Multicanonical Sequential Monte Carlo Sampler for Uncertainty Quantification

Robert Millar∗, Jinglai Li, Hui Li

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

Abstract

In many real-world engineering systems, the performance or reliability of the system is characterised by a scalar parameter. The distribution of this performance parameter 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.

Keywords: Multicanonical Monte Carlo, Sequential Monte Carlo Sampler, Rare Event Simulation, Uncertainty Quantification

1. Introduction

Real-world engineering systems are unavoidably subject to uncertainty, rising from various sources: material properties, geometric parameters, external perturbations and so on. In practice, it is vital to characterise and quantify the impact of the uncertainties on the system performance or reliability, which

∗Corresponding author

Email addresses: r.millar.1@bham.ac.uk (Robert Millar), j.li.10@bham.ac.uk (Jinglai Li), h.li.4@bham.ac.uk (Hui Li)
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 the uncertainty. Within these, the system performance or reliability is often characterised by a scalar parameter \( y \), which we will now refer to as the performance variable. This performance variable can be expressed by a performance function \( y = g(x) \), where \( x \) 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 \( y \) is the deformation of some key components. The distribution of this performance variable is important in many UQ problems, ranging from risk management to utility optimisation. A challenge here is that these UQ problems may demand various statistical information of the performance \( y \); for example, in robust optimisation, the interests are predominantly in the mean and variance \([1]\), in risk management, one is interested in the tail probability as well as some extreme quantiles \([2]\), and in utility optimisation, the complete distribution of the performance parameter is required \([3]\). To this end, methods that can efficiently reconstruct the probability distribution of the performance variable directly are strongly desirable.

In principle the distribution of \( y \) can be estimated by standard Monte Carlo (MC) simulations, however MC can be prohibitively expensive for systems with complex mathematical models. In our previous works \([4, 5]\), 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 \([6, 7]\) to explore the energy landscape of a given physical system. In the MMC method, one splits the state space of the performance parameter 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 parameter, significantly more efficiently than using standard MC. There are other advanced MC techniques developed in reliability engineering, such as the cross-entropy method \([8]\), subset simulation \([9]\), sequential Monte Carlo \([10]\), etc. These methods are designed 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, rather than reconstructing the distribution itself.

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 \([11]\), 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 \([2, 3]\) - which reduce the overall efficiency of the MMC-MCMC method. We propose using an alternative sampling method, namely the Sequential Monte Carlo sampler (SMCS), to draw samples from the IS distri-
butions. The SMCS method, first developed in [12], 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 organized 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 $x$ be a $d$-dimensional random vector following distribution $p_x(\cdot)$, and let $y$ be a scalar variable characterised by a function $y = g(x)$. We want to determine the probability density function (PDF) of $y$, given by $\pi(y)$, where we assume that both $x$ 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 a 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 $\Delta$ centred at the discrete values $\{b_1, ..., b_M\}$, and define the $i$-th 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

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

(1)

is the domain in $X$ that maps into the $i$-th bin $B_i$ (see Fig. 1).

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 $x$-value is in the bin $D_i$ or not. Formally, the indicator function is defined as,

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

(2)

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

$$P_i = \int_{D_i} p(x)dx = \int I_{D_i}(x)p(x)dx = \mathbb{E}[I_{D_i}(x)].$$

(3)
We can estimate $P_i$ via a standard MC simulation. Namely, we draw $N$ i.i.d. samples $\{x_1, ..., x_N\}$ from the distribution $p(x)$, and calculate the MC estimator of $P_i$ as

$$\hat{P}_{i}^{\text{MC}} = \frac{1}{N} \sum_{j=1}^{N} I_{D_i}(x_j) = \frac{N_i}{N}, \quad \text{for } i = 1, ..., M,$$

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 $\Delta$ - 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 $q(x)$, Eq. 3 can be re-written as

$$P_i = \int I_{D_i}(x)[p(x)q(x)]dx = \mathbb{E}_q[I_{D_i}(x)w(x)]$$

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

$$\hat{P}_{i}^{\text{IS}} = \left[ \frac{1}{N} \sum_{j=1}^{N} I_{D_i}(x^j)w(x^j) \right]$$

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

As is well known, key to the successful implementation of IS is identifying a good IS distribution $q(x)$, which is particularly challenging for the present problem, as we are interested in multiple estimates (i.e. $P_1, ..., P_M$) 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, i.e. assuming $x \sim q(x)$,

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

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, $w(x) = \Theta_i$ for all $x \in D_i$,
where $\Theta_i$ 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:

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

where $\Theta(x) = \Theta_i$ for $x \in D_i$, $i = 1, \ldots, M$, and $c_\Theta$ is a normalizing constant. It
is easy to see that,

$$P^*_i = \int_{D_i} q(x) dx = \int_{D_i} \frac{p(x) dx}{c_\Theta \Theta_i} = \frac{P_i}{c_\Theta \Theta_i}. \quad (8)$$

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

2.3. Multicanonical Monte Carlo

The UW-FH distribution, given by Eq. (7), cannot be used directly as $\Theta_i$ de-

pends on the sought-after unknown $P_i$. The MMC method iteratively addresses
this, starting from the original input PDF $p(x)$.

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

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

where $\Theta_t(x) = \Theta_{t,i}$ for $x \in D_i$ and $c_t$ is the normalizing 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 $t$-th iteration, one draws $N$ samples $\{x^j\}_{j=1}^N$ from the current
IS distribution $q_t(x)$, then updates $\{\Theta_{t+1,i}\}_{i=1}^M$ using the following formulas,

$$\hat{H}_{t,i} = \frac{N^*_i}{N} \quad (10a)$$

$$P_{t,i} = \hat{H}_{t,i} \Theta_{t,i} \quad (10b)$$

$$\Theta_{t+1,i} = P_{t,i} \quad (10c)$$

5
where $N_{t,i}^*$ is the number of samples falling into region $D_i$ in the $t$-th iteration. Note that in Eq. (10b) we neglect the normalizing 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. [13]).

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 normalizing 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. [14], 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 each 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 [12], 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 [15], where weighted samples are generated in a sequential manner. Several extensions to this method have been proposed, e.g. [16, 17, 18, 19], with the latest advances being summarised in two recent reviews [20, 21].

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 [12] and [22]. Suppose that at time $t - 1$, we have an IS distribution $\gamma_{t-1}(x_{t-1})$, from which we have or can generate an ensemble of $N$ samples $\{x_{t-1}\}_{j=1}^N$. 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 $x_{t-1}$ and $x_t$ in the form of

$$r_t(x_{t-1}, x_t) = q_t(x_t) L_{t-1}(x_{t-1}|x_t)$$  \tag{11}$$
such that the marginal distribution of $r_t(x_{t-1}, x_t)$ over $x_{t-1}$ is $q_t(x_t)$. Now, using $\gamma_{t-1}(x_{t-1})$ and the forward kernel $K_t(x_t|x_{t-1})$, we can construct an IS distribution for $r_t(x_{t-1}, x_t)$ in the form of

$$\gamma(x_{t-1}, x_t) = \gamma_{t-1}(x_{t-1}) K_t(x_t|x_{t-1})$$  \tag{12}$$
One can draw samples from this joint IS distribution $\gamma(x_{t-1}, x_t)$ using $\{x_{t-1}\}_{j=1}^N$ and the forward kernel $K_t$, and let $\{(x_{t-1}^j, x_t^j)\}_{j=1}^N$ be an ensemble drawn from $\gamma(x_{t-1}, x_t)$. The corresponding weights are computed as

$$w_t(x_{t-1}; t) = \frac{r_t(x_{t-1}, x_t)}{\gamma(x_{t-1}, x_t)} = \frac{q_t(x_t) L_{t-1}(x_{t-1}|x_t)}{\gamma_{t-1}(x_{t-1}) K_t(x_t|x_{t-1})} \tag{13a}$$

where

$$w_{t-1}(x_{t-1}) = \frac{q_{t-1}(x_{t-1})}{\gamma_{t-1}(x_{t-1})},$$

$$\alpha_t(x_{t-1}, x_t) = \frac{q_t(x_t) L_{t-1}(x_{t-1}|x_t)}{q_{t-1}(x_{t-1}) K_t(x_t|x_{t-1})} \tag{13b}.$$}

As such the weighted ensemble $\{x_{t-1}^j, w_t^j\}_{j=1}^N$ follows the joint distribution $r_t(x_{t-1}; t)$, and as such, $\{x_t^j, w_t^j\}_{j=1}^N$ 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 [12], 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. We chose a proposal distribution $k(x_t|x_{t-1})$, and with a sample from the previous iteration $x_{t-1}^j$, we draw a sample $x_t^* \in k(x_t|x_{t-1}^j)$, and then accept (or reject) $x_t^*$ according to the following acceptance probability:

$$a_t(x_t^*|x_{t-1}^j) = \min \left\{ \frac{q_t(x_t^*)}{q_t(x_{t-1}^j)} \frac{k(x_{t-1}^j|x_t^*)}{k(x_{t-1}^j|x_t^*|x_{t-1}^j)} \right\} \tag{14}$$

That is, we set

$$x_t^j = \begin{cases} x_t^*, & \text{with probability } a_t(x_t^*|x_{t-1}^j) \\ x_{t-1}^j, & \text{otherwise.} \end{cases} \tag{15}$$
Once a forward Kernel $K_t(x_t|x_{t-1})$ is chosen, one can determine an optimal choice of $L_{t-1}$ by:

$$L_{t-1}^{opt}(x_{t-1}|x_t) = \frac{q_{t-1}(x_{t-1})K_t(x_t|x_{t-1})}{q_t(x_t)}$$

$$= \frac{q_{t-1}(x_{t-1})K_t(x_t|x_{t-1})}{\int q_{t-1}(x_{t-1})K_t(x_t|x_{t-1})dx_{t-1}}$$

(16)

where the optimality is achieved through yielding the minimal estimator variance [12]. 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):

$$L_{t-1}(x_{t-1}|x_t) = \frac{q_{t}(x_{t-1})K_t(x_t|x_{t-1})}{q_t(x_t)}$$

(17)

the detailed derivation can be found in [12]. When Eq. (17) is used, the incremental weight function $\alpha_t(x_{t-1},x_t)$ in Eq. (13b) reduces to the following:

$$\alpha_t(x_{t-1},x_t) = \frac{q_t(x_{t-1})}{q_{t-1}(x_{t-1})}$$

(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. [23], and are not discussed here. In SMCS, typically resampling is conducted when the effective samples size (ESS) [24] 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 $q_t(\cdot)$.

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
Algorithm 1 Sequential Monte Carlo Sampler

**input:** weighted ensemble \( \{ (x_{j}^{t-1}, w_{j}^{t-1}) \}_{j=1}^{N} \)

**for** \( j = 1 \) to \( N \)

(a) draw \( x_{j}^{t} \) from \( k(\cdot| x_{j}^{t-1} ) \)

(b) calculate the acceptance probability \( a(x_{j}^{*}, x_{j}^{t-1}) \) using Eq. (14)

(c) determine \( x_{j}^{t} \) using Eq. (15) and \( a(x_{j}^{*}, x_{j}^{t-1}) \)

(d) calculate \( \alpha_{j}^{t} \) using Eq. (18)

(e) compute \( w_{j}^{t} = w_{j}^{t-1} \alpha_{j}^{t} \)

**end for**

normalize the weights calculated

calculate ESS

**if** \( ESS < ESS_{\text{min}} \)

resample the ensemble and set \( w_{j}^{t} = 1/N \) for \( j = 1, ..., N \)

**end if**

can be considered as a sequence of distributions. Though the implementation seems straightforward, there are still some issues that need to be addressed with in 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 \( \Theta \)'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,

\[
\hat{H}_{t,i} = \sum_{j=1}^{N} I_{D_{i}(x^{j})} w(x^{j}). \tag{19}
\]

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 \( q_{0}(x) \) and subsequent target distribution \( q_{1}(x) \).

To address this issue, we propose including a simulated tempering process in the method. Namely, we introduce a set of intermediate distributions between \( q_{0} \) and \( q_{t+1} \), which we can apply SMCS too. Note that the difference in the IS distributions, can be attributed to differences in the \( \Theta \)-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 \text{ with } \alpha_0 = 0 \text{ and } \alpha_K = 1, \text{ such that the intermediate } \Theta \text{-functions are }

\Theta_k(x) = \alpha_k \Theta_{t+1}(x) + (1 - \alpha_k) \Theta_t(x). \quad (20)

It follows that the sequence of intermediate distributions \{q_k\}_{k=0}^K \text{ 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). \text{ One can see that when } q_t \text{ and } q_{t+1} \text{ are close to each other, SMCS can efficiently generate samples from } q_{t+1} \text{ via the forward kernel and the samples from } q_t, \text{ 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}(x), \text{ 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(x) - \Theta_{t+1}(x)|| \text{ 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 perspective: that is, the SMCS method is used in a particular problem where the sequence of distributions are constructed via MMC.

5. Numerical Examples

In this section, we provide four 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 parameter 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 \(x_1, \ldots, x_k\) are independent zero-mean Gaussian random variables, with unit variance, then the sum of their squares,

\[ y = \sum_{i=1}^{k} x_i^2, \quad (21) \]

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 $5 \times 10^3$ samples per iteration, to allow for a fair comparison. Within each MMC-MCMC iteration, a single long chain of $5 \times 10^3$ samples with no burn-in period is used, so all samples are utilised.

The results are shown in Figure 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, in this relatively simple example, 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 [8, 25]. 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.

As illustrated in Figure 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:

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

(22)

Following the problem set up of [8, 25], 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.

| Parameter | \( w \) | \( t \) | \( X \) | \( Y \) | \( E \) |
|-----------|--------|--------|--------|--------|--------|
| Mean      | 4      | 4      | 500    | 1000   | \( 2.9 \times 10^9 \) |
| Variance  | 0.001  | 0.0001 | 100    | 100    | \( 1.45 \times 10^6 \) |

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 \( 5 \times 10^4 \) 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. Also in this example, we impose a burn-in period of 15%. 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 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 as has been mentioned, MSMCS has the advantage of parallel implementation.

5.3. Quarter Car Model

In our third example, we consider a further real-world example: a quarter car model studied by Wong et al [26]. 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 Figure 5, we set-up our model following [26], 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:

$$m_s \frac{d^2 x_1}{dt^2} = -k_s (x_1 - x_2)^3 - c \left( \frac{dx_1}{dt} - \frac{dx_2}{dt} \right),$$  \hspace{1cm} (23a)$$

$$m_u \frac{d^2 x_2}{dt^2} = k_s (x_1 - x_2)^3 + c \left( \frac{dx_1}{dt} - \frac{dx_2}{dt} \right) + k_u (z(t) - x_2).$$  \hspace{1cm} (23b)$$

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.

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

$$y = \max_{0 \leq t \leq T} \{|x_1(t) - x_2(t)|\}. \hspace{1cm} (24)$$

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 $P(y > y^*)$ for any value of $y^*$ in the range of interest.

**Implementation Details**

We solve Eqs. (23) numerically using the 4-th 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. (23) to be

\[
x_1(0) = \frac{dx_1}{dt}(0) = 0, \quad x_2(0) = \frac{dx_2}{dt}(0) = 0
\]

(25)

We conduct a standard MC simulation with \( 10^6 \) samples. In both MSMCS and MMC-MCMC, we use 20 iterations with \( 2 \times 10^4 \) 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 \( 2 \times 10^4 \) - 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 \( 2 \times 10^3 \), to provide a fairer comparison to the parallel implementation of MSMCS.

**Results**

The results of all three methods are shown in Figure 6. 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 parameter PDF, however MMC-MCMC with multiple chains (referred to as MMC-MCMC-MC) and therefore enabling 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.

**5.4. 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 variables - 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 [27].

**Problem Set Up**

We follow the problem set up of [27] and [28]. 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 \( i \)-th obligor over the time horizon is \( p_i \in \)
Figure 6: 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 both the linear scale (left column) and the logarithmic scale (right column).
(0, 1), for \( i = 1, \ldots, n \), and that in the event that the \( i \)-th obligor defaults, a fixed and given loss of \( c_i \) monetary units occurs. We begin by introducing a vector of underlying latent variables \( \mathbf{X} = (X_1, \ldots, X_n) \) such that the \( i \)-th 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 \( i \)-th asset, so that \( P(X_i > x_i) = p_i \).

The portfolio loss from defaults is given by

\[
L(\mathbf{X}) = c_1 I_{\{X_1 > x_1\}} + \ldots + c_n I_{\{X_n > x_n\}}
\]  

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,

\[
Z \sim N(0, 1) \quad \text{and} \quad \eta_i \sim N(0, \sigma^2_{\eta_i}), \text{ for } i = 1, \ldots, n.
\]  

We choose \( 0 < p < 1 \) and let

\[
X_i = pZ + \sqrt{1 - p^2}\eta_i, \quad i = 1, \ldots, n,
\]  

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 \( \Gamma \) represents the PDF of the Gamma distribution \([27]\). 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* \([28]\), that is, we set \( \sigma^2_{\eta_i} = 9 \), \( x = \sqrt{n} \times 0.5 \), \( p = 0.25 \), and \( c = 1 \). We conduct a standard MC simulation, with different sample sizes - as detailed in the results tables. In both MMC-MCMC and MSMCS, we use 20 iterations with \( 1 \times 10^4 \) samples in each iteration. We implement MMC-MCMC in two forms, one with a single long chain - as it would typically be implemented - and one with parallel chains (100 chains each of length 100), which provides a fairer comparison to 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 \( l = bn \), 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 with other existing methods, like the Conditional-MC in \([28]\), which would require a new simulation for each \( b \)-value. Our results show that the
Table 3: Copula Results using MC; MSMC; and MMC-MCMC.

(a) \( k = 4 \& n = 250 \)

| Large Loss Threshold | Sample Size | Probability Estimate |
|----------------------|-------------|----------------------|
|                      | MC          | MMC-MCMC-SC          | MMC-MCMC-MC          | MSMC          |
| 0.1                  | 3 \times 10^8 | 7.36 \times 10^{-4}  | 7.21 \times 10^{-4}  | 1.69 \times 10^{-4}  | 7.31 \times 10^{-4}  |
| 0.25                 | 5 \times 10^8 | 1.72 \times 10^{-4}  | 1.63 \times 10^{-4}  | 1.96 \times 10^{-4}  | 1.71 \times 10^{-4}  |
| 0.4                  | 5 \times 10^8 | 8.08 \times 10^{-5}  | 8.13 \times 10^{-5}  | 3.29 \times 10^{-5}  | 8.05 \times 10^{-5}  |
| 0.3                  | 5 \times 10^8 | 3.21 \times 10^{-5}  | 3.24 \times 10^{-5}  | 1.71 \times 10^{-5}  | 3.28 \times 10^{-5}  |

(b) \( k = 8 \& n = 250 \)

| Large Loss Threshold | Sample Size | Probability Estimate |
|----------------------|-------------|----------------------|
|                      | MC          | MMC-MCMC-SC          | MMC-MCMC-MC          | MSMC          |
| 0.1                  | 3 \times 10^8 | 1.45 \times 10^{-5}  | 1.39 \times 10^{-5}  | 2.24 \times 10^{-5}  | 1.42 \times 10^{-5}  |
| 0.25                 | 5 \times 10^8 | 9.49 \times 10^{-5}  | 9.43 \times 10^{-5}  | 1.06 \times 10^{-5}  | 9.49 \times 10^{-5}  |
| 0.4                  | 5 \times 10^8 | 2.38 \times 10^{-5}  | 2.49 \times 10^{-5}  | 4.29 \times 10^{-5}  | 2.46 \times 10^{-5}  |
| 0.3                  | 5 \times 10^8 | 4.04 \times 10^{-6}  | 3.98 \times 10^{-6}  | 1.04 \times 10^{-6}  | 4.01 \times 10^{-6}  |

(c) \( k = 12 \& n = 250 \)

| Large Loss Threshold | Sample Size | Probability Estimate |
|----------------------|-------------|----------------------|
|                      | MC          | MMC-MCMC-SC          | MMC-MCMC-MC          | MSMC          |
| 0.1                  | 3 \times 10^8 | 9.17 \times 10^{-6}  | 9.52 \times 10^{-6}  | 5.96 \times 10^{-6}  | 9.78 \times 10^{-6}  |
| 0.25                 | 5 \times 10^8 | 7.49 \times 10^{-6}  | 7.63 \times 10^{-6}  | 1.04 \times 10^{-6}  | 7.53 \times 10^{-6}  |
| 0.4                  | 5 \times 10^8 | 1.05 \times 10^{-6}  | 1.02 \times 10^{-6}  | 1.22 \times 10^{-6}  | 1.03 \times 10^{-6}  |
| 0.3                  | 5 \times 10^8 | 1.12 \times 10^{-6}  | 1.14 \times 10^{-6}  | 1.80 \times 10^{-6}  | 1.21 \times 10^{-6}  |

(d) \( k = 16 \& n = 250 \)

| Large Loss Threshold | Sample Size | Probability Estimate |
|----------------------|-------------|----------------------|
|                      | MC          | MMC-MCMC-SC          | MMC-MCMC-MC          | MSMC          |
| 0.1                  | 3 \times 10^8 | 9.40 \times 10^{-6}  | 9.36 \times 10^{-6}  | 2.50 \times 10^{-6}  | 9.43 \times 10^{-6}  |
| 0.25                 | 5 \times 10^8 | 6.91 \times 10^{-6}  | 6.90 \times 10^{-6}  | 9.58 \times 10^{-6}  | 6.86 \times 10^{-6}  |
| 0.4                  | 5 \times 10^8 | 6.22 \times 10^{-6}  | 6.18 \times 10^{-6}  | 6.04 \times 10^{-6}  | 6.19 \times 10^{-6}  |
| 0.3                  | 5 \times 10^8 | 4.40 \times 10^{-6}  | 4.31 \times 10^{-6}  | 3.67 \times 10^{-6}  | 4.51 \times 10^{-6}  |

(e) \( k = 20 \& n = 250 \)

| Large Loss Threshold | Sample Size | Probability Estimate |
|----------------------|-------------|----------------------|
|                      | MC          | MMC-MCMC-SC          | MMC-MCMC-MC          | MSMC          |
| 0.1                  | 3 \times 10^8 | 5.23 \times 10^{-6}  | 5.28 \times 10^{-6}  | 1.30 \times 10^{-6}  | 2.76 \times 10^{-6}  |
| 0.25                 | 5 \times 10^8 | 7.98 \times 10^{-7}  | 7.61 \times 10^{-7}  | 1.35 \times 10^{-7}  | 7.73 \times 10^{-7}  |
| 0.4                  | 5 \times 10^8 | 5.40 \times 10^{-7}  | 4.92 \times 10^{-7}  | 2.90 \times 10^{-7}  | 5.32 \times 10^{-7}  |
| 0.3                  | 5 \times 10^8 | 3.12 \times 10^{-7}  | 3.11 \times 10^{-7}  | 1.02 \times 10^{-7}  | 5.63 \times 10^{-7}  |

(f) \( k = 12 \& n = 500 \)

| Large Loss Threshold | Sample Size | Probability Estimate |
|----------------------|-------------|----------------------|
|                      | MC          | MMC-MCMC-SC          | MMC-MCMC-MC          | MSMC          |
| 0.1                  | 3 \times 10^8 | 9.61 \times 10^{-6}  | 9.42 \times 10^{-6}  | 5.08 \times 10^{-6}  | 9.52 \times 10^{-6}  |
| 0.25                 | 5 \times 10^8 | 1.34 \times 10^{-6}  | 1.39 \times 10^{-6}  | 7.15 \times 10^{-6}  | 1.38 \times 10^{-6}  |
| 0.4                  | 5 \times 10^8 | 1.36 \times 10^{-7}  | 1.57 \times 10^{-7}  | 4.37 \times 10^{-7}  | 0.84 \times 10^{-7}  |
| 0.3                  | 5 \times 10^8 | 1.00 \times 10^{-8}  | 1.29 \times 10^{-8}  | 2.54 \times 10^{-8}  | 1.27 \times 10^{-8}  |

(g) \( k = 12 \& n = 1000 \)

| Large Loss Threshold | Sample Size | Probability Estimate |
|----------------------|-------------|----------------------|
|                      | MC          | MMC-MCMC-SC          | MMC-MCMC-MC          | MSMC          |
| 0.1                  | 3 \times 10^8 | 1.96 \times 10^{-6}  | 1.88 \times 10^{-6}  | 2.54 \times 10^{-6}  | 1.91 \times 10^{-6}  |
| 0.25                 | 5 \times 10^8 | 3.67 \times 10^{-7}  | 3.58 \times 10^{-7}  | 6.29 \times 10^{-7}  | 3.72 \times 10^{-7}  |
| 0.4                  | 5 \times 10^8 | 2.39 \times 10^{-7}  | 2.24 \times 10^{-7}  | 4.18 \times 10^{-7}  | 2.28 \times 10^{-7}  |
| 0.3                  | 5 \times 10^8 | 3.25 \times 10^{-8}  | 2.54 \times 10^{-8}  | 7.24 \times 10^{-8}  | 3.19 \times 10^{-8}  |
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, clear illustrating the advantage of MSMCS in the parallel implementation. Finally, as shown by comparison to the standard MC, MMC is very effective method for the purposes of a Copula model and estimating large loss probabilities.

6. Conclusion

In summary, we consider UQ problems where the full distribution of a performance parameter is sought, and we propose a method to do so 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 [29].

References

[1] X. Du, W. Chen, Sequential optimization and reliability assessment method for efficient probabilistic design, Journal of Mechanical Design 126 (2) (2004) 225–233.

[2] R. T. Rockafellar, S. Uryasev, Optimization of conditional value-at-risk, Journal of risk 2 (2000) 21–42.

[3] G. A. Hazelrigg, A framework for decision-based engineering design, Journal of mechanical design 120 (4) (1998) 653–658.
K. Wu, J. Li, A surrogate accelerated multicanonical monte carlo method for uncertainty quantification, Journal of Computational Physics 321 (2016) 1098–1109.

X. Chen, J. Li, A subset multicanonical monte carlo method for simulating rare failure events, Journal of Computational Physics 344 (2017) 23–35.

B. A. Berg, T. Neuhaus, Multicanonical algorithms for first order phase transitions, Physics Letters B 267 (2) (1991) 249–253.

B. A. Berg, T. Neuhaus, Multicanonical ensemble: A new approach to simulate first-order phase transitions, Physical Review Letters 68 (1) (1992) 9.

J. Li, J. Li, D. Xiu, An efficient surrogate-based method for computing rare failure probability, Journal of Computational Physics 230 (24) (2011) 8683–8697.

S. Au, J. Beck, A new adaptive importance sampling scheme for reliability calculations, Struct. Safety 21 (2) (1999) 135–158.

F. Cerou, P. Del Moral, T. Furon, A. Guyader, Sequential monte carlo for rare event estimation, Statistics and Computing 22 (3) (2012) 795–808.

V. Hafych, P. Eller, O. Schulz, A. Caldwell, Parallelizing mcmc sampling via space partitioning, Statistics and Computing 32 (4) (2022) 1–14.

P. Del Moral, A. Doucet, A. Jasra, Sequential monte carlo samplers, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 68 (3) (2006) 411–436.

Y. Iba, N. Saito, A. Kitajima, Multicanonical mcmc for sampling rare events: an illustrative review, Annals of the Institute of Statistical Mathematics 66 (3) (2014) 611–645.

D. N. VanDerwerken, S. C. Schmidler, Parallel markov chain monte carlo, arXiv preprint arXiv:1312.7479 (2013).

M. S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking, IEEE Transactions on signal processing 50 (2) (2002) 174–188.

A. Beskos, A. Jasra, K. Law, R. Tempone, Y. Zhou, Multilevel sequential monte carlo samplers, Stochastic Processes and their Applications 127 (5) (2017) 1417–1440.

J. Heng, A. N. Bishop, G. Deligiannidis, A. Doucet, Controlled sequential monte carlo, The Annals of Statistics 48 (5) (2020) 2904–2929.
[18] P. L. Green, L. Devlin, R. Moore, R. Jackson, J. Li, S. Maskell, Increasing the efficiency of sequential monte carlo samplers through the use of approximately optimal l-kernels, Mechanical Systems and Signal Processing 162 (2022) 108028.

[19] L. South, A. Pettitt, C. Drovandi, Sequential monte carlo samplers with independent markov chain monte carlo proposals, Bayesian Analysis 14 (3) (2019) 753–776.

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

[21] C. Dai, J. Heng, P. E. Jacob, N. Whiteley, An invitation to sequential monte carlo samplers, arXiv preprint arXiv:2007.11936 (2020).

[22] J. Wu, L. Wen, P. L. Green, J. Li, S. Maskell, Ensemble kalman filter based sequential monte carlo sampler for sequential bayesian inference, arXiv preprint arXiv:2012.08848 (2020).

[23] R. Douc, O. Cappé, 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, pp. 64–69.

[24] A. Doucet, A. M. Johansen, A tutorial on particle filtering and smoothing: Fifteen years later, Handbook of nonlinear filtering 12 (656-704) (2009) 3.

[25] Y.-T. Wu, H. Millwater, T. Cruse, Advanced probabilistic structural analysis method for implicit performance functions, AIAA journal 28 (9) (1990) 1663–1669.

[26] J. Y. Wong, Theory of ground vehicles, John Wiley & Sons, 2008.

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

[28] J. C. Chan, D. P. Kroese, Efficient estimation of large portfolio loss probabilities in t-copula models, European Journal of Operational Research 205 (2) (2010) 361–367.

[29] R. M. Neal, Mcmc using hamiltonian dynamics, in: Handbook of Markov Chain Monte Carlo, Chapman and Hall/CRC, 2011, pp. 139–188.