12th International Conference on Applications of Statistics and Probability in Civil Engineering, ICASP12
Vancouver, Canada, July 12-15, 2015

Rare Event Simulation: A Point Process Interpretation With Application In Probability And Quantile Estimation

Clément Walter
Laboratoire de Probabilités et Modèles Aléatoires, Univ. Paris Diderot, Paris, France
CEA, DAM, DIF, F-91297 Arpajon, France

Gilles Defaux
CEA, DAM, DIF, F-91297 Arpajon, France

ABSTRACT: This paper addresses the issue of estimating extreme probability and quantile on the output of complex computer codes. We introduce a new approach to solve this problem in term of a random walk in the output space, leading to two main results: (1) the number of samples required to get a realisation of a random variable in a given domain of probability measure p is drastically reduced, following a Poisson law with parameter $\log 1/p$; and (2) we get parallel algorithms for estimating probabilities and quantiles and especially the optimal parallel Multilevel Splitting algorithm where there is indeed no subset to define anymore.

1. INTRODUCTION

Extreme events simulation and quantification come from the need to insure that undesirable events will not appear. Typically such events are failure of industrial critical systems, *ie.* systems for which failure is regarded as a massive catastrophic situation. Usually the system is a "black box" whose output determines safety/failure domains.

Formally, let $\mathbf{X} \in \mathbb{R}^d$ be a random vector and g be a measurable function from $\mathbb{R}^d$ to $\mathbb{R}$ defining the failure domain $F = \{\mathbf{x} \in \mathbb{R}^d \,|\, \mathrm{g}(\mathbf{x}) > q\}$, we seek to measure F with respect to the distribution of $\mathbf{X}$: $\mu^X(F) = \mathrm{P}[\mathbf{X} \in F] = p$ or to find back q from a given p. The two main difficulties of this problem arise from the order of magnitude of p (say $p < 10^{-5}$) and the computational time of g (from couples of hours to several months).

While the naive Monte Carlo (MC) approach fails to solve this problem because of the rarity of the sought event (its squared coefficient of variation is $\delta^2 \approx 1/(Np)$ with N the sample size), two developments have brought improvement in estimating extreme probability. On the one hand Importance Sampling (Robert and Casella, 2004; Asmussen and Glynn, 2007) modifies the distribution of $\mathbf{X}$ to lower the variance of the naive Monte Carlo estimator; unfortunately this method is intrusive and the search for an appropriate change of measure is not obvious. In particular it is known that the optimal change depends on the quantity of interest and is thus not directly available. On the other hand Multilevel Splitting (MS) methods uses a finite increasing sequence $(q_k)_{k=0..m}$ with $q_0 = -\infty$ and $q_m = q$ such that $F = \{g(\mathbf{X}) > q\}$ can be written as an intersection of nested subsets $F = \bigcap_m \{g(\mathbf{X}) > q_m\}$, each of those subsets being measured by Markov chain Monte Carlo (McMC) (see (Au and Beck, 2001) or (Glasserman et al., 1999) for an in-depth review of splitting techniques). This algorithm was further improved by Cérou et al. (2012) and Guyader et al. (2011) proposed a limit case optimal in terms of computational efficiency but disabling parallel computing.

Recasting results from Guyader et al. (2011) and Huber et al. (2011) we introduce here a general approach to this issue in term of a random walk in the output space, *i.e.* that we focus on the real-valued random variable $Y = g(\mathbf{X}) \in \mathbb{R}$. This approach brings three main results: (1) the number of samples needed to get a realisation of $\mathbf{X}$ in the failure domain follows a Poisson law with param-

eter $\log 1/p$, this has to be compared to a classical Geometric law with parameter p for naive Monte Carlo; (2) we present two new parallel estimators for estimating probability and quantile; (3) we link this framework with the MS algorithm and show it brings its optimal implementation in terms of computational time against variance of the estimator. Especially it is directly related to the optimal MS algorithm by Guyader et al. (2011), which appears to be a specific implementation of the new estimator.

The outline of the paper is as follows: firstly we introduce the new framework and derive the two estimators for probability and quantile; then we compare the probability estimator with the usual Multilevel Splitting algorithm and finally we discuss its implementation and provide examples of probability and quantile estimation on academic test cases.

2. RARE EVENT SIMULATION

We introduce here the key brick of our new estimators. Unlike with usual Monte Carlo methods, we are not going to work with independent and identically distributed (iid) samples of the input random variable of interest $\mathbf{X}$ but with Markov chains defined on the output real-valued random variable $Y = g(\mathbf{X})$. Denote μ^Y its distribution and assume its *cdf* F_Y is continuous.

Definition 1 (Increasing random walk) *Let $Y_0 = -\infty$ and define recursively the Markov sequence $(Y_n)_n$ such that for all $n \in \mathbb{N}$:*

$$\mathrm{P}\left[Y_{n+1} \in A \mid Y_0, \cdots, Y_n\right] = \frac{\mu^Y(A \cap (Y_n, +\infty))}{\mu^Y((Y_n, +\infty))}$$

In other words $(Y_n)_n$ is a random increasing sequence where each element is generated conditionally greater than the previous one. Considering the sequences $(Y_n)_{n\geq 1}$ and $(T_n)_{n\geq 1} \mid T_n = -\log(1 - F_Y(Y_n))$, it can be shown that $(T_n)_{n\geq 1}$ is distributed as the arrival times of a Poisson Process with parameter 1 (Huber et al., 2011; Guyader et al., 2011; Simonnet, 2014). Thus, the counting random variable of the number of events before q: $M_q = \mathrm{card}\{n \geq 1 \mid Y_n \leq q\}$ follows a Poisson law with parameter $-\log(1 - F_Y(q)) = -\log p$.

Firstly, this means that implementing this Markov chain allows us to obtain extreme realisations of a real-valued random variable (*eg* the output of a computer code) much faster than with an (iid) sampling because of this "log attribute": on average $\log 1/p$ samples against $1/p$ with an (iid) sampling. Secondly we have a natural estimator for the quantity $\log 1/p$, namely the maximum likelihood estimator (MLE) of the parameter of a Poisson law: given $(M_q^i)_{i=1..N}$ N random counting variables, one has:

$$\widehat{\log 1/p}_{\mathrm{MLE}} = \frac{1}{N}\sum_{i=1}^{N} M_q^i \stackrel{\mathrm{def}}{=} \frac{1}{N} M_q$$

M_q is then the random counting variable of a Marked Poisson Process with parameter N. Hence for estimating p given q, one can build the following estimator, optimal by theorem of Lehmann-Scheffé:

$$\widehat{p} = \left(1 - \frac{1}{N}\right)^{M_q} \tag{1}$$

and alternatively for estimating q given p, one can consider the $(m = \lfloor N \log 1/p \rfloor)^{th}$ event of the Marked Poisson Process, *i.e.* the m^{th} smallest element of the N Markov chains. Denote q_m this element and q_{m+1} the following one, we therefore consider the estimator for a quantile:

$$\widehat{q} = \frac{1}{2}(q_m + q_{m+1}) \tag{2}$$

Proposition 1 (Statistical properties of $\widehat{p}$)

$$\mathrm{E}[\widehat{p}] = p$$
$$\mathrm{var}[\widehat{p}] = p^2\left(p^{-1/N} - 1\right) \underset{N\to\infty}{\sim} p^2 \frac{\log 1/p}{N}$$

This estimator has a smaller variance than a Monte Carlo one, exhibits a logarithmic efficiency and almost achieves the Cramer-Rao bound $-p^2 \log p/N$ (Walter, 2015).

Proposition 2 (Limit distribution of $\widehat{q}$) *Given Y has a density f_Y, one has:*

$$\sqrt{N}(\widehat{q} - q) \xrightarrow[N\to\infty]{\mathscr{L}} \mathscr{N}\left(0, \frac{p^2 \log 1/p}{f_Y(q)^2}\right)$$

As for a Monte Carlo estimator, it has a bias of order $1/N$, which is in this case centred for the exponential distribution. Furthermore, this estimator has the nice property of providing a confidence interval without any estimation of the density f_Y. Recalling we have considered m instead of the random variable M_q, we can also consider m_- and m_+ such that $\mathrm{P}\left[M_q \in [m_-, m_+]\right] = 0.95$, build $\widehat{q}_-$ and $\widehat{q}_+$ and then compute a confidence interval for $\widehat{q}$ at 95%.

To end up this section, we stress out the fact that these estimators can really be seen as the twin of Monte Carlo ones with a "log attribute", *i.e.* that the statistical properties of $\widehat{p}$ and $\widehat{q}$ are similar but adding a log to the $1/p$ factor: denote $\widehat{p}_{\mathrm{MC}}$ and $\widehat{q}_{\mathrm{MC}}$ the usual Monte Carlo estimators for a probability or a quantile, one has:

$$\mathrm{var}\left[\widehat{p}_{\mathrm{MC}}\right] \approx \frac{p^2}{Np} \rightarrow \mathrm{var}\left[\widehat{p}\right] \approx \frac{p^2 \log 1/p}{N}$$

$$\mathrm{var}\left[\widehat{q}_{\mathrm{MC}}\right] \approx \frac{p^2}{Nf(q)^2 p} \rightarrow \mathrm{var}\left[\widehat{q}\right] \approx \frac{p^2 \log 1/p}{Nf(q)^2}$$

3. Link with Mutlilevel Splitting methods

The purpose of this section is to link this theoretical probability estimator with well known MS algorithms. We first recall the main steps of this kind of algorithms and then point out some of their principal issues. Finally we show that our approach brings the best one could expect from MS methods in terms of variance of the estimator against computational time.

3.1. Definition of Multilevel Splitting

Originally, Multilevel Splitting (MS) methods work with a finite increasing sequence $(q_i)_{i=0..m}$ such that $q_0 = -\infty$, $q_m = q$, write the target probability as a product of conditional ones:

$$\mathrm{P}\left[g(\mathbf{X}) > q\right] = \prod_{i=1}^{m} \mathrm{P}\left[g(\mathbf{X}) > q_i \mid g(\mathbf{X}) > q_{i-1}\right]$$

and then seek to estimate each conditional probability with Markov chain Monte Carlo (McMC). It has been shown (Cérou et al., 2012) that to lower the total variance of the estimator, the sequence $(q_i)_i$ should be chosen so that all conditional probabilities are equal. When it is not possible to infer such a sequence, the idea is to adaptively define it so that each conditional probability is equal to a given constant p_0 ($p_0 = 0.1$ in (Au and Beck, 2001)) and this is known as Adaptive Multilevel Splitting method. While providing an efficient way to implement MS algorithm, it introduces a bias in the estimator. Furthermore the question of a proper choice for p_0 is still open.

Basically the workflow of adaptive MS is as follows:

1. Sample N (iid) random variables $(\mathbf{X}_n)_{n=1..N}$
2. Get q_1 the p_0 quantile; $i = 1$
3. While $q_i < q$
   - Generate $(1 - p_0)$ samples from the p_0 samples such that $g(\mathbf{X}_n) > q_i$ with Markov chain
   - Get q_{i+1} the p_0 quantile; $i = i + 1$
4. Set $q_i = q$
5. Get the final conditional probability p_f
6. $\widehat{p} = p_0^{i-1} \cdot p_f$

The regeneration step is crucial as one requires (iid) samples to estimate properly the conditional probability. To do so, the Metropolis-Hastings algorithm is often used, with a *burn-in* parameter b used to increase the convergence of the Markov chain to its stationary distribution, *i.e.* that b samples are generated for each finally kept $\mathbf{X}_n$.

The bias in this adaptive version comes from the estimation of quantile at each step. Alternatively, it has been proposed to set $p_0 = 1 - k/N$; Bréhier et al. (2014) extensively studied this parametrisation and show unbiasedness whatever k. However, while optimal variance is found for $k = 1$, it disables parallel computing since at each step, only 1 sample is regenerated.

3.2. Effective computing time

We now intend to challenge our new approach against MS methods in terms of effective computing time (ECT). By ECT we mean the unparalleled sequence of simulated samples one requires to reach a given precision on the variance of the final estimator. More precisely, if one considers that n_c cores are available for parallel computation, ECT is the number of samples calculated by each core; for example a naive Monte Carlo estimator with sample size N has an ECT $t_{\mathrm{MC}} = \lceil N/n_c \rceil = \lceil (n_c p \delta^2)^{-1} \rceil$

with δ its coefficient of variation and p the sought probability.

In an adaptive MS algorithm, Cérou et al. (2012) showed that the number of levels m converges almost surely to $\lceil (\log p)/(\log p_0) \rceil$ when $N \to \infty$. If one considers that $p_f \approx p_0$ and that the samples are (iid) for each subset, then one gets for the coefficient of variation δ_{MS}:

$$\delta_{\text{MS}}^2 \approx \frac{\log p}{\log p_0} \frac{1-p_0}{N p_0}$$

The first step of MS requires N simulations, and then there are $N(1-p_0)\cdot b$ samples to be regenerated at each step. This gives:

$$t_{\text{MS}} = \frac{N}{n_c} + \lceil \frac{\log p}{\log p_0} \rceil b \left[\frac{N(1-p_0)}{n_c} \wedge 1 \right]$$

Considering the first term is negligible, we will have either $N(1-p_0) \geq n_c$ and so:

$$t_{\text{MS}} \approx \frac{b(\log p)^2}{n_c \delta_{\text{MS}}^2} \frac{(1-p_0)^2}{p_0 (\log p_0)^2} \tag{3}$$

or $t_{\text{MS}} \approx b \log p / \log p_0$. The first expression decreases in p_0 while the second one increases. On the one hand this brings an optimal value for p_0 depending on n_c, p and targeted precision, on the other hand one has $t_{\text{MS}} \geq b(\log p)^2/(n_c \delta^2)$.

For our algorithm presented in Section 2, one core will generate N/n_c Markov chains, and each Markov chain requires $\approx -\log p$ simulations, which brings $t \approx (-\log p) N/n_c \approx (\log p)^2/(n_c \delta^2)$. As we shall see further in the technical implementation part, we have to include here a *burn-in* parameter too, which gives finally $t \approx b(\log p)^2/(n_c \delta^2)$.

Therefore it is clear that our algorithm brings the best MS method in terms of ECT; basically it allows for taking $p_0 \to 1$ without disabling the parallel computation possibility.

To conclude this section, we point out the fact that this result of optimality is expected to be found each time a MS method is applied (maximum of a stochastic process, counting...) because the only assumptions is that the output of the system is real-valued with continuous *cdf* . Indeed, we argue that this random walk is the underlying mechanism of each MS method.

4. PRACTICAL IMPLEMENTATION

In this section, we precise how to implement the simulation of the random walk and then present results on test cases.

4.1. Simulating conditional distributions

The simulation of the random walks requires to have access to conditional simulators. Indeed the estimator requires to generate the output Y according to the conditional law $\mu^Y(\cdot \mid Y > y)$, *i.e.* to generate the input $\mathbf{X}$ according to $\mu^X(\cdot \mid g(\mathbf{X}) > y)$ for a given y. A general idea is to use convergence properties of an ergodic Markov chain to its unique invariant probability to sample from a given distribution. Assuming μ^X has a *pdf* f_X, it means we intend to generate a Markov chain with stationary *pdf* $\mathbb{1}_{g(\mathbf{x})>y} f_X(\mathbf{x}) / \mathrm{P}[g(\mathbf{X}) > y]$. This implementation is rather simple when a reversible transition kernel K is available and the random walk the Metropolis-Hastings method (Hastings, 1970) provides such a kernel.

Algorithm 1 Metropolis-Hastings method

Generate $\mathbf{W}$ a standard multivariate Gaussian sample or following a symmetric Uniform distribution over a compact set of $\mathbb{R}^d$
$\mathbf{X}^* \leftarrow \mathbf{x} + \sigma \mathbf{W}$
$\rho \leftarrow \min(1, f_X(\mathbf{X}^*)/f_X(\mathbf{x})) \mathbb{1}_{g(\mathbf{X}^*)>y}$
Accept the transition $\mathbf{X}^*$ with probability ρ, return $\mathbf{x}$ otherwise

In the special case where $\mathbf{X} \overset{\mathscr{L}}{\sim} \mathscr{N}(0, I)$ a direct construction in also possible (Cérou et al., 2012).

Algorithm 2 Direct construction for $\mathbf{X} \overset{\mathscr{L}}{\sim} \mathscr{N}(0, I)$

Generate $\mathbf{W}$ a standard multivariate Gaussian sample
$\mathbf{X}^* \leftarrow \dfrac{\mathbf{x} + \sigma \mathbf{W}}{\sqrt{1+\sigma^2}}$
return $\mathbf{X}^*$ if $g(\mathbf{X}^*) > y$, $\mathbf{x}$ otherwise

Remark 1 *Depending on the distribution of Y, it could be possible to use directly (iid) samples. For example if Y is exponentially distributed, then $Y_{n+1} \overset{\mathscr{L}}{\sim} Y_n + Z_n$ with Z_n an (iid) copy of Y; and if Y is a Pareto random variable, then $Y_{n+1} \overset{\mathscr{L}}{\sim} Y_n Z_n$ with Z_n an (iid) copy of Y.*

4.2. Algorithms

We give here pseudo-codes for estimating a probability or a quantile. We want to underline the fact that there is nothing more to do than simulating several random walks. However to make use of Markov chain sampling as presented in Section 4.1, one needs to manipulate several chains simultaneously: the starting point of the Markov chain has to be selected in a population already following the targeted distribution.

Algorithm 3 Generating N random walks

Require: N, q
  Generate N copies $(\mathbf{X}_i)_{i=1..N}$ according to μ^X
  $Y \leftarrow (g(\mathbf{X}_1), \cdots, g(\mathbf{X}_N))$
  $\mathtt{N}_{\text{event}} = (0, \cdots, 0)$
  while $\min Y < q$ **do**
    $\mathtt{ind} \leftarrow \text{which} Y_i < q$
    for i in ind **do**
      $\mathtt{N}_{\text{event}}[i] = \mathtt{N}_{\text{event}}[i] + 1$
      Generate $\mathbf{X}^* \sim \mu^X(\cdot \mid X > g(\mathbf{X}_{\mathtt{i}}))$
      $\mathbf{X}_{\mathtt{i}} \leftarrow \mathbf{X}^*$; $Y_i = g(\mathbf{X}^*)$
    end for
  end while
  return $\mathtt{N}_{\text{event}}$, $(\mathbf{X}_i)_{i=1..N}$, $(Y_i)_{i=1..N}$

For the conditional generation step, it may be necessary to store the history of the positions of the $(\mathbf{X}_i)_{i=1..N}$ to select a starting point among samples generated according to thresholds lower or equal to the current one. If one does not want to do so, then it is possible to adapt the algorithm: "$\mathtt{ind} \leftarrow \text{argmin} Y$" instead of "$\mathtt{ind} \leftarrow \text{which} Y_i < q$". With this slite change, all the current states of the other Markov chains will be good candidates. However, it disables the sequential parallelisation possibility of this algorithm.

The probability estimator only requires the number of events of each random walk and is described in Algorithm 4. Basically it is a wrap-up of Algorithm 3 ; in this scope it is to be noticed that Algorithm 3 returns a vector $\mathtt{N}_{\text{event}}$ containing for each random walk the number of events and not a single integer. The parallel implementation of the quantile estimator is a bit more complicated as the stopping criterion is expressed as a number of events over the total random walk. Indeed to sum several partially simulated point processes, one has to make sure they have all overpassed a given threshold q; then the sum process will be complete until this threshold q. We then suggest a two-fold algorithm: during the first step the stopping criterion is a number of events on each single random walk; then one considers the greatest reached time and relaunch all the random walks until they also reach this time. The issue of choosing a right number of events m_0 for the first step is discussed in Walter (2015). The main idea is that the final number of simulated events at the end of the algorithm should be as close as possible to the targeted one. We give here directly the guideline:

$$m_0 = \lceil N \log 1/p + \beta^2/2 - \beta\sqrt{\Delta}/2 \rceil$$

with:

$$\beta = b_{n_c} - \frac{\log\log 1/\alpha}{\sqrt{2\log n_c}}$$

b_{n_c} being the localisation parameter of the Gaussian law in the framework of Extreme Value Theory: $b_{n_c} = \sqrt{2\log n_c} - (\log\log n_c + \log 4\pi)/2\sqrt{2\log n_c}$, $\Delta = \beta^2 + 4N\log 1/p$ and α a probability of not having enough events in the end (typically $\alpha = 0.05$). Another requirement is to store a copy of vector Y of Algorithm 3 instead of replacing values in it to keep a record of the times of the events and not only their numbers. This is all summed up in Algorithm 5.

Algorithm 4 Probability estimator

Require: N, n_c, q;
  Run n_c Algorithm 3 with parameters N and q
  for i in $1 : n_c$ **do**
    $\mathtt{N}_{\text{event}}[i] = \sum_{j=1}^{N} \mathtt{N}_{\text{event}}(\text{Algorithm}[i])[\mathtt{j}]$
  end for
  Get the total number of events $\mathtt{N}_{\text{event}} = \sum_{i=1}^{n_c} \mathtt{N}_{\text{event}}[\mathtt{i}]$
  $\widehat{p} = (1 - 1/(n_c \cdot N))^{\mathtt{N}_{\text{event}}}$

4.3. Test case

We now present the performance of both Algorithms 4 and 5 on academic examples.

4.3.1. Probability estimator

The two-degrees-of-freedom damped oscillator sketched in Figure 1 was first proposed by Der Ki-

Algorithm 5 Quantile estimator

Require: N, n_c, p

Run n_c Algorithm 3 with statement "ind $\leftarrow \operatorname{argmin} Y$" and stopping criteria "$\sum \mathbb{N}_{\text{event}} > m_0$"

$q \leftarrow \max_{i=1..n_c} \min_{j=1..N} Y_j^{(i)}$

Relaunch the n_c algorithms with original stopping criteria "$\min Y > q$"

Mix all the recorded times of the n_c algorithms and sort them: $(q_i)_i$

$m = \lfloor -(N \cdot n_c) \log p \rfloor$

$\widehat{q} = 0.5\,(q_m + q_{m+1})$

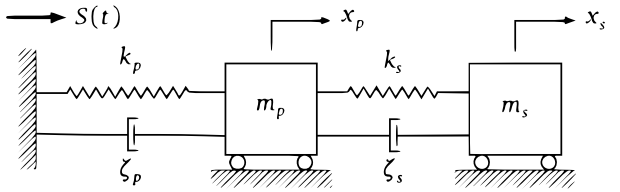

Figure 1: A 2 degrees of freedom damped oscillator (from Dubourg et al. (2011))

ureghian and De Stefano (1991) and then used by Bourinet et al. (2011) and Dubourg et al. (2011). Table 1 presents the physical parameters and the probabilistic model used. Igusa and Der Kiureghian (1985) showed that the mean-squared relative displacement of the secondary spring under a white noise base acceleration with intensity S_0 writes: $\mathrm{E}\left[x_s^2\right] =$

$$\pi \frac{S_0}{4\zeta_s\omega_s^2} \frac{\zeta_a\zeta_s}{\zeta_p\zeta_s(4\zeta_a^2+\theta^2)+\gamma\zeta_a^2} \frac{(\zeta_p\omega_p^3+\zeta_s\omega_s^3)\omega_p}{4\zeta_a\omega_a^4}$$

with $\gamma = m_s/m_p$, $\omega_p^2 = k_p/m_p$, $\omega_s^2 = k_s/m_s$, $\omega_a = (\omega_p+\omega_s)/2$, $\zeta_a = (\zeta_p+\zeta_s)/2$ and $\theta = (\omega_p - \omega_s)/\omega_a$.

Finally, Der Kiureghian and De Stefano (1991) proposed that the limit-state function could write under reasonable approximation as follows:

$$g(\mathbf{x}) = C - \eta\, k_s \sqrt{\mathrm{E}\left[x_s^2\right]}$$

with C the force capacity of the secondary spring and η a peak factor here set to 3 as in Dubourg et al. (2011). Table 1 presents the probabilistic model used. As it uses lognormal distributions and reversible kernel of Algorithm 2 is defined in the standard space, an isoprobalistic transformation is done before each call to the limit-state function g.

Variable	Mean	CV (%)	Description
m_p	1.5	10	masse
m_s	0.01	10	
k_p	1	20	stiffness
k_s	0.01	20	
ζ_p	0.05	40	damping ratio
ζ_s	0.02	50	
C	27.5	10	Force capacity
S_0	100	10	Acceleration

Table 1: Stochastic model of the oscillator; all parameters have log-normal distribution. CV: coefficient of variation

Figure 2 compares the estimator of $\mathrm{P}[g(\mathbf{X}) < 0]$ defined in Eq. (1) with different implementations for a total of $n_c \cdot N = 1000$ random walks to illustrate the effect of the use of Markov chain drawing of Section 4.1. Results are presented as boxplots over 100 simulations, whisker extending to the extreme values. The reference value (red dashed line) is obtained from an usual Subset Simulation (Au and Beck, 2001) with $p_0 = 0.1$ and $N = 4 \times 10^6$ samples and the horizontal grey dotted lines stand for a 95% confidence interval around it. Hence one can say that the estimator converges well and that parameter N of Algorithm 3 should not be too small; as a matter of fact it appears that $N \gtrsim 5$ to 10 is a conservative choice. Then we benchmark Algorithm 4 against usual MS methods: the original *Subset Simulation* (SS) algorithm presented by Au and Beck (2001) with $p_0 = 0.1$ and the optimal non-parallel algorithm *Least Particle Algorithm* (LPA) of Guyader et al. (2011) with $k = 1$. All algorithms require to simulate according to conditional laws and the *burn-in* parameter is set to 20 for all of them. We thus run 100 simulations with $n_C = 100$ cores and Table 2 summarises the results. The total number of calls to the limit-state function N_{call} includes the *burn-in*.

The three algorithms have similar performances in terms of coefficient of variation and a comparison in terms of number of calls to the limit-state function and effective computing time is then rel-

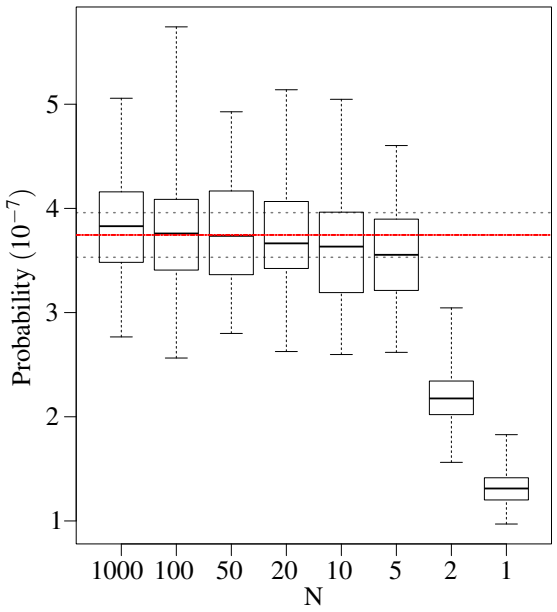

Figure 2: Failure probability estimated with a total of 1000 *random walks simulated by batches of size N using Algorithm 4*

Method	$\widehat{p}$ $(.10^{-7})$	CV	N_{call}	ECT
SS	3.81	0.12	568 800	5 688
LPA	3.85	0.13	292 446	292 446
Algo. 4	3.75	0.12	296 932	3 400

Table 2: Comparison of Algorithm 4 against usual MS methods with $n_c = 100$ cores available. Methods: SS: Au and Beck (2001) with $p_0 = 0.1$ and $N = 4500$; LPA: Guyader et al. (2011) with $N = 1000$; Algo. 4 with batches of size $N = 10$. Results from 100 simulations; $\widehat{p}$ is the empirically averaged probability and CV is the empirical coefficient of variation; N_{call} and ECT are averaged over the 100 simulations

evant. Firstly the number of calls of SS is much greater than for LPA and Algorithm 4: LPA is the optimal MS algorithm in terms of computational efficiency and is also a particular implementation of Algorithm 4 (Walter, 2015). Secondly the ECT of LPA is much greater because it is not parallel, making this implementation inefficient practically speaking. Furthermore, the gain in ECT between Algorithm 4 and SS is in good agreement with the theoretical formula (3): with $p_0 = 0.1$, $(1-p_0)^2/p_0/(\log p_0)^2 \approx 1.53$ and the empirical increase is $5688/3400 \approx 1.67$ Theses results lead to the conclusion that Algorithm 4 is the parallel optimal MS algorithm, being 35% to 40% faster than the original *Subset Simulation* algorithm.

4.3.2. Quantile estimator

The watermarking detection example is used by Cérou et al. (2012) and Guyader et al. (2011). Let $d \in \mathbb{N}^*$ be the dimension of the input space and $\mathbf{u}$ be a unit vector in $\mathbb{R}^d$; the failure domain is regarded as the interior of a double cone of axis $\mathbf{u}$ (see Merhav and Sabbag (2008)):

$$F = \{\mathbf{x} \in \mathbb{R}^d \mid \Phi(\mathbf{x}) = \frac{\mid \mathbf{x}^T\mathbf{u} \mid}{\parallel \mathbf{x} \parallel} > q\} \qquad (4)$$

The analytic relation between p and q writes as follows:

$$p = \mathrm{P}\left[\Phi(\mathbf{X}) > q\right] = 1 - G\left(\frac{(d-1)q^2}{1-q^2}\right)$$

with G the *cdf* of a Fisher variable with $(1, d-1)$ degrees of freedom. We set here $d = 20$ and $p = 4.704\,10^{-11}$ and estimate $q = 0.95$ as in Guyader et al. (2011) and Cérou et al. (2012).

Results are presented as boxplots over 100 simulations, whiskers extending to the extreme values. As for the probability estimator, Figure 3 compares different implementations for a total of $n_c \cdot N = 1000$ random walks.

The same conclusions as for the probability estimator apply while the dimension is more than twice as big. As far as we know, it is the first parallel alternative to naive Monte Carlo and thus allows for a massive gain in quantile estimation.

5. Conclusion

This paper presents a new approach to rare event simulation for real-valued random variables in terms of a random walk and showed its relation with a Poisson process with parameter 1. As soon as the performance function of a complex computer code is real-valued, this framework applies. It has been shown that it is related to Multilevel Splitting algorithms and that it gives the optimal implementation in terms of variance against computational time. This optimality has been proved theoretically and verified in the examples. Especially, it enables

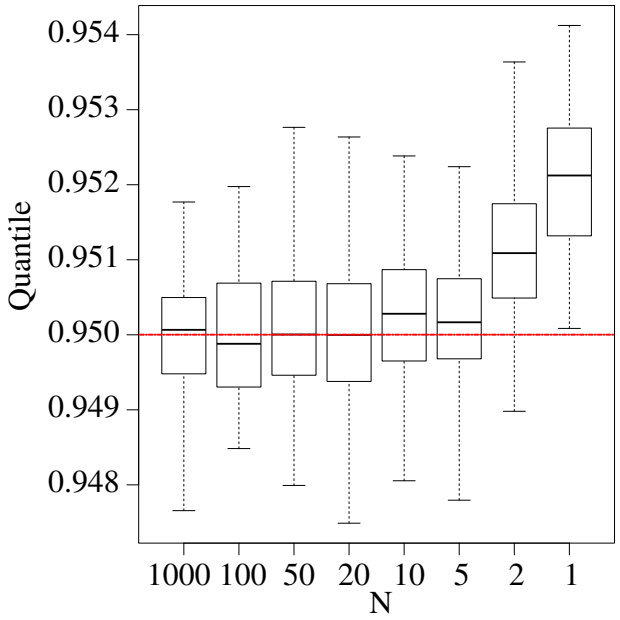

Figure 3: $4.704\,10^{-11}$ *quantile estimated with a total of* 1000 *random walks simulated by batches of size N*

full parallelisation of the estimator. However, it also requires to generate according to conditional distributions and this can bring some limitation in the parallel implementation.

ACKNOWLEDGEMENTS

The author would like to thank his advisor Josselin Garnier (University Paris Diderot) for numerous advices and suggestions.

6. REFERENCES

Asmussen, S. and Glynn, P. W. (2007). *Stochastic Simulation: Algorithms and Analysis: Algorithms and Analysis*, Vol. 57. Springer.

Au, S.-K. and Beck, J. L. (2001). "Estimation of small failure probabilities in high dimensions by subset simulation." *Probabilistic Engineering Mechanics*, 16(4), 263–277.

Bourinet, J.-M., Deheeger, F., and Lemaire, M. (2011). "Assessing small failure probabilities by combined subset simulation and support vector machines." *Structural Safety*, 33(6), 343–353.

Bréhier, C.-E., Lelievre, T., and Rousset, M. (2014). "Analysis of adaptive multilevel splitting algorithms in an idealized case." *arXiv preprint arXiv:1405.1352*.

Cérou, F., Del Moral, P., Furon, T., and Guyader, A. (2012). "Sequential Monte Carlo for rare event estimation." *Statistics and Computing*, 22(3), 795–808.

Der Kiureghian, A. and De Stefano, M. (1991). "Efficient algorithm for second-order reliability analysis." *Journal of engineering mechanics*, 117(12), 2904–2923.

Dubourg, V., Deheeger, F., and Sudret, B. (2011). "Metamodel-based importance sampling for the simulation of rare events." *arXiv preprint arXiv:1104.3476*.

Glasserman, P., Heidelberger, P., Shahabuddin, P., and Zajic, T. (1999). "Multilevel splitting for estimating rare event probabilities." *Operations Research*, 47(4), 585–600.

Guyader, A., Hengartner, N., and Matzner-Løber, E. (2011). "Simulation and estimation of extreme quantiles and extreme probabilities." *Applied Mathematics & Optimization*, 64(2), 171–196.

Hastings, W. K. (1970). "Monte carlo sampling methods using markov chains and their applications." *Biometrika*, 57(1), 97–109.

Huber, M., Schott, S., et al. (2011). "Using tpa for bayesian inference." *Bayesian Statistics 9*, 9, 257.

Igusa, T. and Der Kiureghian, A. (1985). "Dynamic characterization of two-degree-of-freedom equipment-structure systems." *Journal of engineering mechanics*, 111(1), 1–19.

Merhav, N. and Sabbag, E. (2008). "Optimal watermark embedding and detection strategies under limited detection resources." *Information Theory, IEEE Transactions on*, 54(1), 255–274.

Robert, C. P. and Casella, G. (2004). *Monte Carlo statistical methods*. Springer.

Simonnet, E. (2014). "Combinatorial analysis of the adaptive last particle method." *Statistics and Computing*, 1–20.

Walter, C. (2015). "Moving particles: A parallel optimal multilevel splitting method with application in quantiles estimation and meta-model based algorithms." *Structural Safety*, 55(0), 10 – 25.