performance variable is important in many uncertainty quantification problems, ranging from risk management to utility optimisation. In practice, this distribution usually cannot be derived analytically and has to be obtained numerically by simulations. To this end, standard Monte Carlo simulations are often used, however, they cannot efficiently reconstruct the tail of the distribution which is essential in many applications. One possible remedy is to use the Multicanonical Monte Carlo method, an adaptive importance sampling scheme. In this method, one draws samples from an importance sampling distribution in a nonstandard form in each iteration, which is usually done via Markov chain Monte Carlo (MCMC). MCMC is inherently serial and therefore struggles with parallelism. In this paper, we present a new approach, which uses the Sequential Monte Carlo sampler to draw from the importance sampling distribution, which is particularly suited for parallel implementation. With both mathematical and practical examples, we demonstrate the competitive performance of the proposed method.

1. Introduction

Real-world engineering systems are unavoidably subject to uncertainty, rising from various sources: material properties, geometric parameters, external perturbations and so on, and these uncertainty factors can certainly affect the system performance and reliability (i.e., the ability of a system to perform its intended functions). In the context of reliability engineering (RE), it is vital to characterise and quantify the impact of such uncertainties on the system performance or reliability, which constitutes a central task in the field of uncertainty quantification (UQ). Mathematical models and simulations are important tools to assess how engineering systems are impacted by uncertainty. Within these, the system performance or reliability is often characterised by a scalar variable y, which we will now refer to as the *performance variable*. This performance variable can be expressed by a performance function $y = g(\mathbf{x})$, where $\mathbf{x}$ is a multi-dimensional random variable representing all the uncertainty factors affecting the system; the performance function is usually not of analytical form and needs to be evaluated by simulating the underlying mathematical model. A typical example is in structural engineering, where the performance variable y represents the deformation of some key components.

Several advanced Monte Carlo techniques have been developed in reliability engineering to provide a variance-reduced estimator for a specific quantity associated with the distribution of y, such as the probability of a given random event, or failure probability. These include the cross-entropy method [1,2], subset simulation [3], sequential Monte Carlo [4], and Hierarchical partitioning strategy [5].

Whilst many methods exist for the purpose of estimating failure probability, for example, using failure probability functions [6,7], research into techniques for reconstructing the full distribution associated with the performance variable y is limited.

Obtaining the complete distribution of the performance variable is important in many UQ and RE problems, ranging from risk management to utility optimisation, where these problems may demand various statistical information of the performance y: for example, in robust optimisation, the interests are predominantly in the mean and variance [8], in risk management, one is interested in the tail probability as well as some extreme quantiles [9], and in utility optimisation, the complete distribution of the performance variable is required [10]. To this end, methods that can efficiently reconstruct the probability distribution of the performance variable directly are strongly desirable, however, they receive little research, in part because of the high computational cost associated with such methods.

Motivated by this, we propose an adaptation to an existing method for reconstructing the probability distribution of y, which has a significantly lower computational cost than existing methods, by taking advantage of parallel computing.

* Corresponding author.
E-mail addresses: r.millar.1@bham.ac.uk (R. Millar), h.li.4@bham.ac.uk (H. Li), j.li.10@bham.ac.uk (J. Li).

https://doi.org/10.1016/j.ress.2023.109316
Received 22 September 2022; Received in revised form 16 March 2023; Accepted 15 April 2023
Available online 20 April 2023

In our previous works [11,12], we proposed using the Multicanonical Monte Carlo (MMC) method for computing the distribution of y. The MMC method is a special adaptive importance sampling (IS) scheme, which was initially developed by Berg and Neuhaus [13,14] to explore the energy landscape of a given physical system.

In the MMC method, one splits the state space of the performance variable of interest into a set of small bins and then iteratively constructs a so-called flat-histogram distribution that can assign equal probability to each of the bins. This allows for the construction of the entire distribution function of the performance variable, significantly more efficiently than using standard MC.

A key characteristic of MMC is that, within each iteration, samples are drawn from an IS distribution in a nonstandard form, which is usually done via Markov chain Monte Carlo (MCMC). MCMC is inherently serial [15], in that it relies on the convergence of a single Markov chain to its stationary distribution, and therefore often struggles with parallelism. As a result, the MMC method implemented with MCMC (referred to as MMC-MCMC hereafter) cannot take advantage of high-powered parallel computing. There are further limitations to MCMC – detailed in Section 2.4 – which reduce the overall efficiency of the MMC-MCMC method.

Many alternative sampling methods exist, for example, Hamiltonian Monte Carlo [16], and are well summarised in a recent review paper by Tabandeh et al. [17]. However, they are not all readily parallelisable, with the exception of the Sequential Monte Carlo sampler. As such, we propose using the Sequential Monte Carlo sampler (SMCS), to draw samples from the IS distributions in each MMC step. The SMCS method, first developed in [18], can fulfil the same role as MCMC in that, by conducting sequential IS for a sequence of intermediate distributions, it can generate (weighted) samples from an arbitrary target distribution.

The reason that we choose to implement MMC with the SMCS method is two-fold: first, since SMCS is essentially an IS scheme, it is easily parallelisable; second, SMCS can take advantage of a sequence of intermediate distributions, allowing it to be effectively integrated into the MMC scheme. Both points will be elaborated on later.

The rest of the paper is organised as follows. In Section 2, we present the Multicanonical Monte Carlo method and in Section 3, the Sequential Monte Carlo sampler. We bring these techniques together in Section 4 to present the proposed *Multicanonical Sequential Monte Carlo Sampler* and then apply this to various numerical examples in Section 5. Finally, Section 6 provides concluding remarks.

2. Multicanonical Monte Carlo method

2.1. Problem setup and the Monte Carlo estimation

We start with a generic setup of the problems considered here. Let $\mathbf{x}$ be a d-dimensional random vector following distribution $p_{\mathbf{x}}(\cdot)$ and let y be a scalar variable characterised by a function $y = g(\mathbf{x})$. We want to determine the probability density function (PDF) of y, given by $\pi(y)$, where we assume that both $\mathbf{x}$ and y are continuous random variables.

We now discuss how to estimate the PDF using the standard MC simulation. For the sake of convenience, we assume that $\pi(y)$ has bounded support $R_y = [a, b]$ and if the support of $\pi(y)$ is not bounded, we choose the interval $[a, b]$ that is sufficiently large so that $\mathbb{P}(y \in [a, b]) \approx 1$. We first decompose R_y into M bins of equal width Δ centred at the discrete values $\{b_1, \dots, b_M\}$ and define the ith bin as the interval $B_i = [b_i - \Delta/2, b_i + \Delta/2]$. This binning implicitly defines a partition of the input space X into M domains $\{D_i\}_{i=1}^M$, where

$$D_i = \{\mathbf{x} \in X : g(\mathbf{x}) \in B_i\} \tag{1}$$

is the domain in X that maps into the ith bin B_i (see Fig. 1).

A key consideration when using MMC is the optimal choice of bin width. It has been empirically found that adjacent bins should have probability within one order of magnitude [19].

While B_i are simple intervals, the domains D_i are multidimensional regions with possibly tortuous topologies. Therefore, an indicator function is used to classify whether a given $\mathbf{x}$-value is in the bin D_i or not. Formally, the indicator function is defined as,

$$I_{D_i}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \in D_i; \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

or equivalently $\{y = g(\mathbf{x}) \in B_i\}$. By using this indicator function, the probability that y is in the ith bin, i.e. $P_i = \mathbb{P}\{y \in B_i\}$, can be written as an integral in the input space:

$$P_i = \int_{D_i} p(\mathbf{x})d\mathbf{x} = \int I_{D_i}(\mathbf{x})p(\mathbf{x})d\mathbf{x} = \mathbb{E}[I_{D_i}(\mathbf{x})]. \tag{3}$$

We can estimate P_i via a standard MC simulation. Namely, we draw N i.i.d. samples $\{\mathbf{x}^1, \dots, \mathbf{x}^N\}$ from the distribution $p(\mathbf{x})$ and calculate the MC estimator of P_i as

$$\hat{P}_i^{MC} = \frac{1}{N}\sum_{j=1}^{N} I_{D_i}(\mathbf{x}^j) = \frac{N_i}{N}, \quad \text{for } i = 1, \dots, M, \tag{4}$$

where N_i is the number of samples that fall in bin B_i.

Once we have obtained $\{P_i\}_{i=1}^M$, the PDF of y at the point $y_i \in B_i$ – for a sufficiently small Δ – can be calculated as $\pi(y_i) \approx P_i/\Delta$.

2.2. Flat histogram importance sampling

The MC approach can be improved through the use of Importance Sampling. Here IS is used to artificially increase the number of samples falling in the tail bins of the histogram. Given an IS distribution $q(\mathbf{x})$, Eq. (3) can be re-written as

$$P_i = \int I_{D_i}(\mathbf{x})[\frac{p(\mathbf{x})}{q(\mathbf{x})}]q(\mathbf{x})d\mathbf{x} = \mathbb{E}_q[I_{D_i}(\mathbf{x})w(\mathbf{x})] \tag{5}$$

where $w(\mathbf{x}) = p(\mathbf{x})/q(\mathbf{x})$ is the IS weight and $\mathbb{E}_q$ indicates expectation with respect to the IS distribution $q(\mathbf{x})$. The IS estimator for P_i can then be written as follows:

$$\hat{P}_i^{IS} = \left[\frac{1}{N}\sum_{j=1}^{N} I_{D_i}(\mathbf{x}^j)w(\mathbf{x}^j)\right] \tag{6}$$

for each bin $i = 1, \dots, M$.

As is well known, key to the successful implementation of IS is identifying a good IS distribution $q(\mathbf{x})$, which is particularly challenging for the present problem, as we are interested in multiple estimates (i.e. $P_1, \dots, P_M$) rather than a single one, as in conventional IS problems.

The solution provided by MMC is to use the so-called *uniform weight flat-histogram (UW-FH)* IS distribution. The UW-FH IS distribution is designed to achieve the following two goals. First, it should allocate the same probability to each bin, that is, assuming $x \sim q(x)$,

$$P_i^* := \mathbb{P}(y = g(\mathbf{x}) \in B_i) = 1/M,$$

for all i. Intuitively, this property allows all bins to be equally visited by the samples generated from the IS distribution. Second, it should assign a constant weight to all samples falling in the same bin, that is, $w(\mathbf{x}) = \Theta_i$ for all $\mathbf{x} \in D_i$, where Θ_i is a positive constant. Loosely speaking, the second property ensures that all samples falling in the same bin are equally good.

The UW-FH distribution can be expressed in the form of:

$$q(\mathbf{x}) \propto \begin{cases} \frac{p(\mathbf{x})}{c_\Theta \Theta(\mathbf{x})}, & \mathbf{x} \in D, \\ 0, & \mathbf{x} \notin D, \end{cases} \tag{7}$$

where $\Theta(\mathbf{x}) = \Theta_i$ for $\mathbf{x} \in D_i$, $i = 1, \dots, M$, and c_Θ is a normalising constant. It is easy to see that,

$$P_i^* = \int_{D_i} q(\mathbf{x})d\mathbf{x} = \frac{\int_{D_i} p(\mathbf{x})d\mathbf{x}}{c_\Theta \Theta_i} = \frac{P_i}{c_\Theta \Theta_i}. \tag{8}$$

Recall that $P_i^* = 1/M$ for all i, so it follows $\Theta_i \propto P_i$, i.e. Θ_i is proportional to the sought probability P_i, and $c_\Theta = \sum_{i=1}^M \frac{P_i}{\Theta_i}$.

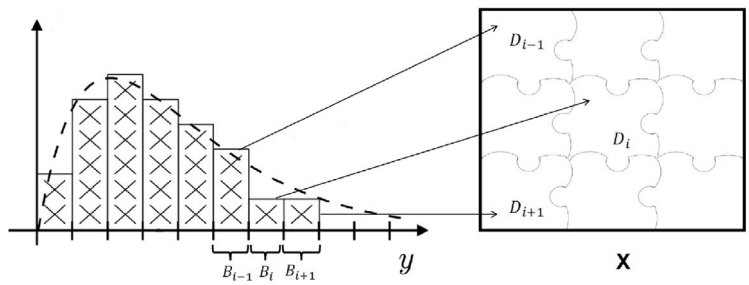

Fig. 1. Schematic illustration of the connection between B_i and D_i.
Source: This figure is reprinted from [11].

2.3. Multicanonical Monte Carlo

The UW-FH distribution, given by Eq. (7), cannot be used directly as Θ_i depends on the sought-after unknown P_i. The MMC method iteratively addresses this, starting from the original input PDF $p(\mathbf{x})$.

Simply put, starting with $q_0(\mathbf{x})$ and $\Theta_{0,i} = p$ for all $i = 1, \dots, M$, where $p = \sum_{i=1}^{M} P_i$, the MMC method iteratively constructs a sequence of distributions (for $t \geq 1$),

$$q_t(\mathbf{x}) \propto \begin{cases} \frac{p(\mathbf{x})}{c_t \Theta_t(\mathbf{x})}, & \mathbf{x} \in D; \\ 0, & \mathbf{x} \notin D. \end{cases} \tag{9}$$

where $\Theta_t(\mathbf{x}) = \Theta_{t,i}$ for $\mathbf{x} \in D_i$ and c_t is the normalising constant for q_t. Ideally, we want to construct q_t in a way that it converges to the actual UW-FH distribution as t increases. The key here is to estimate the values of $\{\Theta_{t,i}\}_{i=1}^{M}$. It is easy to see that when q_t is used as the IS distribution, we have $P_i = c_t P_i^* \Theta_{t,i}$.

That is, in the tth iteration, one draws N samples $\{\mathbf{x}^j\}_{j=1}^{N}$ from the current IS distribution $q_t(\mathbf{x})$, then updates $\{\Theta_{t+1,i}\}_{i=1}^{M}$ using the following formulas,

$$\hat{H}_{t,i} = \frac{N_{t,i}^*}{N} \tag{10a}$$

$$P_{t,i} = \hat{H}_{t,i}\, \Theta_{t,i} \tag{10b}$$

$$\Theta_{t+1,i} = P_{t,i} \tag{10c}$$

where $N_{t,i}^*$ is the number of samples falling into region D_i in the tth iteration. Note that in Eq. (10b) we neglect the normalising constant c_t as it is not needed in the algorithm, which will become clear later. The process is then repeated, until the resulting histogram is sufficiently "flat" (see e.g. [20]).

2.4. The limitation of MCMC

To implement the MMC method, one must be able to generate samples from the IS distribution $q_t(\cdot)$ at each iteration. Typically, this is done using Markov Chain Monte Carlo (MCMC). Simply speaking, MCMC constructs a Markov Chain that converges to the target distribution. It is convenient to use as it only requires the ability to evaluate the target PDF up to a normalising constant (and therefore the knowledge of c_t in Eq. (9) is not needed). The core of MCMC is to construct a single Markov chain converging to its stationary distribution, which often takes a very large number of iterations (known as the burn-in period) to be achieved. The process cannot be easily accelerated by parallel processing. We note here that there are some MCMC variants, e.g. [21], that attempt to exploit parallel implementation; however, to the best of our knowledge, none of these methods can take full advantage of modern parallel computing power. For example, the multi-chain MCMC algorithms can be implemented in parallel but every single chain still requires a long burn-in period before it converges to the target distribution. As a result, MMC-MCMC cannot fully exploit the potential provided by high-performance parallel computing available nowadays. In this work, we want to provide an alternative implementation of MMC, which is based on the sequential Monte Carlo sampler.

3. Sequential Monte Carlo sampler

First proposed in [18], SMCS is an IS method for drawing samples from a sequence of distributions $\{q_t(\cdot)\}_{t=1}^{T}$. It is a generalisation of the particle filter [22], where weighted samples are generated in a sequential manner. Several extensions to this method have been proposed, e.g. [23–26], with the latest advances being summarised in two recent reviews [27,28].

Suppose we have samples following distribution $q_{t-1}(\cdot)$ but want them to follow $q_t(\cdot)$ instead, we can use SMCS. First, a forward Kernel is applied to each of the current samples – sometimes with an acceptance criterion – and then a weight is calculated for each new sample. Finally, if the effective sample size across all the samples is below a certain threshold (usually less than half the total number of samples) the proposed samples are resampled. These new weighted samples follow the distribution $q_t(\cdot)$.

We present the SMCS method in a recursive formulation, largely following the presentation of [18,29]. Suppose that at time $t-1$, we have an IS distribution $\gamma_{t-1}(\mathbf{x}_{t-1})$, from which we can generate, or already have, an ensemble of N samples $\{\mathbf{x}_{t-1}\}_{j=1}^{N}$. To implement SMCS, we first choose two conditional distributions $K_t(\cdot|x_{t-1})$ and $L_{t-1}(\cdot|x_t)$, referred to as the forward and backward kernels respectively. Using $L_{t-1}(\cdot|x_t)$, we are able to construct a joint distribution of $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ in the form of

$$r_t(\mathbf{x}_{t-1}, \mathbf{x}_t) = q_t(\mathbf{x}_t) L_{t-1}(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{11}$$

such that the marginal distribution of $r_t(\mathbf{x}_{t-1}, \mathbf{x}_t)$ over $\mathbf{x}_{t-1}$ is $q_t(\mathbf{x}_t)$. Now, using $\gamma_{t-1}(\mathbf{x}_{t-1})$ and the forward Kernel $K_t(\mathbf{x}_t|\mathbf{x}_{t-1})$, we can construct an IS distribution for $r_t(\mathbf{x}_{t-1}, \mathbf{x}_t)$ in the form of

$$\gamma(\mathbf{x}_{t-1}, \mathbf{x}_t) = \gamma_{t-1}(\mathbf{x}_{t-1}) K_t(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{12}$$

One can draw samples from this joint IS distribution $\gamma(\mathbf{x}_{t-1}, \mathbf{x}_t)$ using $\{\mathbf{x}_{t-1}\}_{j=1}^{N}$ and the forward kernel K_t, to obtain an ensemble $\{(\mathbf{x}_{t-1}^j, \mathbf{x}_t^j)\}_{j=1}^{N}$ from $\gamma(\mathbf{x}_{t-1}, \mathbf{x}_t)$. The corresponding weights are computed as

$$\begin{aligned} w_t(\mathbf{x}_{t-1:t}) &= \frac{r_t(\mathbf{x}_{t-1}, \mathbf{x}_t)}{\gamma(\mathbf{x}_{t-1}, \mathbf{x}_t)} = \frac{q_t(\mathbf{x}_t)\; L_{t-1}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\gamma_{t-1}(\mathbf{x}_{t-1})\; K_t(\mathbf{x}_t|\mathbf{x}_{t-1})} \\ &= w_{t-1}(\mathbf{x}_{t-1}) \alpha(\mathbf{x}_{t-1}, \mathbf{x}_t) \end{aligned} \tag{13a}$$

where

$$\begin{aligned} w_{t-1}(\mathbf{x}_{t-1}) &= \frac{q_{t-1}(\mathbf{x}_{t-1})}{\gamma_{t-1}(\mathbf{x}_{t-1})}, \\ \alpha_t(\mathbf{x}_{t-1}, \mathbf{x}_t) &= \frac{q_t(\mathbf{x}_t)\; L_{t-1}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q_{t-1}(\mathbf{x}_{t-1})\; K_t(\mathbf{x}_t|\mathbf{x}_{t-1})} \end{aligned} \tag{13b}$$

As such the weighted ensemble $\{\mathbf{x}_{t-1:t}^j, w_t^j\}_{j=1}^{N}$ follows the joint distribution $r_t(\mathbf{x}_{t-1:t})$ and as such, $\{\mathbf{x}_t^j, w_t^j\}_{j=1}^{N}$ follows the marginal distribution q_t. By repeating this procedure we can obtain weighted samples from the sequence of distributions $\{q_t\}_{t=1}^{T}$.

For the SMCS method, the choice of forward and backward kernels are essential. While noting that there are a number of existing methods for determining the forward kernel, we adopt the MCMC kernel

proposed in [18], which is closely related to the Metropolis step in MCMC as the name suggests. Specifically, the forward kernel (more precisely the process for generating samples from the forward kernel) is constructed as follows. A proposal distribution $k(\mathbf{x}_t|\mathbf{x}_{t-1})$ is chosen and with a sample from the previous iteration $\mathbf{x}_{t-1}^j$, we draw a sample $\mathbf{x}_t^*$ from $k(\mathbf{x}_t|\mathbf{x}_{t-1}^j)$, and then accept (or reject) $\mathbf{x}_t^*$ according to the following acceptance probability:

$$a_t(\mathbf{x}_t^*|\mathbf{x}_{t-1}^j) = \min\left\{\frac{q_t(\mathbf{x}_t^*)}{q_t(\mathbf{x}_{t-1}^j)}\frac{k(\mathbf{x}_{t-1}^j|\mathbf{x}_t^*)}{k(\mathbf{x}_t^*|\mathbf{x}_{t-1}^j)}, 1\right\}. \tag{14}$$

That is, we set

$$\mathbf{x}_t^j = \begin{cases}\mathbf{x}_t^*, & \text{with probability } a_t(\mathbf{x}_t^*|\mathbf{x}_{t-1}^j)\\ \mathbf{x}_{t-1}^j, & \text{otherwise.}\end{cases} \tag{15}$$

Once a forward Kernel $K_t(\mathbf{x}_t|\mathbf{x}_{t-1})$ is chosen, one can determine an optimal choice of L_{t-1} by:

$$\begin{aligned} L_{t-1}^{opt}(\mathbf{x}_{t-1}|\mathbf{x}_t) &= \frac{q_{t-1}(\mathbf{x}_{t-1})K_t(\mathbf{x}_t|\mathbf{x}_{t-1})}{q_t(\mathbf{x}_t)}\\ &= \frac{q_{t-1}(\mathbf{x}_{t-1})K_t(\mathbf{x}_t|\mathbf{x}_{t-1})}{\int q_{t-1}(\mathbf{x}_{t-1})K_t(\mathbf{x}_t|\mathbf{x}_{t-1})d\mathbf{x}_{t-1}}, \end{aligned} \tag{16}$$

where the optimality is achieved through yielding the minimal estimator variance [18]. In reality, this optimal backward kernel usually cannot be used directly as the integral on the denominator cannot be calculated analytically. However, when the MCMC kernel is used, an approximate optimal kernel can be derived from Eq. (16):

$$L_{t-1}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \frac{q_t(\mathbf{x}_{t-1})K_t(\mathbf{x}_t|\mathbf{x}_{t-1})}{q_t(\mathbf{x}_t)}, \tag{17}$$

the detailed derivation can be found in [18]. When Eq. (17) is used, the incremental weight function $\alpha_t(\mathbf{x}_{t-1}, \mathbf{x}_t)$ in Eq. (13b), reduces to the following:

$$\alpha_t(\mathbf{x}_{t-1}, \mathbf{x}_t) = \frac{q_t(\mathbf{x}_{t-1})}{q_{t-1}(\mathbf{x}_{t-1})}. \tag{18}$$

Note that, interestingly only the previous sample is used in the weight calculation when Eq. (17) is used. In our method, we use the MCMC kernel and Eq. (17) as the forward and backward kernels respectively.

To alleviate sample degeneracy, a key step in SMCS is the resampling of samples according to their associated weights. The resampling algorithms are well documented, e.g. [30], and are not discussed here. In SMCS, typically resampling is conducted when the effective samples size (ESS) [31] is lower than a prescribed threshold value $ESS_{\min}$. To conclude, we provide the complete procedure of SMCS in Algorithm 1, to generate N samples from the target distribution $q_t(\cdot)$.

Algorithm 1 Sequential Monte Carlo Sampler

input: weighted ensemble $\{(x_{t-1}^j, w_{t-1}^j)\}_{j=1}^N$
for $j = 1$ to N
  (a) draw $\mathbf{x}_t^*$ from $k(\cdot|\mathbf{x}_{t-1}^j)$
  (b) calculate the acceptance probability $a(\mathbf{x}_t^*, \mathbf{x}_{t-1}^j)$ using Eq. (14)
  (c) determine $\mathbf{x}_t^j$ using Eq. (15) and $a(\mathbf{x}_t^*, \mathbf{x}_{t-1}^j)$
  (d) calculate α_t^j using Eq. (18)
  (e) compute $w_t^j = w_{t-1}^j \alpha_t^j$
end for
normalize the weights calculated
calculate ESS
if $ESS < ESS_{\min}$
  resample the ensemble and set $w_t^j = 1/N$ for $j = 1, \ldots, N$
end if

As one can see from Algorithm 1, the SMCS algorithm is easily parallelizable, which is the main advantage over MCMC for our purposes. In addition, since SMCS is designed for sampling from a sequence of target distributions, it can naturally take advantage of the similarity between two successive target distributions, like the warped distributions in two consecutive iterations of MMC, which will be further demonstrated in Section 4.

4. Multicanonical sequential Monte Carlo sampler

Our proposed algorithm, termed as the *Multicanonical Sequential Monte Carlo Sampler* (MSMCS) uses SMCS to generate the samples in each MMC iteration. As has been shown in Section 3, SMCS can naturally be used to generate samples from a sequence of target distributions and is therefore well suited for MMC, where the biasing distributions within each MMC iteration can be considered as a sequence of distributions. Though the implementation seems straightforward, there are still some issues that need to be addressed within the proposed MSMCS method.

In the standard MMC method, using MCMC (denoted by MMC-MCMC), the samples generated are unweighted and as such, the update procedure for Θ's – determined by the proportion of samples landing in each bin – is based on the samples being unweighted. However, as SMCS produces weighted samples, we need to adapt the MMC procedure to account for this, by altering the update procedure for the Theta distributions. Specifically, we change how the value of $\hat{H}_{t,i}$ – the estimator of P_i – is determined. The update procedure, when using unweighted samples, is determined by Eq. (10). When SMCS is used, the update procedure needs to be modified, specifically Eq. (10a) becomes,

$$\hat{H}_{t,i} = \sum_{j=1}^{N} I_{D_i}(\mathbf{x}^j)\, w(\mathbf{x}^j). \tag{19}$$

Another issue is that, for SMC to be effective, two successive distributions cannot be too far apart from each other; otherwise, the samples are very likely to be rejected in the Metropolis step. Within the MMC method, there is no guarantee that the IS distributions obtained in two successive iterations are close to each other. For example, in our numerical experiments, we have observed that, for high-dimensional problems, such an issue appears frequently in the first MMC step, due to the difference in the initial distribution $q_0(\mathbf{x})$ and subsequent target distribution $q_1(\mathbf{x})$.

To address this issue, we propose including a simulated tempering process in the method. Namely, we introduce a set of intermediate distributions in between q_t and q_{t+1}, which we can apply SMCS to. Note that the difference in the IS distributions can be attributed to differences in the Θ-functions (i.e. $\Theta_t(x)$ and $\Theta_{t+1}(x)$), as per Eq. (9). We choose a strictly increasing sequence of scalars $\{\alpha_k\}_{k=1}^K$ with $\alpha_0 = 0$ and $\alpha_K = 1$, such that the intermediate Θ-functions are

$$\Theta_k(\mathbf{x}) = \alpha_k\, \Theta_{t+1}(\mathbf{x}) + (1 - \alpha_k)\, \Theta_t(\mathbf{x}). \tag{20}$$

It follows that the sequence of intermediate distributions $\{q_k\}_{k=0}^K$ can be defined accordingly via Eq. (9) and we apply SMCS to this sequence of distributions ultimately yielding samples from the target distribution $q_{t+1}(x)$. One can see that when q_t and q_{t+1} are close to each other, SMCS can efficiently generate samples from q_{t+1} via the forward kernel and the samples from q_t, so this tempering process is not needed. However, for two consecutive IS distributions that are far apart, we found that whilst introducing more intermediate steps increases the computational time for generating samples according to the next target distribution $q_{t+1}(\mathbf{x})$, overall the MMC converges faster, offsetting this increased cost. Therefore, in our algorithm, tempering is only triggered when certain prescribed conditions are satisfied (e.g. $\|\Theta_t(\mathbf{x}) - \Theta_{t+1}(\mathbf{x})\|$ exceeds a threshold value).

We have presented the proposed MSMCS method in a MMC framework: namely, we want to implement MMC for a given problem, where the samples are drawn from the target distribution q_t using SMCS. Alternatively, we can also understand the method from a SMCS

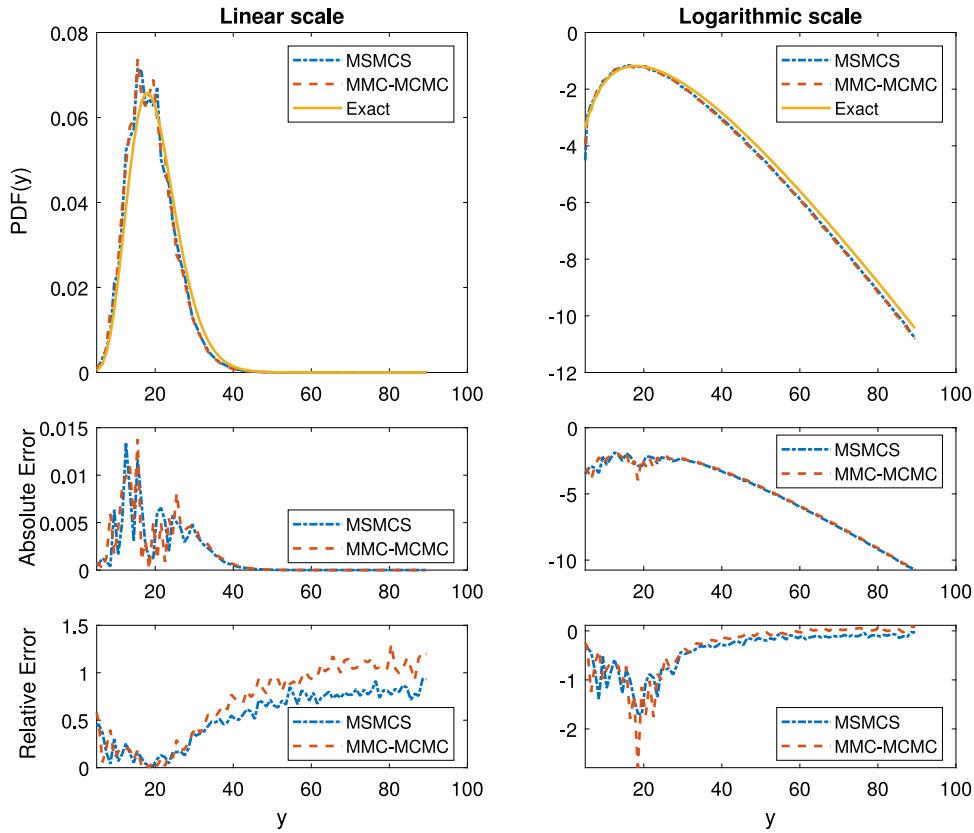

Fig. 2. Chi-square distribution with 20 degrees of freedom computed by MSMCS and MMC-MCMC, compared to the analytical solution. The results are plotted on both the linear scale (left column) and the logarithmic scale (right column). The first row contains the approximated and analytical PDFs of y. The second and third rows show the absolute and relative errors of MMC compared to the analytical solution, respectively.

perspective: that is, the SMCS method is used in a particular problem where the sequence of distributions is constructed via MMC.

Within the MMC method, the computational cost is primarily driven by the sample generation, as opposed to the calculation of the number of weighted samples within each bin—which has a very low associated cost. As such, the cost saving from our proposed algorithm largely depends on the number of cores available to a researcher within their High-powered computing, or parallel computing, facilities. For example, MMC-SMCS implemented using 10 batches on 10 cores – within each iteration – has a computation time c. 10 times lower than standard MMC-MCMC. This cost-saving is demonstrated within the third numerical example (see Section 5.3).

5. Numerical examples

In this section, we provide five numerical examples of increasing complexity to demonstrate the performance of the proposed MSMCS algorithm. By complexity, we are referring to the dimensionality of the problem and the rarity of the performance variable values. Each numerical example also demonstrates a different aspect of the advantages our proposed method has over MMC-MCMC.

5.1. Chi-square distribution

In the first example, we consider the Chi-square distribution, a continuous distribution with k-degrees of freedom, describing the distribution of a sum of squared random variables. In this example, we demonstrate that MMC can be used to reconstruct the Chi-square distribution with very low error compared to the true analytical distribution, using both MCMC and SMCS.

If $x_1, \dots, x_k$ are independent zero-mean Gaussian random variables, with unit variance, then the sum of their squares,

$$y = \sum_{i=1}^{k} x_i^2, \tag{21}$$

is distributed according to the Chi-square distribution with k degrees of freedom, where we often use the notation: $y \sim \chi^2(k)$. In this example, we construct the Chi-square distribution for $k = 20$ degrees of freedom, where the analytical form of the PDF is available.

In both MMC-MCMC and MSMCS, we use 20 iterations with 5×10^3 samples per iteration, to allow for a fair comparison. Within each MMC-MCMC iteration, a single long chain of 5×10^3 samples with no burn-in period is used.

The results are shown in Fig. 2, on both the linear and logarithmic scales. We also show the absolute and relative errors compared to the true analytical solution. The results demonstrate that the MMC method can reconstruct the Chi-square PDF with a low relative error compared to the true analytical solution and that the MMC method can effectively explore the low-probability events with a relatively small total sample size. In addition, the results show that both the MSMCS and MMC-MCMC methods obtain comparable performance with regard to the error measures.

5.2. Cantilever beam problem

We now consider a real-world engineering example: a cantilever beam model studied in [1,32]. In this example, we impose a burn-in period on MCMC, as is often required, to ensure all the samples generated by MCMC follow the MMC distribution in each iteration. As outlined previously, this is not required for SMCS, where all samples can be utilised.

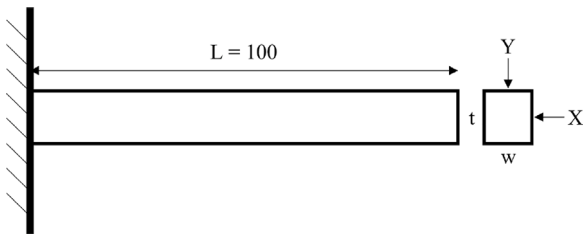

Fig. 3. Cantilever beam problem.

Table 1
The mean and variance of the random parameters.

Parameter	w	t	X	Y	E
Mean	4	4	500	1000	2.9×10^6
Variance	0.001	0.0001	100	100	1.45×10^6

As illustrated in Fig. 3, we define our beam with width w, height t, length L, and elasticity E. We are interested in the beam's reliability when subjected to transverse load Y and horizontal load X. This is a widely adopted testbed problem in reliability analysis, where the failure of the system relates to the maximum deflection of the beam (y), as determined by the following equation:

$$y = \frac{4L^3}{Ewt}\sqrt{\left(\frac{Y}{t^2}\right)^2 + \left(\frac{X}{w^2}\right)^2} \tag{22}$$

Following the problem set up of [1,32], we assume that the beam is of fixed length $L = 100$, with beam width w, height t, applied loads X and Y, and elastic modulus of the material E being random parameters, which are all independently distributed following a normal distribution. The mean and variance of each normally distributed parameter are provided in Table 1.

We compute the PDF of y with three methods: plain MC, MMC-MCMC and MSMCS. In the MC simulation, we use 10^8 full model evaluations. In both MMC-MCMC and MSMCS, we use 20 iterations with 5×10^4 samples in each iteration, to allow for a fair comparison. Within each MMC-MCMC iteration, we use a single long-chain MCMC and as such, it cannot be implemented in parallel. We also impose a burn-in period of 15% on MCMC. We set $R_y = [5.35, 6.80]$ divided into 145 bins, each of width 0.01.

To compare the results, we plot the PDF obtained by the three methods in Fig. 4. First, one can see that the results of the three methods agree very well in the high probability region, indicating that all the methods can correctly reproduce the sought PDF. The two MMC-based methods are substantially more effective in the low probability regions—the plain MC cannot reach the same level of rarity (e.g. at $y = 6.6$) while using 100 times more samples. The two MMC methods yield comparable results in this example but MSMCS has the advantage of parallel implementation.

5.3. Metaball limit-state function

We now consider an example from component reliability analysis [17], which has a changing topological structure. The limit-state function of this example is a metaball, defined as [33]:

$$g(\mathbf{x}) = \frac{30}{[4(x_1+2)^2/9 + x_2^2/25]^2 + 1} + \frac{20}{[(x_1-2.5)^2/4 + (x_2-0.5)^2/25]^2 + 1} - 5 \tag{23}$$

where x_1 and x_2 are statistically independent and identically distributed random variables with standard Gaussian distribution.

The specific geometry of this function is particularly challenging for many sampling methods, in part, because it exhibits multiple regions of high probability, as studied in the paper by Tabandeh et al. [17].

Table 2
The parameter values of the quarter car model.

m_s	m_u	k_s	k_u	c
20	40	400	2000	600

We compute the PDF of $g(\mathbf{x})$ with three methods: plain MC, MMC-MCMC and MSMCS. In both MMC-MCMC and MSMCS, we use 5 iterations with 15,000 samples per iteration and compare this to a MC simulation with 75,000 samples.

To compare the results, we plot the PDF obtained by the three methods in Fig. 5. All three methods perform similarly in high-probability regions, however, only MMC can reproduce the sought PDF in low-probability regions.

One of the main advantages of our proposed method is that the sampling within each MMC iteration can be implemented in parallel, across multiple cores of a high-performance computer. To demonstrate the computational time saved, we provide the computation time for the MSMCS algorithm across varying numbers of cores, using the previously detailed setup. The results of which are shown in Fig. 6.

5.4. Quarter car model

In our fourth example, we consider a further real-world example: a quarter car model studied by Wong et al. [34]. In this example, we implement MMC-MCMC in two alternate ways, to demonstrate the computational efficiency gained by using MSMCS - *see implementation details*.

Problem set up

The quarter-car model is used for vehicle suspension systems to investigate how they respond under a random road profile. As illustrated in Fig. 7, we set up our model following [34], such that the sprung mass m_s and the unsprung mass m_u are connected by a non-linear spring (with stiffness k_s) and a linear damper (with damping coefficient c). The unsprung mass interacts with the road surface via a non-linear spring (with stiffness k_u). The displacement of the wheel $z(t)$ represents the interaction of the quarter-car system with the road surface.

The displacements of the sprung and the unsprung masses are denoted by x_1 and x_2 respectively. Mathematically, the model is described by a two-degree-of-freedom ordinary differential equation (ODE) system:

$$m_s\frac{d^2x_1}{dt^2} = -k_s(x_1-x_2)^3 - c\left(\frac{dx_1}{dt} - \frac{dx_2}{dt}\right), \tag{24a}$$

$$m_u\frac{d^2x_2}{dt^2} = k_s(x_1-x_2)^3 + c\left(\frac{dx_1}{dt} - \frac{dx_2}{dt}\right) + k_u(z(t)-x_2). \tag{24b}$$

In our problem, the uncertainty arises through the random road profile $z(t)$ which is modelled as a zero-mean white Gaussian random force with standard deviation $\sigma = 1$. For the sake of our model, all other parameters are assumed to be fixed, taking the values as given by Table 2.

We are interested in the maximum difference between the displacements of the sprung and unsprung springs in a given interval $[0, T]$, as calculated by:

$$y = \max_{0\le t\le T}\{|x_1(t) - x_2(t)|\}. \tag{25}$$

In extreme scenarios when this displacement exceeds a certain value, say y^*, the car's suspension would break. We want to reconstruct the entire probability density function (PDF) of y. With the PDF, we can estimate the probability $\mathbb{P}(y > y^*)$ for any value of y^* in the range of interest.

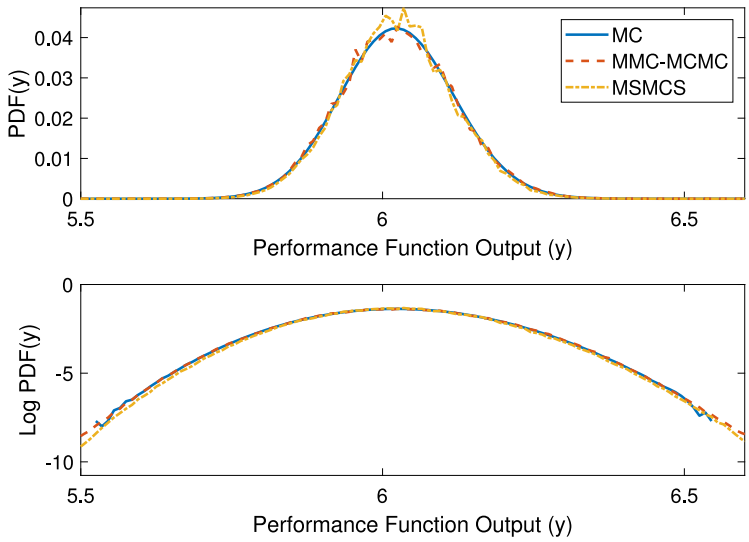

Fig. 4. Cantilever Beam PDF computed by MC, MSMCS and MMC-MCMC. The results are shown on both the linear scale (top) and the logarithmic scale (bottom).

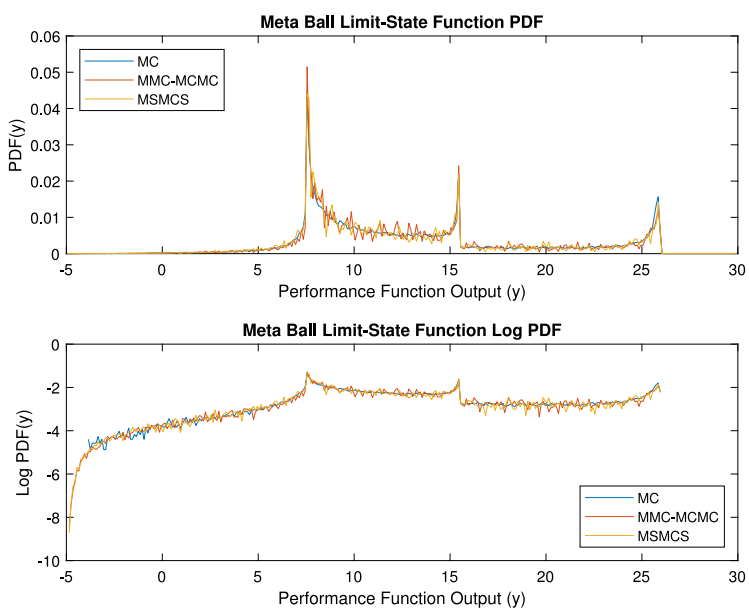

Fig. 5. PDF computed by MC, MSMCS and MMC-MCMC. The results are shown on both the linear scale (top) and the logarithmic scale (bottom).

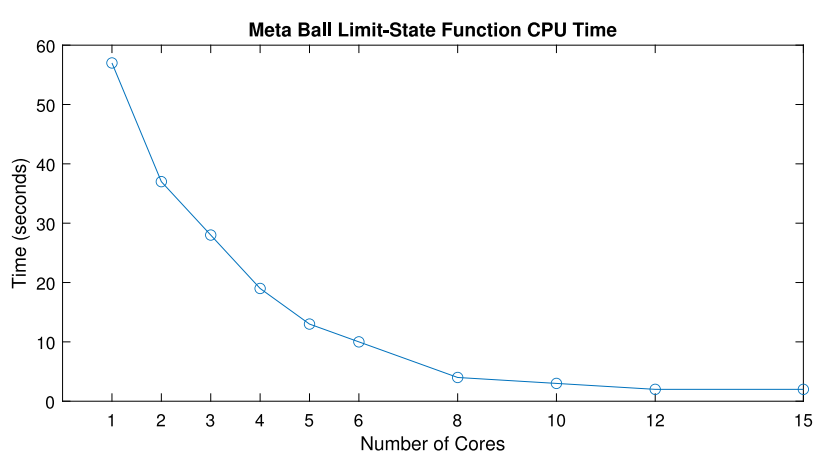

Fig. 6. Time, in seconds, to complete 5 iterations of the MSMCS algorithm with 15 000 samples per iteration, and varying numbers of cores.

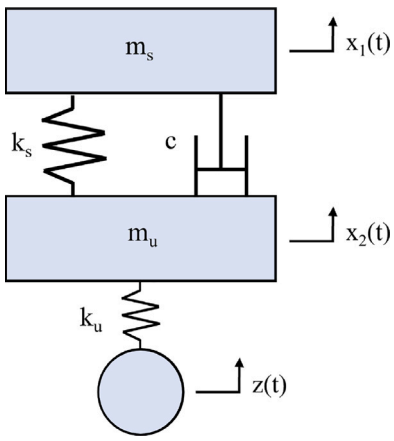

Fig. 7. Quarter car model.

Implementation details

We solve Eqs. (24) numerically using the 4th order Runge–Kutta method where the step size is taken to be $\Delta t = T/100$, so the random variable in this problem is effectively of 100 dimensions. We take $T = 1$ and set initial conditions of Eqs. (24) to be

$$x_1(0) = \frac{dx_1}{dt}(0) = 0, \quad x_2(0) = \frac{dx_2}{dt}(0) = 0 \tag{26}$$

We conduct a standard MC simulation with 10^6 samples. In both MSMCS and MMC-MCMC, we use 20 iterations with 2×10^4 samples in each iteration. The MSMCS method is easily parallelisable, meaning that within each MMC iteration, one can update the new samples completely in parallel according to the target MMC distribution, rather than forming a single long chain—significantly improving the computational efficiency. To provide a fair computational comparison, for this example, we conduct MMC-MCMC in two ways. In the first case, we use a single long chain of length 2×10^4 - the most typical implementation of MCMC, which is also how the MCMC is implemented in the first two examples. In the second case, within each iteration we use 10 chains each of length 2×10^3, to provide a fairer comparison to the parallel implementation of MSMCS.

Results

The results of all three methods are shown in Fig. 8. The MC method only estimated the PDF to the order of 10^{-6} (as expected), while the MSMCS method estimated it to order 10^{-12}. MMC-MCMC with a single chain (referred to as MMC-MCMC-SC), also accurately reconstructed the performance variable PDF, however MMC-MCMC with multiple chains (referred to as MMC-MCMC-MC) with parallel implementation, significantly underestimated the PDF values for values $y > 1.8$. The results indicate that due to the sequential nature of MCMC, running multiple short chains substantially undermines the performance of the method. Therefore, on the basis of parallel implementation, the MSMCS method clearly outperforms MMC-MCMC.

5.5. Copula model

The development of rare event simulation techniques is also critical for the risk management in financial markets. Therefore, the final application we investigate is applying the MMC method to a Copula model—one of the most widely used portfolio risk models. A copula model allows one to separate the dependence structure of the portfolio from the marginal densities of each variable – representing the individual risks of each obligor – which can have different probability distributions. We consider the Student's t-copula model, proposed by Bassamboo et al. [35].

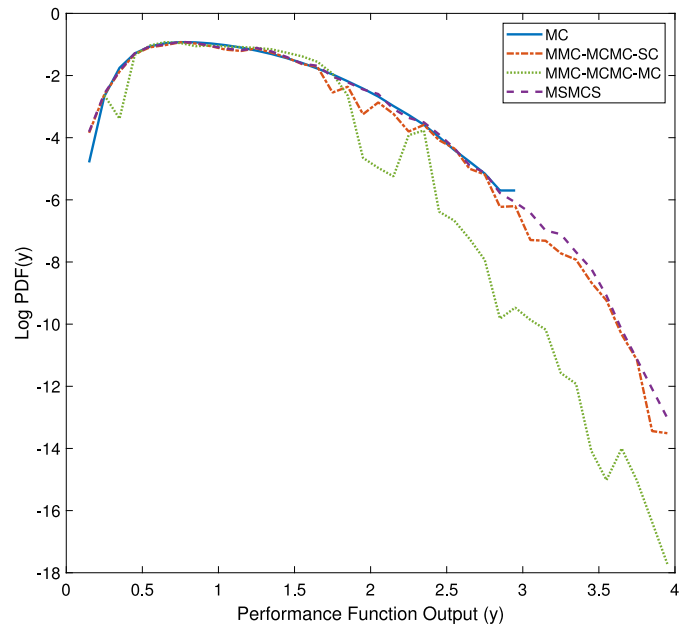

Fig. 8. Quarter car model PDF computed by MC, MSMCS and MMC-MCMC. MMC-MCMC-SC uses a single long chain. MMC-MCMC-MC uses ten shorter chains in parallel. The results are shown on the logarithmic scale.

Problem set up

We follow the problem set up of [35,36]. Consider a portfolio of loans consisting of n obligors, we aim to find the distribution of losses from defaults over a fixed time horizon, from which we can determine large loss probabilities. Suppose the probability of default for the ith obligor over the time horizon is $p_i \in (0,1)$, for $i = 1, \dots, n$, and that in the event that the ith obligor defaults, a fixed and given loss of c_i monetary units occurs. We begin by introducing a vector of underlying latent variables $\mathbf{X} = (X_1, \dots, X_n)$ such that the ith obligor defaults if X_i exceeds a given threshold level x_i. This threshold x_i is set according to the marginal default probability of the ith asset, so that $\mathbb{P}(X_i > x_i) = p_i$.

The portfolio loss from defaults is given by

$$L(\mathbf{X}) = c_1 I_{\{X_1 > x_1\}} + \cdots + c_n I_{\{X_n > x_n\}} \tag{27}$$

where $I_{\{X_i > x_i\}}$ denotes the indicator function, which is equal to 1 if $X_i > x_i$ and 0 otherwise. We let the common risk factor and the individual idiosyncratic risks be independent normally distributed random variables, that is,

$$Z \sim N(0,1) \text{ and } \eta_i \sim N(0, \sigma_\eta^2), \text{ for } i = 1, \dots, n. \tag{28}$$

We choose $0 < p < 1$ and let

$$X_i = \frac{pZ + \sqrt{1-p^2}\eta_i}{T}, i = 1, \dots, n, \tag{29}$$

where T is a non-negative random variable, independent of the other risk factors.

For a positive integer k, let $T = \sqrt{k^{-1}\Gamma(1/2, k/2)}$ where Γ represents the PDF of the Gamma distribution [35]. Therefore, our latent variables follow a multivariate t-distribution, whose dependence structure is given by a t-copula with k degrees of freedom.

Implementation details

We use the same set up as Chan et al. [36], that is, we set $\sigma_\eta^2 = 9$, $x = \sqrt{n}$ x 0.5, $p = 0.25$, and $c = 1$. We conduct a standard MC simulation, with different sample sizes—as detailed in the result tables. In both MMC-MCMC and MSMCS, we use 20 iterations with 1×10^4 samples in each iteration. We implement MMC-MCMC in two forms, one with a single long chain – as it would typically be implemented –

Table 3
Copula results using MC; MSMCS; and MMC-MCMC.

(a) $k = 4$ & $n = 250$					
Large loss threshold (b)	Sample size	Probability estimate			
	MC	MC	MMC-MCMC-SC	MMC-MCMC-MC	MSMCS
0.1	5×10^5	7.36×10^{-2}	7.27×10^{-2}	1.69×10^{-1}	7.31×10^{-2}
0.2	5×10^5	1.72×10^{-2}	1.63×10^{-2}	5.96×10^{-2}	1.71×10^{-2}
0.25	5×10^5	8.08×10^{-3}	8.13×10^{-3}	3.29×10^{-2}	8.05×10^{-3}
0.3	5×10^5	3.21×10^{-3}	3.24×10^{-3}	1.71×10^{-2}	3.28×10^{-3}
(b) $k = 8$ & $n = 250$					
Large loss threshold (b)	Sample size	Probability estimate			
	MC	MC	MMC-MCMC-SC	MMC-MCMC-MC	MSMCS
0.1	5×10^6	1.45×10^{-2}	1.39×10^{-2}	2.24×10^{-3}	1.42×10^{-2}
0.2	5×10^6	9.49×10^{-4}	9.43×10^{-4}	1.66×10^{-4}	9.49×10^{-4}
0.25	5×10^6	2.38×10^{-4}	2.49×10^{-4}	4.29×10^{-5}	2.46×10^{-4}
0.3	5×10^6	4.04×10^{-5}	3.98×10^{-5}	1.04×10^{-5}	4.01×10^{-5}
(c) $k = 12$ & $n = 250$					
Large loss threshold (b)	Sample size	Probability estimate			
	MC	MC	MMC-MCMC-SC	MMC-MCMC-MC	MSMCS
0.1	5×10^7	9.77×10^{-3}	9.82×10^{-3}	5.96×10^{-5}	9.78×10^{-3}
0.2	5×10^7	7.49×10^{-3}	7.63×10^{-3}	1.04×10^{-6}	7.53×10^{-3}
0.25	5×10^7	1.05×10^{-5}	1.02×10^{-5}	1.22×10^{-7}	1.03×10^{-5}
0.3	5×10^7	1.12×10^{-6}	1.34×10^{-6}	1.65×10^{-8}	1.21×10^{-6}
(d) $k = 16$ & $n = 250$					
Large loss threshold (b)	Sample size	Probability estimate			
	MC	MC	MMC-MCMC-SC	MMC-MCMC-MC	MSMCS
0.1	5×10^8	9.40×10^{-4}	9.36×10^{-4}	2.50×10^{-6}	9.43×10^{-4}
0.2	5×10^8	6.91×10^{-6}	6.90×10^{-6}	9.58×10^{-9}	6.86×10^{-6}
0.25	5×10^8	6.22×10^{-7}	6.18×10^{-7}	6.04×10^{-10}	6.19×10^{-7}
0.3	5×10^8	4.40×10^{-8}	4.37×10^{-8}	3.67×10^{-11}	4.51×10^{-8}
(e) $k = 20$ & $n = 250$					
Large loss threshold (b)	Sample size	Probability estimate			
	MC	MC	MMC-MCMC-SC	MMC-MCMC-MC	MSMCS
0.1	5×10^8	2.83×10^{-4}	2.88×10^{-4}	1.39×10^{-7}	2.76×10^{-4}
0.2	5×10^8	7.98×10^{-7}	7.61×10^{-7}	1.35×10^{-10}	7.73×10^{-7}
0.25	5×10^8	5.40×10^{-8}	4.92×10^{-8}	2.99×10^{-12}	5.32×10^{-8}
0.3	5×10^8	0	5.72×10^{-9}	1.02×10^{-13}	5.63×10^{-9}
(f) $k = 12$ & $n = 500$					
Large loss threshold (b)	Sample size	Probability estimate			
	MC	MC	MMC-MCMC-SC	MMC-MCMC-MC	MSMCS
0.1	5×10^8	9.61×10^{-5}	9.42×10^{-5}	5.08×10^{-12}	9.52×10^{-5}
0.2	5×10^8	1.34×10^{-6}	1.39×10^{-6}	7.15×10^{-13}	1.38×10^{-6}
0.25	5×10^8	1.36×10^{-7}	1.57×10^{-7}	4.37×10^{-13}	0.84×10^{-7}
0.3	5×10^8	1.00×10^{-8}	1.29×10^{-8}	2.54×10^{-13}	1.27×10^{-8}
(g) $k = 12$ & $n = 1000$					
Large loss threshold (b)	Sample size	Probability estimate			
	MC	MC	MMC-MCMC-SC	MMC-MCMC-MC	MSMCS
0.1	3×10^8	1.96×10^{-6}	1.88×10^{-6}	2.54×10^{-13}	1.91×10^{-6}
0.2	3×10^8	3.67×10^{-8}	3.58×10^{-8}	6.29×10^{-14}	3.72×10^{-8}
0.25	3×10^8	2.39×10^{-9}	2.24×10^{-9}	4.18×10^{-14}	2.28×10^{-9}
0.3	3×10^8	0	3.25×10^{-10}	7.24×10^{-15}	3.19×10^{-10}

and one with parallel chains (100 chains each of length 100), which provides a fairer comparison to a parallel implementation of MSMCS. Neither MCMC case uses a burn-in period.

Results

We are interested in the probability of large losses, defined as the loss function value $L(\mathbf{X}) > l$, where $l = bn$ for different sample sizes n and different threshold values b. We vary either the degrees of freedom k or the sample size n, and for each of these scenarios, we determine the probability that the loss exceeds $l = b\mathbf{x}n$, for $b = 0.1, 0.2, 0.25, 0.3$. The results are presented in Table 3.

As the MMC method reconstructs the whole loss distribution, we only require seven simulations to be performed, from which the loss probability for any b-value can be obtained. This is a significant computational saving, compared to other existing methods, like the Conditional-MC in [36], which would require a new simulation for each b-value. Our results show that the MMC method – with both MCMC and SMCS – produces significant computational savings for estimating large loss probabilities, given a Copula model. Both MMC-MCMC (with a single long chain, denoted by MMC-MCMC-SC) and MSMCS, are very effective here—although, MMC-MCMC (with multiple parallel chains, denoted by MMC-MCMC-MC) performs poorly, particularly in the high-dimensional setting, clearly illustrating the advantage of MSMCS in the parallel implementation. Finally, as shown by comparison to the standard MC, MMC is a very effective method for a Copula model and estimating large loss probabilities.

6. Conclusion

In summary, we have proposed a new method to obtain the full distribution of a performance variable by incorporating the MMC and SMCS methods. Specifically, the method uses SMCS instead of MCMC to draw samples from the warped distributions in each iteration of MMC. We have demonstrated that the proposed MSMCS method can outperform both the standard MMC-MCMC, in the sense that SMCS is easily parallelisable and so it can take full advantage of parallel high-powered computing, while MCMC, due to its sequential nature, requires a (often very long) burn-in period, which in fact is the reason that the implementation with multiple short chains does not perform well. We believe that our proposed algorithm has wide applicability, improving the computational efficiency associated with finding failure probabilities or reconstructing the whole probability distribution of interest.

One weakness of the proposed method is that MCMC is easier to implement than SMCS and involves simpler computations—so MMC-MCMC is marginally faster than MSMCS to run. However, if one can use a parallel implementation, then MSMCS significantly outperforms MMC-MCMC, as shown in the numerical examples. More importantly, both approaches to MMC can struggle in high-dimensional settings, where the generation of a new sample is likely to get rejected, which should be dealt with by developing and utilising more effective proposal distributions, for example, that based on the Hamiltonian dynamics [37].

CRediT authorship contribution statement

Robert Millar: Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Hui Li:** Writing – review & editing. **Jinglai Li:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Li J, Li J, Xiu D. An efficient surrogate-based method for computing rare failure probability. J Comput Phys 2011;230(24):8683–97.

[2] El Masri M, Morio J, Simatos F. Improvement of the cross-entropy method in high dimension for failure probability estimation through a one-dimensional projection without gradient estimation. Reliab Eng Syst Saf 2021;216:107991.

[3] Au S, Beck J. A new adaptive importance sampling scheme for reliability calculations. Struct Saf 1999;21(2):135–58.

[4] Cerou F, Del Moral P, Furon T, Guyader A. Sequential Monte Carlo for rare event estimation. Stat Comput 2012;22(3):795–808.

[5] Zhou J, Li J. An enhanced method for improving the accuracy of small failure probability of structures. Reliab Eng Syst Saf 2022;228:108784.

[6] Liu W-S, Cheung SH. Reliability based design optimization with approximate failure probability function in partitioned design space. Reliab Eng Syst Saf 2017;167:602–11.

[7] Yuan X, Qian Y, Chen J, Faes MG, Valdebenito MA, Beer M. Global failure probability function estimation based on an adaptive strategy and combination algorithm. Reliab Eng Syst Saf 2023;231:108937.

[8] Du X, Chen W. Sequential optimization and reliability assessment method for efficient probabilistic design. J Mech Des 2004;126(2):225–33.

[9] Rockafellar RT, Uryasev S. Optimization of conditional value-at-risk. J Risk 2000;2:21–42.

[10] Hazelrigg GA. A framework for decision-based engineering design. J Mech Des 1998;120(4):653–8.

[11] Wu K, Li J. A surrogate accelerated multicanonical Monte Carlo method for uncertainty quantification. J Comput Phys 2016;321:1098–109.

[12] Chen X, Li J. A subset multicanonical Monte Carlo method for simulating rare failure events. J Comput Phys 2017;344:23–35.

[13] Berg BA, Neuhaus T. Multicanonical algorithms for first order phase transitions. Phys Lett B 1991;267(2):249–53.

[14] Berg BA, Neuhaus T. Multicanonical ensemble: A new approach to simulate first-order phase transitions. Phys Rev Lett 1992;68(1):9.

[15] Hafych V, Eller P, Schulz O, Caldwel A. Parallelizing MCMC sampling via space partitioning. Stat Comput 2022;32(4):1–14.

[16] Betancourt M. A conceptual introduction to Hamiltonian Monte Carlo. 2017, arXiv preprint arXiv:1701.02434.

[17] Tabandeh A, Jia G, Gardoni P. A review and assessment of importance sampling methods for reliability analysis. Struct Saf 2022;97:102216.

[18] Del Moral P, Doucet A, Jasra A. Sequential monte carlo samplers. J R Stat Soc Ser B Stat Methodol 2006;68(3):411–36.

[19] Bononi A, Rusch L, Ghazisaeidi A, Vacondio F, Rossi N, et al. A fresh look at multicanonical Monte Carlo from a telecom perspective. In: Global telecommunications conference, 2009. GLOBECOM 2009. IEEE. IEEE; 2009, p. 1–8.

[20] Iba Y, Saito N, Kitajima A. Multicanonical MCMC for sampling rare events: an illustrative review. Ann Inst Statist Math 2014;66(3):611–45.

[21] VanDerwerken DN, Schmidler SC. Parallel markov chain monte carlo. 2013, arXiv preprint arXiv:1312.7479.

[22] Arulampalam MS, Maskell S, Gordon N, Clapp T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Trans Signal Process 2002;50(2):174–88.

[23] Beskos A, Jasra A, Law K, Tempone R, Zhou Y. Multilevel sequential monte carlo samplers. Stochastic Process Appl 2017;127(5):1417–40.

[24] Heng J, Bishop AN, Deligiannidis G, Doucet A. Controlled sequential monte carlo. Ann Statist 2020;48(5):2904–29.

[25] Green PL, Devlin L, Moore R, Jackson R, Li J, Maskell S. Increasing the efficiency of Sequential Monte Carlo samplers through the use of approximately optimal L-kernels. Mech Syst Signal Process 2022;162:108028.

[26] South L, Pettitt A, Drovandi C. Sequential monte carlo samplers with independent markov chain monte carlo proposals. Bayesian Anal 2019;14(3):753–76.

[27] Chopin N, Papaspiliopoulos O, et al. An introduction to sequential Monte Carlo, Vol. 4. Springer; 2020.

[28] Dai C, Heng J, Jacob PE, Whiteley N. An invitation to sequential Monte Carlo samplers. 2020, arXiv preprint arXiv:2007.11936.

[29] Wu J, Wen L, Green PL, Li J, Maskell S. Ensemble Kalman filter based Sequential Monte Carlo sampler for sequential Bayesian inference. 2020, arXiv preprint arXiv:2012.08848.

[30] Douc R, Cappé O. Comparison of resampling schemes for particle filtering. In: ISPA 2005. Proceedings of the 4th international symposium on image and signal processing and analysis, 2005. IEEE; 2005, p. 64–9.

[31] Doucet A, Johansen AM. A tutorial on particle filtering and smoothing: Fifteen years later. In: Handbook of nonlinear filtering, Vol. 12. 2009, p. 3, (656–704).

[32] Wu Y-T, Millwater H, Cruse T. Advanced probabilistic structural analysis method for implicit performance functions. AIAA J 1990;28(9):1663–9.

[33] Breitung K. The geometry of limit state function graphs and subset simulation: Counterexamples. Reliab Eng Syst Saf 2019;182:98–106.

[34] Wong JY. Theory of ground vehicles. John Wiley & Sons; 2008.

[35] Bassamboo A, Juneja S, Zeevi A. Portfolio credit risk with extremal dependence: Asymptotic analysis and efficient simulation. Oper Res 2008;56(3):593–606.

[36] Chan JC, Kroese DP. Efficient estimation of large portfolio loss probabilities in t-copula models. European J Oper Res 2010;205(2):361–7.

[37] Neal RM. MCMC using Hamiltonian dynamics. In: Handbook of Markov Chain Monte Carlo. Chapman and Hall/CRC; 2011, p. 139–88.