université
PARIS DIDEROT
PARIS 7

Thèse

présentée pour obtenir le grade de

Docteur de l'Université Paris Diderot

❧

Laboratoire de Probabilités et Modèles Aléatoires
École doctorale Mathématiques Paris Centre

Discipline : Mathématiques Appliquées

Using Poisson processes for rare event simulation

Par : **Clément Walter**

Sous la direction de Josselin Garnier

Soutenue publiquement le 21 octobre 2016 devant un jury composé de :

Examinateur	Stéphane Boucheron, Professeur, Université Paris Diderot
Examinateur	Gilles Defaux, Ingénieur, CEA
Directeur	Josselin Garnier, Professeur, École Polytechnique
Examinateur	Arnaud Guyader, Professeur, Université Pierre et Marie Curie
Rapporteur	François Le Gland, Directeur de Recherche, INRIA
Examinateur	Tony Lelièvre, Professeur, École des Ponts ParisTech
Examinateur	Éric Moulines, Professeur, École Polytechnique
Rapporteur	Daniel Straub, Professeur, TU München

C'est chose étroite qu'un couteau et le fruit qu'il tranche, on n'en rejoindra pas les parts

Paul Claudel – *Le Partage de Midi*

Remerciements

C'est donc ici que tout commence et tout finit. Le moment de chercher et rendre compte des causes et éléments qui ont contribué, directement ou non, à la création du manuscrit qui suit. Avril 2013 - octobre 2016 : trois ans et demi autour des événements rares, rétrospective d'une trajectoire aléatoire.

Lundi 8 avril 2013. Après une très pleine saison de ski et une semaine de retard sur la rentrée officielle, le CEA et Gilles Defaux m'accueillent pour 4 mois avec option "thèse" sur les conseils de Nicolas Petit, professeur aux Mines. Toujours présent pour répondre à mes doutes sur mon orientation et mes activités extra-scientifiques, c'est donc un peu grâce à lui que je suis arrivé ici, même s'il reconnaissait volontiers : "Les probabilités des évènements rares sont un domaine scientifique pointu, d'intérêt majeur depuis longtemps (même si je ne sais pas personnellement pourquoi). C'est un excellent sujet." S'il lit cette thèse, j'espère qu'il trouvera quelques réponses à cette question.

Le sujet du stage, mélanger les *Subset Simulation* avec les méthodes de krigeage, est une idée de Gilles. Son sens pratique des algorithmes m'a donné de partir immédiatement sur une piste intéressante. Les résultats obtenus à la fin des ces quatre mois ont été le réel point de départ des premiers travaux de recherche en novembre 2013.

A ce moment là, après de grandes grandes vacances en Amérique, je commence véritablement à travailler avec Josselin Garnier, mon directeur de thèse (merci Gilles et le CEA pour la prime de stage, Josselin pour avoir accepté cette *insensément longue* pause). C'est lui qui me pousse à étudier de manière théorique et précise les pistes proposées à la fin du stage ; et pousse entre mes mains la pierre essentielle de tous mes travaux futurs, l'article d'Arnaud Guyader *et al.* : "Simulation and estimation of extreme quantiles and extreme probabilities". J'ai eu l'occasion de le rencontrer par la suite et il m'a fait l'honneur de participer à mon jury de thèse, Arnaud est sans aucune doute, mais peut-être sans en être si conscient, un élément déterminant de ce doctorat. C'est comme s'il avait apporté, *ready-made*, tous les outils théoriques dont j'avais besoin pour justifier mes bricolages algorithmiques.

Ce premier cycle s'achève le 13 mai 2014 par la soumission d'un article et des vacances au Canada. Au retour rien ne sera plus pareil, et pour la première fois j'éprouve le sentiment de la vacuité de ces recherches ; tout lasse. Première période de vide, je remercie l'équipe du CEA de me laisser papillonner sur différents thèmes et Guillaume Perrin de m'accompagner en Italie étudier les sensibilités.

Je retiens la date du 24 juillet 2014 parmi un de ces rares jours où *tout s'éclaire*, où l'on veut crier *eurêka*, où l'on court à son bureau à 19h pour essayer et écrire. Quatre jours, quatre moments, quatre chapitres à cette thèse. Ceux qui étaient présent sur les

pelouses des Invalides s'en souviennent peut-être. Le travail sur ce qui s'appelait en fait le *nested sampling* prend une nouvelle tournure. Grâce au soutient technique précis et puissant de Josselin et aux excellentes remarques d'Arnaud Guyader, il terminera plus d'un an plus tard dans un article de *Statistics and Computing*.

A la rentrée, toujours radical et pressé, je décide de ne plus aller au CEA ou d'arrêter. Grâce à la complaisance de Gilles, j'obtiens l'autorisation de m'installer trois jours par semaine à l'université Paris Diderot. Nouvelle année nouveau départ. Parallèlement le Komet Football Club est créé avec Josselin Decroix ; toute une équipe tous les lundi pour taper la balle sur les différents parcs interdépartementaux de la petite couronne. Des victoires, des montées, des résumés et des likes Facebook, on respire. Le 24 octobre je termine également ma première chemise, le début d'une autre histoire.

La deuxième année devait être celle de mon départ pour l'étranger, elle sera celle de la rupture tout court. Une année *standard* où les activités extra-thèses se développent à mesure que les mois passent. Je pense toucher le fond en mai ; les travaux que je mène n'intéressent pas directement le CEA, et indirectement personne. Julien Bect m'invite néanmoins à venir présenter des résultats à Supélec et ceci marque le début d'une collaboration qui aboutira aux résultats du chapitre 5. Fin juin, l'étude sur les discontinuités prend sa forme finale à la suite d'une de ces journées *éclair* ; un papier est envoyé avant de partir pour un mois et demi de vacances. Août est l'occasion de retravailler sur le *nested sampling* d'après les retours des rapporteurs. Une fois de plus, la confiance de Gilles me permet d'effectuer ce travail dans les meilleures conditions, depuis la Bretagne. C'est à ce moment-là qu'Arnaud Drizard m'initie au cluster Amazon EC2 et au calcul parallèle, un ultime virage pour la dernière année.

Les épreuves renvoyées, la rentrée arrive avec l'envie de tout sauf avancer cette thèse. Challenger Deep m'accueille pour deux mois, avant que Calchemise ne devienne une activité à temps plein. Noël passé, la science reprend sa place, retour des travaux avec Julien Bect. Rapidement néanmoins, les considérations pratiques débordent sur le reste : il faut faire tourner ces méthodes sur de vrais cas, sur de vrais supercalculateurs, ou ça n'a pas de sens. Près de trois moi pour développer du calcul parallèle sur Airain et il est déjà temps de commencer à rédiger le manuscrit. Ultime épreuve, ultime passion ; les travaux sur le krigeage se finiront simultanément à la rédaction du chapitre. Le 10 juillet à 3h20 du matin, Professeur Garnier met fin à l'exercice. Trois jours plus tard je pars pour deux mois ; merci à celles qui m'ont recueilli en complète déflation !

Lorsque ma peine s'arrête commence celle des rapporteurs. Je remercie chaleureusement messieurs François Le Gland et Daniel Straub d'avoir accepté d'emporter cet ouvrage en vacances, de le lire et de le commenter. Le 21 octobre, ils sont tous les deux présents accompagnés de Messieurs Stéphane Boucheron, Arnaud Guyader, Tony Lelièvre, Eric Moulines ainsi que mes encadrants Gilles Defaux et Josselin Garnier pour constituer le jury de ma soutenance. Merci à eux d'avoir conclu avec pertinence et bienveillance cette entreprise. Tout spécialement merci à Josselin pour avoir toujours répondu à mes questions

sans jamais les devancer et à Gilles pour m'avoir toujours fait confiance et apporté un soutien inconditionnel.

J'ai pu mentionner au fil des dates certaines personnes ; évidemment d'autres aussi sont là tout le temps qui ne sont pas entrées dans cette énumération. Il y a bien sûr les membres de l'équipe incertitude du CEA, Jean "le boss" Giorla qui a du accepter mes originalités de procédures, Guillaume Perrin, arrivé quasiment en même temps que moi et qui m'a initié à la FSGT, Claire Cannamela et le petit nouveau, Philippe Mellinger. J'éviterai la longue liste de tous les stagiaires et doctorants du bâtiment B, mais je les remercie de m'avoir accueilli quand je voulais bien moi faire une pause avec eux. Cette remarque vaut également pour tous les camarades de P7. Je pense également à Bertrand Iooss qui est à l'origine du projet mistral ; Nathalie Bergame et Valérie Juvé qui m'ont plus d'une fois débloqué à Sophie Germain et Claire Boulier conduit à Brétigny. Et l'extra scientifique : Philippe Masse, mon professeur en MP, pour m'avoir initié aux math ; Matthieu Rougé pour ses encouragements ; tous mes camarades des Mines ou d'ailleurs, en particulier Martin de Gourcuff ; Calchemise en général ; tous ceux qui ont un jour franchi le palier de la rue des Mathurins, pour un soir ou plusieurs mois, dont Marc Lafont présent seul de bout en bout ; vin chaud for ever ; Paris.

La conclusion appartient à tous ceux qui ont fait l'effort de venir le jour même à 9h30 pour passer deux heures à me voir parler sans rien comprendre (comme d'habitude), en particulier ma famille au complet.

Contents

Context

This thesis deals with the simulation and the analysis of *rare* events. By *rare* events, we mean events which do not appear often but whose realisation is important enough to justify their study. Such events represent different kinds of risk: situations engaging human beings (aviation or rail safety, civil engineering), environmental issues (failure of a nuclear plant, dam robustness), financial risks (probability of ruin of an insurer); and some other applications like telecommunication networks (loss of data at a saturated node) or molecular dynamics (transition from one metastable state to another).

When these situations are modelled with numerical codes, the rare event is often expressed as a level that a given real-valued output of the code should not overpass. For instance the design of a reliable mechanical system may require that its maximum strain should not exceed a given value, or that the internal pressure of an body not become greater than a security threshold. In a financial setting, this would be that the total insurance claims stay lower than the available funds. For network reliability, failure means that the total required memory for a given node becomes greater than its physical capacity because of the arriving of new parcels.

Throughout manuscript, the numerical code will be denoted by g a function taking some input parameters $\mathbf{x}$ in a given set $\mathbb{X}$ and returning a real value $y \in \mathbb{R}$. In the more general setting, $\mathbb{X}$ is finite- or infinite dimensional: if $\mathbf{x}$ is a set of some parameters of the code (for instance a stiffness coefficient, the Young's modulus, ...), $\mathbb{X} \subset \mathbb{R}^d$ for some $d \geq 1$. On the other hand $\mathbf{x}$ can be a path and thus $\mathbf{x} = (\mathbf{x}_t)_t$ and $\mathbb{X} \subset (\mathbb{R}^d)^{\mathbb{R}}$ is a space of time dependent trajectories in $\mathbb{R}^d$. Formally, the so-called failure domain F writes

$$F = \{\mathbf{x} \in \mathbb{X} \mid g(\mathbf{x}) > q\}$$

for a given $q \in \mathbb{R}$.

The computer code may be a coupling of several codes modelling different physical phenomena (hydrodynamic, thermodynamic, mechanic, ...) and has no analytical expression: the underlying physical phenomena have no analytical solutions. This is often called a *black-box* model because for any $\mathbf{x}$ in $\mathbb{X}$ it is possible to evaluate $y = g(\mathbf{x})$, but no other knowledge of "what happens" is available.

While the numerical code g is deterministic, the uncertainty on the inputs (*exact* values of physical parameters for instance, solution of a stochastic differential equation, ...) leads to a probabilistic approach to the problem. For a given input $\mathbf{x} \in \mathbb{X}$, the response *safety/failure*, *i.e.* $\mathbf{x} \notin F$ or $\mathbf{x} \in F$, is deterministic. On a given underlying probability space $(\Omega, \mathcal{F}, \mathrm{P})$, one can instead consider the random variable $\mathbf{X}$ with known distribution μ^X modelling the uncertainty on the parameters. The firm response *safety/failure* becomes

a probability that $\mathbf{X}$ be in F:

$$\mathrm{P}\left[\mathbf{X} \in F\right] = \mathrm{P}\left[g(\mathbf{X}) > q\right].$$

Since the system is supposed to be designed to operate in safe conditions (in some sense above mentioned), this probability is expected to be very low. Yet one wants to be able to estimate it to eventually guarantee that the system is *safe with great confidence*. The level of required confidence in our applications gives a probability of failure typically lower than 10^{-6}, or one over one million of trials.

Furthermore these numerical codes are often very time-consuming and can take several hours to several days to run. Hence very few trials will be possible to estimate the sought probability. In this context, the well-known Monte Carlo method, which basically consists in repeating independent trials and counting the number of failures found is not an option and advanced methods have to be used. To fix the idea, in our industrial setting, the question is somehow to insure with few hundred samples that a system is unsafe with a probability lower than 10^{-6}.

Note that the extreme value theory is not well suited for this class of problems. Essentially it makes use of available data and is able to estimate a probability that a rare event *according to these data* can arise. In our context we can generate the data and the question is rather "what data should be simulated to give the most precise estimation with the less samples?".

We introduce the general concepts and methods used to address this issue in Part I. The first chapter is inspired by the book of Rubino et al. [2009]; the reader is referred to it and references therein for more details about Monte Carlo methods for rare events. General introductions to Monte Carlo methods can also be found in the books by Robert and Casella [2004] or Rubinstein and Kroese [2011]. In Chapter 1 we present the more usual Monte Carlo methods. We start by presenting the original Monte Carlo method in Section 1.1, now referred to as the *crude* Monte Carlo method, based on the idea of repeating independent and identically distributed (*iid.*) trials [Metropolis and Ulam, 1949]. Then in Section 1.2 we present the Importance Sampling strategy, which consists in using another distribution for $\mathbf{X}$ making the failure event *less rare*. In Section 1.3 we come up with the Splitting method. With origins dating back to [Kahn and Harris, 1951] in the dynamical setting $\mathbb{X} \subset (\mathbb{R}^d)^{\mathbb{R}}$ and revisited in the static setting by Au and Beck [2001], it aims at *splitting* the rare event into a sequence of *less rare* events such that the failure domain is progressively reached with conditional simulations. Section 1.4 gives a brief description of the main concept and limitations of the nested sampling method [Skilling, 2006]. This method has been developed in a Bayesian framework for evidence estimation and we will show how it is linked with rare event simulation tools in Chapter 4. Finally we give in Section 1.5 the usual criteria used for comparing rare event estimators.

In Chapter 2 we address the issue of using surrogate models for extreme event simulation. Indeed, the above mentioned industrial setting (computational time of g) does not allow

for the use of the advanced statistics described in Chapter 1 because they are still too time consuming. In this context we present in Section 2.1 the classes of metamodels mostly used in the rare event setting, following [Sudret, 2012]. The issue of approximating a computer code from some of its input-output couples is rather general and falls into the field of *Machine learning* [DasGupta, 2011]. When the samples are generated by the user/algorithm itself, one speaks of *Computer experiments*. A general introduction to these techniques can be found in the books by Sacks et al. [1989], Santner et al. [2003], Fang et al. [2005] or Forrester et al. [2008]. Specific to the *Computer experiments* setting is the creation of the so-called *Design of Experiments (DoE)*, *i.e.* the question of where to call the code g. We present in Section 2.2 the different strategies used in the rare event case. Finally, Section 2.3 summarises the different possible ways to use the metamodel to produce an estimator of the sought probability.

Parallel computers have led to a paradigm shift for building algorithms. While the power of one given CPU (Central Processing Unit) does not increase much any more, computer clusters have become the standard for High Performance Computing (HPC). The main concept is that instead of having one single *core* (CPU) available, one can use *simultaneously* several (hundreds of) computers. For instance, the supercomputer Airain used for the numerical applications of this thesis allows us to run tasks on more than 2000 cores at 2.7 Ghz each.

In order to use these computational facilities, new algorithms have to be designed to be used *in parallel*. This means that they have to enable parallel computing, in other words that their different operations should be made in parallel. These considerations led us to define the point process framework for rare event simulation presented in Chapter 3. In Section 3.2 we define the main theoretical tool used throughout this thesis, it is a Poisson process associated with any continuous real-valued random variable. Using this Poisson process we are able to derive parallel probability (Section 3.3) and quantile (Section 3.4) estimators. Especially we show that the optimal Multilevel Splitting estimator which is totally sequential is indeed a particular implementation of this new parallel estimator. We further relax the continuity hypothesis of the *cdf* in Section 3.5 and are able to provide the distributions of the counting random variables of the point process (not Poisson any more). We also provide corrected probability estimators. Especially one of them has the same distribution as in the continuous case. Numerical examples of Section 3.6 illustrate the efficiency of the estimators as well as the impact of discontinuities.

The point process framework defined in Chapter 3 not only produces an estimator of the probability $\mathrm{P}\left[g(\mathbf{X}) > q\right]$ but of the whole cumulative distribution function (*cdf*) of $g(\mathbf{X})$ over $(-\infty, q]$. This property lets us define an estimator for the (conditional) mean of any real-valued random variable of the form $Y = g(\mathbf{X})$ which does not require the finiteness of the second order moment to have a finite variance (Section 4.2). Indeed in some applications one can be interested in estimating conditional moments of the form $\mathrm{E}\left[Y \mid Y > q\right]$, for instance the Mean Excess Loss $\mathrm{E}\left[Y - q \mid Y > q\right]$ in finance (also called

the Mean Residual Life in insurance). This estimator is indeed related to the nested sampling method and brings a new theoretical support for this algorithm. Furthermore we propose optimal weights removing its bias and minimising its variance. In Section 4.3 we use recent results on randomised estimators [McLeish, 2011, Rhee and Glynn, 2015] to address the issue of nested sampling termination. Especially we show that with a specific randomising procedure given in Section 4.3.1, the nested sampling method can remain unbiased with an almost surely finite number of terms. Furthermore this new estimator only doubles the variance of the ideal one with an infinite number of terms. In Section 4.4 we thoroughly study the special case where $Y = g(\mathbf{X})$ is heavy-tailed. Finally we discuss the parallel implementation and present numerical results in Section 4.5.

All these methods still require an important number of simulations and are not usable in some contexts such as the industrial one addressed here. In Chapter 5 we apply the point process framework to metamodel-based algorithms. We focus on kriging [Krige, 1951, Matheron, 1963, 1969, Wackernagel, 2013, Chiles and Delfiner, 2009] and show that all the results derived with the deterministic code g remain valid using instead a random process ξ with known distribution. We especially focus on Gaussian Process regression [Rasmussen and Williams, 2006] and are able in this context to use the Poisson process framework to monitor and drive efficiently the *learning* of the model with an easy computation of the Stepwise Uncertainty Reduction (SUR) strategy [Bect et al., 2012]. We also suggest new SUR criteria well suited for extreme events simulation. In Section 5.2 we develop the general framework of an algorithm combining Poisson processes for rare event simulations and metamodeling. Finally we apply this new algorithm in Section 5.3 to academic test cases as well as to an industrial problem from CEA: the assessment of the reliability of a containment vessel under dynamical loading. Finally, Appendix A gathers all the technical results on parallel implementation and a study of the impact of parallel computing on the properties of the estimators.

Part I

Introduction

Chapter 1

Monte Carlo methods for rare events

Throughout this thesis, we consider $g : \mathbf{x} \in \mathbb{X} \mapsto y \in \mathbb{R}$ a deterministic function standing for the computer code. $\mathbf{x} \in \mathbb{X}$ is a finite- or infinite dimensional vector: $\mathbf{x}$ is either a set of parameters of the computer code g and thus $\mathbb{X} \subset \mathbb{R}^d$ for some $d \geq 1$ or a time-dependent trajectory $\mathbf{x} = (\mathbf{x}_t)_t$ and $\mathbb{X} \subset (\mathbb{R}^d)^{\mathbb{R}}$. We also consider the so-called failure domain $F = \{\mathbf{x} \in \mathbb{X} \mid g(\mathbf{x}) > q\}$ for some $q \in \mathbb{R}$. The possible variations of the model parameters are represented by a probability distribution over $\mathbb{X}$. In the sequel we then consider random inputs $\mathbf{X}$ defined on an underlying probability space $(\Omega, \mathcal{F}, \mathrm{P})$ with induced measure μ^X over $\mathbb{X}$. g is assumed to be measurable and $Y = g(\mathbf{X})$ is hence a real-valued random variable with distribution μ^Y. While the law μ^X is always supposed to be known, the random variable Y is only accessible through the computer code g and is then always unknown. Formally, the problem of estimating the probability of failure p of the system modelled by g reduces to measuring the set F:

$$p = \mu^X(F) = \mathrm{P}\left[g(\mathbf{X}) > q\right] = \int_{\mathbb{X}} \mathbb{1}_{g(\mathbf{x})>q} \mathrm{d}\mu^X(\mathbf{x}) \tag{1.1}$$

with $\mathbb{1}_{\text{cond}}$ the indicator function returning 1 if the condition is verified and 0 otherwise.

This integration problem cannot be handled with usual quadrature rules because of the dimension of $\mathbb{X}$ and the low value of p. A usual tool to estimate Eq. (1.1) is the crude Monte Carlo method and we introduce it in Section 1.1. We discuss its statistical properties and a possible implementation is given as well as an illustration on a toy example.

However this method is not well suited when both the probability is low and the computer code expensive to evaluate. In this scope we present in section 1.2 the *Importance Sampling* method which aims at building an other random variable $\tilde{Y}$ with the same expectation as $\mathbb{1}_{Y>q}$ but a lower variance. While this method can be very efficient it requires also some knowledge on the computer code. This cannot be verified here and in the worst case, this can even turn this algorithm into a poorer method than the crude Monte Carlo. In our setting it is not applicable and we come up with the Splitting method in Section 1.3.

This method goes back to Kahn and Harris [1951] and has been widely used and studied since then. Amongst other approaches we choose here to present it from the *computational budget* point of view. Roughly speaking it means that we focus on the precision of an estimator for a given fixed number of generated samples. This makes it perhaps a little bit different to what is seen elsewhere, but in the end we come up with

the more up-to-date references and results for these algorithms. The reader interested in different approaches (Fixed Effort, Fixed Splitting, Fixed Success for instance) is referred to [Diaconis and Holmes, 1995, Glasserman et al., 1996, 1999, Garvels, 2000, Amrein and Künsch, 2011, Rubinstein et al., 2012] and references therein. Also somehow related to Splitting but specific to the dynamic case, the RESTART method [Villén-Altamirano and Villén-Altamirano, 1991] will not be addressed here.

The simulation methods presented before are designed for the estimation of extreme probability. Yet there is a more general way to use them, precisely to estimate the expectation of any real-valued random variable of the form $Y = g(\mathbf{X})$. Widely used in Bayesian analysis, the nested sampling method [Skilling, 2006] indeed implements a specific case of the splitting method described in Section 1.3 and is presented in Section 1.4.

In Section 1.5 we present the usual notions used in rare event simulation to compare algorithms and comment practical implementation with a special focus on parallel computing.

Throughout the manuscript, a given estimator of p will be denoted by $\widehat{p}$, its standard deviation will be σ and its coefficient of variation $\delta = \sigma / \mathrm{E}\left[\widehat{p}\right]$. When it is unbiased, δ is also the relative Root Mean Squared Error (rRMSE) of the estimator.

1.1 Crude Monte Carlo method

The crude Monte Carlo method is a general tool to estimate integrals. It writes the sought quantity as the expectation of a given random variable and only requires to be able to generate such random variables. In this framework, Eq. (1.1) becomes:

$$p = \mathrm{P}\left[g(\mathbf{X}) > q\right] = \mathrm{E}\left[\mathbb{1}_{g(\mathbf{X})>q}\right]$$

and the random variable of interest is simply the indicator function of the model being actually defective or not, *i.e.* a Bernoulli random variable with probability of success p the sought value. From a practical point of view it can be generated by first sampling a random $\mathbf{X}$ according to μ^X and then running the code g onto $\mathbf{X}$.

1.1.1 Theoretical definition

Let us now consider $N \geq 1$ independent and identically distributed (*iid.*) samples $(\mathbf{X}_i)_{i=1}^N \sim \mu^X$ and the following statistic:

$$\widehat{p}_{\mathrm{MC}} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{g(\mathbf{X}_i)>q}. \tag{1.2}$$

$\widehat{p}_{\mathrm{MC}}$ is the crude Monte Carlo estimator of p. It is unbiased:

$$\mathrm{E}\left[\widehat{p}_{\mathrm{MC}}\right] = \frac{1}{N}\sum_{i=1}^{N}\mathrm{E}\left[\mathbb{1}_{g(\mathbf{X}_i)>q}\right] = p,$$

and since the $(\mathbf{X}_i)_i$ are independent, one has the following expression for its variance:

$$\mathrm{var}\left[\widehat{p}_{\mathrm{MC}}\right] = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{var}\left[\mathbb{1}_{g(\mathbf{X}_i)>q}\right] = \frac{p(1-p)}{N}. \tag{1.3}$$

Hence the standard deviation $\sigma_{\mathrm{MC}} = \sqrt{\mathrm{var}\left[\widehat{p}_{\mathrm{MC}}\right]}$ of the crude Monte Carlo estimator decreases as $1/\sqrt{N}$ and so achieves a squared-root convergence rate. Furthermore, this estimator is supported by two main theorems, it is the Strong Law of Large Numbers:

$$\mathrm{P}\left[\lim_{N\to\infty}\widehat{p}_{\mathrm{MC}} = p\right] = 1$$

and the Central Limit Theorem:

$$\sqrt{N}\,\frac{\widehat{p}_{\mathrm{MC}} - p}{\sqrt{p(1-p)}} \xrightarrow[N\to\infty]{\mathcal{L}} \mathcal{N}(0,1).$$

In other words the Strong Law of Large Numbers insures that the more samples used for $\widehat{p}_{\mathrm{MC}}$, the more probable it is that the statistic be equal the sought probability p, while the Central Limit Theorem gives the limit distribution of the estimator, which can be useful for building confidence intervals. Indeed, for N large enough, it is often assumed that:

$$\widehat{p}_{\mathrm{MC}} \overset{\mathcal{L}}{\sim} \mathcal{N}\left(p, \sigma^2_{\mathrm{MC}}\right). \tag{1.4}$$

Now, letting α be a given probability (often $\alpha = 0.05$ or $\alpha = 0.01$) one can build $\alpha-$confidence intervals for p, *i.e.* random intervals $\mathcal{I}_\alpha$ based on $\widehat{p}_{\mathrm{MC}}$ such that $\mathrm{P}\left[p \in \mathcal{I}_\alpha\right] \geq \alpha$. From the approximate law of $\widehat{p}_{\mathrm{MC}}$, one has:

$$\mathrm{P}\left[-Z(\alpha) < \frac{\widehat{p}_{\mathrm{MC}} - p}{\sigma_{\mathrm{MC}}} < Z(\alpha)\right] = \mathrm{P}\left[\widehat{p}_{\mathrm{MC}} - \sigma_{\mathrm{MC}}Z(\alpha) < p < \widehat{p}_{\mathrm{MC}} + \sigma_{\mathrm{MC}}Z(\alpha)\right] = 1 - \alpha$$

with $Z_{1-\alpha/2}$ the quantile of order $1-\alpha/2$ of a standard Gaussian random variable. Usual values are $Z_{0.975} = 1.96$ and $Z_{0.995} = 2.58$. Since σ_{MC} is unknown, it is also estimated with $\widehat{\sigma_{\mathrm{MC}}} = \sqrt{\widehat{p}_{\mathrm{MC}}(1-\widehat{p}_{\mathrm{MC}})/(N-1)}$. This estimator almost surely converges toward σ_{MC}. The confidence interval eventually writes [Rubino et al., 2009]:

$$\mathcal{I}_\alpha = \left(\widehat{p}_{\mathrm{MC}} - \sqrt{\frac{\widehat{p}_{\mathrm{MC}}(1-\widehat{p}_{\mathrm{MC}})}{N-1}}Z_{1-\alpha/2},\, \widehat{p}_{\mathrm{MC}} + \sqrt{\frac{\widehat{p}_{\mathrm{MC}}(1-\widehat{p}_{\mathrm{MC}})}{N-1}}Z_{1-\alpha/2}\right).$$

1.1.2 Limitation

As N goes to infinity the standard deviation σ_{MC} goes to zero and the interval becomes narrower: its width is $2\sigma_{\text{MC}}Z(\alpha) \propto 1/\sqrt{N}$. However this width is to be compared with p to see how far the estimator can depart from the sought value. More precisely one has:

$$\delta^2_{\text{MC}} = \frac{\sigma^2_{\text{MC}}}{p^2} = \frac{p(1-p)}{Np^2} = \frac{1-p}{Np}. \tag{1.5}$$

When $p \ll 1$, Eq. (1.5) gives:

$$N = \frac{1-p}{p\delta^2_{\text{MC}}} \approx \frac{1}{p\delta^2_{\text{MC}}}.$$

This means that the required number of samples N to achieve a given target precision δ_{MC} on the Monte Carlo estimator is directly inversely proportional to the sought probability times the squared precision. A relatively rough precision of 10% hence gives $N = 10^2 p^{-1}$. In the applications considered further in thesis this is clearly not affordable; as a matter of comparison with a computer code running in 5 minutes on a cluster and a failure probability of 10^{-6}, this would make ≈ 951 years to compute.

To circumvent this limitation, low variance estimators have been proposed and will be further introduced in Sections 1.2 and 1.3.

1.1.3 Practical implementation

Despite this relatively slow convergence, the crude Monte Carlo estimator can be efficiently used when either the probability is not so small or the code is much faster to run. Furthermore it is also the base tool of the advanced estimators introduced further in Section 1.2 and 1.3 and serves as a reference in academic test cases when no analytical solution is available.

We then present in Algorithm 1 a sequential basic approach to implement a Monte Carlo estimator.

Algorithm 1 A sequential algorithm for crude Monte Carlo estimator

Require: N_{MC} a total number of samples or δ_{MC} a given coefficient of variation

Require: a sampler of μ^X

$N = 0; \delta = \infty; \widehat{p}_{\text{MC}} = 0$

while $N < N_{\text{MC}}$ or $\delta > \delta_{\text{MC}}$ **do**

Sample $\mathbf{X} \sim \mu^X$; $N \leftarrow N + 1$

$\widehat{p}_{\text{MC}} = \widehat{p}_{\text{MC}} + \mathbb{1}_{g(\mathbf{X})>q}$

$\delta = \sqrt{(N - \widehat{p}_{\text{MC}})/((N-1)\widehat{p}_{\text{MC}})}$ ▷ $N = 1$ or $\widehat{p}_{\text{MC}} = 0$ gives $\delta = \infty$

end while

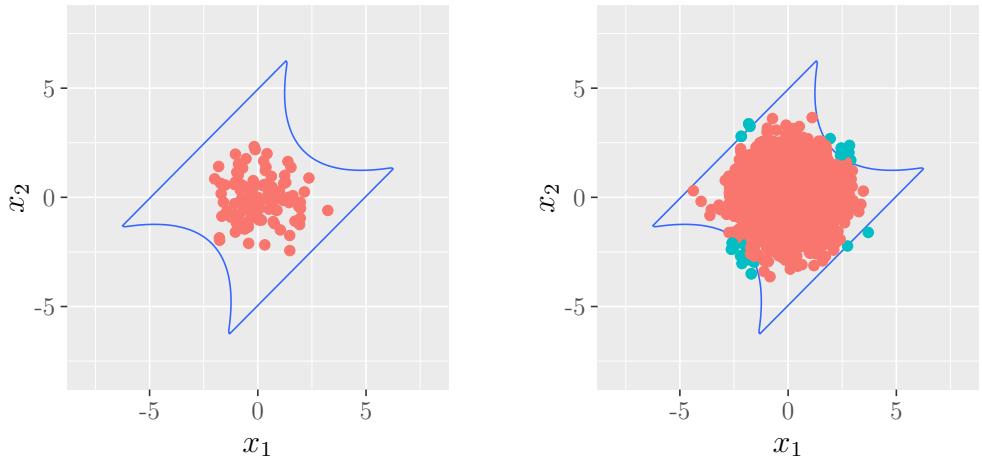
$\widehat{p}_{\text{MC}} = \widehat{p}_{\text{MC}}/N$

Remark 1.1. *In Algorithm 1, the variance of $\widehat{p}_{MC}$ is estimated with $\widehat{p}_{MC}(1-\widehat{p}_{MC})/(N-1)$ instead of $\widehat{p}_{MC}(1-\widehat{p}_{MC})/N$. This classical correction applied to get an unbiased estimator of the variance is sometimes omitted because N is supposed to be large. Eventually the estimator of the coefficient of variation does not provide any guarantee. Generally speaking, the estimation of the variance is even more sensitive to the rare event than the probability itself. Further details on this topic can be found in [Glynn and Whitt, 1992, Rubino et al., 2009].*

Figure 1.1 below shows an example of the use of the crude Monte Carlo method to estimate $p = \mathrm{P}\left[g(\mathbf{X}) < q\right]$ with:

$$g : \mathbf{x} \in \mathbb{R}^2 \mapsto \min \begin{cases} 3 + \dfrac{(x_1 - x_2)^2}{10} - \dfrac{|x_1 + x_2|}{\sqrt{2}} \\ -|x_1 - x_2| + 7/\sqrt{2} \end{cases} \tag{1.6}$$

[originally defined by Waarts, 2000], $\mathbf{X} \sim \mathcal{N}(0, \mathbf{I})$ a standard Gaussian vector ($\mathbf{I}$ is the 2×2 identity matrix) and $q = 0$.



(a) $N = 100$ (b) $N = 10^4$

Figure 1.1: Example of a crude Monte Carlo estimation on the standard Gaussian input space with different sample sizes N with a limit-state function defined in Eq. (1.6). The dots are the N *iid.* samples $(\mathbf{X}_i)_{i=1}^N \sim \mu^X$: blue dots are failing samples while red ones are in the safety domain. The boundary, *i.e.* the level set $\{\mathbf{x} \in \mathbb{R}^2 \mid g(\mathbf{x}) = 0\}$ (unknown in real settings) is added for visual purpose.

The corresponding estimated probabilities in Figure 1.1 are $\widehat{p}_{\mathrm{MC}}(100) = 0/100 = 0$ and $\widehat{p}_{\mathrm{MC}}(10^4) = 24/10^4 = 2.4 \times 10^{-3}$ with estimated variances 0 and 2.39×10^{-7} respectively (see Table 1.1) . The first estimator with only 100 samples is clearly not able to recover the target probability while with an order of magnitude of 10^{-3} the estimator $\widehat{p}_{\mathrm{MC}}(10^4)$ gives a precision of order 10%.

N	$\widehat{p}_{\mathrm{MC}}$	$\widehat{\delta_{\mathrm{MC}}}$
10^2	0	∞
10^4	2.4×10^{-3}	0.204124

Table 1.1: Estimation of a moderately low probability with the crude Monte Carlo method.

1.2 Importance Sampling

The limitation seen in the crude Monte Carlo estimator comes from the fact that the standard deviation of a Bernoulli random variable with parameter p becomes much larger than its expectation when the parameter p is small. In this context, the idea behind Importance Sampling is to consider an other random variable with the same mean but a lower variance. Reader is referred to [Glynn and Iglehart, 1989, Juneja and Shahabuddin, 2006] and references therein for more information about Importance Sampling.

Let Y be a real-valued random variable with distribution μ^Y and finite expectation $\mathrm{E}[Y] < \infty$; and $\tilde{Y}$ be a real-valued random variable with distribution $\mu^{\tilde{Y}}$ such that μ^Y is absolutely continuous with respect to $\mu^{\tilde{Y}}$:

$$\forall B \in \mathcal{B}(\mathbb{R}),\ \mu^{\tilde{Y}}(B) = 0 \Rightarrow \mu^Y(B) = 0.$$

The Radon-Nikodym theorem states that μ^Y has a derivative L against $\mu^{\tilde{Y}}$, such that:

$$\forall B \in \mathcal{B}(\mathbb{R}),\ \mu^Y(B) = \int_B L \mathrm{d}\mu^{\tilde{Y}}.$$

If one assumes that both μ^Y and $\mu^{\tilde{Y}}$ have a density with respect to the Lebesgue measure, f and $\tilde{f}$ respectively, then $L = f/\tilde{f}$ and so it is often called the *likelihood ratio* of μ^Y with respect to $\mu^{\tilde{Y}}$. Finally, the problem of estimating the expectation of Y can be rewritten:

$$\mathrm{E}[Y] = \int_{\mathbb{R}} y \mathrm{d}\mu^Y(y) = \int_{\mathbb{R}} yL(y) \mathrm{d}\mu^{\tilde{Y}}(y) = \mathrm{E}\left[\tilde{Y}L(\tilde{Y})\right].$$

Hence it is possible to estimate $\mathrm{E}[Y]$ with a Monte Carlo estimator and samples generated according to $\mu^{\tilde{Y}}$ given that the generated samples are multiplied by the likelihood ratio. The question is now to find the optimal distribution $\mu^{\tilde{Y}}$ – *i.e.* the distribution which will minimize the variance of the Monte Carlo estimator based on $\tilde{Y}$ – such that the domination condition remains true. In this context, the Cauchy-Schwarz inequality stands:

$$\mathrm{E}\left[\left(\tilde{Y}L(\tilde{Y})\right)^2\right] \geq \mathrm{E}\left[\left|\tilde{Y}L(\tilde{Y})\right|\right]^2$$

with equality *iff.* $1 \propto |\tilde{Y}L(\tilde{Y})|$. This equality is solved using the fact that a probability

measure sums up to 1 and one finally finds that the optimal distribution is:

$$\forall B \in \mathcal{B}(\mathbb{R}),\ \mu^{\tilde{Y}}(B) = \int_B \frac{|y|}{\int |y| \mathrm{d}\mu^Y(y)} \mathrm{d}\mu^Y(y).$$

Then the optimal distribution strongly depends on the problem at stake and if $Y \geq 0$ the optimal Importance Sampling estimator has a null variance. More precisely in our setting, $y = \mathbb{1}_{g(\mathbf{x})>q} \in \{0, 1\}$ and one finds:

$$\forall y \in \mathbb{R},\ \mathrm{d}\mu^{\tilde{Y}}(y) = \frac{y}{p} \mathrm{d}\mu^Y(y).$$

In other words, the optimal importance distribution only samples *successes* of the Bernoulli random variable. One can write this in terms of distributions over $\mathbb{X}$ and the optimal importance distribution $\mu^{\tilde{X}}$ eventually writes:

$$\forall \mathbf{x} \in \mathbb{X},\ \mathrm{d}\mu^{\tilde{X}}(\mathbf{x}) = \frac{\mathbb{1}_{g(\mathbf{x})>q}}{p} \mathrm{d}\mu^X(\mathbf{x}).$$

While not of great practical use, it lets define some adaptive strategies to approximate it, for example by selecting an importance distribution amongst a family of *well-chosen* ones minimizing a given distance to the optimal distribution. The well-known cross-entropy method is one of those and the interested reader is referred to [Rubinstein, 1999, Rubino et al., 2009, Rubinstein, 2009a] for more information.

Indeed in our setting, no assumption has to be made on the computer code. In this context it may be very difficult to find an appropriate importance distribution and a bad choice may lead to an estimator with an even greater variance. Even more difficult while impossible is to insure that the absolute continuity assumption is verified. For all these reasons this estimator will not be considered as an option later in this thesis.

1.3 Splitting

Splitting, originally called multilevel splitting [Kahn and Harris, 1951] in a dynamic case ($\mathbf{X}$ is a trajectory) and rediscovered in the static case by Au and Beck [2001] under the name *Subset Simulation* is based on the idea that the rare event should be split into several events such that going from one to another such events is *less rare*.

Remark 1.2. *In the sequel we will use indifferently the terms* splitting *and* multilevel splitting *while* Subset Simulation *refers to a specific implementation developed in Section 1.3.2.*

More precisely, Splitting makes use of a finite sequence of nested subsets $(F_i)_{i=0}^m$ such that $F_0 = \mathbb{X}$ and $F_m = F$ the original failure domain. Then using the Bayes' rule it writes

the sought probability as a product of conditional ones:

$$\mathrm{P}\left[\mathbf{X} \in F\right] = \prod_{i=1}^{m} \mathrm{P}\left[\mathbf{X} \in F_i \mid \mathbf{X} \in F_{i-1}\right] = \prod_{i=1}^{m} p_i \tag{1.7}$$

with $\forall i \in [\![1, m]\!]$, $p_i = \mathrm{P}\left[\mathbf{X} \in F_i \mid \mathbf{X} \in F_{i-1}\right]$.

1.3.1 Ideal splitting

From Eq. (1.7) the goal is then to estimate each conditional probability *independently* using crude Monte Carlo estimation. Let us consider that each p_i is estimated with a number of samples N_i. Denote by $\widehat{p}_i$ the corresponding estimators, one has:

$$\forall i \in [\![1, m]\!],\ \mathrm{E}\left[\widehat{p}_i\right] = p_i \text{ and } \mathrm{var}\left[\widehat{p}_i\right] = \frac{p_i(1-p_i)}{N_i}.$$

Then, the multilevel splitting estimator $\widehat{p}_{\mathrm{MS}}$ is defined as follows:

$$\widehat{p}_{\mathrm{MS}} = \prod_{i=1}^{m} \widehat{p}_i. \tag{1.8}$$

Since the $\widehat{p}_i$ are independent, one can derive its mean and coefficient of variation:

$$\mathrm{E}\left[\widehat{p}_{\mathrm{MS}}\right] = \mathrm{E}\left[\prod_{i=1}^{m} \widehat{p}_i\right] = \prod_{i=1}^{m} \mathrm{E}\left[\widehat{p}_i\right] = \prod_{i=1}^{m} p_i = p$$

and:

$$\delta^2_{\mathrm{MS}} = \frac{\prod_{i=1}^{m} \mathrm{E}\left[{\widehat{p}_i}^2\right] - p^2}{p^2} = \prod_{i=1}^{m} \left(\frac{\mathrm{var}\left[\widehat{p}_i\right] + p_i^2}{p_i^2}\right) - 1 = \prod_{i=1}^{m} (\delta_i^2 + 1) - 1.$$

For a given number of subsets m and a total number of samples N one can look for the parametrisation (choice of the subsets and sample size for the probability estimations) minimising the squared coefficient of variation:

$$\underset{\substack{p_i \in (0,1],\, i \in [\![1,m]\!] \\ N_i \geq 1,\, i \in [\![1,m]\!]}}{\operatorname{argmin}} \prod_{i=1}^{m} (\delta_i^2 + 1) - 1 \ \ s.t. \ \ \prod_{i=1}^{m} p_i = p;\ \sum_{i=1}^{m} N_i = N. \tag{1.9}$$

Here as in the crude Monte Carlo method the parameter N is supposed to be fixed and represents somehow the total computational cost one can afford. This point will be further developed in Section 1.5. Let λ_p and λ_N be the Lagrange multipliers for the constrained optimisation problem (1.9). In addition to the two constrained equalities, the system eventually reduces to m pairs of equations:

$$\begin{cases} \Delta_i = N_i p_i p \lambda_p \\ \Delta_i (1 - p_i) = N_i^2 p_i \lambda_N \end{cases}$$

for all $i \in [\![1, m]\!]$, with $\Delta_i = 2(\delta_{\text{MS}}^2 + 1)\delta_i/(\delta_i^2 + 1)$. Hence the couples $(p_i, N_i)_{i=1}^m$ satisfy all the same equations and are equal to some (p_0, N_0). Finally, using the constraints one finds:

$$\begin{cases} \prod_{i=1}^{m} p_i = p & \Rightarrow \forall i \in [\![1, m]\!],\ p_i = p_0 = p^{\frac{1}{m}} \\ \sum_{i=1}^{m} N_i = N & \Rightarrow \forall i \in [\![1, m]\!],\ mN_i = mN_0 = N \end{cases}. \tag{1.10}$$

Note that this theoretical optimisation is done on the augmented domain $\forall i \in [\![1, m]\!],\ N_i \in \mathbb{R}_+^*$. If the total budget N cannot be split into m equal parts, solution (1.10) is not feasible. However this result means that all the conditional probabilities have to be estimated with the same *precision*, *i.e.* with coefficient of variations δ_i of the same order of magnitude δ_0.

$$\delta_{\text{MS}}^2 = \sum_{i=1}^{m} \delta_i^2 + o(\delta_0^2),$$

which means that for any $N_i \leq N_j$, $(i, j) \in [\![1, m]\!]$, one should have $p_i \geq p_j$ such that $\delta_i = \delta_j$. In any cases, assuming that this solution is feasible leads to the following result:

$$\delta_{\text{MS}}^2 = (\delta_0^2 + 1)^m - 1 \tag{1.11}$$

with $\delta_0^2 = (p^{-1/m} - 1)m/N$. One can now wonder what is the best way to split the failure domain for a given number of samples N, *i.e.* if there is an optimal number $m \geq 1$ of subsets for a given N. If one looks for $m \in \mathbb{R}_+$, one has:

$$\frac{\partial \delta_{\text{MS}}^2}{\partial m} = \log\left(\delta_0^2 + 1\right)(\delta_0^2 + 1)^m + m(\delta_0^2 + 1)^{m-1}\frac{\partial \delta_0^2}{\partial m}. \tag{1.12}$$

Eq. (1.12) is intractable analytically. However, if one assumes that $N \gg 1$, it simplifies to:

$$\begin{aligned} 0 &= \delta_0^2 + m\frac{\partial \delta_0^2}{\partial m} \\ 0 &= (p^{-1/m} - 1)m + m(p^{-1/m} - 1 + \frac{\log p}{m}p^{-1/m}) \\ 0 &= 2(1 - p_0) + \log p_0. \end{aligned}$$

This latter equation has two solutions: either $p_0 = 1$ which is not possible, or $p_0^* \approx 0.2032$. Since the function $p_0 \in (0, 1) \mapsto 2(1 - p_0) + \log p_0$ is increasing until $p_0 = 0.5$, it is negative before 0.2 and positive afterwards. Then this solution is a minimum. This result is consistent with the approximation used: with this latter value $\delta_0^2 = -\log p/(2p_0^* N)$. In general settings, $p \lesssim 10^{-5}$ and $N \geq 1000$, which gives $\delta_0^2 \lesssim 10^{-2}$. Eventually we can conclude that the optimal splitting method with fixed total number of samples N is the

following [see Rubino et al., 2009, Section 3.3.2]:

$$\begin{cases} p_i = p_0^* \approx 0.2032 & \forall i \in [\![1, m]\!] \\ m = \dfrac{\log p}{\log p_0^*} \approx -0.63 \log p & \\ N_i = \dfrac{N}{m} = \dfrac{N \log p_0^*}{\log p} \approx 1.59 \dfrac{N}{-\log p} & \forall i \in [\![1, m]\!] \\ \delta_i^2 = \delta_0^2 = \dfrac{(p_0^*)^{-1} - 1}{N} \dfrac{\log p}{\log p_0^*} \approx 2.46 \dfrac{-\log p}{N} & \forall i \in [\![1, m]\!] \\ \delta_{\text{MS}}^2 = (\delta_0^2 + 1)^m - 1 = \dfrac{(\log p)^2}{N} \dfrac{1 - p_0^*}{p_0^* (\log p_0^*)^2} + o\left(\dfrac{1}{N}\right) & \\ \quad \approx 1.54 \dfrac{(\log p)^2}{N} + o\left(\dfrac{1}{N}\right) & \end{cases} \tag{1.13}$$

This optimality result has been found neglecting the fact that N and m are integers. Furthermore this optimal solution depends on p the sought value and is thus not implementable. However as for the Importance Sampling estimator (see Section 1.2) it can give guidelines for a practical suboptimal implementation.

1.3.2 Adaptive splitting

The optimal settings for the splitting method found in Section 1.3.1 are not usable in practice. Indeed they require to know in advance the target probability p and the *cdf* of $g(\mathbf{X})$ to define the sequence $(F_i)_{i=1}^m$. Moreover it is not possible to generate *iid.* conditional samples from scratch. While it could be possible to use an oracle sequence $(q_i)_{i=1}^m$ to define suboptimal subsets, or to get such a sequence from a first pilot run [Botev and Kroese, 2008, 2012], an other option is to define the splitting on-the-fly while the algorithm is running. These strategies are referred to as adaptive.

Subset Simulation The basic idea is to select the subsets iteratively with a heuristic approach based on a current *population*, *i.e.* a set of *iid.* samples. The optimality result for the conditional probability p_0^* gives a possible choice for this heuristic method: get q_i as the crude Monte Carlo estimator of the p_0 quantile, with p_0 a given arbitrary value (often set to 0.1). A direct application of this principle leads to the *Subset Simulation* method (see Algorithm 2).

On lines 4 and 9 of Algorithm 2, q_{m+1} is the crude Monte Carlo estimator of a quantile of order $1 - p_0$, *i.e.* the $\lceil 1 - p_0 \rceil$ ordered statistic of the N_0-samples [see for example Arnold et al., 1992]. The sequential approach of this algorithm does not only serve the definition of the subsets but also to sample from the conditional distributions (line 7). Indeed its link with Sequential Monte Carlo methods [Chopin, 2002, Del Moral et al., 2006, Le Gland, 2007, Vergé et al., 2013, Smith et al., 2013, Beskos et al., 2016] and Interacting

Algorithm 2 *Subset Simulation* [Au and Beck, 2001, Cérou et al., 2012]

Require: $N_0 \geq 1$ ▷ the Monte Carlo population size for each subset

Require: $p_0 \in (0,1)$ ▷ a probability for defining the thresholds

Require: a sampler of $\mu^X(\cdot \mid g(\mathbf{X}) > y), \forall y < y_R$ the right endpoint of Y.

$\widehat{p}_{\text{SS}} = 1$, $m = 0$, $q_m = -\infty$

Sample N_0 *iid. particles* $(\mathbf{X}_j)_{j=1}^{N_0} \sim \mu^X(\cdot \mid g(\mathbf{X}) > q_m)$

$\forall j \in [\![1, N_0]\!]$, $Y_j = g(\mathbf{X}_j)$

$q_{m+1} = Y_{(\lceil N_0(1-p_0) \rceil)}$

while $q_{m+1} < q$ **do**

  $m \leftarrow m + 1$

  Resample the $N_0(1-p_0)$ $\mathbf{X}_j$ such that $Y_j < q_m$ according to $\mu^X(\cdot \mid g(\mathbf{X}) > q_m)$

  Evaluate g on the new samples

  $q_{m+1} = Y_{(\lceil N_0(1-p_0) \rceil)}$

  $\widehat{p}_{\text{SS}} \leftarrow \widehat{p}_{\text{SS}} \times p_0$

end while

$q_{m+1} \leftarrow q$

$\widehat{p}_{r_0} = \dfrac{1}{N_0} \sum_{i=1}^{N_0} \mathbb{1}_{g(\mathbf{X}_i) > q}$

$\widehat{p}_{\text{SS}} = \widehat{p}_{\text{SS}} \times \widehat{p}_{r_0}$

Particles System [Del Moral, 2004] lets obtain the following result [Cérou et al., 2012]:

$$m \xrightarrow[N_0 \to \infty]{a.s.} m_0 \stackrel{\text{def}}{=} \lfloor \frac{\log p}{\log p_0} \rfloor. \tag{1.14}$$

Furthermore, assuming that the *cdf* of $Y = g(\mathbf{X})$ is continuous, the estimator $\widehat{p}_{\text{SS}}$ supports a Central Limit Theorem:

$$\sqrt{N_0}(\widehat{p}_{\text{SS}} - p) \xrightarrow[N_0 \to \infty]{\mathcal{L}} \mathcal{N}(0, \sigma^2_{\text{SS}}) \tag{1.15}$$

with $\sigma^2_{\text{SS}} = p^2 \left(m_0(p_0^{-1} - 1) + p_{r_0}^{-1} - 1 \right)$ and $p_{r_0} = p p_0^{-m_0}$.

The adaptive version of the splitting method succeeds in producing evenly spaced subsets for all but the last one. These convergence results show that the adaptive version of the algorithm converges toward the optimal one in some of its properties, namely the number of subsets and the variance of the estimator. Yet it is biased:

$$N_0 \frac{\mathrm{E}\left[\widehat{p}_{\text{SS}}\right] - p}{p} \xrightarrow[N_0 \to \infty]{} m_0(p_0^{-1} - 1). \tag{1.16}$$

This positive bias is a direct consequence of the use of a Monte Carlo based quantile estimator at each iteration. Furthermore, these results require the continuity hypothesis

of the *cdf* of $g(\mathbf{X})$, which cannot be verified in practice.

Adaptive Multilevel Splitting An other heuristic approach used to defined iteratively the sequence $(q_i)_{i=1}^m$ is to select the $k-$ordered statistic. We will refer to this implementation as the Adaptive Multilevel Splitting (AMS) method; it is described in Algorithm 3.

Algorithm 3 Adaptive Multilevel Splitting [Cérou and Guyader, 2007, Bréhier et al., 2015a]

Require: $N_0 \geq 1$, $k \in [\![1, N_0 - 1]\!]$

Require: a sampler of $\mu^X(\cdot \mid g(\mathbf{X}) > y), \forall y < y_R$ the right endpoint of Y.

$M = 0$; $q_M = -\infty$; $\widehat{p}_{\text{AMS}} = 1$

Sample N_0 *iid. particles* $(\mathbf{X}_j)_{j=1}^{N_0} \sim \mu^X(\cdot \mid g(\mathbf{X}) > q_M)$

3: $\forall j \in [\![1, N_0]\!]$, $Y_j = g(\mathbf{X}_j)$

$q_{M+1} = Y_{(k)}$

while $q_{M+1} < q$ **do**

6: $M \leftarrow M + 1$

$\widehat{p}_{\text{AMS}} \leftarrow \widehat{p}_{\text{AMS}} \times \dfrac{1}{N_0} \sum_{i=1}^{N_0} \mathbb{1}_{g(\mathbf{X}_i) > q_M}$

Resample the particles such that $Y_j \leq q_M$ according to $\mu^X(\cdot \mid g(\mathbf{X}) > q_M)$

9: Evaluate g on the new samples

$q_{M+1} = Y_{(k)}$

end while

12: $q_{M+1} \leftarrow q$

$\widehat{p}_{\text{AMS}} = \widehat{p}_{\text{AMS}} \times \dfrac{1}{N_0} \sum_{i=1}^{N_0} \mathbb{1}_{g(\mathbf{X}_i) > q}$

Before the work of Bréhier et al. [2015a] the algorithm used at line 7 the ready-made formula $\widehat{p}_i = 1 - k/N$. However if the *cdf* of $g(\mathbf{X})$ is not continuous the ordered statistic may not be unique and the estimator is not consistent. The issue of possible discontinuities in the *cdf* of $g(\mathbf{X})$ has gained a lot of attention recently [Simonnet, 2016] and will be further discussed in Section 3.5. This correction insures the unbiasedness of the estimator in any case.

With the continuity hypothesis, Bréhier et al. [2015b,c] showed a Central Limit Theorem and gave theoretical formulae for the variance, whatever k. However one of the main differences between Algorithms 2 and 3 lies in the number of iterations: while it converges toward a deterministic value in the first case (see Eq. 1.14) it remains here a non-degenerate random variable and is thus denoted by capital M. We report here the results from [Bréhier

et al., 2015c, Section 5.2] with our notations:

$$\mathrm{E}\left[M\right] = \frac{N_0}{k}\left(-\log p - \frac{(k-1)(-\log p - 1)}{2N_0} + o\left(\frac{1}{N_0}\right)\right) \tag{1.17}$$

$$\mathrm{var}\left[\widehat{p}_{\mathrm{AMS}}\right] = \frac{p^2}{N_0}\left(-\log p + \frac{(\log p)^2 + (k-1)|\log p|}{2N_0} + o\left(\frac{1}{N_0}\right)\right). \tag{1.18}$$

While the expected number of iterations decreases with k (see Eq. 1.17), the variance of the estimator is the smallest with $k = 1$. An optimal setting can be found when looking at the variance against the total number of generated samples. Precisely the cost of an estimator is defined as the product of its coefficient of variation and the computing effort required to generate it (see also further Section 1.5 and Section 4.3):

$$\begin{aligned}\frac{\mathrm{var}\left[\widehat{p}_{\mathrm{AMS}}\right]}{p^2} \times \left(k\,\mathrm{E}\left[M\right] + N_0\right) = \left(|\log p| + |\log p|^2\right)\left(1 + \frac{|\log p|}{2N_0}\right) \\ + \frac{|\log p|(k-1)}{N_0} + o\left(\frac{1}{N_0}\right).\end{aligned} \tag{1.19}$$

Last Particle Algorithm Equations (1.18) and (1.19) show that the optimal (minimal variance and cost) choice for the adaptive multilevel splitting method is to set $k = 1$. With the continuity hypothesis this can be understood because this choice leads to evenly spaced subsets: all of them will be estimated with the same value $1 - 1/N$. This special case is referred to as the *Last Particle Algorithm* [Guyader et al., 2011, Simonnet, 2016]: at each iteration of Algorithm 3, only the *particle* with the smallest Y_j is resampled (see line 8; a complete description of the Last Particle Algorithm is given in Appendix A, Algorithm 22).

It can be noticed that with this setting and in the idealised case where conditional simulations are performed exactly, the random number of iterations M follows a Poisson distribution with parameter $-N_0 \log p$ [Huber and Schott, 2011, Guyader et al., 2011]. This result lets have the *exact* distribution of the estimator $\widehat{p}_{\mathrm{LPA}}$ whatever N_0 and so allows for building non-asymptotic confidence intervals. Finally the properties of the Last Particle Algorithm with the continuity hypothesis are the following ones:

$$\begin{cases} M \sim \mathcal{P}\left(-N_0 \log p\right) \\ \mathrm{E}\left[M\right] = \mathrm{var}\left[M\right] = -N_0 \log p \\ \widehat{p}_{\mathrm{LPA}} = \left(1 - \dfrac{1}{N_0}\right)^M \\ \mathrm{E}\left[\widehat{p}_{\mathrm{LPA}}\right] = p \\ \delta^2_{\mathrm{LPA}} = p^{-1/N_0} - 1 = \dfrac{-\log p}{N_0} + o\left(\dfrac{1}{N_0}\right) \end{cases}. \tag{1.20}$$

Inverting the relation between $\widehat{p}_{\mathrm{LPA}}$ and M, Guyader et al. [2011] also defined a quantile

estimator:

$$\widehat{q}_{\text{LPA}} = q_{M_0} \tag{1.21}$$

with $M_0 = \lceil \frac{\log p}{\log(1 - N_0^{-1})} \rceil$ and q_m the minimum found at iteration m of Algorithm 3 with $k = 1$. This estimator will be further discussed in Chapter 3, Section 3.4.

Comparison between ideal and adaptive splitting The adaptive strategy lets define two practical Splitting algorithms. In these implementations it is not possible to insure a total fixed number of generated samples N. The *Subset Simulation* method (Algorithm 2) lets approach it because its random number of iterations converges almost surely toward a constant. However it is biased.

The other implementation, referred to as Adaptive Multilevel Splitting (Algorithm 3) makes use of the $k-$ordered statistic to define iteratively the thresholds and then uses a crude Monte Carlo. It is unbiased and can handle discontinuities in the *cdf* of $g(\mathbf{X})$ (see also Section 3.5 on this latter issue). Amongst all the possible choices for $k \in [\![1, N-1]\!]$, $k = 1$ (Last Particle Algorithm) has been shown to be optimal in terms of total variance of the final estimator against expected total number of generated samples. Furthermore the distribution of the Last Particle Algorithm estimator is well-determined. In table 1.2 we summarise these results.

	Ideal splitting	Optimal adaptive splitting
Subset definition	$p_0^* \approx 0.2032$	$k = 1$
Nbr. of subsets	$m = \log p / \log p_0^*$	$M \sim \mathcal{P}(-N_0 \log p)$
Nbr. of samples N	$N_0 \times m$ deterministic	$M + N_0$ random
Coef. of var. δ^2	$\frac{-\log p}{N_0} \frac{(p_0^*)^{-1} - 1}{-\log p_0^*} + o\left(\frac{1}{N}\right)$	$\frac{-\log p}{N_0} + o\left(\frac{1}{N_0}\right)$
N VS δ^2	$\frac{(\log p)^2}{\delta^2} \frac{(p_0^*)^{-1} - 1}{(\log p_0^*)^2} + o\left(\frac{1}{\delta^2}\right)$	$\mathcal{P}\left(\frac{(\log p)^2}{\log(\delta^2 + 1)}\right) + \frac{-\log p}{\log(\delta^2 + 1)}$

Table 1.2: Comparison between ideal and adaptive splitting when the *cdf* of $g(\mathbf{X})$ is continuous. Note that $\frac{(p_0^*)^{-1} - 1}{-\log p_0^*} \approx 2.46$ and that $\frac{(p_0^*)^{-1} - 1}{(\log p_0^*)^2} = 1.54$ with $p_0^* = 0.2032$.

In a sequential strategy such as the one described in Algorithm 2, the fixed parameter is not N the total number of generated samples but N_0 the number of samples per step. In this context Table 1.2 shows that the optimal choice would be $p_0 \to 1$ in order to have $\delta^2_{\text{MS}} \approx -\log p_0 / N_0$. This is achieved with the Last Particle Algorithm, which shows that it is the best adaptive strategy for the Multilevel Splitting method. On the other hand, when N is fixed, one can calculate the probability that the total number of samples generated by

the optimal adaptive strategy is lower than the one with optimal $p_0^* \approx 0.2$. Using Normal approximation of a Poisson distribution and an asymptotic expansion in $1/\delta^2$ one has:

$$\begin{aligned} \mathrm{P}\left[N_{\mathrm{LPA}} < N_{\mathrm{MS}}\right] &= \mathrm{P}\left[\left(\frac{\log p}{\delta}\right)^2 + \left(\frac{\log p}{\delta}\right) U + \frac{|\log p|}{\delta^2} < \left(\frac{\log p}{\delta}\right)^2 1.54\right] \\ &= \mathrm{P}\left[U < \frac{1}{\delta}\left(0.54|\log p| - 1\right)\right] \end{aligned}$$

with $U \sim \mathcal{N}(0,1)$. Standard values of $p \lesssim 10^{-5}$ and $\delta = 10\%$ give $\left(0.54|\log p| - 1\right)\delta^{-1} \approx 56.77$ and so one can conclude that the optimal Adaptive Splitting will *always* (*i.e.* with great probability) generate less samples than the optimal Splitting with fixed number of samples N. This justifies the fact that in the sequel we will focus on adaptive Splitting even if the original industrial problem is often defined with a fixed computational budget.

1.3.3 Conditional sampling

In practice the adaptive splitting uses Markov chains to simulate according to the conditional distributions. At least this will increase the variance of the estimator. Very recently this issue has been tackled by Bréhier et al. [2015a] who show that the estimator remains unbiased in the dynamic case $\mathbf{X} = (\mathbf{X}_t)_t$ using the Markov property of the trajectory and successive appropriate filtrations; and by Cérou and Guyader [2016] who proved the same Central Limit Theorem as their previous result in the ideal case [see Cérou et al., 2012].

In this section we present commonly used tools to perform these simulations for both the static $\mathbf{X} \in \mathbb{X} \subset \mathbb{R}^d$ and the dynamic case $(\mathbf{X}_t)_t \in \mathbb{X} \subset (\mathbb{R}^d)^{\mathbb{R}}$. While a lot of ongoing research is devoted to the improvement of such methods and more advanced algorithms are now available, this thesis focuses on finding an optimal (in some sense specified throughout the manuscript) estimator given these conditional simulations are possible. In a sense, we focus on optimising the estimators while these researches strive to make them more robust. These two approaches are complementary and this distinction justifies the fact that the skilled reader may find that this section lacks depth.

Metropolis-Hastings method When $\mathbb{X} \subset \mathbb{R}^d$ for some $d \geq 1$, one looks indeed for a simulator of the truncated distributions $\mu^X(\cdot \mid g(\mathbf{X}) > y)$ for all y in the support of μ^Y. Assuming that μ^X has a density π with respect to the Lebesgue measure, it means that one wants to simulate $\mathbf{X} \sim \mathbb{1}_{g(\mathbf{x})>y}\pi(\mathbf{x})/\,\mathrm{P}\left[g(\mathbf{X}) > y\right]$. A general idea for simulating from a given distribution known up to a given constant is to use the convergence property of a Markov chain to its unique invariant measure. This strategy is referred to as the Metropolis-Hastings algorithm in the literature, from the pioneering work of Metropolis and Ulam [1949] further extended by Hastings [1970]. This presentation follows [Delmas and Jourdain, 2006].

The idea of this algorithm is to select an irreducible transition kernel and to modify its

dynamic to manipulate *indeed* a transition kernel with stationary distribution the one we look for. More precisely let f be a target distribution density and Q be transition kernel probability density such that $\forall(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{X}^2$, $Q(\mathbf{x}_1, \mathbf{x}_2) = 0 \Leftrightarrow Q(\mathbf{x}_2, \mathbf{x}_1) = 0$ we build a function $\rho : \mathbb{X}^2 \to (0, 1]$ such that f is reversible for the *modified* kernel ρQ:

$$\forall(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{X}^2,\ f(\mathbf{x}_1)\rho(\mathbf{x}_1, \mathbf{x}_2)Q(\mathbf{x}_1, \mathbf{x}_2) = f(\mathbf{x}_2)\rho(\mathbf{x}_2, \mathbf{x}_1)Q(\mathbf{x}_2, \mathbf{x}_1). \tag{1.22}$$

The function ρ is considered as a probability of acceptance and should verify the equation:

$$\rho(\mathbf{x}_1, \mathbf{x}_2) = \gamma\left(\frac{f(\mathbf{x}_2)Q(\mathbf{x}_1, \mathbf{x}_2)}{f(\mathbf{x}_1)Q(\mathbf{x}_2, \mathbf{x}_1)}\right),\ \forall(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{X}^2 \mid Q(\mathbf{x}_1, \mathbf{x}_2) > 0$$

with $\gamma : \mathbb{R}_+^* \to (0, 1]$ verifying $\gamma(u) = u\gamma(1/u)$. The case $\gamma(u) = \min(1, u)$ corresponds to the Metropolis-Hastings algorithm while $\gamma(u) = u/(1 + u)$ stands for the Boltzmann one for instance.

Then, from a given $\mathbf{X}_0$ taking values in the support of f, a sequence $\mathbf{X}_1, \cdots, \mathbf{X}_n$ is built as follows (see also Algorithm 16):

1. generate $\mathbf{X}_{n+1} \sim Q(\mathbf{X}_n, \cdot)$;
2. sample $U \sim \mathcal{U}[0, 1]$ independently;
3. if $U > \rho(\mathbf{X}_n, \mathbf{X}_{n+1})$, $\mathbf{X}_{n+1} \leftarrow \mathbf{X}_n$;
4. $n \leftarrow n + 1$.

Hence $(\mathbf{X}_n)_n$ is a Markov chain with transition kernel defined by:

$$P(\mathbf{x}_1, \mathbf{x}_2) = \rho(\mathbf{x}_1, \mathbf{x}_2)Q(\mathbf{x}_1, \mathbf{x}_2)$$

for $\mathbf{x}_2 \neq \mathbf{x}_1$ and:

$$P(\mathbf{x}_1, \mathbf{x}_1) = 1 - \int_{\mathbb{X}\setminus\{\mathbf{x}_1\}} P(\mathbf{x}_1, \mathbf{x}_2)\mathrm{d}\mathbf{x}_2$$

for all $(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{X}^2$. Eq. (1.22) implies that $(\mathbf{X}_n)_n$ is a reversible irreducible Markov chain for f since Q is supposed to be irreducible. So its invariant distribution is f. If furthermore $(\mathbf{X}_n)_n$ is aperiodic, it converges in law to f.

In our practical implementations $f(\mathbf{x}) \propto \mathbb{1}_{g(\mathbf{x})>y}\pi(\mathbf{x})$ for some y in the support of μ^Y and Q is always chosen to be a symmetric kernel such that ρ reduces to:

$$\rho(\mathbf{x}_1, \mathbf{x}_2) = \min\left(1, \mathbb{1}_{g(\mathbf{x}_2)>y}\frac{\pi(\mathbf{x}_2)}{\pi(\mathbf{x}_1)}\right) = \mathbb{1}_{g(\mathbf{x}_2)>y}\min\left(1, \frac{\pi(\mathbf{x}_2)}{\pi(\mathbf{x}_1)}\right). \tag{1.23}$$

This latter equality shows that the Metropolis-Hastings algorithm can be computed in two steps: 1) the drawing of a sample which has indeed π as target distribution, 2) the acceptance-rejection scheme to insure that it is in the right domain. Eventually,

if μ^X is a distribution for which a ready-made transition kernel K is available, then the Metropolis-Hastings algorithm can be simplified:

1. generate $\mathbf{X}_{n+1} \sim K(\mathbf{X}_n, \cdot)$;
2. if $g(\mathbf{X}_{n+1}) < y$, $\mathbf{X}_{n+1} \leftarrow \mathbf{X}_n$;
3. $n \leftarrow n + 1$.

For instance if π is the standard multivariate normal distribution, then a possible choice for K is [Guyader et al., 2011, Cérou et al., 2012]:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\frac{1+\sigma^2}{2\pi\sigma^2}} \exp\left(-\frac{1+\sigma^2}{2\sigma^2}\left(\mathbf{x}_2 - \frac{\mathbf{x}_1}{\sqrt{1+\sigma^2}}\right)^2\right) \tag{1.24}$$

which practically speaking means that:

$$\mathbf{X}_{n+1} = \frac{\mathbf{X}_n + \sigma\mathbf{U}}{\sqrt{1+\sigma^2}}$$

with $\mathbf{U}$ an independent standard Gaussian vector. The parameter $\sigma > 0$ has to be tuned. It is often initialised at 0.3 and updated on-the-go to get an acceptance ratio close to 30% (lowered or augmented by a factor of 10% iterations after iterations) [Guyader et al., 2011]. In a more general setting, usual choices for Q are based on $\mathbf{U}$ a centred Uniform or Gaussian random variable and the proposed transition:

$$\mathbf{X}_{n+1} = \mathbf{X}_n + \sigma\mathbf{U}$$

accepted with probability $\rho(\mathbf{X}_n, \mathbf{X}_{n+1})$ with ρ given by Eq. (1.23).

Gibbs sampler The Gibbs sampler also works with finite dimensional vectors and is specifically designed for high dimensional input spaces $\mathbb{X}$. Indeed it makes an iterative sampling of the one-dimensional conditional distributions where all but one coordinates are fixed [see for example Ross, 2013].

Algorithm 4 A systematic Gibbs sampler

Require: $\mathbf{X} \sim f$ the target density
  draw $X_1^* \sim f(x_1 \mid X_2, \cdots, X_d)$
  for $i \in [\![2, d-1]\!]$ **do**
    draw $X_i^* \sim f(x_i \mid X_1^*, \cdots, X_{i-1}^*, X_{i+1}, \cdots, X_d)$
  end for
  draw $X_d^* \sim f(x_d \mid X_1^*, \cdots, X_{d-1}^*)$
return $\mathbf{X}^* \sim f$

We consider $\mathbf{X} = (X_1, \cdots, X_d)$ a random vector with distribution f. In a systematic Gibbs sampler (see Algorithm 4), the coordinates of $\mathbf{X}$ are resampled iteratively according to their index. In a random Gibbs sampler, each coordinate is first chosen at random with the draw of a uniform random variable over the discrete set $\{1, \cdots, d\}$. Another variant, namely the Metropolis-within-Gibbs sampler, makes use of a Metropolis-Hastings transition kernel to generate according to the conditional distributions (lines 1, 3 and 5). This has been shown to improve upon the standard Metropolis-Hastings algorithm when the dimension of the input space is *high* [Au and Beck, 2001].

Dynamic case Recall that in the dynamic case, $\mathbb{X} \subset (\mathbb{R}^d)^{\mathbb{R}}$, the goal is to estimate the probability that a trajectory – a random process often the solution of a stochastic differential equation – enters a given set B before another set A. These sets are metastable and the probability to escape from one to another is very low. In this context the function g which was given in the static case is now somehow an artefact used to quantify the *proximity* of one trajectory to the target set. This function is referred to as the *reaction coordinate* or the *importance function*. The effectiveness of the splitting algorithm may depend strongly on a proper choice for g [Glasserman et al., 1998, Bréhier, 2015].

On the other hand, the conditional simulation of the trajectory is almost straightforward for Markov Process: from a given trajectory $\mathbf{X} = (\mathbf{X}_t)_t$ for which $g(\mathbf{X}) > y$ for some y in the support of μ^Y one wants to sample above, one only has to replicate the trajectory until the first time t_y it reaches $\{\mathbf{X} \in \mathbb{X} \mid g(\mathbf{X}) > y\}$ and then to simulate the original dynamic from this entrance state $\mathbf{X}_{t_y}$ (the stochastic differential equation for instance). The possible side effects of time discretisation have been taken into account by Bréhier et al. [2015a] and this algorithm remains unbiased.

Finally, note that all these algorithms for simulating conditional distributions do not output *iid.* samples. Usually the first samples are discarded to reach the stationary distribution and this is referred to as the *burn-in*. The correlation between samples can then be reduced by selecting only a fraction of the chain: the *thinning* is the ratio of selected samples, for instance one sample each b steps. Note that when the initial sample already follows the target distribution, the *burn-in* indeed only serves independence purpose.

1.4 Nested sampling

Nested sampling was introduced in the Bayesian framework by Skilling [2006] as a method for "estimating directly how the likelihood function relates to prior mass". Formally, it builds an approximation for the evidence:

$$Z = \int_\Theta L(\theta)\pi(\theta)\mathrm{d}\theta,$$

where π is the prior distribution, L the likelihood, and $\Theta \subset \mathbb{R}^d$. It is somehow a quadrature formula but in the $[0,1]$ interval rather than in the original multidimensional space Θ:

$$Z = \int_0^1 Q(P)\mathrm{d}P,$$

where Q is the quantile function, which is the generalised inverse of:

$$P(\lambda) = \int_{L(\theta)>\lambda} \pi(\theta)\mathrm{d}\theta. \tag{1.25}$$

Hence the name *nested sampling* because the initial input space is divided into nested subsets $\{\theta \in \Theta \mid L(\theta) > \lambda\}$. Indeed this latter quantity (Eq. 1.25) is exactly the one estimated in rare event simulation for a given λ.

The recent Bayesian Updating with Structural reliability (BUS) method [Straub and Papaioannou, 2014, Straub et al., 2016] exploits this link and proposes to plug several reliability tools into this integral (FORM/SORM, Line Sampling and Subset Simulation for instance). The Generalised Adaptive Multilevel Splitting method of Bréhier et al. [2015a] focuses on the branching of the adaptive multilevel splitting while the practical implementation suggested by Skilling [2006] is indeed the Last Particle Algorithm. These links and further results will be presented in Chapter 4.

For now, we just come up with the formulation of Skilling [2006]: let $(L(\mathbf{X}_i))_i$ be the sequence of the successive minima of the Last Particle Algorithm with N_0 particles, then form the following estimator:

$$\tilde{Z} = \sum_{i=1}^{T} L(\mathbf{X}_i)\left(e^{\frac{i-1}{N_0}} - e^{\frac{i}{N_0}}\right) \tag{1.26}$$

with T a given stopping *time* which has to be tuned or selected according to some given criterion. The convergence of the approximation error toward a Gaussian distribution has been proven [Chopin, 2002] assuming that Q is twice continuously differentiable with its two first derivatives bounded over $[\varepsilon, 1]$ for some $\varepsilon > 0$. Yet the termination rule of the nested sampling algorithm is an open issue and Chapter 4 will bring some new insight onto it using Multilevel Monte Carlo (MLMC) methods. We will also propose corrected weights for sum (1.26) and give other theoretical results on the nested sampling estimator.

1.5 Efficiency of the estimators

Two different notions are usually used to quantify the *quality* of an estimator in the rare event context: 1) its robustness, *i.e.* how it behaves when the probability becomes smaller; and 2) its reliability, *i.e.* which confidence intervals can be built and how *reliable* they are. Indeed, confidence intervals often require to estimate not only the sought probability but also a variance, and this may result in very bad confidence intervals. These notions

will be presented here but the reader interested in more details and examples is referred to [Rubino et al., 2009, Section 4] and [L'Ecuyer et al., 2010] for even more advanced concepts.

Remember that the main problem of this thesis is the estimation of the quantity (1.1), it is: $p = \mathrm{P}\left[g(\mathbf{X}) > q\right]$ for a given q. Assume that we have defined an estimator $\widehat{p}$ of p for any q, one considers the relative moment of order k of the estimator $\widehat{p}$:

$$m_k(q) = \frac{\mathrm{E}\left[\widehat{p}^k\right]}{p^k}. \tag{1.27}$$

The estimator $\widehat{p}$ is said to have a Bounded Relative Moment of order k (BRM_k) if:

$$\limsup_{q\to\infty} m_k(q) < \infty. \tag{1.28}$$

Note that by usual properties of the moments of a random variables, one has $\forall k < k'$, $BRM_{k'} \Rightarrow BRM_k$. Especially $\widehat{p}$ has a Bounded Relative Error (BRE) if its coefficient of variation remains finite when p goes to 0:

$$\limsup_{p\to 0} \frac{\sigma}{p} < \infty.$$

In Section 1.1.2 we have derived confidence intervals for the crude Monte Carlo estimator. Precisely their width are directly proportional to the coefficient of variation of the estimator. Hence the BRE means that the width of the interval decreases at least as fast as the quantity p itself.

This property is quite stringent and it can happen that no estimator is found satisfying it in some applications. In this context the weaker Logarithmic Efficiency (LE) is used. An estimator will have the LE property of order $k \geq 1$ if:

$$\frac{\log \mathrm{E}\left[\widehat{p}^k\right]}{k \log p} \xrightarrow[p\to 0]{} 1. \tag{1.29}$$

This means that the estimator goes to 0 as fast as the sought quantity p. This is the best possible rate for an unbiased estimator since one has: $\mathrm{E}\left[\widehat{p}^k\right] - p^k \geq 0$ from Jensen's inequality.

As an illustration, the optimal adaptive multilevel splitting algorithm defined in Section 1.3.2, it is the Last Particle Algorithm, does not have a bounded relative error:

$$\delta^2_{\mathrm{LPA}} = p^{-1/N_0} - 1 \xrightarrow[p\to 0]{} \infty.$$

However it asymptotically achieves the logarithmic efficiency for all $k \geq 1$:

$$\frac{\log \mathrm{E}\left[\widehat{p}^k_{\mathrm{LPA}}\right]}{k \log p} = \frac{N_0}{k}\left[1 - \left(1 - \frac{1}{N_0}\right)^k\right] = 1 - \frac{k-1}{2N_0} + o\left(\frac{1}{N_0}\right).$$

Another interesting attribute of the Last Particle Algorithm is that the estimation of the variance is much less sensitive than in a crude Monte Carlo algorithm thanks to the Poisson distribution of the random number of iterations M [Huber and Schott, 2011]. While crude Monte Carlo requires the estimator of the expectation to build an estimator of the variance, in the Last Particle Algorithm one has straight away an estimator of the relative variance:

$$\delta^2_{\mathrm{LPA}} = \frac{-\log p}{N_0} + o\left(\frac{1}{N_0}\right) = \mathrm{E}\left[\frac{M}{N_0^2}\right] + o\left(\frac{1}{N_0}\right).$$

Hence M/N_0^2 is a biased estimator of δ^2_{LPA} whose variance $\mathrm{var}\,[M/N_0^2] = -\log p/N_0^3$ is not more sensitive to the extreme event that the one of $\widehat{p}_{\mathrm{LPA}}$.

Chapter 2

Rare event simulation and surrogate models

We have defined in Chapter 1 advanced statistics to estimate a probability of exceeding a threshold (see Eq. 1.1). However it can happen that the computational budget is so small that there are still too many calls to the computer code using these methods. As a matter of fact in some industrial settings, and especially the one we are interested in, the goal is to estimate $p \lesssim 10^{-5}$ with $N \approx 100$ to 1000 calls to the code.

In this chapter we restrict ourselves to the case where $\mathbb{X} = \mathbb{R}^d$ or $\mathbb{X} \subset \mathbb{R}^d$ is a hypercube for some $d \geq 1$. While this assumption may not be necessary for defining the mathematical tools below, it makes it simpler and it corresponds indeed to our *real* industrial setting: $\mathbf{x}$ is a set of parameters of a computer experiment $g : \mathbf{x} \in \mathbb{X} \mapsto y \in \mathbb{R}$. In this context, g is called the *limit-state function* defining the failure domain $F = \{\mathbf{x} \in \mathbb{X} \mid g(\mathbf{x}) > q\}$, for some $q \in \mathbb{R}$. The random parameters are modelled with $\mathbf{X}$ a random vector defined on an underlying probability space $(\Omega, \mathcal{F}, \mathrm{P})$ with induced measure μ^X over $\mathbb{X}$ and g is assumed to be measurable. Like in the previous chapter, the probability of failure is hence defined by $p = \mu^X(F)$.

The problem of approximating a computer code g seen as a black-box from a set of its input-output couples $(\mathbf{x}_i, g(\mathbf{x}_i))_{i=1}^N$, $N \geq 1$ is rather usual and the reader interested in an overview of this topic is referred to books by Santner et al. [2003], Hastie et al. [2005], Fang et al. [2005], Rasmussen and Williams [2006], Forrester et al. [2008] or recent PhD thesis by Chevalier [2013] and Le Gratiet [2013]. More dedicated results on uncertainty quantification can be found in [Ghanem et al., 2017].

In the rare event setting, it means that one has to rely on cheap-to-compute approximations, either of the boundary, which is equal to $\{\mathbf{x} \in \mathbb{X} \mid g(\mathbf{x}) = q\}$ provided g is continuous (construction of a classifier) or of g over the whole domain $\mathbb{X}$. Then the computational budget can be spent *enriching* these *metamodels* and final computations be performed only with them. The use of such techniques requires to answer mainly three questions: 1) what kind of metamodels should be used? 2) how to make the metamodel $\widetilde{g}$ *closer* to the real model g or to the boundary? and 3) how to use the metamodel? Some surrogate models do not give any precise information on the *closeness* of $\widetilde{g}$ to g or to the boundary while others, like Kriging [Krige, 1951], define pointwise distribution of the error. Hence enrichment strategies will also depend on the metamodel used.

We review here some of the most usual surrogate models used for rare event estimation

following the three questions above mentioned. Another general introduction can be found in [Sudret, 2012]. Reader is referred to the proceedings of the most recent international conferences [Deodatis et al., 2014, Haukaas, 2015] for last updates and improvements of practical algorithms.

2.1 Usual surrogate models

As specified in the introduction, a metamodel is then an analytical function belonging to a specific class of functions (its *type*) characterised by a set of parameters tuned using the available data. It is expected to be fast to compute and so to be used in the statistical methods seen in Chapter 1.

2.1.1 First/Second order reliability method

The First-Order Reliability method (FORM), defined for standard Gaussian input spaces: $\mathbb{X} = \mathbb{R}^d$ and $\mathbf{X} \sim \mathcal{N}(0, \mathbf{I}_d)$ with $\mathbf{I}_d$ the identity matrix in dimension d, is based on a linear approximation of the limit-state function around the so-called Most Probable Failure Point (MPFP). Indeed this method assumes that the origin is not in the failure domain and so the MPFP is the failure point $\mathbf{x}^*$ that is the closest to the origin:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{X}}{\operatorname{argmin}} \, ||\mathbf{x}||_2 \text{ s.t. } g(\mathbf{x}) > q \tag{2.1}$$

with $|| \cdot ||_2$ a Euclidean norm. In the original FORM method, this point is supposed to be unique, *i.e.* that the problem (2.1) has a unique solution. Then, provided g is smooth enough, the Taylor expansion at $\mathbf{x}^*$ writes:

$$g(\mathbf{x}) = g(\mathbf{x}^*) + \nabla g(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) + o\left(||\mathbf{x} - \mathbf{x}^*||_2\right) \tag{2.2}$$

such that if one only retains the first order approximation for the probability of failure, one finds:

$$\begin{aligned} p &\approx \mathrm{P}\left[g(\mathbf{x}^*) + \nabla g(\mathbf{x}^*)^\top (\mathbf{X} - \mathbf{x}^*) > q\right] \\ &\approx \mathrm{P}\left[\nabla g(\mathbf{x}^*)^\top \mathbf{X} > g(\mathbf{x}^*)^\top \mathbf{x}^*\right] \\ &\approx \mathrm{P}\left[\sum_{i=1}^{d} \frac{\nabla g(\mathbf{x}^*)_i}{||\nabla g(\mathbf{x}^*)||_2} X_i > \frac{\nabla g(\mathbf{x}^*)^\top \mathbf{x}^*}{||\nabla g(\mathbf{x}^*)||_2}\right]. \end{aligned}$$

Now recall that the problem is supposed to be defined in the standard Gaussian space, and so the left-hand side is indeed a sum of weighted *iid.* standard Gaussian random variables. Hence it is a Gaussian random variable with mean 0 and variance $\sum_{i=1}^{d} \nabla g(\mathbf{x}^*)_i^2 / ||\nabla g(\mathbf{x}^*)||_2^2 = 1$. Furthermore, the gradient $\nabla g(\mathbf{x}^*)$ and $\mathbf{x}^*$ are parallel.

Eventually the probability rewrites:

$$p_{\text{FORM}} = 1 - \Phi\left(\beta_{\text{HL}}\right) \tag{2.3}$$

with Φ the *cdf* of a standard Gaussian random variable and:

$$\beta_{\text{HL}} = \frac{\nabla g(\mathbf{x}^*)^\top \mathbf{x}^*}{||\nabla g(\mathbf{x}^*)||_2} = ||\mathbf{x}^*||_2$$

the so-called the Hasofer-Lind reliability index. Practically speaking, a FORM algorithm requires to solve a quadratic optimisation problem under non-linear constraint. We refer the reader to the book by Ditlevsen and Madsen [1996] or Zhang and Der Kiureghian [1995] for standard algorithms to solve this problem.

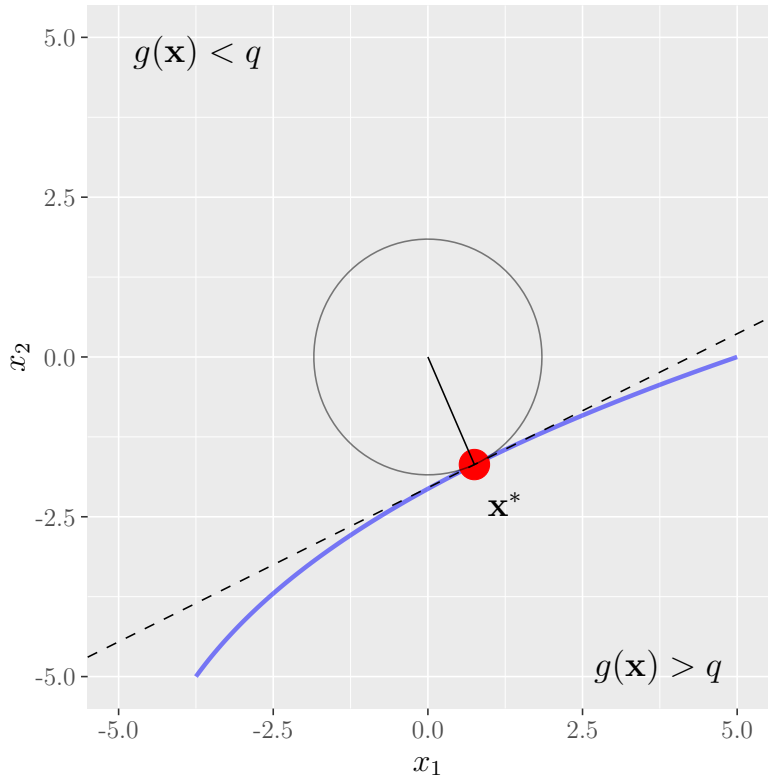


Figure 2.1: First-Order Reliability Method (FORM). The limit-state function (blue line) is approximated with a hyperplane at the Most Probable Failure Point $\mathbf{x}^*$. $\mathbf{X} = (X_1, X_2) \sim \mathcal{N}(0, \mathbf{I}_2)$.

Since the first-order approximation may be too rough, a higher order expansion of the limit-state function has been considered, which leads to the Second-Order Reliability Method (SORM) [Breitung, 1984]. It just goes one step ahead in the Taylor expansion of the limit-state function around the MPFP:

$$g(\mathbf{x}) = g(\mathbf{x}^*) + \nabla g(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \nabla^2 g(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) + o\left(||\mathbf{x} - \mathbf{x}^*||_2^2\right) \tag{2.4}$$

with $\nabla^2 g(\mathbf{x}^*)$ the Hessian matrix of g at $\mathbf{x}^*$. Using again only the first orders instead of the function for the probability estimation:

$$p_{\text{SORM}} = \mathrm{P}\left[\nabla g(\mathbf{x}^*)^\top(\mathbf{X}-\mathbf{x}^*) + \frac{1}{2}(\mathbf{X}-\mathbf{x}^*)^\top \nabla^2 g(\mathbf{x}^*)(\mathbf{X}-\mathbf{x}^*) > 0\right]$$

Breitung [1984] showed the following approximation in the limit case $\beta_{\text{HL}} \to \infty$:

$$p_{\text{SORM}} \underset{\beta_{\text{HL}}\to\infty}{\sim} \Phi(-\beta_{\text{HL}}) \prod_{i=1}^{d-1} \frac{1}{\sqrt{1+\beta_{\text{HL}}\kappa_i}} \tag{2.5}$$

with $(\kappa_i)_{i=1}^{d-1}$ the principal curvatures of the limit-state surface, *i.e.* the eigenvalues of the Hessian matrix [Koyluoglu and Nielsen, 1994, Ditlevsen and Madsen, 1996]. This approximation is used in practice when $\beta_{\text{HL}} \gtrsim 3$.

The limitations of these methods are twofold: firstly it requires that the failure domain be a connected space; secondly it makes assumptions on the shape of the limit-state function without any possible control onto it. To circumvent the first limitation, some multi-points FORM/SORM methods have been proposed [Der Kiureghian and Dakessian, 1998] but the problem remains the same in essence. Ultimately the search for the MPFP is not trivial and an other source of uncertainty.

2.1.2 Support-Vector Machine

Support Vector Machine (SVM) finds its root in the field of statistical learning applied to image recognition. Originally developed for classification, it is now a common tool in machine learning for regression. It has recently been used for rare event simulation and reliability analysis by several authors [Rocco and Moreno, 2002, Most, 2007, Xiukai et al., 2009, Basudhar and Missoum, 2010, Bourinet et al., 2011, Bourinet, 2016]. In this Section we present the main concepts of SVM for classification focusing on the reliability setting, *i.e.* classification with two labels, namely "failure" and "safety" modes. Reader is referred to the tutorial paper of Smola and Schölkopf [2004] for a general description of Support Vector Machine in regression.

Main concept The main idea of Support Vector Machine is to build a linear classifier in the input space $\mathbb{X}$ according to some labels attached to the points in $\mathbb{X}$, *i.e.* to build an hyperplane $\mathcal{H}$ separating the input space $\mathbb{X}$ in two parts such that each one corresponds to one label. More precisely to each $\mathbf{x} \in \mathbb{X}$ is associated a deterministic value y in $\{-1, 1\}$ defining its state. In our context, $y = \operatorname{sign}(g(\mathbf{x}) - q)$ and $y = 1$ means that $\mathbf{x}$ is in the failure domain while $y = -1$ means that it is in the safety one.

It lies in the field of supervised learning as it builds this frontier from a given set $(\mathbf{x}_i, y_i)_{i=1}^N \in \mathbb{X} \times \{-1, 1\}$ of known points. Let $\omega \in \mathbb{X}$ and $b \in \mathbb{R}$, the hyperplane is then of

the form:

$$\mathcal{H} = \{\mathbf{x} \in \mathbb{X} \mid \langle \omega, \mathbf{x} \rangle + b = 0\}$$

with $\langle \cdot, \cdot \rangle$ the natural inner product in $\mathbb{X}$. On the other hand the constraint on the right classification of the points means that on each side of the hyperplane, all the samples have the same label, *i.e.* the same sign. Even if that implies to switch the sign of the labels, this can be written:

$$\forall i \in [\![1, N]\!], \text{ sign}(\langle \mathbf{x}_i, \omega \rangle + b) = y_i = \pm 1.$$

The constraint on the right classification for each point can then be rewritten:

$$\forall i \in [\![1, N]\!], \ (\langle \mathbf{x}_i, \omega \rangle + b) y_i \geq 1.$$

With this setting, the *margin* of the classifier, *i.e.* the minimum distance between the two parallel hyperplanes around $\mathcal{H}$ such that no sample is in between them, is at least $2/||\omega||_2$. Finally, if one wants to maximize this margin one should minimize $||\omega||_2$ and the SVM optimisation problem rewrites:

$$\min_{\omega, b} \frac{1}{2} ||\omega||_2^2 \ s.t. \ \forall i \in [\![1, N]\!], \ y_i(\langle \omega, \mathbf{x}_i \rangle + b) \geq 1. \tag{2.6}$$

This optimisation problem can be solved with standard calculation. Let $(\lambda_i)_{i=1}^N$ be the Lagrange multipliers and $\mathcal{L}(\omega, b)$ the corresponding dual function, one has:

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial \omega} = \omega + \sum\limits_{i=1}^{N} \lambda_i y_i \mathbf{x}_i = 0 \\ \dfrac{\partial \mathcal{L}}{\partial b} = \sum\limits_{i=1}^{N} \lambda_i y_i = 0 \end{cases} .$$

Furthermore, the Karush-Kuhn-Tucker (KKT) conditions [see for example Boyd and Vandenberghe, 2004] writes:

$$\forall i \in [\![1, N]\!], \ \lambda_i \left(y_i(\langle \mathbf{x}_i, \omega \rangle + b) - 1 \right) = 0.$$

On the one hand ω writes as a linear combination of the data points $(\mathbf{x}_i)_{i=1}^N$; on the other hand the KKT conditions mean that only the points right on the borders (active constraint) will have non-zero multipliers. All together, this means that the classifier is ultimately defined only by the vectors on the margin. Figure 2.2 shows an example of such a classifier.

This *active* property means also that new samples outside of the margin will not change the classifier. In other words, if one intend to *learn better* the boundary between the failure and the safety region, one has to sample only into the margin. This property will be further developed in Section 2.2.

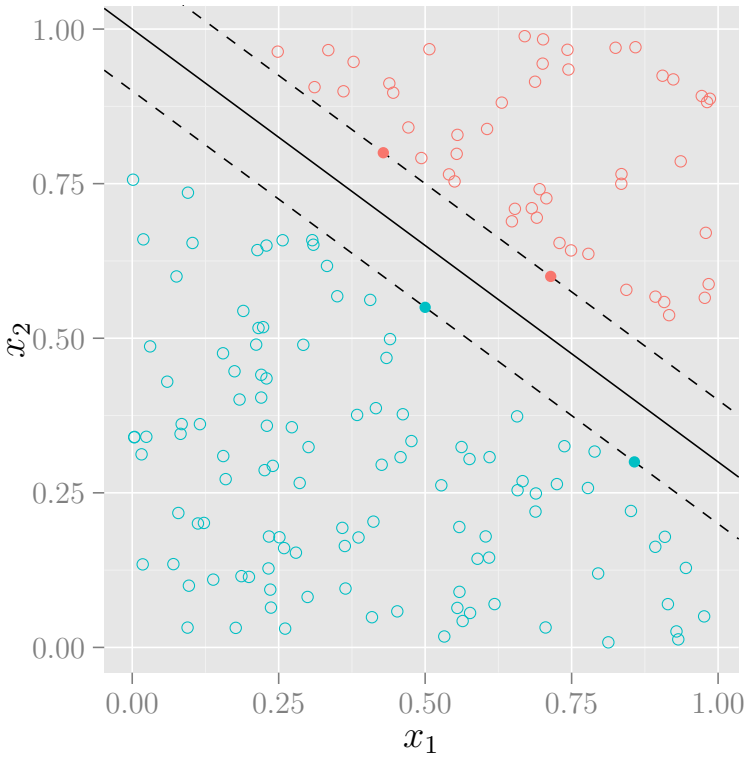


Figure 2.2: Example of a SVM classifier with a linear kernel. Plains dots are the *active* samples

Non-linear classifier The approach and methodology seen above is the very first formulation of what was then called the *Portrait method* [Vapnik, 1963, Vapnik and Chervonenkis, 1964]. However it happens often that the data are not linearly separable, and here comes what is referred to as the *kernel trick*. In few words, it replaces the natural inner product of the space $\mathbb{X}$ by another function called a *kernel*. Without digging too much into the Reproducing Kernel Hilbert Space theory [Aronszajn, 1950] and its link with SVM [Wahba et al., 1999] we give the main idea of such a transformation.

A non-negative kernel $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ is a symmetric continuous function such that for any $(a_1, \cdots, a_n) \in \mathbb{R}^n$ and distinct $(\mathbf{x}_1, \cdots, \mathbf{x}_n) \in \mathbb{X}^n$, $n \geq 1$, one has:

$$\sum_{i,j=1}^{n} a_i a_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \tag{2.7}$$

with equality *iff.* $\forall i \in [\![1, N]\!]$, $a_i = 0$. The Mercer theorem lets clarify the link between the original *linear* SVM, *i.e.* the SVM with a linear kernel (the natural scalar product of $\mathbb{X}$) and this new version. Let T_k be the linear integral operator on $L_2(\mathbb{X})$ the space of square-integrable functions such that:

$$T_k : \phi \in L_2(\mathbb{X}) \mapsto \int_{\mathbb{X}} k(\mathbf{x}, \cdot)\phi(\mathbf{x}) \mathrm{d}\mu(\mathbf{x})$$

with μ a measure on $\mathbb{X}$ such that $\mu(\mathbb{X})$ is finite and $\mathrm{supp}(\mu) = \mathbb{X}$.

Theorem 2.1 (Mercer's theorem [Mercer, 1909, Ferreira and Menegatto, 2009])**.** *If k is a continuous symmetric positive kernel such that $\sup_{x\in\mathbb{X}} k(\mathbf{x},\mathbf{x}) < \infty$, then there is an orthonormal basis $(\phi_i)_i$ of $L_2(\mathbb{X})$ consisting of eigenfunctions of T_k such that the corresponding sequence of eigenvalues $(\lambda_i)_i$ is non-negative. The eigenfunctions corresponding to non-zero eigenvalues are continuous on $\mathbb{X}$ and k has the representation:*

$$\forall(\mathbf{x}_1,\mathbf{x}_2)\in\mathbb{X}^2,\ k(\mathbf{x}_1,\mathbf{x}_2)=\sum_{j=1}^{\infty}\lambda_j\,\phi_j(\mathbf{x}_1)\,\phi_j(\mathbf{x}_2).$$

Theorem 2.1 shows that using a kernel k instead of the natural inner product can be seen as using the natural inner product in another space mapped with the eigenfunctions of T_k. This auxiliary space is referred to as the *feature* space and is practically speaking never precisely characterised. Instead, one chooses directly a kernel amongst a set of known admissible options and speaks about the *kernel trick* because it allows for mapping $\mathbb{X}$ into another space where the problem is linearly separable, but without specifying exactly this space.

As a matter of fact, the kernels implemented in the `R` [R Core Team, 2015] package `e1071` [Meyer et al., 2015] wrapping the reference `libsvm` C++ library by Chang and Lin [2011] are given in Table 2.1.

Kernel	Expression	Parameters
Linear	$k(\mathbf{x}_1,\mathbf{x}_2)=\mathbf{x}_1^\top\mathbf{x}_2$	
Polynomial	$k(\mathbf{x}_1,\mathbf{x}_2)=(\gamma\mathbf{x}_1^\top\mathbf{x}_2+\beta)^n$	$\gamma>0$, $\beta\in\mathbb{R}$, $n>0$
Gaussian	$k(\mathbf{x}_1,\mathbf{x}_2)=e^{-\gamma\lvert\lvert\mathbf{x}_2-\mathbf{x}_1\rvert\rvert_2^2}$	$\gamma>0$
Sigmoid	$k(\mathbf{x}_1,\mathbf{x}_2)=\tanh\left(\gamma\mathbf{x}_1^\top\mathbf{x}_2+\beta\right)$	$\gamma>0$, $\beta\in\mathbb{R}$

Table 2.1: Usual kernels for a SVM classifier.

Figure 2.3 shows an example of a SVM classifier using a Gaussian kernel. On the one hand it is able to create a frontier between samples not linearly separable. On the other hand one can see that the margin is not empty any more (there are samples in between the two dashed lines). Indeed it often happens that there is no feasible solution to Problem (2.6); to circumvent this limitation *slack variables* $(\xi_i)_i\in\mathbb{R}_+$ are introduced to lower the constraints:

$$\forall i\in[\![1,N]\!],\ c_i\left(\langle\omega,\mathbf{x}_i\rangle+b\right))\geq 1\leftarrow c_i\left(\langle\omega,\mathbf{x}_i\rangle+b\right))\geq 1-\xi_i. \tag{2.8}$$

This can even allow for misclassification of some points and these parameters as well as the ones in the kernel have to be tuned. As mentioned by Meyer et al. [2015]:

> The correct choice of kernel parameters is crucial for obtaining good results, which practically means that an extensive search must be conducted on the parameter space before results can be trusted.

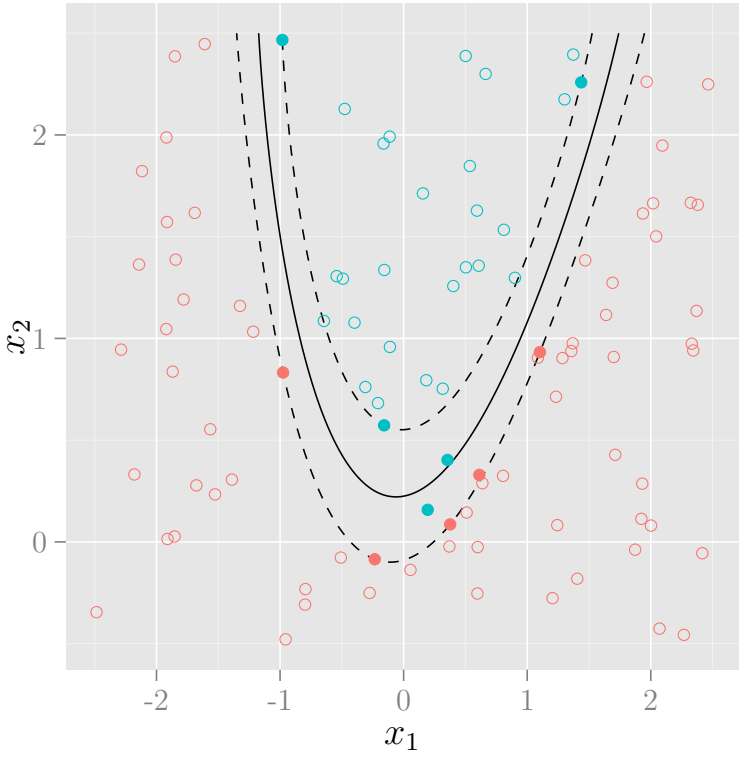


Figure 2.3: Example of a SVM classifier with a Gaussian kernel generated with `R` package `e1071`. Plain dots are the *active* samples.

2.1.3 Polynomial-Chaos expansion

Polynomial Chaos expansion has been introduced by Ghanem and Spanos [1990] for stochastic finite elements methods. It can be seen as a linear projection of g onto a specific orthonormal basis based on the distribution μ^X of the random input. With this specification the function g should be well-approximated in the areas of high probability.

Framework Let us consider that $\mathbf{X} \sim \mu^X$ the random input has independent coordinates with densities $(f_{X_i})_{i=1}^d$ (for the sake of simplicity of this presentation; this is indeed not a requirement, see the generalised Polynomial Chaos expansion framework [Ernst et al., 2012]). Recall that we have assumed that either $\mathbb{X} = \mathbb{R}^d$ or $\mathbb{X} \subset \mathbb{R}^d$ is an hypercube, let us write: $\mathbb{X} = \mathbb{X}_1 \times \cdots \times \mathbb{X}_d$. Then for each $i \in [\![1, d]\!]$, a family of polynomials $(P_j^{(i)})_j$, orthonormal with respect to the inner product defined by f_{X_i} is built [Soize and Ghanem, 2004]:

$$\langle P_j^{(i)}, P_k^{(i)} \rangle = \int_{\mathbb{X}_i} P_j^{(i)}(x) P_k^{(i)}(x) f_{X_i}(x) \mathrm{d}x = \delta_{jk}$$

with δ_{jk} the Kronecker symbol equals to 1 if $j = k$ and 0 otherwise. The multidimensional polynomials are then built by tensorisation: with each $d-$tuple $\alpha = (\alpha_1, \cdots, \alpha_d) \in \mathbb{N}^d$ one

builds the polynomial $\Psi_\alpha(\mathbf{x})$, $\mathbf{x} \in \mathbb{X}$:

$$\Psi_\alpha(\mathbf{x}) = \prod_{i=1}^{d} P_{\alpha_i}^{(i)}(x_i)$$

so that they inherit the orthonormal property:

$$\mathrm{E}\left[\Psi_\alpha(\mathbf{X})\Psi_\beta(\mathbf{X})\right] = \delta_{\alpha\beta}$$

thanks to the independence of the coordinates of $\mathbf{X}$. Then the linear projection of g onto this orthonormal basis is:

$$g(\mathbf{x}) = \sum_{\alpha \in \mathbb{N}^d} a_\alpha \Psi_\alpha(\mathbf{x}). \tag{2.9}$$

Table 2.2 summarises usual polynomials used according to the distribution of $\mathbf{X}$ (namely its marginals).

Distribution	*pdf*	Orthogonal basis	Orthonormal basis
Uniform	$\mathbb{1}_{(-1,1)}(x)/2$	Legendre $P_k(x)$	$P_k(x)(2k+1)^{-0.5}$
Gaussian	$(2\pi)^{-0.5}e^{-x^2/2}$	Hermite $H_k(x)$	$H_k(x)/\sqrt{k!}$
Gamma	$x^a e^{-x}\mathbb{1}_{x>0}$	Laguerre $L_k^a(x)$	$L_k^a(x)/\sqrt{\frac{\Gamma(k+a+1)}{k!}}$

Table 2.2: Usual polynomial families for Polynomial Chaos expansion.

Metamodelisation From Eq. (2.9) the Polynomial Chaos expansion appears as a specific case of a linear projection over a basis selected according to the distribution of the input random variable $\mathbf{X}$. In this context the decomposition can be truncated to serve as a cheap surrogate model for g. As a rule of thumb the truncation is often limited to all polynomials with a total order $|\alpha| = \sum_{i=1}^{d} \alpha_i \leq r$ with $r = 2$ or 3:

$$\widetilde{g}(\mathbf{x}) = \sum_{\alpha \in \mathcal{A}_r} a_\alpha \Psi_\alpha(\mathbf{x}) \tag{2.10}$$

with $\mathcal{A}_r = \{\alpha \in \mathbb{N}^d \mid |\alpha| \leq r\}$. The total order r can also be chosen adaptively according to a given target precision, the error being estimated with cross-validation [Blatman and Sudret, 2010, 2011].

The last case to handle is the estimation of the coefficients (a_α) in Eq. (2.10). Using the projection property, one has:

$$\forall \alpha \in \mathcal{A}_r,\ a_\alpha = \int_{\mathbb{X}} g(\mathbf{x})\Psi_\alpha(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) \mathrm{d}\mathbf{x}.$$

These integrals can then be evaluated using quadrature rules [Ghiocel and Ghanem, 2002, Le Maître et al., 2002, Keese and Matthies, 2005], stochastic collocation methods [Xiu,

2009] or regression methods [Blatman and Sudret, 2010, 2011]. This latter is essentially a least-square regression: from an experimental design set $(\mathbf{x}_i)_{i=1}^N \in \mathbb{X}^N$ one solves the minimisation problem:

$$\mathbf{a}^* = \underset{\mathbf{a}\in\mathbb{R}^{\operatorname{card}\mathcal{A}_r}}{\operatorname{argmin}} \sum_{i=1}^{N} \left[g(\mathbf{x}_i) - \sum_{\alpha\in\mathcal{A}_r} a_\alpha \Psi_\alpha(\mathbf{x}_i) \right]^2$$

with $\mathbf{a} = (a_{\alpha_1}, \cdots, a_{\alpha_{\operatorname{card}\mathcal{A}_r}})$. A rule of thumb for the size N of the design set is to keep $N \geq 2 - 3 \times \operatorname{card}\mathcal{A}_r$ [Blatman, 2009, Sudret, 2012]. In our setting, typical parameters would be $r = 3$ and $d = 10$, which makes $N \geq 2 - 3 \times 10^3/3! \approx 300 - 500$. This is still quite a lot of calls to the computer code g and furthermore Polynomial Chaos expansion does not lead to easy enrichment strategies (see Section 2.2 below) because it tries to learn the function especially over the most probable regions of the input space. Note however that it has been recently used in conjunction with Kriging [Schöbi et al., 2016] for trend estimation (see Section 2.1.4).

2.1.4 Kriging

Gaussian Process regression has been receiving a lot of attention recently in computer experiment [Sacks et al., 1989, Fang et al., 2005, Rasmussen and Williams, 2006]. It is a special case of Kriging [Krige, 1951, Matheron, 1963, 1969, Chiles and Delfiner, 2009], a statistical approach based on the use of random processes to approximate an unknown function, where the random process is actually assumed to be Gaussian. It has then been used in rare event estimation with main contributions from Bichon et al. [2008], Bect et al. [2012], Dubourg et al. [2011] and Echard et al. [2011]. In this introduction we present Kriging from the geostatistical point of view following Chauvet [2008], *i.e.* without any assumption of Gaussianity for the random process.

Kriging framework

Random process Kriging is based on the idea that the *unknown* computer code g is a realisation of a random process ξ defined over a given probability space $(\Omega_0, \mathcal{F}_0, \mathrm{P}_0)$ and indexed by $\mathbf{x} \in \mathbb{X}$:

$$\xi : (\mathbf{x} \times \omega) \in (\mathbb{X} \times \Omega_0) \mapsto \xi(\mathbf{x}, \omega) \in \mathbb{R}. \tag{2.11}$$

Here, for all $\mathbf{x} \in \mathbb{X}$, $\xi(\mathbf{x})$ is then a real-valued random variable and one implicitly assumes that:

$$\exists\, \omega \in \Omega_0 \mid \forall \mathbf{x} \in \mathbb{X},\ g(\mathbf{x}) = \xi(\mathbf{x}, \omega).$$

We now assume that the random process ξ is stationary with a finite covariance k:

$$\forall \mathbf{x} \in \mathbb{X},\ \mathrm{E}_0\left[\xi(\mathbf{x})\right] = m(\mathbf{x}) = m \tag{2.12}$$

$$\forall (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{X}^2,\ \operatorname{cov}_0\left[\xi(\mathbf{x}_1), \xi(\mathbf{x}_2)\right] = k(\mathbf{x}_1, \mathbf{x}_2) < \infty. \tag{2.13}$$

The stationary hypothesis means that the law of the random process does not depend on the origin of the plane. Consequently the mean is constant (Eq. 2.12) and the covariance $k(\mathbf{x}_1, \mathbf{x}_2)$ only depends on the relative position $\mathbf{x}_1 - \mathbf{x}_2$ of one point to the other (Eq. 2.13).

Kriging Kriging is the idea of approximating a linear functional of ξ with a linear combination of ξ at other given locations $(\mathbf{x}_1, \cdots, \mathbf{x}_N) \in \mathbb{X}^N$, $N \geq 1$. Such linear functional can be for instance the mean:

$$m = \mathrm{E}_0\left[\xi(\mathbf{x})\right], \mathbf{x} \in \mathbb{X},$$

or simply the process ξ at another location $\mathbf{x} \in \mathbb{X}$. Let us write Q this quantity, one is going to consider an estimator $\widehat{Q}$ of Q of the form:

$$\widehat{Q} = \sum_{i=1}^{N} \lambda_i \xi(\mathbf{x}_i)$$

with $(\lambda_i)_{i=1}^N \in \mathbb{R}^N$ the so-called kriging weights. Then one enforces the estimator to be unbiased:

$$\mathrm{E}_0\left[\widehat{Q}\right] = \sum_{i=1}^{N} \lambda_i \,\mathrm{E}\left[\xi(\mathbf{x}_i)\right] = m \sum_{i=1}^{N} \lambda_i = \mathrm{E}_0\left[Q\right].$$

If m is known, one can consider from the beginning $\xi(\mathbf{x}) \leftarrow \xi(\mathbf{x}) - m$, *i.e.* $m = 0$ and so the constraint rewrites $0 = 0$. This case is often referred to as the *Simple Kriging* (stationary process, null mean and known covariance). On the other hand if m is unknown, then the constraint is active. Especially, if $Q = m$ or the random process at some other location, $\mathrm{E}_0\left[Q\right] = m$ and it becomes:

$$\sum_{i=1}^{N} \lambda_i = 1.$$

This case is referred to as the *Ordinary Kriging* (or kriging of the mean if $Q = m$) and is the framework usually used in the rare event setting. Finally, one looks for the estimator whose error is optimal (minimal variance) and so solves the optimisation problem:

$$\underset{(\lambda_i)_{i=1}^N \in \mathbb{R}^N}{\operatorname{argmin}} \operatorname{var}_0\left[\widehat{Q} - Q\right] \;\; s.t. \;\; \sum_{i=1}^{N} \lambda_i = 1. \tag{2.14}$$

Ordinary Kriging solution The minimisation problem defined in Eq. (2.14) rewrites:

$$\underset{\boldsymbol{\lambda} \in \mathbb{R}^N,\, \mu \in \mathbb{R}}{\operatorname{argmin}} \; \boldsymbol{\lambda}^\top \mathbf{K} \boldsymbol{\lambda} + \operatorname{var}_0\left[Q\right] - 2\mathbf{k}_Q^\top \boldsymbol{\lambda} + 2\mu\left(\mathbf{1}^\top \boldsymbol{\lambda} - 1\right)$$

with:

$\mathbf{K}$ the covariance matrix $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{1 \leq i,j \leq N}$ of the random variables $\xi(\mathbf{x}_1), \cdots, \xi(\mathbf{x}_N)$;

$\mathbf{k}_Q$ the vector $(\mathrm{cov}_0[Q, \xi(\mathbf{x}_1)], \cdots, \mathrm{cov}_0[Q, \xi(\mathbf{x}_N)])^\top$;

$\boldsymbol{\lambda}$ the column vector of the kriging weights;

$\mathbf{1}$ a column vector of dimension N full of 1;

μ the Lagrange multiplier for the constraint.

Then the solution is given by the system:

$$\begin{cases} \mathbf{K}\boldsymbol{\lambda} + \mu\mathbf{1} = \mathbf{k}_Q \\ \mathbf{1}^\top\boldsymbol{\lambda} = 1 \end{cases} \tag{2.15}$$

which rewrites:

$$\begin{pmatrix} \mathbf{K} & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \mu \end{pmatrix} = \begin{pmatrix} \mathbf{k}_Q \\ 1 \end{pmatrix}. \tag{2.16}$$

From Eq. (2.16) one notices that the kriging weights $\boldsymbol{\lambda}$ do not depend on the known data $(\xi(\mathbf{x}_i))_{i=1}^N$ but only on their locations $(\mathbf{x}_i)_{i=1}^N$ through the covariance function. Furthermore, Eqs. (2.15) and (2.16) give:

$$\boldsymbol{\lambda}^\top\mathbf{K}\boldsymbol{\lambda} = \boldsymbol{\lambda}^\top\mathbf{k}_Q - \mu = 2\boldsymbol{\lambda}^\top\mathbf{k}_Q - (\boldsymbol{\lambda}^\top, \mu)\begin{pmatrix} \mathbf{k}_Q \\ 1 \end{pmatrix}$$

$$\mathrm{var}_0\left[Q - \widehat{Q}\right] = \mathrm{var}_0[Q] - (\mathbf{k}_Q^\top, 1)\begin{pmatrix} \mathbf{K} & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{pmatrix}^{-1}\begin{pmatrix} \mathbf{k}_Q \\ 1 \end{pmatrix}.$$

The prediction variance does not depend on $\boldsymbol{\xi} = (\xi(\mathbf{x}_1), \cdots, \xi(\mathbf{x}_N))^\top$ and so it is possible to estimate the precision of an estimation given a dataset $(\mathbf{x}_1, \cdots, \mathbf{x}_N)$ without actually computing the model at these locations. Kriging only makes use of the relative position from one point to another through the covariance (see Eq. 2.17). This specific property will be used in Section 2.2.3 to derive strategies for enriching the design of experiments.

Finally the system (2.15) is solved with:

$$\begin{cases} \boldsymbol{\lambda} = \mathbf{K}^{-1}\mathbf{k}_Q + \mu\mathbf{K}^{-1}\mathbf{1} \\ \mu = \dfrac{1 - \mathbf{1}^\top\mathbf{K}^{-1}\mathbf{k}_Q}{\mathbf{1}^\top\mathbf{K}^{-1}\mathbf{1}} \end{cases}. \tag{2.17}$$

and the quantity $\widehat{Q}$ verifies:

$$\widehat{Q} = \boldsymbol{\lambda}^\top\boldsymbol{\xi} = \mathbf{k}_Q^\top\mathbf{K}^{-1}\boldsymbol{\xi} + \mathbf{1}^\top\mathbf{K}^{-1}\boldsymbol{\xi}\frac{1 - \mathbf{1}^\top\mathbf{K}^{-1}\mathbf{k}_Q}{\mathbf{1}^\top\mathbf{K}^{-1}\mathbf{1}} \tag{2.18}$$

or equivalently:

$$\widehat{Q} = \mathbf{k}_Q^\top\mathbf{K}^{-1}\left(\boldsymbol{\xi} - \mathbf{1}\frac{\mathbf{1}^\top\mathbf{K}^{-1}\boldsymbol{\xi}}{\mathbf{1}^\top\mathbf{K}^{-1}\mathbf{1}}\right) + \frac{\mathbf{1}^\top\mathbf{K}^{-1}\boldsymbol{\xi}}{\mathbf{1}^\top\mathbf{K}^{-1}\mathbf{1}}. \tag{2.19}$$

This formulation simplifies when $Q = m$ because $\mathbf{k}_Q = (0, \cdots, 0)^\top$, which gives:

$$\widehat{m} = \frac{\mathbf{1}^\top \mathbf{K}^{-1} \boldsymbol{\xi}}{\mathbf{1}^\top \mathbf{K}^{-1} \mathbf{1}}; \tag{2.20}$$

$$\widehat{Q} = \mathbf{k}_Q^\top \mathbf{K}^{-1} (\boldsymbol{\xi} - \mathbf{1}\widehat{m}) + \widehat{m}. \tag{2.21}$$

On the other hand, the solution of the Simple Kriging with known mean m can be obtained from Eq. (2.15) by writing $\mu = 0$ and discarding the constraint:

$$\widehat{Q} - m = \mathbf{k}_Q^\top \mathbf{K}^{-1} (\boldsymbol{\xi} - \mathbf{1}m). \tag{2.22}$$

All together, Eqs. (2.21) and (2.22) show that it is equivalent to apply the Kriging framework to the random process with unknown mean or to first estimate the mean and then to perform a simple kriging plugging-in the estimated mean.

Universal Kriging This paragraph addresses the issue of the stationary hypothesis. To this end the Universal Kriging refers to the case where the mean is expressed as a finite sum of basis functions:

$$m(\mathbf{x}) = \sum_{j=1}^{J} \alpha_j f_j(\mathbf{x}) = \mathbf{f}_\mathbf{x}^\top \boldsymbol{\alpha},\ J \geq 1$$

with $(f_j)_{j=1}^J$ the basis functions: $\forall j \in [\![1, J]\!]$, $f_j \in \mathbb{R}^\mathbb{X}$, $\mathbf{f}_\mathbf{x} = (f_1(\mathbf{x}), \cdots, f_J(\mathbf{x}))^\top$ and $(\alpha_j)_{j=1}^J \in \mathbb{R}^J$ unknown coefficients (otherwise the mean would be known and simple kriging apply). The underlying assumption is that the non-stationary part of the process will be *captured* by the mean (then often called the *trend*) and that the remaining random process is thus stationary. In this context the constraint on the bias becomes:

$$\forall \boldsymbol{\alpha} \in \mathbb{R}^J,\ \boldsymbol{\lambda}^\top \mathbf{F}\boldsymbol{\alpha} - \mathbf{f}_\mathbf{x}^\top \boldsymbol{\alpha} = 0,\ \textit{i.e.}\ \mathbf{F}^\top \boldsymbol{\lambda} - \mathbf{f}_\mathbf{x} = 0$$

with $\mathbf{F}$ the matrix gathering the values of the basis functions on the dataset $\mathbf{F}^\top = (\mathbf{f}_{\mathbf{x}_1}, \cdots, \mathbf{f}_{\mathbf{x}_N})$ and $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_J)^\top$. There are finally J equality constraints and so system (2.16) becomes:

$$\begin{pmatrix} \mathbf{K} & \mathbf{F} \\ \mathbf{F}^\top & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\mu} \end{pmatrix} = \begin{pmatrix} \mathbf{k}_Q \\ \mathbf{f}_\mathbf{x} \end{pmatrix} \tag{2.23}$$

with $\boldsymbol{\mu} \in \mathbb{R}^J$ the vector of the Lagrange multipliers. If this system is feasible the same calculations and conclusions as above for the ordinary kriging stand, substituting $\mathbf{F}$ to $\mathbf{1}$:

$$\boldsymbol{\alpha} = (\mathbf{F}^\top \mathbf{K}^{-1} \mathbf{F})^{-1} \mathbf{F}^\top \mathbf{K}^{-1} \boldsymbol{\xi} \tag{2.24}$$

$$\widehat{Q} = \mathbf{k}_Q^\top \mathbf{K}^{-1} (\boldsymbol{\xi} - \mathbf{F}\boldsymbol{\alpha}) + \mathbf{f}_\mathbf{x}^\top \boldsymbol{\alpha} \tag{2.25}$$

$$\mathrm{var}_0\left[\widehat{Q} - Q\right] = \mathrm{var}_0[Q] - (\mathbf{k}_Q^\top, \mathbf{f}_\mathbf{x}^\top) \begin{pmatrix} \mathbf{K} & \mathbf{F} \\ \mathbf{F}^\top & 0 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{k}_Q \\ \mathbf{f}_\mathbf{x} \end{pmatrix}. \tag{2.26}$$

A necessary condition for the feasibility of the system is that the family of basis functions be linearly independent over the data. Usually they are chosen amongst usual basis functions of $L_2(\mathbb{X})$ (*e.g.* polynomial functions) and this condition is always satisfied. Finally, it is important to notice that Kriging interpolates the data: systems (2.16) and (2.23) have a trivial solution if $Q = \xi(\mathbf{x}_i)$ for some $i \in [\![1, N]\!]$, precisely $\lambda_i = 1$ and $\forall j \neq i$, $\lambda_j = 0$.

Gaussian process framework

Gaussian process regression While we have been able to solve the kriging system without further assumptions that the covariance of the random process is finite, in computer experiments it is always assumed that ξ is a Gaussian process. There are mainly three reasons for that:

1. the covariance was supposed to be known. Indeed it is chosen amongst several acceptable kernels like in Section 2.1.2 and some parameters have to be estimated, especially the variance $\sigma_0^2 = k(\mathbf{x}, \mathbf{x}), \forall \mathbf{x} \in \mathbb{X}$. Assuming that ξ is Gaussian lets have the joint density of the $N-$tuple $(\xi(\mathbf{x}_1), \cdots, \xi(\mathbf{x}_N))$ and apply a Bayesian approach to parameter estimation or use Maximum Likelihood Estimation (MLE);

2. while kriging gives an expression of the error variance, it does not give the distribution of the prediction. Assuming that ξ is Gaussian lets build confidence intervals; and

3. Kriging seeks for the minimal variance linear combination of the data. Indeed it is known that the best way (minimal variance) to approximate a given random variable with some others is to use the conditional expectation. While in traditional Geostatistics it has led to the Disjunctive Kriging, the Gaussian hypothesis insures that the Universal Kriging is also the conditional expectation, *i.e.* the best estimator one can produce given the data.

In this context we are going to use further in this thesis the notations:

$\mathcal{F}_N$ the filtration generated by the N random variables $\xi(\mathbf{x}_1), \cdots, \xi(\mathbf{x}_N)$;

$\xi_N(\mathbf{x})$ the conditional expectation $\xi_N(\mathbf{x}) = \mathrm{E}_0\left[\xi(\mathbf{x}) \mid \mathcal{F}_N\right]$ and

$\sigma_N^2(\mathbf{x})$ the conditional variance $\mathrm{var}_0\left[\xi(\mathbf{x}) \mid \mathcal{F}_N\right]$.

With the Gaussian hypothesis, ξ_N corresponds to the kriging predictor given by Eq. (2.25) and σ_N^2 is the kriging error (see Eq. 2.26). In the sequel, quantities evaluated conditionally to the filtration $\mathcal{F}_N$ will be subscripted by N:

$$\begin{aligned}
\mathrm{E}_N\left[\cdot\right] &= \mathrm{E}_0\left[\,\cdot \mid \mathcal{F}_N\right] \\
\mathrm{var}_N\left[\cdot\right] &= \mathrm{var}_0\left[\,\cdot \mid \mathcal{F}_N\right] \\
\mathrm{P}_N\left[\cdot\right] &= \mathrm{P}_0\left[\,\cdot \mid \mathcal{F}_N\right].
\end{aligned}$$

Parameter estimation In the previous parts we have assumed that the covariance was known. Indeed, as for Support-Vector Machine in Section 2.1.2 the covariance is chosen amongst a family of usual kernels whose parameters have to be estimated. Furthermore it is also more common to specify such kernels as correlation instead of covariance kernels:

$$\forall (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{X}^2,\ k(\mathbf{x}_1, \mathbf{x}_2) = \sigma_0^2 r(\mathbf{x}_1, \mathbf{x}_2)$$

with r depending only on a set of parameters $\boldsymbol{\theta} \in \mathbb{R}_+^d$ representing the typical correlation lengths in each dimension and σ_0^2 the stationary variance of the random process. Then the multidimensional correlation kernel r is usually defined by tensorisation of unidimensional kernels [Rasmussen and Williams, 2006]:

$$r(\mathbf{x}_1, \mathbf{x}_2) = \prod_{i=1}^{d} r_i(x_{1,i}\,, x_{2,i}).$$

Let $(\mathbf{x}_i)_{i=1}^N \in \mathbb{X}^N$ with $N \geq 1$, the Gaussian hypothesis lets have the likelihood of the random vector $\boldsymbol{\xi} = (\xi(\mathbf{x}_1), \cdots, \xi(\mathbf{x}_N))$ given the parameters $\boldsymbol{\alpha}$ (the coefficients of the trend), σ_0^2 (the variance of the random process) and $\boldsymbol{\theta}$ (the parameters of the correlation kernel):

$$f(\boldsymbol{\xi} \mid \boldsymbol{\alpha}, \sigma_0^2, \boldsymbol{\theta}) = \frac{1}{(2\pi\sigma_0^2)^{N/2}\sqrt{|\det \mathbf{R}(\boldsymbol{\theta})|}} \exp\left(-\frac{1}{2}\frac{(\boldsymbol{\xi} - \mathbf{F}\boldsymbol{\alpha})^\top \mathbf{R}(\boldsymbol{\theta})^{-1}(\boldsymbol{\xi} - \mathbf{F}\boldsymbol{\alpha})}{\sigma_0^2}\right) \quad (2.27)$$

with $\mathbf{R}(\boldsymbol{\theta})$ the correlation matrix of the data $\mathbf{R}(\boldsymbol{\theta}) = (r(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^N$. Then, maximising first this quantity against $\boldsymbol{\alpha}$ conditionally to σ_0^2 and $\boldsymbol{\theta}$ results in:

$$\boldsymbol{\alpha}(\boldsymbol{\theta}) = (\mathbf{F}^\top \mathbf{R}(\boldsymbol{\theta})^{-1}\mathbf{F})^{-1}\mathbf{F}^\top \mathbf{R}(\boldsymbol{\theta})^{-1}\boldsymbol{\xi} \quad (2.28)$$

which is already the result found in Eq. (2.24) solving the Universal Kriging system (2.23). Injecting this value into Eq. (2.27) and maximising it against σ_0^2 for a given $\boldsymbol{\theta}$ hence produces:

$$\sigma_0^2(\boldsymbol{\theta}) = \frac{1}{N}(\boldsymbol{\xi} - \mathbf{F}\boldsymbol{\alpha}(\boldsymbol{\theta}))^\top \mathbf{R}(\boldsymbol{\theta})^{-1}(\boldsymbol{\xi} - \mathbf{F}\boldsymbol{\alpha}(\boldsymbol{\theta})) \quad (2.29)$$

and so finally for the likelihood against $\boldsymbol{\theta}$:

$$f(\boldsymbol{\xi} \mid \boldsymbol{\theta}) = \frac{e^{-N/2}}{(2\pi\sigma_0^2(\boldsymbol{\theta}))^{N/2}\sqrt{\det \mathbf{R}(\boldsymbol{\theta})}}. \quad (2.30)$$

A solution can be found by minimising the opposite of the log-likelihood:

$$\forall i \in [\![1, N]\!],\ \frac{\partial}{\partial \theta_i}(-\log f(\boldsymbol{\xi} \mid \boldsymbol{\theta})) = \\ -\frac{1}{\sigma_0^2}\mathbf{z}(\boldsymbol{\theta})^\top \mathbf{R}(\boldsymbol{\theta})^{-1}\frac{\partial \mathbf{R}(\boldsymbol{\theta})}{\partial \theta_i}\mathbf{R}(\boldsymbol{\theta})^{-1}\mathbf{z}(\boldsymbol{\theta}) + \operatorname{tr}\left(\mathbf{R}(\boldsymbol{\theta})^{-1}\frac{\partial \mathbf{R}(\boldsymbol{\theta})}{\partial \theta_i}\right) \quad (2.31)$$

with $\mathbf{z}(\boldsymbol{\theta}) = \boldsymbol{\xi} - \mathbf{F}\boldsymbol{\alpha}(\boldsymbol{\theta})$. This MLE estimation is the most commonly used and the default one in `R` package `DiceKriging` [Roustant et al., 2012]. However it is specific to the Gaussian random process case while kriging has been theoretically defined for any random process with finite covariance (or such that the Universal Kriging framework applies).

To circumvent this limitation, an other technique from [Bachoc, 2013] is based on cross-validation (CV). CV splits the learning database $(\xi(\mathbf{x}_1), \cdots, \xi(\mathbf{x}_N))$ into two sets such that the first one is used to predict the values of the other one. By comparing the predicted values to the true ones this technique aims at estimating the predictive capacity of a model. A specific case of such methods is the Leave-One-Out cross-validation (LOO-CV) when all-but-one samples are used to predict the value at the *removed* one. By iterating over all the possible combinations of *training/predicted* samples, a statistic is built and minimised according to the parameter $\boldsymbol{\theta}$: let ε_i be the error in estimating $\xi(\mathbf{x}_i)$ with $(\xi(\mathbf{x}_j))_{\substack{j=1\\j\neq i}}^{N}$, then [see Le Gratiet, 2013, for a detailed calculation]:

$$\varepsilon_i(\boldsymbol{\theta}) = \bar{r_i}\left[\mathbf{R}(\boldsymbol{\theta})^{-1}(\mathbf{z}(\boldsymbol{\theta}) - \mathbf{F}\boldsymbol{\alpha}_{-i}(\boldsymbol{\theta}))\right]_{[i]}$$

with $\bar{r_i} = (\mathbf{R}(\boldsymbol{\theta})^{-1}_{[i,i]})^{-1}$ and $\boldsymbol{\alpha}_{-1}(\boldsymbol{\theta})$ given by Eq. (2.24) where the i^{th} data point was removed. Note that we have used computer-like notations for the matrix indices, *i.e.* brackets indicating which coordinates are kept or dropped (negative sign).

$\boldsymbol{\theta}$ is estimated by minimising the sum of the error terms, which does not depend on σ_0^2:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\mathbb{R}_+^d}{\operatorname{argmin}} \sum_{i=1}^{N} \varepsilon_i^2(\boldsymbol{\theta}). \tag{2.32}$$

Still σ_0^2 has to be estimated; Bachoc [2013] suggests to use:

$$\hat{\sigma_0^2} = \frac{1}{N}\sum_{i=1}^{N} \frac{\varepsilon_i^2}{\widetilde{\sigma_i^2}(\boldsymbol{\theta}^*)}$$

with:

$$\widetilde{\sigma_i^2}(\boldsymbol{\theta}) = \bar{r_i} + \bar{r_i}^2(\mathbf{R}^{-1}\mathbf{F})_{[i,]}(\mathbf{F}^\top_{[-i,]}\mathbf{R}^{-1}_{[-i,-i]}\mathbf{F}_{[-i,]})^{-1}(\mathbf{R}^{-1}\mathbf{F})^\top_{[i,]}.$$

Covariance kernel This is the last requirement of Kriging. We have mentioned in the previous paragraph that there were often tensorised unidimensional correlation kernels chosen amongst a family of usual ones. We give in Table 2.3 directly the most usual choices implemented in `R` package `DiceKriging`. The choice of the kernel will impact the smoothness of the random process. Indeed, when $\mathbb{X} = \mathbb{R}^d$, a stationary Gaussian process $\xi(\mathbf{x})$, $\mathbf{x} \in \mathbb{X}$, with covariance $\operatorname{cov}_0[\xi(\mathbf{x}_1), \xi(\mathbf{x}_2)] = k(\mathbf{x}_1 - \mathbf{x}_2)$ is mean square continuous if and only if k is continuous at $\mathbf{h} = 0$; and admits partial mean square derivatives of order $j : \partial^j\xi(\mathbf{x})/\partial x_{i_1}\cdots\partial x_{i_j}$ if and only if $\partial^{2j}k(\mathbf{h})/\partial^2 h_{i_1}\cdots\partial^2 h_{i_j}$ exists and is finite for $\mathbf{h} = 0$.

To conclude this section, we stress out the fact that Kriging can be conduced without the Gaussian hypothesis, which may be difficult to verify in practice. As for any machine

Kernel	Expression: r(x, x') =	Parameters
Gauss	$\exp\left(-\frac{1}{2}\frac{\lvert x-x'\rvert^2}{\theta^2}\right)$	$\theta > 0$
Exponential	$r\exp\left(-\frac{\lvert x-x'\rvert}{\theta}\right)$	$\theta > 0$
Power-exponential	$\exp\left(-\left(\frac{\lvert x-x'\rvert}{\theta}\right)^{\gamma}\right)$	$\theta > 0$, $0 < \gamma \leq 2$
Matern(3/2)	$\left(1+\sqrt{3}\frac{\lvert x-x'\rvert}{\theta}\right)e^{-\sqrt{3}\frac{\lvert x-x'\rvert}{\theta}}$	$\theta > 0$
Matern(5/2)	$\left(1+\sqrt{5}\frac{\lvert x-x'\rvert}{\theta}+\frac{5}{3}\left(\frac{\lvert x-x'\rvert}{\theta}\right)^2\right)e^{-\sqrt{5}\frac{\lvert x-x'\rvert}{\theta}}$	$\theta > 0$

Table 2.3: Usual 1−dimensional kernels for Kriging.

learning methods, some parameters have to be tuned to produce accurate results. However in the case of Kriging, the *physical meaning* of the correlation lengths, *i.e.* more or less the hypersphere of influence of a data point, can give insights on the pertinence of the estimation. Especially when the input space is the standard Gaussian one: $\mathbb{X} = \mathbb{R}^d, d \geq 1$ and $\mathbf{X} \sim \mathcal{N}(0, \mathbf{I}_d)$ with $\mathbf{I}_d$ the identity matrix in $\mathbb{R}^d$, the impact of correlation lengths on probability estimation is *easily* interpretable: for instance a correlation length $\theta_i \gtrsim 10$ means that for each data point $\mathbf{x} \in \mathbb{X}$, the whole line $\mathbf{x} + \mathbb{R}\mathbf{e}_i$ (with $\mathbf{e}_i$ the i^{th} coordinate vector of $\mathbb{R}^d$) will be in the same domain as $\mathbf{x}$ (failure of safety) with great probability. In addition to this first remark, the kriging mean does not depend on the variance parameter σ_0^2. For all these reason Kriging will be a preferred choice in Chapter 5.

2.2 Design of Experiments

2.2.1 First Design of Experiments

This first Design of Experiments (DoE) is a set of points initially evaluated at the beginning of an algorithm. They are chosen in order to acquire a *global* knowledge of the function before trying to focus on the rare events. Indeed the importance of the DoE should not be underrated: as explained by Echard et al. [2011] for its AKMCS method, its is very unlikely that the algorithm converges toward the sought value if no failing samples are embedded into the first DoE.

This section presents usual concepts and methods for generating a first DoE following Pronzato and Müller [2012]. Other general introductions to the design of computer experiments can be found in books by Santner et al. [2003], Fang et al. [2005] or Kleijnen [2008]. A recent comparative study has also been performed by Damblin et al. [2013].

As expressed by the fact that one looks for a *global* knowledge of the computer code, one is going to consider so-called *space-filling* designs, *i.e.* designs which spread out *uniformly* across the input space $\mathbb{X}$. Recall that we have assumed $\mathbb{X} = \mathbb{R}^d$ or $\mathbb{X} \subset \mathbb{R}^d$ is an hypercube, and since there is no uniform distribution over $\mathbb{R}$, one is going to consider a

compact subset $\mathbb{X}_0 \subset \mathbb{X}$. This subset $\mathbb{X}_0$ is often chosen to be an hypercube in $\mathbb{R}^d$ and with appropriate renormalisation we consider $\mathbb{X}_0 = [0,1]^d$. Other approaches based on hyperspheres are also proposed in the literature for standard Gaussian input spaces for instance [Dubourg et al., 2011]. The distance considered in the following is the Euclidean distance and is denoted by $||\cdot||_2$. The distance of $\mathbf{x} \in \mathbb{X}_0$ to a given subset $\mathcal{X} \subset \mathbb{X}_0$ is defined by $d(\mathbf{x}, \mathcal{X}) = \min_{\mathbf{x}' \in \mathcal{X}} ||\mathbf{x} - \mathbf{x}'||_2$.

Let us consider $\mathcal{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_{N_{\text{DoE}}}) \in \mathbb{X}_0^{N_{\text{DoE}}}$ an initial DoE. The size $N_{\text{DoE}} \geq 1$ of this first evaluated batch will not be discussed here. Guidelines can be found in [Loeppky et al., 2012]; as a rule of thumb $N_{\text{DoE}} \approx 5$ to $10d$, with d the dimension of the input space (note that for parameters estimation of Section 2.1.4, one should have $N_{\text{DoE}} \geq d+1$).

Two main different geometric criteria are used to characterise space-filling DoE over $\mathbb{X}_0$ [Johnson et al., 1990]:

1. the maximin criterion, which aims at maximising the minimum distance $\phi_m(\mathcal{X})$ between two samples of the DoE:

$$\max_{\mathcal{X} \in \mathbb{X}_0^{N_{\text{DoE}}}} \phi_m(\mathcal{X}) = \max_{\mathcal{X} \in \mathbb{X}_0^{N_{\text{DoE}}}} \min_{(\mathbf{x}_i \neq \mathbf{x}_j) \in \mathcal{X}^2} ||\mathbf{x}_i - \mathbf{x}_j||_2;$$

2. the minimax criterion which minimises the maximum distance $\phi_M(\mathcal{X})$ between any sample in $\mathbb{X}_0$ and $\mathcal{X}$:

$$\min_{\mathcal{X} \in \mathbb{X}_0^{N_{\text{DoE}}}} \phi_M(\mathcal{X}) = \min_{\mathcal{X} \in \mathbb{X}_0^{N_{\text{DoE}}}} \max_{\mathbf{x} \in \mathbb{X}_0} d(\mathbf{x}, \mathcal{X}).$$

However in addition to good space-filling properties one also looks for good projection properties, *i.e.* that the projection of $\mathcal{X}$ on each dimension should also be a space-filling design for the one-dimensional subset. This is useful when focusing on main important variables while some others play no substantial role in the variation of the output; see for instance [Saltelli et al., 2008] or [Iooss and Lemaître, 2015] for a review of sensitivity analysis methods, which will not be addressed in this thesis. In any case, this leads to focusing on Latin Hypercube Sampling (LHS), which have the property that any of their one-dimensional projections leads to the maximin distance sequence [Pronzato and Müller, 2012] defined by:

$$\forall i \in [\![1, N]\!],\ x_i = \frac{i-1}{N-1}.$$

Practically speaking, finding an optimal LHS (according to maximin and/or minimax criterion) can be computationally intensive. Indeed there are $(N_{\text{DoE}}!)^{d-1}$ possible LHS, which rapidly discards any attempt to perform an exhaustive search over all the solutions. Most of the algorithms are of exchange-type, *i.e.* that starting from a random LHS they iteratively switch the i^{th} coordinates of two samples and update the current value of the criterion to accept or reject the *transition*.

All these geometrical properties however do not take into account the fact that this first DoE may have to serve also the estimation of the model parameters. Basically space-filling designs will produce somehow evenly spaced samples while estimating correlation lengths requires some *diversity*. This issue of kriging-parameter estimation is addressed by Pronzato and Müller [2012]. Also the `R` package `DiceDesign` [Dupuy et al., 2015] implements the Strauss-Gibbs process, historically introduced to represent repulsion between charged particles, and used here to generate suitable DoE for kriging [Dupuy et al., 2011]. The Strauss-Gibbs potential is of the form:

$$\pi(\mathcal{X}) \propto \exp\left(-\beta \sum_{1\leq i,j\leq N_{\text{DoE}}} \phi(||\mathbf{x}_i - \mathbf{x}_j||_2)\right)$$

with $\gamma = e^{-\beta} \in (0,1]$ a repulsion parameter and $\phi : \mathbb{R} \to \mathbb{R}$ a decreasing continuous function such that $\phi(0) = 1$ and $\phi(x) \to 0$ when $x \to \infty$. More specifically, ϕ is parametrised by $\alpha \geq 0$ and $r > 0$ such that:

$$\phi_{\alpha,r}(x) = \begin{cases} \left(1 - \dfrac{x}{r}\right)^{\alpha} & 0 \leq x \leq r \\ 0 & \text{otherwise} \end{cases}.$$

$\alpha = 0$ corresponds to a Strauss process and in this case the target distribution becomes:

$$\pi(\mathcal{X}) \propto \gamma^{s(\mathcal{X})}$$

with $s(\mathcal{X})$ the number of pairs of points in $\mathcal{X}$ closer than the radius r.

Once a first DoE $\mathcal{X}$ is sampled and a first metamodel is built according to the calculated outputs $(g(\mathbf{x}_1), \cdots, g(\mathbf{x}_{N_{\text{DoE}}}))$, algorithms focus on refining the DoE according to some given target criteria defined on purpose. These procedures are iterative in order to benefit progressively from the calculations to refine the metamodel.

Remark 2.1. *Batch sequential strategies are sometimes used when computer clusters are available and so allow for simultaneous computations of several points. Note however that these strategies are in essence suboptimal regarding the total number of calculated samples N because they use* less information *than totally sequential ones. In the latter the $(n+1)^{th}$ point is selected taking into account the n already sampled data; with batches of size $k \geq 2$ samples $n+1, \cdots, n+k$ are generated with the information gathered with the n first points, i.e. that points $n+2, \cdots, n+k$ are generated with* less information *than their sequential counterparts. Furthermore heavy computer codes are usually already designed to use parallel computing facilities and so a trade-off has to be found between parallelisation of one given run of the code g and running in parallel several calls to g.*

2.2.2 Model-oriented designs

In this section we present some of the main criteria used in practical algorithms for refining the metamodel. We first handle the case of a SVM metamodel (see Section 2.1.2) and then come up with criteria based on random processes, and especially criteria for kriging using the Gaussian hypothesis (see Section 2.1.4). Theoretical and numerical comparisons of criteria presented in this Section can be found in [Bect et al., 2012] and [Dubourg, 2011, Chapter 2].

As mentioned in Section 2.1.2, SVM classifiers in their original setting only depend on the so-called *support-vectors*, *i.e.* vectors which define the classifier and so are onto the margin, defined by the constraints of Eq. (2.6):

$$\mathcal{M}_1 = \{\mathbf{x} \in \mathbb{X} \mid |k(\omega, \mathbf{x}) + b| < 1\}. \tag{2.33}$$

In this setting, adding points to the DoE out of the margin will not change the classifier. Even though the introduction of slack variables in the optimisation problem (Eq. 2.8) makes it slightly more complex to predict the behaviour of the classifier when adding a new point to the DoE, this consideration leads to a heuristic strategy called the *margin shrinking concept* [Hurtado, 2013] used for example by Deheeger [2008] and Bourinet et al. [2011] for SMART and ^{2}SMART algorithms:

1. get samples into the margin, either from the DoE, or by a crude Monte Carlo sampling;
2. generate a Monte Carlo population conditionally to be into the margin by using for instance MCMC algorithms (see Section 1.3.3);
3. select a batch of $k \geq 1$ points by clustering [see for instance Duda et al., 2012]; and
4. evaluate the model g and train it with the augmented DoE.

Due to the definition of a SVM, the margin concept is natural for this class of models. However it carries the idea that in the context of rare event simulation, the metamodel does not need to be precise far from the boundary. The only useful information for a Monte Carlo estimator for instance (see Chapter 1) is whether or not the sample lies into the failure domain. These considerations lead to some variations of this concept applied to the Gaussian process regression. Recall that $\xi(\mathbf{x}) \mid \mathcal{F}_n \sim \mathcal{N}(\xi_n(\mathbf{x}), \sigma_n(\mathbf{x}))$, Echard et al. [2011] define the so-called U statistic as a normalised algebraic distance to the failure threshold q:

$$\forall \mathbf{x} \in \mathbb{X},\ U(\mathbf{x}) = \frac{q - \xi_n(\mathbf{x})}{\sigma_n(\mathbf{x})}. \tag{2.34}$$

Similarly to Eq. (2.33), the margin is defined as $\mathcal{M}_k = \{\mathbf{x} \in \mathbb{X} \mid |U(\mathbf{x})| < k\}$. A usual value for k is 2, meaning that the margin is composed by points classified with less than $\approx 97.5\%$ confidence. Building upon this result, Dubourg [2011] proposes a smoother

version of this concept with the so-called *margin probability* function MP. Instead of using directly the kriging mean $\xi_n(\mathbf{x})$ in Eq. (2.34), it considers the random process ξ itself. The U statistic becomes:

$$\forall \mathbf{x} \in \mathbb{X},\ \widetilde{U}(\mathbf{x}) = \frac{q - \xi(\mathbf{x})}{\sigma_n(\mathbf{x})}$$

with conditional mean $\mathrm{E}_n\left[\widetilde{U}(\mathbf{x})\right] = U(\mathbf{x})$ and variance $\mathrm{var}_n\left[\widetilde{U}(\mathbf{x})\right] = 1$. Considering the *hard-margin* indicator $\mathbb{1}_{\mathbf{x}\in\mathcal{M}_k}$, one finally has:

$$\mathbb{1}_{\mathbf{x}\in\mathcal{M}_k} = \mathbb{1}_{|U(\mathbf{x})|<k} \leftarrow \mathrm{MP}(\mathbf{x}) = \mathrm{E}_n\left[\mathbb{1}_{|\widetilde{U}(\mathbf{x})|<k}\right] = \mathrm{P}_n\left[|\widetilde{U}(\mathbf{x})| < k\right]. \tag{2.35}$$

Since ξ is a Gaussian random process, the distribution of $\widetilde{U}(\mathbf{x})$ given $\mathcal{F}_n$ is known and one has:

$$\begin{aligned}
\mathrm{MP}(\mathbf{x}) &= \Phi\left(U(\mathbf{x}) + k\right) - \Phi\left(U(\mathbf{x}) - k\right) \\
\mathrm{MP}(\mathbf{x}) &= \Phi\left(\frac{q - \xi_n(\mathbf{x}) + k\sigma_n(\mathbf{x})}{\sigma_n(\mathbf{x})}\right) - \Phi\left(\frac{q - \xi_n(\mathbf{x}) - k\sigma_n(\mathbf{x})}{\sigma_n(\mathbf{x})}\right)
\end{aligned} \tag{2.36}$$

with Φ the *cdf* of a standard Gaussian random variable. Both the *hard-margin* indicator and the *margin probability* function can then be used to weight usual criteria for Gaussian processes such as the Integrated Mean-Squared Error (IMSE):

$$\mathrm{IMSE} = \int_{\mathbb{X}} \sigma_n^2(\mathbf{x}) \mathrm{d}\mu^X(\mathbf{x}).$$

For instance, Picheny et al. [2010] defined the targeted IMSE as a weighted IMSE designed to focus on some regions of interest:

$$\mathrm{tIMSE} = \int_{\mathbb{X}} \sigma_n^2(\mathbf{x}) w(\mathbf{x}) \mathrm{d}\mu^X(\mathbf{x}) \tag{2.37}$$

with w a given weighting function. Picheny et al. [2010] suggested to use either a *hard-margin* indicator $w(\mathbf{x}) = \mathrm{E}_n\left[\mathbb{1}_{|\xi(\mathbf{x})-q|<\varepsilon}\right]$ for a given $\varepsilon > 0$ or a smoother version with the *pdf* $\phi_{\sigma_\varepsilon}$ of a Gaussian standard random variable $\mathcal{N}(0, \sigma_\varepsilon^2)$, $\sigma_\varepsilon > 0$: $w(\mathbf{x}) = \mathrm{E}_n\left[\phi_{\sigma_\varepsilon}(\xi(\mathbf{x}) - q)\right]$. The *hard-margin* indicator $\mathrm{E}_n\left[\mathbb{1}_{|\xi(\mathbf{x})-q|<\varepsilon}\right]$ is closely related to the *margin probability* function MP: this latter is a special case of the former, parametrised with $\varepsilon = \sigma_n(\mathbf{x})k$, $k > 0$. Hence it admits also a closed-form expression:

$$\mathrm{E}_n\left[\mathbb{1}_{|\xi(\mathbf{x})-q|<\varepsilon}\right] = \Phi\left(\frac{q - \xi_n(\mathbf{x}) + \varepsilon}{\sigma_n(\mathbf{x})}\right) - \Phi\left(\frac{q - \xi_n(\mathbf{x}) - \varepsilon}{\sigma_n(\mathbf{x})}\right).$$

On the other hand the *soft-margin* indicator $\mathrm{E}_n\left[\phi_{\sigma_\varepsilon}(\xi(\mathbf{x}) - q)\right]$ is indeed the convolution of two Gaussian random variables and one has [Picheny et al., 2010]:

$$\mathrm{E}_n\left[\phi_{\sigma_\varepsilon}(\xi(\mathbf{x}) - q)\right] = \frac{1}{\sqrt{2\pi(\sigma_\varepsilon^2 + \sigma_n^2(\mathbf{x}))}} e^{-\frac{1}{2}\frac{(\xi_n(\mathbf{x})-q)^2}{\sigma_\varepsilon^2 + \sigma_n^2(\mathbf{x})}}.$$

Then this tIMSE criterion is used in a one-step-look-ahead strategy: since the kriging variance does not depend on the random variables $(\xi(\mathbf{x}_i))_{i=1}^n$ but only on their locations $(\mathbf{x}_i)_{i=1}^n$, one can update it assuming that a new point $\mathbf{x}^*$ is added to the DoE: $\sigma^2_{n,\mathbf{x}^*}(\mathbf{x}) = \mathrm{var}_0\left[\xi(\mathbf{x}) \mid \mathcal{F}_n, \mathbf{X}_{n+1} = \mathbf{x}^*\right]$ and calculate:

$$\mathrm{tIMSE}(\mathbf{x}^*) = \int_{\mathbb{X}} \sigma^2_{n,\mathbf{x}^*}(\mathbf{x}) w(\mathbf{x}) \mathrm{d}\mu^X(\mathbf{x}).$$

Then a possible refinement strategy is to select the point minimising the *forthcoming* tIMSE, *i.e.*:

$$\mathbf{x}_{n+1} = \underset{\mathbf{x}^* \in \mathbb{X}}{\mathrm{argmin}}\, \mathrm{tIMSE}(\mathbf{x}^*).$$

2.2.3 Stepwise Uncertainty Reduction

The idea of using a criterion based on the *next* metamodel is at the very heart of Stepwise Uncertainty Reduction (SUR) strategies. But instead of considering an arbitrary measure of uncertainty (for instance the margin and the IMSE in the previous section), it rather focuses on the error between the chosen estimator and the sought quantity. Bect et al. [2012] developed this general concept with a Bayesian decision-theoretic framework and applied it to rare event estimation. We focus here on the rare event setting and refer the reader to the original paper for a more general presentation of SUR strategies.

Basically, a SUR strategy is as follows:

1. select α_n a $\mathcal{F}_n-$measurable function as an estimator of the sought quantity α;
2. select a loss function $\epsilon(\alpha_n, \alpha)$ quantifying the error between α_n and α; and
3. select iteratively the next sample as the one minimising the *forthcoming* error:

$$\mathbf{x}_{n+1} = \underset{\mathbf{x}^* \in \mathbb{X}}{\mathrm{argmin}}\, \mathrm{E}_n\left[\epsilon(\alpha_{n+1}, \alpha) \mid \mathbf{X}_{n+1} = \mathbf{x}^*\right].$$

This strategy is referred to as a one-step-look-ahead SUR because it outputs only one sample at a time. Similarly r-steps-look-ahead SUR can be defined. However according to Remark 2.1 this is not obviously suitable (if the parameters of the covariance are estimated for instance) while increasing drastically the complexity of the optimum research. In the sequel we stick to the one-step-look-ahead strategy but all the formulations remain true for any $r \geq 1$.

In the context of rare event estimation, Bect et al. [2012] adopt a Bayesian approach, *i.e.* that they put a Gaussian process prior on the *unknown* function g: $\xi(\mathbf{x})$ is a Gaussian random process indexed by $\mathbf{x} \in \mathbb{X}$ with covariance kernel k. Thus they introduce the *augmented* problem:

$$p = \int_{\mathbb{X}} \mathbb{1}_{g(\mathbf{x})>q} \mathrm{d}\mu^X(\mathbf{x}) \leftarrow \alpha = \int_{\mathbb{X}} \mathbb{1}_{\xi(\mathbf{x})>q} \mathrm{d}\mu^X(\mathbf{x}). \tag{2.38}$$

The term "augmented" refers to the fact that this simply adds a dimension to the original problem (1.1), according to Eq. (2.11). This approach will be further developed in Chapter 5. In this setting the quantity of interest α is not a deterministic value any more but a random variable. Given a filtration $\mathcal{F}_n$, an optimal estimator (minimal variance) of α is the conditional expectation $\alpha_n = \mathrm{E}_n[\alpha]$ and so the loss function ϵ is chosen to be the quadratic loss:

$$\epsilon(\alpha_n, \alpha) = (\alpha_n - \alpha)^2.$$

Finally, the criterion to be minimised at each iteration writes:

$$J^{\alpha}_{n,n+1}(\mathbf{x}^*) = \mathrm{E}_n\left[(\alpha_{n+1} - \alpha)^2 \mid \mathbf{X}_{n+1} = \mathbf{x}^*\right] = \mathrm{E}_n\left[\mathrm{var}_{n+1}[\alpha]\right]. \tag{2.39}$$

In the end, the SUR strategy means that one iteratively select the point minimising the conditional variance of the random variable of interest, *i.e.* which minimises the *remaining uncertainty* in the approximation of α by $\mathrm{E}_n[\alpha]$. Chevalier et al. [2014] proposed a closed-form expression for this formula for any $r \geq 1$, with $\mathbf{x}^* \in \mathbb{X}^k$, $\mathbf{x}^* = (\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r})$:

$$J^{\alpha}_{n,n+r}(\mathbf{x}^*) = \gamma_n - \int_{\mathbb{X}\times\mathbb{X}} \Phi_2\left(\begin{pmatrix} a(\mathbf{x}_1) \\ a(\mathbf{x}_2) \end{pmatrix}, \begin{pmatrix} c(\mathbf{x}_1) & d(\mathbf{x}_1, \mathbf{x}_2) \\ d(\mathbf{x}_1, \mathbf{x}_2) & c(\mathbf{x}_2) \end{pmatrix}\right) \mathrm{d}\mu^X(\mathbf{x}_1)\mathrm{d}\mu^X(\mathbf{x}_2) \tag{2.40}$$

with:

- $a(\mathbf{x}) = (\xi_n(\mathbf{x}) - q)/\sigma_{n+r}(\mathbf{x})$;
- $\mathbf{b}(\mathbf{x}) = \boldsymbol{\Sigma}^{-1}(k_n(\mathbf{x}, \mathbf{x}_{n+1}), \cdots, k_n(\mathbf{x}, \mathbf{x}_{n+r}))^\top / \sigma_{n+r}(\mathbf{x})$;
- $c(\mathbf{x}) = 1 + \mathbf{b}(\mathbf{x})^\top \boldsymbol{\Sigma} \mathbf{b}(\mathbf{x}) = \sigma^2_n(\mathbf{x})/\sigma^2_{n+r}(\mathbf{x})$;
- $d(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{b}(\mathbf{x}_1)^\top \boldsymbol{\Sigma} \mathbf{b}(\mathbf{x}_2)$;
- γ_n a value not depending on $(\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r})$;
- $\boldsymbol{\Sigma}$ the $r \times r$ covariance matrix of $(\xi(\mathbf{x}_{n+1}), \cdots, \xi(\mathbf{x}_{n+r}))^\top$ conditional on $\mathcal{F}_n$; and
- $\Phi_2(\cdot, M)$ the *cdf* of the centred bivariate Gaussian with covariance matrix M.

This quantity can indeed be rewritten:

$$J^{\alpha}_{n,n+r}(\mathbf{x}^*) = \int_{\mathbb{X}\times\mathbb{X}} \Delta \mathrm{P}_{n+r}(\mathbf{x}_1, \mathbf{x}_2) \mathrm{d}\mu^{\mathbf{X}}(\mathbf{x}_1)\mathrm{d}\mu^{\mathbf{X}}(\mathbf{x}_2) \tag{2.41}$$

with:

$$\Delta \mathrm{P}_{n+r}(\mathbf{x}_1, \mathbf{x}_2) = \mathrm{P}_n\left[\xi(\mathbf{x}_1) > q, \xi(\mathbf{x}_2) > q\right] - \mathrm{P}_n\left[U^{(1)}_{n+r}(\mathbf{x}_1) > q, U^{(2)}_{n+r}(\mathbf{x}_2) > q\right], \tag{2.42}$$

where:

$$\begin{pmatrix} U_{n+r}^{(1)}(\mathbf{x}_1) \\ U_{n+r}^{(2)}(\mathbf{x}_2) \end{pmatrix} \mid \mathcal{F}_n \sim \mathcal{N}\left(\begin{pmatrix} \xi_n(\mathbf{x}_1) \\ \xi_n(\mathbf{x}_2) \end{pmatrix}, \begin{pmatrix} \sigma_n^2(\mathbf{x}_1) & \mathrm{cov}_n\left[\xi_{n+r}(\mathbf{x}_1), \xi_{n+r}(\mathbf{x}_2)\right] \\ \mathrm{cov}_n\left[\xi_{n+r}(\mathbf{x}_1), \xi_{n+r}(\mathbf{x}_2)\right] & \sigma_n^2(\mathbf{x}_2) \end{pmatrix} \right). \tag{2.43}$$

The two Gaussian couples in Eq. (2.42) differ only in their covariance: in this latter formula the conditional expectation ξ_{n+r} is used instead of the original random process ξ. All these formulae remain true for any $r \geq 0$:

$$\mathrm{cov}_n\left[U_{n+r}^{(1)}(\mathbf{x}_1), U_{n+r}^{(2)}(\mathbf{x}_2)\right] = \mathrm{cov}_n\left[\xi_{n+r}(\mathbf{x}_1), \xi_{n+r}(\mathbf{x}_2)\right]. \tag{2.44}$$

On the one hand for $r = 0$, one finds back that conditionally to $\mathcal{F}_n$, $U_n^{(1)}(\mathbf{x}_1)$ and $U_n^{(2)}(\mathbf{x}_2)$ are independent random variables with distribution $\mathcal{N}(\xi_n(\mathbf{x}_1), \sigma_n^2(\mathbf{x}_1))$ and $\mathcal{N}(\xi_n(\mathbf{x}_2), \sigma_n^2(\mathbf{x}_2))$ respectively. This gives:

$$J_n^{\alpha} = \mathrm{E}_n\left[\mathrm{var}_n\left[\alpha\right]\right] = \mathrm{var}_n\left[\alpha\right].$$

One the other hand, the greater r the *closer* ξ_{n+r} to ξ in the quadratic mean, *i.e.* the closer $\mathrm{cov}_n\left[\xi_{n+r}(\mathbf{x}_1), \xi_{n+r}(\mathbf{x}_2)\right]$ to $\mathrm{cov}_n\left[\xi(\mathbf{x}_1), \xi(\mathbf{x}_2)\right]$. Eventually the criterion goes to 0 as $r \to \infty$: if the size of the next batch of points to be estimated goes to infinity then the error goes to 0.

From a practical point of view, only the second part of Eq. (2.40) depends on the proposed point $\mathbf{x}^*$ and needs to be evaluated. This integration over $\mathbb{X} \times \mathbb{X}$ of a bivariate *cdf* of Gaussian random variables can be numerically demanding. However Chevalier et al. [2014] showed that it is numerically almost as efficient (in terms of relative Root-Mean-Squared Error of the estimation of p) to use an upper bound of $J_{n,n+r}^{\alpha}$ based on an upper bound of $\mathrm{var}_n\left[\alpha\right]$:

$$\begin{aligned} \mathrm{var}_n\left[\alpha\right] &= \mathrm{var}_n\left[\int_{\mathbb{X}} \mathbb{1}_{\xi(\mathbf{x})>q} \mathrm{d}\mu^{\mathbf{X}}(\mathbf{x})\right] \\ &\leq \left(\int_{\mathbb{X}} \sqrt{\mathrm{var}_n\left[\mathbb{1}_{\xi(\mathbf{x})>q}\right]} \mathrm{d}\mu^{\mathbf{X}}(\mathbf{x})\right)^2 \\ &\leq \int_{\mathbb{X}} \mathrm{var}_n\left[\mathbb{1}_{\xi(\mathbf{x})>q}\right] \mathrm{d}\mu^{\mathbf{X}}(\mathbf{x}). \end{aligned} \tag{2.45}$$

This upper bound reduces the integration over $\mathbb{X} \times \mathbb{X}$ to an integration over $\mathbb{X}$ only. Denote $p_n^q(\mathbf{x}) = \mathrm{P}_n\left[\xi(\mathbf{x}) > q\right], \forall n \geq 0$, this latter formulation gives for the criterion:

$$J_{n,n+r}^{\alpha}(\mathbf{x}^*) \leq J_{n,n+r}^{\Gamma}(\mathbf{x}^*) \stackrel{\mathrm{def}}{=} \int_{\mathbb{X}} \mathrm{E}_n\left[p_{n+r}^q(\mathbf{x})\left(1 - p_{n+r}^q(\mathbf{x})\right)\right] \mathrm{d}\mu^{\mathbf{X}}(\mathbf{x}), \tag{2.46}$$

which can be rewritten [Chevalier et al., 2014]:

$$J_{n,n+r}^{\Gamma}(\mathbf{x}^*) = \int_{\mathbb{X}} \Phi_2\left(\begin{pmatrix} a(\mathbf{x}) \\ -a(\mathbf{x}) \end{pmatrix}, \begin{pmatrix} c(\mathbf{x}) & 1 - c(\mathbf{x}) \\ 1 - c(\mathbf{x}) & c(\mathbf{x}) \end{pmatrix}\right) \mathrm{d}\mu^{X}(\mathbf{x}). \tag{2.47}$$

Once again this quantity can be rewritten using the same Gaussian couple $(U^{(1)}_{n+r}, U^{(2)}_{n+r})$:

$$J^{\Gamma}_{n,n+r}(\mathbf{x}^*) = \int_{\mathbb{X}} \mathrm{P}_n \left[U^{(1)}_{n+r}(\mathbf{x}) > q, U^{(2)}_{n+r}(\mathbf{x}) < q \right] \mathrm{d}\mu(\mathbf{x}). \tag{2.48}$$

Even though criteria (2.39) and (2.46) are now expressed as the integral of a bivariate Gaussian *cdf* whose dimension does not depend on r, a direct Monte Carlo approach to quantify Eqs. (2.40) and (2.47) may fail to produce a correct estimation because of the rarity of the event $\{\xi(\mathbf{x}) > q\}$. Chevalier et al. [2014] proposed to use a combination of crude Monte Carlo and Importance Sampling to circumvent this limitation. In Chapter 5 we will show how results on Poisson processes presented further in Chapter 3 can be used to address this issue with reformulations (2.41) and (2.48), reducing this calculation to the average of a univariate Gaussian *cdf* on a suitable population.

To conclude this section on designs of experiments, we stress out the fact that SUR strategies appear as the most conservative and theoretically founded strategies. However the price to pay in terms of numerical complexity can become relatively consequent and a trade-off between criterion complexity and computational time of the computer code g has to be found. This will be illustrated in the numerical examples of Section 5.3.

2.3 Metamodels and estimators

In this section, we address the issue of the selection of an estimator based on the metamodel used. Indeed, while in Section 2.1.1 we have seen that the First Order Reliability Method leads itself to an analytical expression of the probability of failure, the other approaches need to be used in addition to Monte Carlo estimators such as the ones defined in Chapter 1.

In all this section, we hence consider that a surrogate model has been built with a given number of calls to the real computer code g and we describe several possibilities for building the estimator of $p = \mathrm{P}\left[g(\mathbf{X}) > q\right]$.

2.3.1 Crude Monte Carlo estimator

The crude Monte Carlo method presented in Section 1.1 combines the advantages of supporting a Central Limit Theorem, being consistent, easy to implement, parallelisable and insensitive to the dimension of the problem. We saw that its main and only drawback is that it is not well suited for rare event estimation as its squared coefficient of variation typically scales like $1/(Np)$ with N the number of generated samples and p the sought probability. However, if the metamodel is fast-to-compute, this number N can be high at almost *no cost* comparing the to original code g.

For the sake of clarity we rewrite here the definition of the crude Monte Carlo estimator:

$$\widehat{p}_{\mathrm{MC}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{g(\mathbf{X}_i)>q} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{\mathbf{X}_i \in F} \tag{2.49}$$

with $(\mathbf{X}_i)_{i=1}^N$ $N \geq 1$ *iid.* random variables with distribution μ^X and $F = \{\mathbf{x} \in \mathbb{X} \mid g(\mathbf{x}) > q\}$ the failure domain. Hence a crude Monte Carlo estimator only requires to know whether a given random sample $\mathbf{X} \sim \mu^X$ lies into the failure domain or not.

Hard classifiers, like SORM or a SVM, are especially designed to answer to this question instead of the true computer code g. Without loss of generality, one considers a classifier $\widetilde{g} : \mathbb{X} \to \{0, 1\}$ such that it returns 1 if the sample $\mathbf{x}$ is classified as failing and 0 otherwise. This definition leads to the so-called plug-in approach:

$$\widehat{p} = \frac{1}{N}\sum_{i=1}^{N} \widetilde{g}(\mathbf{X}_i)$$

which gives an unbiased estimate of $\widetilde{p}_{\mathrm{HC}} = \mathrm{P}\left[\mathbf{X} \in \widetilde{F}_{\mathrm{HC}}\right]$ with $\widetilde{F}_{\mathrm{HC}} = \{\mathbf{x} \in \mathbb{X} \mid \widetilde{g}(\mathbf{x}) = 1\}$. However there is no guarantee that $\widetilde{p}_{\mathrm{HC}}$ is close to p. Indeed one has:

$$\widetilde{p}_{\mathrm{HC}} - p = \int_{\mathbb{X}} \left(\widetilde{g}(\mathbf{x}) - \mathbb{1}_{g(\mathbf{x})>q}\right) \mathrm{d}\mu(\mathbf{x}).$$

If the classifier does not provide any information on its *proximity* to the true limit-state surface, the only way to estimate this quantity is to rely on a crude Monte Carlo estimation, which means ending up with doing a crude Monte Carlo on the original computer code g. This is clearly not an option and thus these approaches require to *trust* the classifier without any possible control on it. This strategy is used for instance by Deheeger [2008] in its algorithm called SMART based on a SVM (and so ^{2}SMART too).

With a Bayesian perspective, it is also possible to define a classifier based on the random process ξ. Recall that in this context the quantity of interest becomes a random variable: $\alpha = \int_{\mathbb{X}} \mathbb{1}_{\xi(\mathbf{x})>q} \mathrm{d}\mu^X(\mathbf{x})$, one looks for a classifier minimising the probability of misclassification of a sample $\mathbf{x} \in \mathbb{X}$ [Bect et al., 2012]:

$$\begin{aligned}
\mathrm{P}_0\left[\widetilde{g}(\mathbf{x}) \neq \mathbb{1}_{\xi(\mathbf{x})>q}\right] &= \mathrm{E}_0\left[\left(\widetilde{g}(\mathbf{x}) - \mathbb{1}_{\xi(\mathbf{x})>q}\right)^2\right] \\
&= \widetilde{g}(\mathbf{x})^2\,\mathrm{P}_0\left[\xi(\mathbf{x}) \leq q\right] + \left(\widetilde{g}(\mathbf{x}) - 1\right)^2 \mathrm{P}_0\left[\xi(\mathbf{x}) > q\right] \\
&= \widetilde{g}(\mathbf{x})\left(1 - \mathrm{P}_0\left[\xi(\mathbf{x}) > q\right]\right) + \left(1 - \widetilde{g}(\mathbf{x})\right)\mathrm{P}_0\left[\xi(\mathbf{x}) > q\right].
\end{aligned}$$

This error is minimised for:

$$\widetilde{g} : \mathbf{x} \in \mathbb{X} \mapsto \widetilde{g}(\mathbf{x}) = \begin{cases} 1 & \mathrm{P}_0\left[\xi(\mathbf{x}) > q\right] \geq 0.5 \\ 0 & \mathrm{P}_0\left[\xi(\mathbf{x}) > q\right] < 0.5 \end{cases},$$

i.e. $\widetilde{g}(\mathbf{x}) = \mathbb{1}_{\overline{\xi(\mathbf{x})}>q}$ with $\overline{\xi(\mathbf{x})}$ the median of $\xi(\mathbf{x})$. If ξ is a Gaussian process: $\forall \mathbf{x} \in$

$\mathbb{X}, \xi(\mathbf{x}) \sim \mathcal{N}(m(\mathbf{x}), \sigma^2(\mathbf{x}))$, then the mean and the median are the same and one finds:

$$\widetilde{g}(\mathbf{x}) = \mathbb{1}_{m(\mathbf{x})>q}.$$

In other words the classifier built from a kriging metamodel minimising the probability of misclassification is the classifier obtained by substituting the true computer code g with the kriging mean. This *hard-classifier* is used for instance by Echard et al. [2011] for their AKMCS (Active learning using Kriging and Monte Carlo Simulation) method.

On the other hand we have presented in Section 2.1.4 the conditional expectation of α as a *natural* estimator for p, *i.e.* the one minimising the quadratic error. From a more pragmatic point of view, one can simply consider that the Bayesian formulation just adds a dimension to problem (2.11) but that it remains essentially the same, *i.e.* the estimation of the probability that a deterministic real-valued function be above a threshold given the distribution of its inputs:

$$p = \mathrm{P}_{\mu^X}\left[g(\mathbf{X}) > q\right] \leftarrow \widetilde{p}_{\mathrm{aug}} = \mathrm{P}_{\mu^X \otimes \mathrm{P}_0}\left[\xi(\mathbf{X}, \omega) > q\right]. \tag{2.50}$$

The integration over P_0 can be calculated, which gives:

$$\widetilde{p}_{\mathrm{aug}} = \int_{\mathbb{X}\times\Omega_0} \mathbb{1}_{\xi(\mathbf{x},\omega)>q} \mathrm{d}\mu^X(\mathbf{x}) \mathrm{d}\,\mathrm{P}_0(\omega) = \int_{\mathbb{X}} \mathrm{P}_0\left[\xi(\mathbf{x}) > q\right] \mathrm{d}\mu^X(\mathbf{x}). \tag{2.51}$$

This latter quantity is referred to as the *augmented failure probability* by Dubourg [2011] because on top of the uncertainty of the parameters of the model embedded in the random vector $\mathbf{X}$, one adds the uncertainty on the code itself. If the law of the random process ξ is known, then this quantity can be estimated with a crude Monte Carlo for instance: for all $\mathbf{x} \in \mathbb{X}$, let $\Phi_{\mathbf{x}}$ be the complementary *cdf* of $\xi(\mathbf{x})$, then:

$$\widehat{p} = \frac{1}{N} \sum_{i=1}^{N} \Phi_{\mathbf{X}_i}(q)$$

is an unbiased estimator of $\widetilde{p}_{\mathrm{aug}} = \mathrm{E}_{\mu^X}\left[\Phi_{\mathbf{X}}(q)\right]$. It is used for instance by Picheny et al. [2010] and Bect et al. [2012]. One the one hand, $\widetilde{p}_{\mathrm{HC}}$ appears as a sort of thresholded version of $\widetilde{p}_{\mathrm{aug}}$:

$$\widetilde{p}_{\mathrm{HC}} = \int_{\mathbb{X}} \mathbb{1}_{\Phi_{\mathbf{x}}(q)>0.5} \mathrm{d}\mu^X(\mathbf{x}) = \int_{\mathbb{X}} \left[\Phi_{\mathbf{x}}(q)\right] \mathrm{d}\mu^X(\mathbf{x})$$

with $[\cdot]$ the rounding operator. On the other hand there is still no guarantee that $\widetilde{p}_{\mathrm{aug}}$ is close to p and the same problem as for the *hard classifier* still holds.

2.3.2 Importance sampling-based procedures

The issue of the uncontrolled bias is inherent to the use of a metamodel as a direct proxy for the true computer code g. To circumvent this limitation, the idea, first developed by Dubourg [2011], is to use the metamodel to define an importance distribution.

We recall that the main concept of Importance Sampling is to use another distribution $\mu^{\tilde{X}}$ such that μ^X is absolutely continuous with respect to $\mu^{\tilde{X}}$. For the sake of simplicity of this section and since it is always the case in our practical examples, we further assume that both μ^X and $\mu^{\tilde{X}}$ have a density with respect to a reference measure, for instance, Lebesgue measure, π and $\widetilde{\pi}$ respectively. We refer to Section 1.2 for a more general introduction to Importance Sampling. The absolute continuity hypothesis means indeed that $\forall \mathbf{x} \in \mathbb{X}$, $\widetilde{\pi}(\mathbf{x}) = 0 \Rightarrow \pi(\mathbf{x}) = 0$ and the Importance Sampling scheme writes:

$$p = \int_{\mathbb{X}} \mathbb{1}_{g(\mathbf{x})>q} \pi(\mathbf{x}) \mathrm{d}\mathbf{x} = \int_{\mathbb{X}} \mathbb{1}_{g(\mathbf{x})>q} \frac{\pi(\mathbf{x})}{\widetilde{\pi}(\mathbf{x})} \widetilde{\pi}(\mathbf{x}) \mathrm{d}\mathbf{x} = \mathrm{E}_{\mu^{\tilde{X}}} \left[\mathbb{1}_{g(\widetilde{\mathbf{X}})>q} \frac{\pi(\widetilde{\mathbf{X}})}{\widetilde{\pi}(\widetilde{\mathbf{X}})} \right].$$

The importance distribution $\mu^{\tilde{X}}$ minimising the variance of the Monte Carlo estimator has the following density:

$$\forall \mathbf{x} \in \mathbb{X},\ \widetilde{\pi}^*(\mathbf{x}) = \frac{\mathbb{1}_{g(\mathbf{x})>q} \pi(\mathbf{x})}{\int_{\mathbb{X}} \mathbb{1}_{g(\mathbf{x})>q} \pi(\mathbf{x}) \mathrm{d}\mathbf{x}} = \frac{\mathbb{1}_{g(\mathbf{x})>q} \pi(\mathbf{x})}{p},$$

i.e. it is the original distribution truncated to the failure domain. Especially it depends on the unknown quantity p. While it may not be possible to use a classifier to define an importance density satisfying the absolute continuity condition (if the classifier says that an area of the input space is safe while it is not), the soft-classifier $\mathrm{P}_0\left[\xi(\mathbf{x}) > q\right] = \Phi_{\mathbf{x}}(q)$ is never null when ξ is a Gaussian process for instance. This leads to the idea of approximating the optimal $\widetilde{\pi}^*$ with the soft-classifier:

$$\forall \mathbf{x} \in \mathbb{X},\ \widetilde{\pi}(\mathbf{x}) = \frac{\mathrm{P}_0\left[\xi(\mathbf{x}) > q\right] \pi(\mathbf{x})}{\int_{\mathbb{X}} \mathrm{P}_0\left[\xi(\mathbf{x}) > q\right] \pi(\mathbf{x}) \mathrm{d}\mathbf{x}} = \frac{\mathrm{P}_0\left[\xi(\mathbf{x}) > q\right] \pi(\mathbf{x})}{\widetilde{p}_{\mathrm{aug}}}. \tag{2.52}$$

On the one hand $\widetilde{p}_{\mathrm{aug}}$ can be estimated with a crude Monte Carlo as explained in Section 2.3.1. On the other hand p can be estimated with $(\widetilde{\mathbf{X}}_i)_{i=1}^N$ $N \geq 1$ *iid.* samples with distribution $\mu^{\tilde{X}}$ by:

$$\widehat{p} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{g(\widetilde{\mathbf{X}}_i)>q} \frac{\pi(\widetilde{\mathbf{X}}_i)}{\widetilde{\pi}(\widetilde{\mathbf{X}}_i)} = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbb{1}_{g(\widetilde{\mathbf{X}}_i)>q}}{\mathrm{P}_0\left[\xi(\widetilde{\mathbf{X}}_i) > q\right]} \widetilde{p}_{\mathrm{aug}}. \tag{2.53}$$

Such Importance Sampling schemes are used for example by Dubourg et al. [2011], Balesdent et al. [2013] or Cadini et al. [2014]. Also Bect et al. [2015] recently showed that in a Bayesian framework, the optimal importance density is indeed:

$$\forall \mathbf{x} \in \mathbb{X},\ \widetilde{\pi}(\mathbf{x}) = \frac{\sqrt{\mathrm{E}_0\left[\mathbb{1}^2_{\xi(\mathbf{x})>q}\right]} \pi(\mathbf{x})}{\int_{\mathbb{X}} \sqrt{\mathrm{E}_0\left[\mathbb{1}^2_{\xi(\mathbf{x})>q}\right]} \pi(\mathbf{x})} = \frac{\sqrt{\mathrm{P}_0\left[\xi(\mathbf{x}) > q\right]} \pi(\mathbf{x})}{\int_{\mathbb{X}} \sqrt{\mathrm{P}_0\left[\xi(\mathbf{x}) > q\right]} \pi(\mathbf{x}) \mathrm{d}\mathbf{x}}. \tag{2.54}$$

Their numerical study seems to show a little improvement in terms of Mean Squared Error against the importance density of Eq. (2.52). However from a practical point of view

these methods require to simulate at least a small number of times the heavy computer code g for the estimator (2.53). This means stopping the enrichment step before having spent all the computational budget available. The question of a good repartition of the computational budget between the *exploration* phase and the Importance Sampling scheme in order to minimise the Root Mean Squared Error of the estimator is still unclear.

Furthermore the variance of the Importance Sampling estimator may be very high if the metamodel misses a part of the failure domain, *i.e.* if it considers that an area is safe with great probability while it is not. This issue is the main drawback of Importance Sampling, namely that the variance may be even greater than the crude Monte Carlo one. For such algorithms, the metamodel should be very conservative, *i.e.* that it is very important not to consider a part of the input space $\mathbb{X}$ as safe while it is not. Unfortunately there is actually no criterion for such property. Some usual techniques for Importance Sampling using mixtures of densities [Owen and Zhou, 2000] may be used but this will increase again the number of required samples for this last step.

In the end, since these algorithms intend to spend few computational budget on this final Importance Sampling estimator, there is almost no luck that a potential *ill-learnt* metamodel be detected (it will not sample in areas considered as safe with great probability, and so not visit areas which may be indeed in the failure domain). Eventually, in the numerical examples the Importance Sampling sum $\widehat{p}/\widetilde{p}_{\text{aug}}$ in Eq. (2.53) is always very close to 1.

All these considerations lead us to consider that this *correction* of $\widetilde{p}_{\text{aug}}$ as presented by Dubourg [2011] and visible in Eq. (2.53) has to be manipulated with caution.

2.3.3 Subset Simulation

All the estimators presented previously are based on a crude Monte Carlo estimator, either for $\widetilde{p}_{\text{HC}}$ or for $\widetilde{p}_{\text{aug}}$ and the importance sampling based correction. However, for the same reasons as in Section 1.2, a crude Monte Carlo estimator may struggle to produce a reliable estimate.

The problem is indeed twofold: 1) enrichment strategies based on an *iid.* sampling of the input random vector may fail to *discover* the failure domain and to propose relevant samples; and 2) the estimation of the selected estimator may require a too large sample size for processing capacities or be very inaccurate. This limitation is especially severe in an algorithm like AKMCS [Echard et al., 2011] and all its variants [Echard et al., 2013, Fauriat and Gayton, 2014] where an initial *iid.* sampling is considered for the whole algorithm as a discretised version of μ^X. Chevalier et al. [2014] also relied on crude Monte Carlo sampling to find samples in the failure domain for the evaluation of the SUR criterion and so were limited to relatively big probabilities ($p \approx 10^{-2}$).

The Subset Simulation algorithm, which lets go sequentially with a sequence of intermediate thresholds to the failure domain, has been used by Bourinet et al. [2011] to address both issues simultaneously. Remember that a Subset Simulation algorithm works

typically as follows:

1. $i = 0$; $q_i = -\infty$; $\widehat{p} = 1$;
2. sample an *iid.* population with distribution $\mu^X(\cdot \mid g(\mathbf{X}) > q_i)$;
3. select a threshold q_{i+1} based on this population (the $1 - p_0$ empirical quantile of the population or the k^{th} ordered statistic for instance);
4. if $q_{i+1} > q$, set $q_{i+1} = q$;
5. $\widehat{p} = \widehat{p} \times \mathrm{P}\left[g(\mathbf{X}) > q_{i+1} \mid g(\mathbf{X}) > q_i\right]$, estimated with the current *iid.* population;
6. $i \leftarrow i + 1$;
7. if $q_i < q$, go to step 2.

In a metamodel based Subset Simulation, it is proposed to replace step 5 with a learning step. In this context, the *iid.* population of step 2 only serves the definition of the next threshold and the algorithm is modified as follows:

5a. run a metamodel-based algorithm on the intermediate failure event $\widetilde{F}_{i+1} = \{\mathbf{x} \in \widetilde{F}_i \mid g(\mathbf{x}) > q_{i+1}\}$;
5b. $\widehat{p} = \widehat{p} \times \widetilde{p}$ with $\widetilde{p}$ the output probability of the metamodel step.

Note that in these new steps and according to step 2 the input distribution is not μ^X but $\mu^X(\cdot \mid \mathbf{X} \in \widetilde{F}_i)$ with $\widetilde{F}_i$ the previous intermediate failure domain defined with the previous metamodel learnt on the previous threshold: Subset Simulation can be seen as a Sequential Monte Carlo sampler *modifying* iteratively the input distribution to output samples following the truncated distribution $\mu^X(\cdot \mid g(\mathbf{X}) > q)$.

Since the pioneering work of Deheeger [^{2}SMART, 2008], this strategy has been applied to the AKMCS method [AK-SS, Huang et al., 2016] and to the Bayesian approach of rare event simulation [Bayesian Subset Simulation, Li et al., 2012, Bect et al., 2016]. In this latter version, it is shown an improvement of several orders of magnitude against the ^{2}SMART method, and even very recent improved version of ^{2}SMART [Bourinet, 2016] does not seam to achieve the same performances.

To conclude this Chapter, we emphasize that our main goal was not to present an exhaustive list of all the metamodel-based algorithms used by practitioners to estimate a probability of failure. The interested reader is referred to [Dubourg, 2011] or `R` package `mistral` [Bousquet et al., 2015] for such a list. Beyond very practical parameters, which can indeed deeply change the behaviour of an algorithm, we think that it is important to have a general knowledge of what can be used under which hypothesis. Without underrating the real difficulties of practical implementations, we argue that it is important to consider algorithms which can be at least theoretically analysed and justified. To the best of our knowledge, the Bayesian framework is the more suitable for this specification and will be further considered in this thesis. In Chapter 5 we will show how the Poisson process framework can lead to an improvement of the Bayesian Subset Simulation similarly to what it did for the Subset Simulation.

Part II

Contribution to rare event simulation

Chapter 3

Point process for rare event simulation

3.1 Introduction

The estimation of extreme quantile or probability is a main concern in reliability analysis. Indeed one goal of uncertainty quantification is to estimate the probability of failure of a given system and inversely, quantile estimation helps defining guidelines to insure a *good* behaviour of the system with a given probability of failure. Usually, the system is considered as a *blackbox* (often a complex numerical code denoted by g in the previous chapters) which returns a real value defining its *state*. According to this output, it is then considered as working properly or not.

Formally, the problem can be written as follows: let $\mathbf{X}$ be a random finite- or infinite-dimensional vector with known distribution μ^X and g a *performance function* (the computer code for instance), one seeks for estimating p given q (or q given p) such that $p = \mathrm{P}\left[g(\mathbf{X}) > q\right]$. The main difficulties arise from the fact that 1) the sought probability or quantile is extreme, say $p < 10^{-6}$ and 2) the computer code is very time consuming.

In Chapter 1 we have presented statistical tools to estimate such quantity. Indeed, recall that the crude Monte Carlo estimator (see Section 1.1):

$$\widehat{p}_{\mathrm{MC}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{g(\mathbf{X}_i)>q} \tag{3.1}$$

with $(\mathbf{X}_i)_{i=1}^N$ N *iid* samples with distribution μ^X is not an option because $\mathrm{CV}\left[\widehat{p}_{\mathrm{MC}}\right]^2 \approx (Np)^{-1}$, which means that one would require $N = 10^2/p$ to get a coefficient of variation of 10%. To circumvent this limitation, the Splitting method (Section 1.3) rewrites the sought probability using the Bayes' rule and a finite sequence of increasing thresholds $(q_i)_{i=0}^m$, $m \geq 1$, such that $q_0 = -\infty$ and $q_m = q$:

$$p = \mathrm{P}\left[g(\mathbf{X}) > q_m \mid g(\mathbf{X}) > q_{m-1}\right] \times \cdots \times \mathrm{P}\left[g(\mathbf{X}) > q_2 \mid g(\mathbf{X}) > q_1\right] \mathrm{P}\left[g(\mathbf{X}) > q_1\right]. \tag{3.2}$$

From Eq. (3.2) the goal is then to estimate *independently* each conditional probability with a crude Monte Carlo estimator. The variance of the estimator depends on the choice of this sequence $(q_i)_{i=0}^m$ and especially it is known that the conditional probabilities should be all equal to minimize it (see Section 1.3.1, Eq. 1.10). A typical MS algorithm works as

follows:

1. sample a Monte-Carlo population $(\mathbf{X}_i)_{i=1}^N$ of size N; $j = 0$;
2. estimate the conditional probability $\mathrm{P}\left[g(\mathbf{X}) > q_{j+1} \mid g(\mathbf{X}) > q_j\right]$;
3. resample the $\mathbf{X}_i$ such that $g(\mathbf{X}_i) \leq q_{j+1}$ conditionally to be greater than q_{j+1} (the other ones do not change);
4. $j \leftarrow j + 1$ and repeat from step 2 until $j = m$.

The sequence $(q_j)_{j=1}^m$ is usually defined *on-the-fly* while the algorithm is running and this is known as Adaptive Splitting (see Section 1.3.2). The sequence is built either by fixing the conditional probabilities to be all equal to some given value $p_0 \in (0.1, 0.5)$ [Subset Simulation, Au and Beck, 2001], or by using the k^{th} order statistic [Adaptive Multilevel Splitting (AMS), Cérou and Guyader, 2007, Cérou et al., 2012]. In the first case, $(q_j)_{j=1}^m$ is then a sequence of quantiles estimated with crude Monte Carlo. Since the Monte Carlo estimator of a quantile has a bias of order $1/N$ (see Section 3.4.2) this bias is also found in the Subset Simulation estimator. In the second case the sequence is determined with a given statistic on the *iid.* population, precisely the k^{th} ordered statistic. However the conditional probability is still estimated *as usual*, *i.e.* writing $\sum_{i=1}^N \mathbb{1}_{g(\mathbf{X}_i)>q_j}$, which produces an unbiased estimator for any k [Bréhier et al., 2015c]. Note that if the *cdf* of Y is continuous, then $\mathrm{P}\left[g(\mathbf{X}_i) > q_j\right] = 1 - k/N$ and so at each iteration the conditional probability is simply estimated by $1 - k/N$.

The special case of the Last Particle Algorithm ($k = 1$) has gained a lot of attention recently. It has the smallest variance amongst all AMS (see Section 1.3.2); especially Guyader et al. [2011], Huber and Schott [2011] and Simonnet [2016] showed that the random number of iterations of the algorithm follows a Poisson law when the *cdf* of $g(\mathbf{X})$ is continuous. However, it is totally sequential and does not allow for parallel computation.

In this chapter, we define the point process framework for rare event simulation. Indeed, we focus on a random walk defined on the real-valued random variable $Y = g(\mathbf{X})$, *i.e.* we focus on the output Y instead of the input $\mathbf{X}$. We first address the case where the *cdf* of Y is continuous and find back the results from [Guyader et al., 2011, Huber and Schott, 2011, Simonnet, 2016], *i.e.* that the random walk is a Poisson process with mean measure depending on the distribution of Y (Section 3.2). The main difference is that we do not study a particular algorithm but rather derive general properties for this random walk. Especially we present in Section 3.3 the Last Particle Algorithm estimator as the Minimal Variance Unbiased Estimator (MVUE) of the exponential of a parameter of a Poisson law with *iid.* replicas of the Poisson random variable. As a consequence it turns the Last Particle Algorithm into the optimal (minimal variance) *parallel* Multilevel Splitting estimator [Walter, 2015a,c].

In Section 3.4 we review the quantile estimator proposed by Guyader et al. [2011] in the light of the point process framework and present a slightly modified version of

their estimator which improves its bias. Then in Section 3.5 we remove the continuity hypothesis of the *cdf* and derive three unbiased estimators for the probability of exceeding of threshold. One of them, the less robust, has also been proposed simultaneously by Bréhier et al. [2015a] in the Splitting framework. These results are illustrated on numerical examples in Section 3.6. All practical details on parallel implementations are postponed to Appendix A.

3.2 The increasing random walk

Let us consider $Y = g(\mathbf{X}) \in \mathbb{R}$ a real-valued random variable with distribution μ^Y where g is a deterministic function (for instance the output of a computer code) and $\mathbf{X}$ a random finite- or infinite-dimensional vector with known distribution μ^X. In this section, we assume that the *cdf* F_Y of Y is continuous. This hypothesis will be further removed in Section 3.5.

Definition 3.1 (Increasing random walk)**.** *Let Y be a real-valued random variable with continuous cdf F_Y, $Y_0 = -\infty$; we call the increasing random associated with Y the Markov sequence $(Y_n)_n$ such that:*

$$\forall n \in \mathbb{N},\ \mathrm{P}\left[Y_{n+1} \in A \mid Y_0, \cdots, Y_n\right] = \frac{\mathrm{P}\left[Y \in A \cap (Y_n, +\infty)\right]}{\mathrm{P}\left[Y \in (Y_n, +\infty)\right]}. \tag{3.3}$$

In other words $(Y_n)_n$ is an increasing sequence where each element is randomly generated conditionally greater than the previous one: $Y_{n+1} \sim \mu^Y(\cdot \mid Y > Y_n)$.

Remark 3.1. *When Y is continuous, the random walk can alternatively be defined with a non-strict inequality: $Y_{n+1} \sim \mu^Y(\cdot \mid Y \geq Y_n)$, i.e.:*

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

because $\forall y \in \mathbb{R}, \mathrm{P}\left[Y = y\right] = 0$. This continuity hypothesis will be further investigated and removed in Section 3.5.

Theorem 3.1. *The increasing random walk associated with Y a real-valued random variable with continuous cdf F_Y is indeed a Poisson process with mean measure:*

$$\forall y \in \mathbb{R},\ \lambda((-\infty, y]) = -\log \mathrm{P}\left[Y > y\right] = -\log\left(1 - \mu^Y((-\infty, y])\right). \tag{3.4}$$

Remark 3.2. *As stated by Kingman [1992, p. 12-13] there is no standard term for λ. In this manuscript we use their convention by naming the measure* mean measure *and its derivative against the Lebesgue measure, if it exists, the* intensity *or* rate *of the Poisson process. Especially a Poisson process with mean measure λ and parameter $N \geq 1$ is indeed a Poisson process with mean measure $N\lambda$.*

In particular, a homogeneous Poisson process is a Poisson process with constant rate 1 *and so mean measure:* $\forall y > 0,\ \lambda([0,y]) = \int_0^y \mathrm{d}t = y$.

Proof. Let $(Y_n)_{n\geq 0}$ be an increasing random walk. We consider the associated sequence $(T_n)_{n\geq 0}$ such that $\forall n \geq 0,\ T_n = -\log(\mathrm{P}\left[Y > Y_n\right])$. Especially, note that $T_0 = 0$ since $Y_0 = -\infty$.

The function $y \in \mathbb{R} \mapsto -\log \mathrm{P}\left[Y > y\right] \in \mathbb{R}_+$ is increasing over $\mathbb{R}$. Since the sequence $(Y_n)_{n\geq 0}$ is also increasing, so is the sequence $(T_n)_{n\geq 0}$. We now show that $(T_n)_{n\geq 1}$ is a homogeneous Poisson process with parameter 1, which means by definition that inter-arrival times are independent and follow an exponential law with parameter 1. Considering $n \in \mathbb{N}$ we have:

$$\begin{aligned} T_{n+1} - T_n &= -\log(\mathrm{P}\left[Y > Y_{n+1}\right]) + \log(\mathrm{P}\left[Y > Y_n\right]) \\ &= -\log\left(\frac{\mathrm{P}\left[Y > Y_{n+1}\right]}{\mathrm{P}\left[Y > Y_n\right]}\right) \\ &= -\log\left(\mathrm{P}\left[Y > Y_{n+1} \mid Y > Y_n\right]\right). \end{aligned}$$

Let $\mathcal{F}_n$ be the σ-algebra generated by $(T_j)_{j\leq n}$ and F_n be the *cdf* of the distribution $\mu^Y(\cdot \mid Y > Y_n)$. Knowing $\mathcal{F}_n$, F_n is the *cdf* of Y_{n+1} and thus $F_n(Y_{n+1})$ follows a uniform law on $[0,1]$. Finally we get:

$$\begin{aligned} \forall t \in \mathbb{R}_+, \mathrm{P}\left[T_{n+1} - T_n < t \mid \mathcal{F}_n\right] &= \mathrm{P}\left[-\log(1 - F_n(Y_{n+1})) < t \mid \mathcal{F}_n\right] \\ &= \mathrm{P}\left[F_n(Y_{n+1}) < 1 - \exp(-t) \mid \mathcal{F}_n\right] \\ &= 1 - \exp(-t). \end{aligned}$$

Thus the inter-arrival times are independent and follow an exponential law with parameter 1. $(T_n)_{n\geq 1}$ is then a homogeneous Poisson process with parameter 1. Let $y \in \mathbb{R}$ and M_y be the counting random variable of the number of events before y, one has:

$$M_y = \mathrm{card}\left\{n \geq 1 \mid Y_n \leq y\right\} = \mathrm{card}\left\{n \geq 1 \mid T_n \leq -\log \mathrm{P}\left[Y > y\right]\right\}. \tag{3.5}$$

Let $t_y = -\log \mathrm{P}\left[Y > y\right]$. Since $(T_n)_{n\geq 1}$ is a homogeneous Poisson process with parameter 1, its counting random variable a time $t_y > 0$ follows a Poisson law with parameter t_y. Then the equality of Eq. (3.5) lets conclude:

$$M_y \sim \mathcal{P}\left(t_y\right) = \mathcal{P}\left(-\log \mathrm{P}\left[Y > y\right]\right),$$

which means that $(Y_n)_{n\geq 1}$ is a Poisson process with mean measure λ defined on $\mathcal{B}(\mathbb{R})$ by:

$$\forall y \in \mathbb{R}, \lambda((-\infty, y]) = -\log \mathrm{P}\left[Y > y\right] = -\log\left(1 - \mu^Y((-\infty, y])\right).$$

□

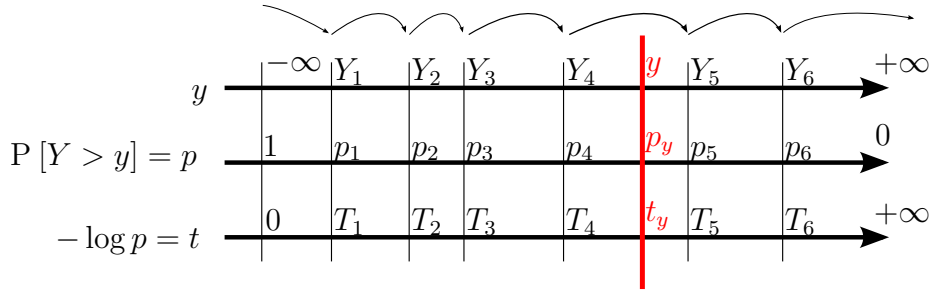


Figure 3.1: The increasing random walk and the associated homogeneous Poisson process.

Corollary 3.1. *The random number of simulations required to get a realisation of Y above a given threshold y is $1 + M_y$, with $M_y \sim \mathcal{P}(\log 1/\mathrm{P}[Y > y])$ the random number of events* before *reaching y. This is to be compared with a usual iid. sampling where it follows a geometric law with parameter $\mathrm{P}[Y > y]$. In expectation, it is $1 + \log 1/\mathrm{P}[Y > y]$ samples instead of $1/\mathrm{P}[Y > y]$. Since:*

$$\forall p \in (0, 1],\ 1 + \log \frac{1}{p} \leq \frac{1}{p}$$

the increasing random walk is on average always faster than the iid. sampling to get a realisation of Y above a given threshold. This can be understood in the sense that each new state of the random walk can be seen as a new trial to get a sample above the threshold, but with a greater probability of success.

Remark 3.3. *Simulating an increasing random walk requires to be able to generate according to conditional distributions. Section 1.3.3 gives practical details on how to achieve this when only a generator of the original random variable is available.*

This feature is especially interesting when y is far in the tail of Y; as a matter of fact Figure 3.2 plots the function $p \in (0, 1] \mapsto 1/p$ and $p \in (0, 1] \mapsto 1 - \log p$ as well as the *cdf* of the distributions $\mathcal{P}(-\log 10^{-2}) = \mathcal{P}(4.61)$ and $\mathcal{G}(10^{-2})$.

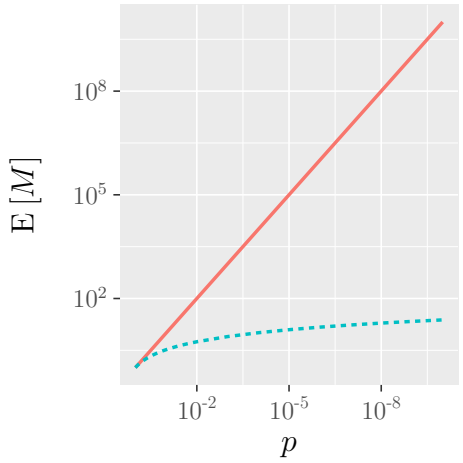
Corollary 3.2. *The renewal property of the Poisson process insures that:*

$$\forall y \in \mathbb{R},\ Y_{M_y+1} \sim \mu^Y(\cdot \mid Y > y)$$

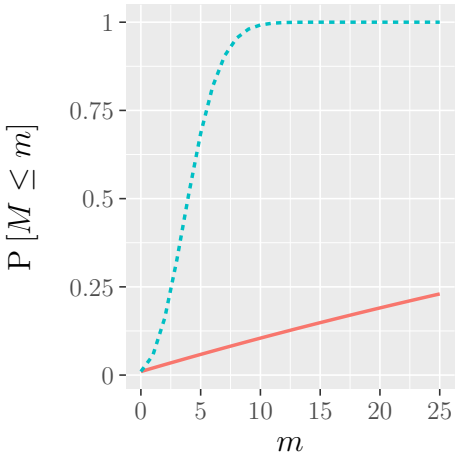
with M_y the counting random variable of the number of events before y.

This means that given a threshold q, simulating several independent random walks until they reach q produces an *iid.* population with distribution $\mu^Y(\cdot \mid Y > q)$. This property will be later used in Chapter 5 for SUR criteria estimation (see Section 2.2.3).

Remark 3.4. *Note also that generating the random walk until a given threshold y means indeed generating the random walk for all the thresholds $y_0 \leq y$, i.e. generating a realisation of all the counting random variables M_{y_0}, $y_0 \leq y$.*



(a) Means of the Poisson (dotted blue line) and the Geometric (solid red line) distribution against p.



(b) Cumulative distribution functions of $\mathcal{G}(10^{-2})$ (solid red line) and $\mathcal{P}(-\log 10^{-2}) + 1$ (dotted blue line).

Figure 3.2: Comparison of the random number of samples M required to get a realisation of a real-valued random variable Y with continuous *cdf* above of given threshold $y \in \mathbb{R}$ using an *iid.* sampling or simulating an increasing random walk.

3.3 Probability estimator

3.3.1 Minimal variance unbiased estimators

In the sequel, we then consider for all $y \in \mathbb{R}$ the associated numbers $p_y = \mathrm{P}[Y > y]$ and $t_y = -\log p_y$. Hence, for all $y \in \mathbb{R}$, the counting random variable of the number of events before y: $M_y = \mathrm{card}\{n \geq 1 \mid Y_n \leq y\}$ follows a Poisson law with parameter $t_y = -\log p_y$.

Lemma 3.1. *Let $y \in \mathbb{R}$ and $(M_y^i)_{i=1}^N$ be N iid. counting random variables at time y, the Minimal Variance Unbiased Estimator (MVUE) of t_y is:*

$$\widehat{t}_y = \frac{\sum_{i=1}^{N} M_y^i}{N} \tag{3.6}$$

and satisfies:

$$\mathrm{var}\left[\widehat{t}_y\right] = \frac{t_y}{N}.$$

Furthermore this estimator is fully efficient (it reaches the Cramér-Rao bound).

Proof. On the one hand $\sum_{i=1}^{N} M_y^i \sim \mathcal{P}\left(\sum_{i=1}^{N} t_y\right) = \mathcal{P}(Nt_y)$ because the sum of independent Poisson random variables is a Poisson random variable with parameter the sum of the parameters. Then one has:

$$\mathrm{E}\left[\widehat{t}_y\right] = \frac{Nt_y}{N} = t_y$$

and:

$$\mathrm{var}\left[\widehat{t}_y\right] = \frac{Nt_y}{N^2} = \frac{t_y}{N}.$$

On the other hand the log-likelihood $\log L(\mathbf{m}, t_y)$ of the N-sample $(M_y^i)_{i=1}^N$ depending on t_y writes (with $\mathbf{m} = (m_1, \cdots, m_N) \in \mathbb{N}^N$):

$$\log L(\mathbf{m}, t_y) = \log \left(\prod_{i=1}^{N} e^{-t_y} \frac{t_y^{m_i}}{m_i!} \right) = -N t_y + \log t_y \sum_{i=1}^{N} m_i - \sum_{i=1}^{N} \log (m_i!)$$

and so:

$$\frac{\partial^2 \log L(\mathbf{m}, t_y)}{\partial t_y^2} = -\frac{\sum_{i=1}^{N} m_i}{t_y^2}.$$

The Cramér-Rao bound for t_y is then:

$$\left[-\operatorname{E} \left[-\frac{\sum_{i=1}^{N} M_y^i}{t_y^2} \right] \right]^{-1} = \left[\frac{N t_y}{t_y^2} \right]^{-1} = \frac{t_y}{N}.$$

Hence $\widehat{t}_y$ achieves the Cramér-Rao bound and is the MVUE of t_y. □

However, $e^{-\widehat{t}_y}$ is not an unbiased estimator for $p_y = -\log t_y$:

$$\operatorname{E} \left[e^{-\widehat{t}_y} \right] = \operatorname{E} \left[(\exp -1/N)^{\sum_{i=1}^{N} M_y^i} \right] = e^{-(N t_y)(1 - \exp(-1/N))} = p_y^{N(1 - e^{-1/N})}$$

where we have used the moment generating function of a Poisson distribution to evaluate the expectation. This is an illustration of the fact that given a non-constant function $h : \mathbb{R} \to \mathbb{R}_+$ and real-valued unbiased estimators $(\widehat{t}_N)_{N \geq 1}$ of a given $t \in \mathbb{R}$, there is no algorithm yielding almost surely non-negative unbiased estimators of $h(t)$, as recently stated by Jacob et al. [2015].

To obtain the MVUE of p_y, we rely instead on the Lehmann-Scheffé theorem: given M a sufficient and complete statistic for a given parameter p_y and $\widehat{p}$ an unbiased estimator of p_y, it states that $\operatorname{E} [\widehat{p} \mid M]$ is the MVUE of p_y.

Proposition 3.1 (Poisson process estimator)**.** *Let $y \in \mathbb{R}$ and $(M_y^i)_{i=1}^N$ be N iid. counting random variables at time y, the Minimal Variance Unbiased Estimator (MVUE) of $p_y = e^{-t_y}$ is:*

$$\widehat{p}_y = \left(1 - \frac{1}{N} \right)^{\sum_{i=1}^{N} M_y^i}. \tag{3.7}$$

Proof. We consider the statistic $\bar{M}_y = \sum_{i=1}^{N} M_y^i$. Hence $\bar{M}_y \sim \mathcal{P}(N t_y)$. From the proof of Lemma 3.1 we know that it is sufficient. We then show that it is complete. Let $h : \mathbb{N} \to \mathbb{R}$ be a function, one has:

$$\operatorname{E} \left[h(\bar{M}_y) \right] = \sum_{m=0}^{\infty} h(m) e^{-N t_y} \frac{(N t_y)^m}{m!} = p_y^N \sum_{m=0}^{\infty} \frac{h(m)}{m!} N^m t_y^m.$$

Define $\forall m \geq 0$, $\alpha_m = N^m h(m)/m!$, one finds:

$$\forall p_y \in (0,1],\ \mathrm{E}\left[h(\bar{M}_y)\right] = 0 \Rightarrow \forall t_y \in \mathbb{R}_+,\ \sum_{m=0}^{\infty} \alpha_m (t_y)^m = 0.$$

Hence the power series $t \mapsto \sum_{m=0}^{\infty} \alpha_m t^m$ is identically null on $\mathbb{R}_+$ and so $\forall m \in \mathbb{N}, \alpha_m = 0$. This is equivalent to $\forall m \in \mathbb{N}$, $h(m) = 0$, which implies:

$$\forall p_y \in (0,1],\ \mathrm{P}\left[h(\bar{M}_y) = 0\right] = 1,$$

i.e. that the statistic is complete. Now consider $N \geq 2$ and $\widehat{p}_y^{\,1} = \mathbb{1}_{M_y^1 = 0}$ as an estimator of p_y. It is unbiased: $\mathrm{E}\left[\widehat{p}_y^{\,1}\right] = 1 \times \mathrm{P}\left[M_y^1 = 0\right] = p_y$. Then the Lehmann-Scheffé theorem insures that $\mathrm{E}\left[\widehat{p}_y^{\,1} \mid \bar{M}_y\right]$ is the MVUE of p_y: let $m \geq 0$,

$$\begin{aligned}
\mathrm{E}\left[\widehat{p}_y^{\,1} \mid \bar{M}_y = m\right] = \mathrm{P}\left[M_y^1 = 0 \mid \bar{M}_y = m\right] &= \frac{\mathrm{P}\left[M_y^1 = 0, \bar{M}_y = m\right]}{\mathrm{P}\left[\bar{M}_y = m\right]} \\
&= \frac{\mathrm{P}\left[M_y^1 = 0, \sum_{i=2}^{N} M_y^i = m\right]}{\mathrm{P}\left[\bar{M}_y = m\right]} = \frac{\mathrm{P}\left[\sum_{i=2}^{N} M_y^i = m\right]}{\mathrm{P}\left[\bar{M}_y = m\right]} \mathrm{P}\left[M_y^1 = 0\right] \\
&= p_y^{N-1} \frac{\left((N-1)\, t_y\right)^m}{m!} \frac{m!}{p_y^N (N t_y)^m} p_y \\
\mathrm{E}\left[\widehat{p}_y^{\,1} \mid \bar{M}_y = m\right] &= \left(1 - \frac{1}{N}\right)^m.
\end{aligned}$$

Hence $\widehat{p}_y = (1 - 1/N)^{\bar{M}_y}$ is the MVUE of p_y. □

Remark 3.5. *The LPA produces an estimator of the form:*

$$\widehat{p}_{LPA} = \left(1 - \frac{1}{N}\right)^M$$

with M the random number of iterations of the algorithm. Guyader et al. [2011] and Simonnet [2016] showed that in the case of a continuous cdf of $g(\mathbf{X})$, $M \sim \mathcal{P}(-N \log p_y)$. This means that the LPA is only a possible way to generate this MVUE. Especially, it is totally sequential. In Appendix A we will discuss different possible parallel implementations of this MVUE.

Proposition 3.2 (Statistical properties of the probability estimator)**.**

$$\widehat{p}_y \xrightarrow[N \to \infty]{a.s} p_y. \tag{3.8}$$

$$\mathrm{var}\left[\widehat{p}_y\right] = p_y^2 \left(p_y^{-1/N} - 1\right). \tag{3.9}$$

Proof. The almost sure convergence comes from the fact that $\bar{M}_y/N$ almost surely converges toward $-\log p_y$ thanks to the Strong Law of Large Numbers. On the other hand the

expression of the variance is calculated with the moment generating function of a Poisson random variable. □

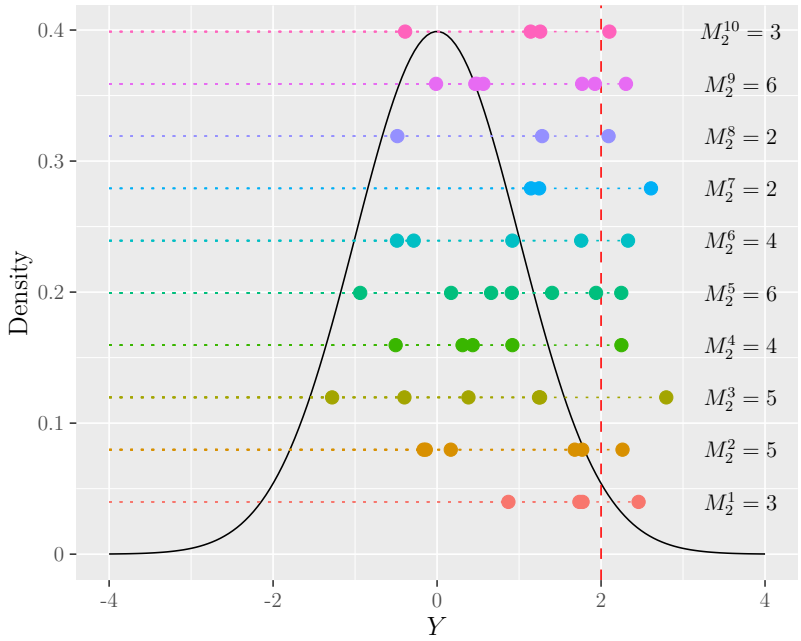


Figure 3.3: $N = 10$ simulations of a random walk associated with a standard normal Gaussian random variable. The statistic $\bar{M}_2 = 40$ and so the estimated probability is $\widehat{p}_2 = (1 - 1/10)^{40} \approx 1.48 \times 10^{-2}$ with CV $[\widehat{p}_y] \approx 0.43$. Animated figure online.

Corollary 3.3. *Remark 3.4 and Proposition 3.2 let have a Glivenko-Cantelli like result:*

$$\sup_{y_0 \leq y} |F_N(y_0) - F_Y(y_0)| \xrightarrow[N\to\infty]{a.s.} 0 \tag{3.10}$$

with $F_N(y_0) = 1 - \left(1 - \frac{1}{N}\right)^{\bar{M}_{y_0}}$ *and* $\bar{M}_{y_0}$ *the sum of* N *iid. counting random variables at state* y_0.

Proof. The proof relies on the same argument as the one of the Glivenko-Cantelli theorem. Let us consider that the increasing random walks have been generated until a given $y \in \mathbb{R}$ such that $1 - F_Y(y) = \mathrm{P}[Y > y] = p > 0$. We show that:

$$\sup_{y_0 \leq y} |F_N(y_0) - F_Y(y_0)| \xrightarrow[N\to\infty]{a.s} 0.$$

Let $m \geq 1$ and define the sequence $(y_{i,m})_{i=1}^m$ such that $\forall i \in [\![1, m]\!], y_{i,m} = G_Y(\frac{i}{m}(1-p))$ with G_Y the generalised inverse of F_Y, defined on $(0, 1)$ by:

$$G_Y(u) = \inf\{y \in \mathbb{R} \mid F_Y(y) \geq u\}.$$

We set $y_{0,m} = y_L \in \bar{\mathbb{R}}$ the left endpoint of F_Y such that $F_Y(y_{0,m}) = 0$ and $\mathrm{P}[F_N(y_{0,m}) = 0] = 1$. Since F_Y is supposed to be continuous, one also has: $\forall i \in [\![1, m]\!]$, $F_Y(y_{i,m}) = (1-p)i/m$.

For all $y_0 \in (y_L, y]$, there exists $i \in [\![0, m-1]\!]$ such that $y_{i,m} < y_0 \leq y_{i+1,m}$. Let $(\Omega, \mathcal{F}, \mathrm{P})$ be the underlying probability space and $\omega \in \Omega$. Since the functions $F_N(\cdot, \omega)$ and F_Y are increasing, one has:

$$\begin{aligned} F_N(y_{i,m}, \omega) \leq F_N(y_0, \omega) &\leq F_N(y_{i+1,m}, \omega) \\ F_Y(y_{i,m}) \leq F_Y(y_0) \quad &\leq F_Y(y_{i+1,m}) \end{aligned}$$

which gives the following bounds:

$$\begin{aligned} F_N(y_{i,m}, \omega) - F_Y(y_{i+1,m}) \leq F_N(y_0, \omega) - F_Y(y_0) &\leq F_N(y_{i+1,m}, \omega) - F_Y(y_{i,m}) \\ F_N(y_{i,m}, \omega) - F_Y(y_{i,m}) - \frac{1-p}{m} \leq F_N(y_0, \omega) - F_Y(y_0) &\leq F_N(y_{i+1,m}, \omega) - F_Y(y_{i+1,m}) + \frac{1-p}{m}. \end{aligned}$$

Eventually this means:

$$\sup_{y_0 \leq y} |F_N(y_0, \omega) - F_Y(y_0)| \leq \sup_{1 \leq i \leq m} |F_N(y_{i,m}, \omega) - F_Y(y_{i,m})| + \frac{1-p}{m}.$$

Let: $\forall y_0 \leq y$, $A_{y_0} = \left\{ \omega \in \Omega \mid \lim_N F_N(y_0, \omega) = F_Y(y_0) \right\}$ and $\Omega_m = \cap_{i=1}^m A_{y_{i,m}}$. For all $\omega \in \Omega_m$:

$$\limsup_{N \to \infty} \sup_{y_0 \leq y} |F_N(y, \omega) - F_Y(y)| \leq \frac{1-p}{m}.$$

The almost sure convergence of the estimator gives that the sets $(A_{y_{i,m}})_{i=1}^m$ are of probability 1 and so is Ω_m as a finite intersection of such sets. Finally $\bar{\Omega} = \cap_{m \geq 1} \Omega_m$ is of probability 1 as a countable intersection of such sets and:

$$\forall \omega \in \bar{\Omega}, \limsup_{N \to \infty} \sup_{y_0 \leq y} |F_N(y_0, \omega) - F_Y(y_0)| \leq \inf_{m \geq 1} \frac{1-p}{m} = 0$$

which concludes the proof. □

In other words, generating an estimator of p_y for a given $y \in \mathbb{R}$ means indeed generating an estimator of the whole *cdf* of Y until this threshold y, similarly to the empirial *cdf* built from an *iid.* Monte Carlo sampling.

3.3.2 Efficiency of the estimator

Thanks to Lehmann-Scheffé theorem, we have been able to derive the MVUE of p_y. However it does not achieve the Cramér-Rao bound: with similar calculations as in the proof of Lemma 3.1 but making the derivative of the log-likelihood against p_y instead of t_y, we find that the bound is: $-p_y^2 \log p_y / N$. This bound is at least asymptotically achieved for sufficiently large N:

$$\mathrm{var}\left[\widehat{p}_y\right] = p_y^2 \left(p_y^{-1/N} - 1 \right) = \frac{-p_y^2 \log p_y}{N} + o\left(\frac{1}{N}\right). \tag{3.11}$$

We also recall the specific criteria defined in Section 1.5 for rare event estimators [Rubino et al., 2009]. An estimator is said to have a Bounded Relative Moment of order $k \geq 1$ (BRM_k) if:

$$\limsup_{p_y \to 0} \frac{\mathrm{E}\left[\widehat{p}_y^k\right]}{p_y^k} < \infty \tag{3.12}$$

and a Logarithmic Efficiency of order $k \geq 1$ (LE_k) if:

$$\frac{\log \mathrm{E}\left[\widehat{p}_y^k\right]}{k \log p_y} \xrightarrow[p_y \to 0]{} 1. \tag{3.13}$$

Let $k \geq 1$, we have:

$$\frac{\mathrm{E}\left[\widehat{p}_y^k\right]}{p_y^k} = p_y^{N\left(1-(1-1/N)^k\right)-k} \tag{3.14}$$

$$\frac{\log \mathrm{E}\left[\widehat{p}_y^k\right]}{k \log p_y} = \frac{N}{k}\left[1 - \left(1 - \frac{1}{N}\right)^k\right] = 1 - \frac{k-1}{2N} + o\left(\frac{1}{N}\right). \tag{3.15}$$

Hence for any $k \geq 2$, Eq. (3.12) is not satisfied because $N(1-(1/N)^k) - k < 0$. On the other hand Eq. (3.13) is asymptotically achieved (large N) for any $k \geq 1$ with Eq. (3.15) not depending on p_y.

3.3.3 Confidence intervals

The distribution of the discrete random variable $\widehat{p}_y$ is fully determined through a Poisson distribution with parameter $-N \log p_y$. Furthermore the Poisson distribution is known to be well approximated with a Gaussian random variable as soon as its parameter is greater than 5 to 10. Hence even for small N we fall into the range of validity of this approximation: for instance considering $N \geq 10$ and $p \leq 10^{-1}$ leads to $-N \log p \geq 23$.

This means that $\widehat{p}_y$ approximately follows a log-normal distribution:

$$\log \widehat{p}_y \sim \mathcal{N}(\mu, \sigma^2) \text{ with } \begin{cases} \mu = -N \log p_y \log\left(1 - \dfrac{1}{N}\right) & = \log p_y + O\left(\dfrac{1}{N}\right) \\ \sigma^2 = -N \log p_y \left(\log\left(1 - \dfrac{1}{N}\right)\right)^2 & = \dfrac{-\log p_y}{N} + O\left(\dfrac{1}{N^2}\right) \end{cases} \tag{3.16}$$

so that one can build approximate confidence intervals based on the standard Gaussian distribution.

Proposition 3.3. *Given $\alpha \in (0,1)$ and $Z_{1-\alpha/2}$ the quantile of order $1-\alpha/2$ of the standard normal distribution:* $\mathrm{P}\left[-Z_{1-\alpha/2} < \mathcal{N}(0,1) < Z_{1-\alpha/2}\right] = 1-\alpha$, *one has:*

$$\liminf_{N \to \infty} \mathrm{P}\left[\sqrt{N}|p_y - \widehat{p}_y| < Z_{1-\alpha/2}\widehat{p}_y\sqrt{-\log \widehat{p}_y}\right] \geq 1 - \alpha.$$

Proof. Since $\bar{M}$ is the sum of *iid.* Poisson random variables, the Central Limit Theorem gives:

$$\sqrt{\frac{N}{-\log p_y}}\left(\frac{\bar{M}}{N} - (-\log p_y)\right) \xrightarrow[N\to\infty]{\mathcal{L}} \mathcal{N}(0,1).$$

Let $\widehat{t}_y = -\log \widehat{p}_y = -\bar{M}\log(1-1/N)$ and $t_y = -\log p_y$, the above equation rewrites:

$$\sqrt{\frac{N}{-\log p_y}}\left(\frac{\widehat{t}_y}{-N\log(1-1/N)} - t_y\right) \xrightarrow[N\to\infty]{\mathcal{L}} \mathcal{N}(0,1)$$
$$\sqrt{\frac{N}{-\log p_y}}\left(\widehat{t}_y - t_y - \widehat{t}_y\left(1 - \frac{1}{-N\log(1-1/N)}\right)\right) \xrightarrow[N\to\infty]{\mathcal{L}} \mathcal{N}(0,1).$$

On the one hand $1 - \dfrac{1}{-N\log(1-1/N)} = 1/(2N) + o(1/N)$ and $\widehat{t}_y$ converges almost surely to t_y. This gives:

$$\sqrt{\frac{N}{-\log p_y}}\widehat{t}_y\left(1 - \frac{1}{-N\log(1-1/N)}\right) \xrightarrow[N\to\infty]{a.s.} 0.$$

Then Slutsky's theorem gives that:

$$\sqrt{\frac{N}{-\log p_y}}\left(\widehat{t}_y - t_y\right) \xrightarrow[N\to\infty]{\mathcal{L}} \mathcal{N}(0,1).$$

Now recall that $\log \widehat{p}_y$ converges almost surely toward $\log p_y$, Slutsky's theorem eventually gives:

$$\sqrt{\frac{N}{-\log \widehat{p}_y}}\left(\widehat{t}_y - t_y\right) \xrightarrow[N\to\infty]{\mathcal{L}} \mathcal{N}(0,1).$$

Denote by $Z_{1-\alpha/2}$ the quantile of order $1-\alpha/2$ of the standard normal distribution, one gets:

$$\mathrm{P}\left[\frac{|\widehat{t}_y - t_y|}{\sqrt{-\log \widehat{p}_y}}\sqrt{N} < Z_{1-\alpha/2}\right] \xrightarrow[N\to\infty]{} 1-\alpha$$
$$\mathrm{P}\left[\exp\left(-Z_{1-\alpha/2}\sqrt{\frac{-\log \widehat{p}_y}{N}}\right) < \frac{p_y}{\widehat{p}_y} < \exp\left(Z_{1-\alpha/2}\sqrt{\frac{-\log \widehat{p}_y}{N}}\right)\right] \xrightarrow[N\to\infty]{} 1-\alpha.$$

Finally the asymptotic expansion of the exponential lets conclude the proof. □

In this section we developed the *point process framework* for rare event simulation. In this framework, the Last Particle Algorithm [Guyader et al., 2011, Simonnet, 2016] appears only as a particular sequential implementation of the MVUE defined in Eq. (3.7), it is the MVUE of the exponential of a Poisson parameter with *iid.* replicas of such Poisson random variables. This more general *iid. representation* paves the way for parallel implementation of the estimator (see Appendix A). Since the LPA estimator is optimal (minimal variance against number of generated samples) amongst all Adaptive Splitting

estimators (see Section 1.3.2), this means that we have enabled the parallel implementation of the optimal Multilevel Splitting algorithm.

Furthermore the point process framework lets us show a Glivenko-Cantelli like theorem: like with a crude Monte Carlo estimator, one not only produces an estimator of the sought probability but of the whole *cdf* of $Y = g(\mathbf{X})$. In other words, one could say that the optimal splitting *does not make any splitting anymore*. The Poisson process framework appears as a true counterpart of the crude Monte Carlo estimator with a similar behaviour but a coefficient $\log 1/p_y$ instead of $1/p_y$ in all the considered formulas. Table 3.1 summarises these properties.

	crude Monte Carlo	Poisson process
Sampling: N *iid.* replicas of	$\mathbf{X} \sim \mu^X$	$M_y \sim \mathcal{P}(-\log p_y)$
Statistic	$\bar{M} = \sum_{i=1}^{N} \mathbb{1}_{g(\mathbf{X}_i)>y}$	$\bar{M} = \sum_{i=1}^{N} M_y^i$
Estimator	$\bar{M}/N$	$(1-1/N)^{\bar{M}}$
Mean	p_y	p_y
Variance σ^2	$\frac{p_y(1-p_y)}{N}$	$p_y^2(p_y^{-1/N}-1)$
Asymptotic variance $\lim_{N\to\infty} \sigma^2 N$	$p_y(1-p_y)$	$-p_y^2 \log p_y$
Squared coef. of variation	$\frac{1-p_y}{Np_y} + o\left(\frac{1}{N}\right)$	$\frac{1}{N}\log\frac{1}{p_y} + o\left(\frac{1}{N}\right)$
Width of conf. interval at $(1-\alpha)\%$	$2Z_{1-\alpha/2}\,\sigma$	$2Z_{1-\alpha/2}\,\sigma$

Table 3.1: Summary of the properties of the crude Monte Carlo and the Poisson process estimators.

In the next section, we go one step further with the development of this similarity by defining a quantile estimator based on the increasing random walk. Guyader et al. [2011] indeed already proposed a quantile estimator based on the Last Particle Algorithm. Apart from its parallelisation, we will show that the Poisson process framework also slightly modifies this estimator, resulting in simplified bounds on the bias.

3.4 Quantile estimator

3.4.1 Description of the estimator

Recall that we were interested in estimating $p = \mathrm{P}[g(\mathbf{X}) > q]$ for a given $q \in \mathbb{R}$ with $g(\mathbf{X})$ a real-valued random variable with continuous *cdf*, we now assume that one instead wants to estimate q for a given p:

$$q = \inf\{y \in \mathbb{R} \mid \mathrm{P}[g(\mathbf{X}) > y] \leq p\}.$$

Let us first consider $(T_m)_{m\geq 0}$ a homogeneous Poisson process with parameter $N \geq 1$ ($T_0 = 0$) and denote by $\bar{M}_q$ the counting random variables at time $t = -\log p > 0$. We have $\bar{M}_q \sim \mathcal{P}(-N \log p) = \mathcal{P}(Nt)$.

Lemma 3.2 (Laws of random variables $T_{\bar{M}_q}$ and $T_{\bar{M}_q+1}$)**.** *Let us denote by $F_{\bar{M}_q}$ and $F_{\bar{M}_q+1}$ the cdf associated to $T_{\bar{M}_q}$ and $T_{\bar{M}_q+1}$ respectively. The following result holds:*

$$\forall(\alpha,\beta) \in \mathbb{R}_+^2, \mathrm{P}\left[(T_{\bar{M}_q+1} - t > \alpha/N) \cap (t - T_{\bar{M}_q} \geq \beta/N)\right] = e^{-\alpha}e^{-\beta}\,\mathbb{1}_{[0,Nt)}(\beta). \quad (3.17)$$

Proof. Given $(\alpha,\beta) \in \mathbb{R}_+^2$, we write $\forall k \in \mathbb{N}, \Delta_k = \{(T_{k+1} - t > \alpha/N) \cap (t - T_k \geq \beta/N)\}$. We have:

$$\mathrm{P}\left[\Delta_{\bar{M}_q}\right] = \sum_{k=0}^{\infty} \mathrm{P}\left[\Delta_{\bar{M}_q} \cap \{\bar{M}_q = k\}\right].$$

Noticing here that $\{\bar{M}_q = k\} = \{T_k \leq t < T_{k+1}\}$ we have:

$$\mathrm{P}\left[\Delta_{\bar{M}_q}\right] = \sum_{k=0}^{\infty} \mathrm{P}\left[\Delta_k\right].$$

From standard results on Poisson processes, we have the joint density of (T_k, T_{k+1}):

$$\forall k \in \mathbb{N}^*, f(t_k, t_{k+1}) = N^{k+1}e^{-Nt_{k+1}}t_k^{k-1}/(k-1)!\,\mathbb{1}_{0<t_k<t_{k+1}}.$$

Then we get:

$$\forall k \in \mathbb{N}, \mathrm{P}\left[\Delta_k\right] = \frac{(N(t-\beta/N))^k}{k!}e^{-Nt-\alpha}\,\mathbb{1}_{[0,t)}\left(\frac{\beta}{N}\right)\mathbb{1}_{[0,+\infty)}(\alpha)$$

and so the result announced:

$$\mathrm{P}\left[(T_{\bar{M}_q+1} - t > \alpha/N) \cap (t - T_{\bar{M}_q} \geq \beta/N)\right] = e^{-\alpha}\mathbb{1}_{[0;+\infty)}(\alpha)\,e^{-\beta}\mathbb{1}_{[0;Nt)}(\beta).$$

□

Note that, for any N and t, the distribution of $(NT_{\bar{M}_q}, NT_{\bar{M}_{q+1}})$ depends only of Nt.

Corollary 3.4. *The center of the interval $[T_{\bar{M}_q}; T_{\bar{M}_q+1}]$ converges toward a random variable centred in t with symmetric pdf, i.e.:*

$$N\left(\frac{T_{\bar{M}_q} + T_{\bar{M}_q+1}}{2} - t\right) \xrightarrow[Nt\to\infty]{\mathcal{L}} Z \quad (3.18)$$

with Z a random variable with pdf $f_Z(z) = e^{-2|z|}$.

Proof. From the joint probability distribution of $T_{\bar{M}_q}$ and $T_{\bar{M}_q+1}$ we have:

$$\forall(\alpha,\beta)\in\mathbb{R}_+^2, \mathrm{P}\left[(T_{\bar{M}_q+1}-t>\alpha/N)\cap(t-T_{\bar{M}_q}\geq\beta/N)\right]\xrightarrow[Nt\to\infty]{\mathcal{L}} e^{-\alpha}\,e^{-\beta}$$

i.e.:

$$N(T_{\bar{M}_q+1}-t, t-T_{\bar{M}_q})\xrightarrow[Nt\to\infty]{} (Z_1,Z_2)$$

with Z_1 and Z_2 *iid.* random variables with Exponential distribution $\mathcal{E}(1)$. The difference between the center of the interval and t is thus the difference between two Exponential random variables with parameter 2, which gives the result. □

Lemma 3.2 and Corollary 3.4 show that a homogeneous Poisson process is such that for any given $t>0$, the first events before and after t form a random interval centred at t. Going back to the increasing random walk, it suggests to define the following quantity of interest:

$$\widetilde{q}=\frac{Y_{\bar{M}_q}+Y_{\bar{M}_q+1}}{2}$$

with $(Y_m)_m$ the superposed process of N *iid.* random walks, *i.e.* a Poisson process with mean measure: $\forall y\in\mathbb{R}$, $\lambda((-\infty,y])=-N\log\mathrm{P}\left[Y>y\right]$. In other words $(Y_m)_m$ is such that $\forall m\geq 0, T_m=-\log\mathrm{P}\left[Y>Y_m\right]$: it is the merged and sorted sequence of the events of N *iid.* increasing random walks.

Denote by $\bar{F_Y}^{-1}$ the complementary quantile function of $Y=g(\mathbf{X})$, *i.e.* the generalised inverse of its complementary *cdf* $\bar{F_Y}$ [see for example Embrechts et al., 1997, Resnick, 2013]:

$$\bar{F_Y}^{-1}(p)=\inf\{y\in\mathbb{R}\mid \mathrm{P}\left[Y>y\right]\leq p\}$$

one has: $\forall m\geq 0$, $Y_m=\bar{F_Y}^{-1}(e^{-T_m})$. By assuming that $Y=g(\mathbf{X})$ has a *pdf* f_Y continuous and strictly positive at q, we can make the following Taylor expansion around $t=-\log p$:

$$\begin{aligned}Y_{\bar{M}_q}&=q+\left(T_{\bar{M}_q}-t\right)\frac{p}{f_Y(q)}+o_P\left(T_{\bar{M}_q}-t\right)\\ Y_{\bar{M}_q+1}&=q+\left(T_{\bar{M}_q+1}-t\right)\frac{p}{f_Y(q)}+o_P\left(T_{\bar{M}_q+1}-t\right).\end{aligned}$$

Corollary 3.4 lets have the convergence in distribution of $\widetilde{q}$:

$$N(\widetilde{q}-q)\xrightarrow[N\to\infty]{\mathcal{L}}\frac{p}{f_Y(q)}Z,$$

where Z is a random variable with *pdf* $f_Z(z)=e^{-2|z|}$. Unfortunately we cannot observe $\bar{M}_q$: in general there is no information on when the estimated quantile is *crossed* by the random walk. However we know that $\bar{M}_q\overset{\mathcal{L}}{\sim}\mathcal{P}(-N\log p)$; writing $m_q=\lfloor -N\log p\rfloor$, we then choose as an estimator for q:

$$\widehat{q}=\frac{Y_{m_q}+Y_{m_q+1}}{2}. \tag{3.19}$$

3.4.2 Statistical analysis of the estimator

Let $N \geq 1$ be the total number of random walks and $m_q = \lfloor -N \log p \rfloor$. We first present some asymptotic results for an estimator Y_{m_q+k} with a given $k \in \mathbb{Z}$ as $N \to +\infty$. Then we study the property of a linear combination of Y_{m_q+k} around m_q. From now on, let us suppose that $g(\mathbf{X})$ has *cdf* F_Y and *pdf* f_Y continuous at q.

Proposition 3.4 (Central Limit Theorem)**.** *If $f_Y(q) \neq 0$, then:*

$$\sqrt{N}\left(Y_{m_q+k} - q\right) \xrightarrow[N\to\infty]{\mathcal{L}} \mathcal{N}\left(0, \frac{-p^2 \log p}{f_Y(q)^2}\right). \tag{3.20}$$

Proof. This proof comes mainly from [Guyader et al., 2011]. There exists $N_0 \in \mathbb{N} \mid \forall N \geq N_0$, $m_q + k > 0$. Let $N \geq N_0$, we have $Y_{m_q+k} = \bar{F_Y}^{-1}(e^{T_{m_q+k}})$ where $T_{m_q+k} \sim \Gamma_{m_q+k}/N$ with Γ_{m_q+k} a Gamma random variable with parameter $m_q + k$. From the definition of $m_q = \lfloor -N \log p \rfloor$ we get:

$$\frac{m_q + k}{N} \xrightarrow[N\to\infty]{} -\log p$$

which lets us rewrite the Central Limit Theorem for the Gamma random variable as follows:

$$\sqrt{N}\,\frac{T_{m_q+k} - (-\log p)}{\sqrt{-\log p}} \xrightarrow[N\to\infty]{\mathcal{L}} \mathcal{N}(0,1).$$

One eventually concludes by making a Taylor expansion of $y \in \mathbb{R} \mapsto -\log \mathrm{P}\left[Y > y\right]$ around q:

$$T_{m_q+k} - (-\log p) = (Y_{m_q+k} - q) f_Y(q)/p + o_P(Y_{m_q+k} - q).$$

Then:

$$\sqrt{N}(Y_{m_q+k} - q) \xrightarrow[N\to\infty]{\mathcal{L}} \mathcal{N}\left(0, \frac{-p^2 \log p}{f_Y(q)^2}\right).$$

□

Proposition 3.5 (Bounds on bias)**.** *If f_Y continuously differentiable and positive at q, we get the following boundaries for the bias:*

$$\begin{aligned}
\mathrm{E}\left[Y_{m_q+k}\right] - q &\geq \frac{p}{N f_Y(q)}\left(\frac{\log p}{2}\left(1 + \frac{f_Y'(q)p}{f_Y(q)^2}\right) + k - 1\right) + o\left(\frac{1}{N}\right) \\
\mathrm{E}\left[Y_{m_q+k}\right] - q &\leq \frac{p}{N f_Y(q)}\left(\frac{\log p}{2}\left(1 + \frac{f_Y'(q)p}{f_Y(q)^2}\right) + k\right) + o\left(\frac{1}{N}\right).
\end{aligned} \tag{3.21}$$

Proof. Let us write the Taylor expansion of Y_{m_q+k} around $t = -\log p = -\log \bar{F_Y}(q)$:

$$\begin{aligned}
Y_{m_q+k} = \bar{F_Y}^{-1}(e^{-T_{m_q+k}}) = q &+ \frac{p}{f_Y(q)}(T_{m_q+k} - t) \\
&+ \frac{(T_{m_q+k} - t)^2}{2}\left(-\frac{p}{f_Y(q)}\right)\left(1 + \frac{p f_Y'(q)}{f_Y(q)^2}\right) + o_P\left((T_{m_q+k} - t)^2\right).
\end{aligned}$$

We have:

$$\mathrm{E}\left[T_{m_q+k} - t\right] = \frac{m_q + k}{N} - t = \frac{1}{N}(m_q - Nt + k) \in \left(\frac{k-1}{N}, \frac{k}{N}\right]$$

and:

$$\mathrm{E}\left[(T_{m_q+k} - t)^2\right] = \mathrm{var}\left[T_{m_q+k}\right] + (\frac{m_q + k}{N} - t)^2 = \frac{m_q}{N^2} + \frac{k}{N^2} + \frac{1}{N^2}(m_q - Nt + k)^2$$
$$\mathrm{E}\left[(T_{m_q+k} - t)^2\right] \in \frac{t}{N} + \frac{k}{N^2}\left(k-1, k+1\right],$$

which concludes the proof. □

Proposition 3.6 (Confidence interval)**.** *Writing $Z_{1-\alpha/2}$ the $(1-\alpha/2)$ quantile of a standard Gaussian distribution, $m_- = \lfloor m_q - Z_{1-\alpha/2}\sqrt{m_q} \rfloor$ and $m_+ = \lceil m_q + Z_{1-\alpha/2}\sqrt{m_q} \rceil$, we have:*

$$\mathrm{P}\left[q \in [Y_{m_-}, Y_{m_+}]\right] \xrightarrow[N\to\infty]{} 1 - \alpha. \tag{3.22}$$

Proof. Considering the approximation of a Poisson distribution by a Normal distribution: $M_q \sim \mathcal{N}(m_q, m_q)$ and writing $Z_{1-\alpha/2}$ the $1 - \alpha/2$ quantile of a standard Gaussian law:

$$\mathrm{P}\left[M_q \in [m_q - Z_{1-\alpha/2}\sqrt{m_q}, m_q + Z_{1-\alpha/2}\sqrt{m_q}]\right] = 1 - \alpha.$$

We conclude by noticing that the sequence $(q_i)_i$ is strictly increasing. □

Hence it is possible to produce a confidence interval for q without estimating the *pdf* of $g(\mathbf{X})$ in contrast with a crude Monte Carlo estimation.

Proposition 3.7 (Multidimensional Central Limit Theorem)**.** *Let us now consider the vector $(Y_{m_q}, Y_{m_q+1}, \ldots, Y_{m_q+k})$ with $k \in \mathbb{Z} \mid k \geq -m_q + 1$. If $f_Y(q) \neq 0$, we can write the following multidimensional central limit theorem:*

$$\sqrt{N}\left[\begin{pmatrix} Y_{m_t} \\ \vdots \\ Y_{m_t+k} \end{pmatrix} - q\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}\right] \xrightarrow[N\to\infty]{\mathcal{L}} \mathcal{N}\left(\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, \frac{-p^2 \log p}{f_Y(q)^2}\begin{pmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \ldots & 1 \end{pmatrix}\right). \tag{3.23}$$

Proof. The $(T_{m_q+k})_k$ are the times of a homogeneous Poisson process with parameter N so that we know their joint probability distribution and we get:

$$\forall k \in \mathbb{Z} \mid m_q + k > 0, \mathrm{cov}\left[T_{m_q}, T_{m_q+k}\right] = \frac{m_q}{N^2}.$$

Let us now define $\phi : t \mapsto= \bar{F_Y}^{-1}(e^{-t})$ and calculate the covariance matrix between the $(Y_{m_t+k})_k$. Since $Y_{m_q+k} = q + (T_{m_q+k} - t)\phi'(t) + o(T_{m_q+k} - t)$ and $\phi'(t) = p/f_Y(q)$, we find:

$$\mathrm{cov}\left[Y_{m_q}, Y_{m_q+k}\right] = \left(\frac{p}{f_Y(q)}\right)^2 \frac{m_q}{N^2} + o\left(\frac{1}{N}\right)$$

from which it comes:

$$N \operatorname{cov}\left[Y_{m_q}, Y_{m_q+k}\right] \xrightarrow[N\to\infty]{} \frac{-p^2 \log p}{f_Y(q)^2}$$

which concludes the proof. □

Proposition 3.8. *Assuming that f_Y is twice continuously differentiable and $f_Y(q) \neq 0$, the estimator $\widehat{q} = \dfrac{Y_{m_q} + Y_{m_q+1}}{2}$ introduced in Eq. (3.19) has the following properties:*

$$\sqrt{N}\left(\widehat{q} - q\right) \xrightarrow[N\to\infty]{\mathcal{L}} \mathcal{N}\left(0, \frac{-p^2 \log p}{f_Y(q)^2}\right) \tag{3.24}$$

and:

$$\begin{aligned} \mathrm{E}\left[\widehat{q}\right] - q &\geq \frac{p}{2N f_Y(q)}\left(\log p\left(1 + \frac{f_Y'(q)p}{f_Y(q)^2}\right) - 1\right) + o\left(\frac{1}{N}\right) \\ \mathrm{E}\left[\widehat{q}\right] - q &\leq \frac{p}{2N f_Y(q)}\left(\log p\left(1 + \frac{f_Y'(q)p}{f_Y(q)^2}\right) + 1\right) + o\left(\frac{1}{N}\right). \end{aligned} \tag{3.25}$$

Remark 3.6. *For von Mises distributions [see Embrechts et al., 1997, Section 3.3.3] one has:*

$$\frac{f_Y'(q)p}{f_Y(q)^2} \xrightarrow[q\to\infty]{} -1.$$

For instance for Weibull distributions, i.e. distributions with cdf $F_Y(y) = e^{-\lambda y^k}$, $y \geq 0$, $\lambda > 0$ and $k > 0$, one has

$$\log p\left(1 + \frac{f_Y'(q)p}{f_Y(q)^2}\right) = 1 - \frac{1}{k}$$

so that the bounds on bias simplify as follow:

$$-\frac{1}{2k}\frac{p}{N f_Y(q)} \leq \mathrm{E}\left[\widehat{q}\right] - q \leq \left(1 - \frac{1}{2k}\right)\frac{p}{N f_Y(q)}. \tag{3.26}$$

Eventually, it is centred for $k = 1$.

Remark 3.7. *Guyader et al. [2011] defined the quantile estimator by* inverting *Eq. (3.7), i.e.:*

$$\widehat{q}_{LPA} = Y_{m_{LPA}} \tag{3.27}$$

with $m_{LPA} = \lceil \log p / \log\left(1 - 1/N\right) \rceil$. While this does not change the convergence in distribution, its modifies the bounds on the bias. As a matter of fact, with the same assumptions he got:

$$\begin{aligned} \mathrm{E}\left[\widehat{q}\right] - q &\geq \frac{p}{N f_Y(q)}\left(\log p\left(1 + \frac{f_Y'(q)p}{2f_Y(q)^2}\right) + \frac{f_Y'(q)p}{f_Y(q)^2}\right) + o\left(\frac{1}{N}\right) \\ \mathrm{E}\left[\widehat{q}\right] - q &\leq \frac{p}{N f_Y(q)}\left(\log p\left(1 + \frac{f_Y'(q)p}{2f_Y(q)^2}\right) + 1 - \frac{f_Y'(q)p}{f_Y(q)^2}\right) + o\left(\frac{1}{N}\right). \end{aligned} \tag{3.28}$$

assuming $f'(q) < 0$. *Especially these bounds change with the sign of* $f'(q)$ *and the interval is wider than the one given in Eq.* (3.25).

3.5 Discontinuous random variables

All the previous results assume that the *cdf* of $Y = g(\mathbf{X})$ is continuous. This if often the case in the literature and little is known about the impact of using splitting strategies if this assumption does not hold. Recall that $\mathbf{X}$ is a random finite- or infinite-dimensional vector with known distribution μ^X and g a real-valued measurable performance function. Y can be discontinuous if for example there is some threshold effect in g and/or if $\mathbf{X}$ is discrete or mixed discrete/continuous [Cérou et al., 2011, Rubinstein, 2009b, 2010, Rubinstein et al., 2012]. Also if $\mathbf{X}$ is a random path, the discretisation of the solution of a Stochastic Differential Equation (SDE) can lead to some accumulations points (see Section 3.6 below).

Recently, Simonnet [2016] showed that in the case of the Last Particle Algorithm the random number of iterations is indeed a mixture of independent Poisson and negative binomial laws while in the continuous case, it is only a Poisson law. Unfortunately he could not derive a general unbiased estimator from this result [Simonnet, 2016, Theorems 4 and 5]. The main problem comes from the fact that the equality: $\forall y \in \mathbb{R},\ \mathrm{P}\left[Y > y\right] = \mathrm{P}\left[Y \geq y\right]$ does not hold any more (see Remark 3.1). Rubinstein [2009b] already noticed that one should pay attention to the fact that the root q_i of $\mathrm{P}\left[g(\mathbf{X}) \geq q_i \mid g(\mathbf{X}) \geq q_{i-1}\right] = p_0$ may not be unique [Rubinstein, 2009b, Remark 6.1], [Botev and Kroese, 2008, Remark 2.6]. He then derived some guidelines for an appropriate adaptive choice of the $(q_i)_i$ for Splitting methods (see Section 1.3.2) in this case but concludes that the algorithm can eventually fail to estimate the sought probability [it stops at an intermediate level, see Rubinstein, 2009b, Remark 6.3] or returns 0 [Amrein and Künsch, 2011]).

Finally, Cérou et al. [2011] suggested to use for the Boolean SATisfiability Problem (SAT problem) an auxiliary continuous random variable $\widetilde{Y}$ such that $\mathrm{P}\left[\widetilde{Y} > q\right] = \mathrm{P}\left[g(\mathbf{X}) > q\right]$ and showed practical improvement. This idea of using an auxiliary continuous random variable is also used by Huber and Schott [2011] for the Ising model. Eventually there are always case specific transformations. Skilling [2006] also mentions this issue and proposes to add a uniform random variable on a *tiny* interval but one lacks of justifications and clear guidelines and consequences.

Following the random walk framework developed in Section 3.2 the goal of this section is to fill this gap by providing both the distribution of the number of iterations (the distribution of the counting random variables) and the Minimal Variance Unbiased Estimators (MVUE) in the two alternative definitions of the increasing random walk (see Remark 3.1): the one with strict inequality and the one with non-strict inequality. While they are the same with probability 1 if $Y = g(\mathbf{X})$ is continuous, they may differ if it is not. Especially Bréhier et al. [2015a] recently derive an unbiased estimator for the strict

inequality case in the usual AMS framework.

However the distributions of these estimators are less simple than in the continuous case. In this scope we also suggest a third estimator based on the increasing random walk with non-strict inequality which has the same statistical properties as in the continuous case, *i.e.* with or without discontinuities. Practically speaking it is not necessary to know in advance if Y is actually continuous or not and in this latter case, the three estimators become the same.

3.5.1 The increasing random walk for discontinuous random variables

From now on, we assume that Y is a real-valued random variable, continuous or not. We extend the definition of the increasing random walk (Definition 3.1) as follows:

Definition 3.2 (Increasing random walk with non-strict inequality)**.** *Let $Y_0^{\geq} = 0$; the increasing random walk with non-strict inequality associated with Y is the Markov sequence $(Y_n^{\geq})_n$ such that:*

$$\forall n \in \mathbb{N},\ \mathrm{P}\left[Y_{n+1}^{\geq} \in A \mid Y_0^{\geq}, \cdots, Y_n^{\geq}\right] = \frac{\mathrm{P}\left[Y \in A \cap [Y_n^{\geq}, +\infty)\right]}{\mathrm{P}\left[Y \in [Y_n^{\geq}, +\infty)\right]}. \tag{3.29}$$

Definition 3.3 (Increasing random walk with strict inequality)**.** *Let $Y_0^{>} = 0$; the increasing random walk with strict inequality associated with Y is the Markov sequence $(Y_n^{>})_n$ such that:*

$$\forall n \in \mathbb{N},\ \mathrm{P}\left[Y_{n+1}^{>} \in A \mid Y_0^{>}, \cdots, Y_n^{>}\right] = \frac{\mathrm{P}\left[Y \in A \cap (Y_n^{>}, +\infty)\right]}{\mathrm{P}\left[Y \in (Y_n^{>}, +\infty)\right]}. \tag{3.30}$$

In other words $(Y_n^{\geq})_n$ is an increasing sequence where each element is randomly generated conditionally greater or equal than the previous one: $Y_{n+1}^{\geq} \sim \mu^Y(\cdot \mid Y \geq Y_n^{\geq})$ while $(Y_n^{>})_n$ is an increasing sequence where each element is randomly generated conditionally strictly greater than the previous one : $Y_{n+1}^{>} \sim \mu^Y(\cdot \mid Y > Y_n^{>})$. In the sequel, the superscripts $^{>}$ and $^{\geq}$ will be used to denote quantities based on the increasing random walk with strict (resp. non-strict) inequality.

Let D be the set of the atoms of Y. According to Froda's theorem [Froda, 1929] it is countable. As in Section 3.2 we consider for all $y \in \mathbb{R}$ the counting random variable at time $y \in \mathbb{R}$: $M_y^{>} = \mathrm{card}\{n \geq 1 \mid Y_n^{>} \leq y\}$ and $M_y^{\geq} = \mathrm{card}\{n \geq 1 \mid Y_n^{\geq} \leq y\}$. The next two propositions aim at giving the law of $M_y^{>}$ and $M_y^{\geq}$. In both Propositions 3.9 and 3.10, we consider $y \in \mathbb{R} \mid p_y = \mathrm{P}[Y > y] > 0$, $D_y = D \cap (-\infty, y]$ and define:

$$\forall d \in D,\ \Delta_d = \frac{\mathrm{P}[Y > d]}{\mathrm{P}[Y \geq d]}. \tag{3.31}$$

Proposition 3.9 (Law of the counting random variable for the non-strict random walk)**.** *$M_y^{\geq}$ is a mixture of independent Poisson and Geometric random variables such that:*

$$M_y^{\geq} \sim \mathcal{P}\left(-\log \frac{p_y}{\prod\limits_{d\in D_y} \Delta_d}\right) \oplus \sum_{d\in D_y} \mathcal{G}\left(\Delta_d\right) \tag{3.32}$$

with $\mathcal{G}$ a Geometric law counting the number of failures before success. In other words, $M_y^{\geq}$ is the sum of independent random variables, the first one being a Poisson random variable with parameter $-\log p_y / \prod_{d\in D_y} \Delta_d$ and the other ones independent Geometric random variables with parameter Δ_d, $d \in D_y$ respectively.

Proof. The distribution of $M_y^{\geq}$ has already been proved by Simonnet [2016] assuming that $\operatorname{card} D_y < \infty$. We extend this result to the possible infinite countable number of discontinuities.

Let $S_n = \{y \in \mathbb{R} \mid 0 < \mathrm{P}\left[Y = y\right] < 1/n\}$ be the set of the jump points of F_Y with jump amplitudes smaller than $1/n$ and $Y^{(n)} = Y \mathbb{1}_{Y \notin S_n}$. $Y^{(n)}$ has a finite number of jump points and accumulates the (possibly infinite) number of jump points $d \in D \mid \mathrm{P}\left[Y = d\right] < 1/n$ at 0. Since it has a finite number of discontinuities, the law of its associated counting random variables is known.

Furthermore it verifies:

$$\forall y \in \mathbb{R},\ \mathrm{P}\left[Y^{(n)} \leq y\right] = \mathrm{P}\left[Y \leq y\right] + \mathbb{1}_{y\geq 0} \sum_{\substack{d\in S_n \\ d>y}} \mathrm{P}\left[Y = d\right] - \mathbb{1}_{y<0} \sum_{\substack{d\in S_n \\ d\leq y}} \mathrm{P}\left[Y = d\right].$$

Hence $Y^{(n)} \xrightarrow[n\to\infty]{\mathcal{L}} Y$, which implies $M_y^{(n)} \xrightarrow[n\to\infty]{\mathcal{L}} M_y^{\geq}$ with $M_y^{(n)}$ the counting random variable at state y associated with an increasing random walk with non-strict inequality on $Y^{(n)}$.

Moreover, one has: $\forall y \in \mathbb{R},\ \forall n \geq 1,\ M_y^{(n)} \sim \mathcal{P}\left(-\log \dfrac{\mathrm{P}\left[Y^{(n)} > x\right]}{\prod\limits_{d\in D_y\setminus S_n} \Delta_d}\right) + \sum\limits_{d\in D_y\setminus S_n} \mathcal{G}\left(\Delta_d\right).$

Finally, this gives:

$$M_y^{\geq} \sim \mathcal{P}\left(-\log \frac{\mathrm{P}\left[Y > y\right]}{\prod\limits_{d\in D_y} \Delta_d}\right) \oplus \sum_{d\in D_y} \mathcal{G}\left(\Delta_d\right).$$

We also show that the first and second order moments of $M_y^{\geq}$ remain finite even when $\operatorname{card}(D_y) = \infty$. One has:

$$0 \leq \sum_{d\in D_y} \left(\frac{1}{\Delta_d} - 1\right) = \sum_{d\in D_y} \frac{\mathrm{P}\left[Y = d\right]}{\mathrm{P}\left[Y > d\right]} \leq \frac{1}{\mathrm{P}\left[Y > y\right]} \sum_{d\in D_y} \mathrm{P}\left[Y = d\right] \leq \frac{1 - p_y}{p_y},$$

and:

$$1 \geq \prod_{d \in D_y} \Delta_d \geq \prod_{d \in D_y} e^{-\left(\frac{1}{\Delta_d}-1\right)} \geq e^{-\sum\limits_{d \in D_y}\left(\frac{1}{\Delta_D}-1\right)} \geq e^{-\frac{1-p_y}{p_y}} > 0.$$

All together, theses inequalities give the result:

$$\mathrm{E}\left[M_y^{\geq}\right] = -\log \frac{p_y}{\prod\limits_{d \in D_y} \Delta} + \sum_{d \in D_y} \left(\frac{1}{\Delta_d} - 1\right) \leq -\log p_y + \frac{1-p_y}{p_y}$$

$$\mathrm{var}\left[M_y^{\geq}\right] = -\log \frac{p_y}{\prod\limits_{d \in D_y} \Delta} + \sum_{d \in D_y} \frac{1}{\Delta_d}\left(\frac{1}{\Delta_d} - 1\right) \leq -\log p_y + \frac{1-p_y}{p_y^2}.$$

□

It is part of the proof above that the distribution of $M_y^{\geq}$ is well defined with finite mean and variance even when $\mathrm{card}(D_y) = \infty$, extending the result of Simonnet [2016] who proved it with a combinatorial analysis assuming that $\mathrm{card}(D) < \infty$. Indeed it can be understood using the renewal property of a Poisson process: the number of events corresponding to the *continuous* part, *i.e.* events $Y_n \notin D$, follows a Poisson law with parameter $-\log p_y - \sum_d(-\log \Delta_d) = -\log(p_y / \prod_d \Delta_d)$. On the other hand each jump point leads to a random number of iterations following a Geometric law with probability of success $\mathrm{P}\left[Y > d\right] / \mathrm{P}\left[Y \geq d\right] = \Delta_d$.

Proposition 3.10 (Law of the counting random variable for the strict random walk)**.** *$M_y^{>}$ is a mixture of independent Poisson and Bernoulli random variables such that:*

$$M_y^{>} \sim \mathcal{P}\left(-\log \frac{p_y}{\prod\limits_{d \in D_y} \Delta_d}\right) \oplus \sum_{d \in D_y} \mathcal{B}\left(1 - \Delta_d\right) \tag{3.33}$$

with $\mathcal{B}$ a Bernoulli distribution. $M_y^{>}$ is then the sum of independent random variables, the first one being a Poisson random variable with parameter $-\log p_y / \prod_{d \in D_y} \Delta_d$ and the other ones independent Bernoulli random variables with parameter $1 - \Delta_d$, $d \in D_y$ respectively.

Proof. Using the renewal property of the Poisson process the number of events in the *continuous* part will be the same as the one in the non-strict case. Indeed the only difference with the non-strict random walk comes from the *behaviour* of the random walk when $Y_n \in D_y$. In this latter case, while the non-strict inequality repeats the trial until success, the strict inequality do it only once. Hence the Geometric law is replaced by a Bernoulli one. □

Since the Geometric law counts the number of failures while the Bernoulli one gives 1 in case of success, the parameter is the opposite. Furthermore, both Eqs. (3.32) and (3.33)

are equal when $D = \varnothing$, *i.e.* when Y is continuous. In this latter case, one finds back the pure Poisson distribution $M_y^> \sim M_y^\geq \sim M_y \sim \mathcal{P}(-\log p_y)$.

3.5.2 Probability estimators

As noticed by Simonnet [2016], formulas (3.32) and (3.33) are not very useful to derive an unbiased probability estimator. However we do not generate only *iid.* copies of the counting random variables but the random walks themselves. Hence if one can afford storing all the *states* of each random walk, then it is possible to build a MVUE in both cases.

Preliminary results

Lemma 3.3 (MVUE for a Geometric distribution)**.** *Let $G \sim \mathcal{G}(p)$ be a Geometric random variable counting the number of failures before success with probability of success p and $(G_i)_{i=1}^N$ N iid. copies of G, then the minimal variance unbiased estimator for p is:*

$$\widehat{p} = \frac{N-1}{N-1+\sum\limits_{i=1}^{N} G_i}. \tag{3.34}$$

Proof. One is going to use Lehmann-Scheffé theorem with the statistic $T = \sum\limits_{i=1}^{N} G_i$. As the sum of N independent Geometric random variables with parameter p, T follows a Negative Binomial law: $\forall t \in \mathbb{N},\ \mathrm{P}\left[T = t\right] = \binom{N+t-1}{t} p^N (1-p)^t$. T is sufficient:

$$\mathcal{L}(g_1, \cdots, g_N, p) = \prod_{i=1}^{N} \mathrm{P}\left[G_i = g_i\right] = (1-p)^{\sum\limits_{i=1}^{N} g_i} p^N = (1-p)^t p^N.$$

T is also complete: let $\phi : \mathbb{N} \to \mathbb{R}$ be a function, one has:

$$\begin{aligned} \forall p \in (0,1),\ \mathrm{E}\left[\phi(T)\right] = 0 &\Rightarrow \forall p \in (0,1),\ p^N \sum_{t=0}^{\infty} \binom{t+N-1}{t} \phi(t)(1-p)^t = 0 \\ &\Rightarrow \forall \theta \in (0,1),\ \sum_{t=0}^{\infty} \alpha_t \theta^t \end{aligned}$$

with $\alpha_t = \binom{t+N-1}{t}\phi(t)$ and $\theta = 1-p$. Furthermore $p = 1$ *i.e.* $\theta = 0$ gives $\phi(0) = 0$ and $\theta = 1$, *i.e.* $p = 0$ gives $\mathrm{P}\left[T < \infty\right] = 0$. Hence the power series $\theta \mapsto \sum \alpha_t \theta^t$ is identically null on its radius of convergence $[0,1)$ and so $\forall t \in \mathbb{N},\ \alpha_t = 0$, which means $\forall t \in \mathbb{N},\ \phi(t) = 0$ and T is complete.

We now consider the estimator $R = \dfrac{1}{N} \sum\limits_{i=1}^{N} \mathbb{1}_{G_i=0}$. R is unbiased because $\mathrm{E}\left[\mathbb{1}_{G_i=0}\right] = \mathrm{P}\left[G_1 = 0\right] = p$. Then the Lehmann-Scheffé theorem states that $\mathrm{E}\left[R \mid T\right]$ is the MVUE of

p. This gives:

$$\begin{aligned}
\mathrm{E}\left[R \mid T=t\right] &= \frac{1}{N}\sum_{i=1}^{N}\mathrm{E}\left[\mathbb{1}_{G_i=0} \mid \sum_{i=1}^{N} G_i = t\right] = \mathrm{P}\left[G_1 = 0 \mid \sum_{i=1}^{N} G_i = t\right] \\
&= \frac{\mathrm{P}\left[G_1 = 0, \sum_{i=2}^{N} G_i = t\right]}{\mathrm{P}\left[\sum_{i=1}^{N} G_i = t\right]} = \mathrm{P}\left[G_1 = 0\right] \frac{\binom{t+N-2}{t} p^{N-1}(1-p)^t}{\binom{t+N-1}{t} p^N (1-p)^t} \\
\mathrm{E}\left[R \mid T=t\right] &= \frac{N-1}{N-1+t}.
\end{aligned}$$

Hence, $\widehat{p} = \mathrm{E}\left[R \mid T\right] = (N-1)/(N-1+T)$ is the MVUE of p. □

Lemma 3.4 (MVUE for a Bernoulli distribution)**.** *Let $B \sim \mathcal{B}(1-p)$ be a Bernoulli random variable with probability of failure p and $(B_i)_{i=1}^{N}$ N iid. copies of B, then the minimal variance unbiased estimator for p is:*

$$\widehat{p} = 1 - \frac{\sum_{i=1}^{N} B_i}{N}. \tag{3.35}$$

Definition 3.4 (Run-length encoding)**.** *Let $\mathbf{v} = (v_1, \cdots, v_m) \in \mathbb{R}^m$, $m \geq 1$, be a vector such that $\forall i \in [\![1, m-1]\!]$, $v_i \leq v_{i+1}$. We call the* run-length encoding *of $\mathbf{v}$ the vector $\mathbf{r}$ of the lengths of runs of equal values in $\mathbf{v}$.*

In other words, the run-length encoding counts for any non decreasing sequence the number of times each value is repeated: for example if $\mathbf{v} = (0.5, 2.1, 2.1, 2.1, \pi)$ then $\mathbf{r} = (1, 3, 1)$. Especially, if Y is continuous the RLE of the states of a realisation of the increasing random walk $(Y_1, \cdots, Y_m)$, is $\mathbf{r} = (1, \cdots, 1) \in \mathbb{N}^m$ with probability 1 while on the contrary discontinuities will produce repeated values with non-zero probability. More precisely, the number of times each value is repeated corresponds to the number of *failures* while sampling above a threshold. With this consideration we are now in position to define the probability estimators.

In the sequel we assume that for a given $y \in \mathbb{R} \mid \mathrm{P}\left[Y > y\right] > 0$, $(Y_i)_{i=1}^{\bar{M}_y}$ is the merged and sorted sequence of the states of N (non-)strict inequality random walks generated until state y; $\bar{M}_y = \sum_{i=1}^{N} M_y^i$ is the sum of the counting random variables of each random walk, $\mathbf{r}$ is the RLE of $(Y_1, \cdots, Y_{\bar{M}_y})$, and l is its length.

Non-strict random walk

Proposition 3.11. *The MVUE for the non-strict inequality random walk is:*

$$\widehat{p_y^{\geq}} = \prod_{i=1}^{l} \frac{N-1}{N-1+r_i}. \tag{3.36}$$

It verifies:

$$p_y^2\left(p_{pois}^{-1/N}-1\right) \leq \operatorname{var}\left[\widehat{p}_y^{\geq}\right] \leq p_y^2\left(p_{pois}^{-1/N}\left(\frac{N-1}{N-2}\right)^{\#D_y}-1\right) \tag{3.37}$$

with $p_{pois}=\dfrac{p_y}{\prod\limits_{d\in D_y}\Delta_d}$ *and* $\#D_y=\operatorname{card}(D_y)$.

Proof. On the one hand, for all $a<b$ such that Y is continuous on (a,b), $\mathrm{P}\left[Y>b \mid Y>a\right]$ can be estimated by $(1-1/N)^{\#\{Y_n\in(a,b)\}}$ with $\#\{Y_n\in(a,b)\}$ the number of events of the superposed process in (a,b). Moreover, since $b\mapsto \mathrm{P}\left[Y\geq b\right]$ is left-continuous, it is also a MVUE of $\mathrm{P}\left[Y\geq b \mid Y>a\right]$ and this relation remains true if $b\to a$ since $\operatorname{card}(\varnothing)=0$. Using the fact that the RLE of $(Y_1,\cdots,Y_{\bar{M}_y})$ equals $(1,...,1)$ with probability 1 when Y is continuous, $(1-1/N)^{\#\{Y_n\in(a,b)\}}=\prod_i(N-1)/(N-1+r_i)$ with probability 1.

On the other hand $\forall d\in D_y,\ \Delta_d=\mathrm{P}\left[Y>d\right]/\mathrm{P}\left[Y\geq d\right]$ can be estimated with the MVUE defined in Lemma 3.3. The first state of each chain non smaller than d can be considered as an *iid.* sample of $Y \mid Y\geq d$. The number of times d is found in $(Y_1,\cdots,Y_{\bar{M}_y})$ is then the sum of N realisations of a Geometric random variable with parameter Δ_d. Hence, Δ_d is estimated with $(N-1)/(N-1+r_{i_d})$ with r_{i_d} the number of times d is found in $(Y_1,\cdots,Y_{\bar{M}_y})$.

Since D is countable, we consider $\mathbb{R}\setminus D=\bigcup_i I_i$ with $(I_i)_i$ a sequence of disjoint open intervals. Note that some subsequence of $(I_n)_n$ may converge toward the empty set if D is infinite countable with some accumulation points. However Y is continuous on $\mathbb{R}\setminus D$. Using the renewal property of the Poisson process, one can consider that $\prod_d(N-1)/(N-1+r_{i_d})$ is a product of independent MVUE estimators. Especially, denoting by M_{pois} the number of 1 in $\mathbf{r}$, $(1-1/N)^{M_{\text{pois}}}$ is a MVUE of $p_{\text{pois}}:=p_y/\prod_d\Delta_d$.

We now explicit the calculation for the bounds on the variance. One has:

$$\mathrm{E}\left[(\widehat{p}_y^{\geq})^2\right]=p_{\text{pois}}^2 p_{\text{pois}}^{-1/N}\prod_{d\in D_y}\mathrm{E}\left[\left(\frac{N-1}{N-1+T_d}\right)^2\right]$$

with $T_d\sim NegBin(N,\Delta_d)$. For a given $d\in D_y$, one has:

$$\begin{aligned}
\mathrm{E}\left[\left(\frac{N-1}{N-1+T_d}\right)^2\right] &= \sum_{t=0}^{\infty}\binom{t+N-1}{t}(1-\Delta_d)^t\Delta_d^N\left(\frac{N-1}{N-1+t}\right)^2\\
&= \sum_{t=0}^{\infty}\binom{t+N-2}{t}(1-\Delta_d)^t\Delta_d^N\frac{N-1}{N-1+t}\\
&= \Delta_d\sum_{t=0}^{\infty}\binom{t+N-2}{t}(1-\Delta_d)^t\Delta_d^{N-1}\frac{N-2}{N-2+t}\frac{N-1}{N-2}\frac{N-2+t}{N-1+t}.
\end{aligned}$$

Furthermore, $\forall t \geq 0$, $\dfrac{N-1}{N-2}\dfrac{N-2+t}{N-1+t} \in [1, (N-1)/(N-2)]$, which gives:

$$\Delta_d^2 \leq \mathrm{E}\left[\left(\frac{N-1}{N-1+T_d}\right)^2\right] \leq \Delta_d^2 \frac{N-1}{N-2}.$$

Eventually the variance writes:

$$p_y^2\left(p_{\text{pois}}^{-1/N}-1\right) \leq \mathrm{var}\left[\widehat{p}_y^{>}\right] = \mathrm{E}\left[(\widehat{p}_y^{>})^2\right] - p_y^2 \leq p_y^2\left(p_{\text{pois}}^{-1/N}\prod_{d\in D_y}\frac{N-1}{N-2}-1\right).$$

□

It is interesting to notice here that in a case of a discrete random variable, $p_{\text{pois}} = 1$ and the bounds on the variance become:

$$0 \leq \frac{\mathrm{var}\left[\widehat{p}_y^{>}\right]}{p_y^2} \leq \left(\frac{N-1}{N-2}\right)^{\#D_y} - 1 = \frac{\#D_y}{N} + o\left(\frac{1}{N}\right). \tag{3.38}$$

It means that the coefficient of variation is bounded by a quantity which does not depend on the size but only on the number of jumps. Since this quantity is likely to be known with good confidence and does not vary much with both y (if one looks at different thresholds) and $\#D_y$ (the number of discontinuities before the given y), this can provide a robust upper bound for the coefficient of variation.

Strict random walk

Proposition 3.12. *The MVUE for the strict inequality random walk is:*

$$\widehat{p}_y^{>} = \prod_{i=1}^{l}\left(1-\frac{r_i}{N}\right). \tag{3.39}$$

It verifies:

$$\mathrm{var}\left[\widehat{p}_y^{>}\right] = p_y^2\left(p_y^{-1/N}\prod_{d\in D_y}\nu(\Delta_d, N) - 1\right) \tag{3.40}$$

with $\nu : (\Delta, N) \mapsto \Delta^{1/N}\left(1+\dfrac{1-\Delta}{N\Delta}\right)$.

Proof. The same reasoning as for the proof of Proposition 3.11 applies where the MVUE of a Geometric law is replaced by the one of a Bernoulli distribution. For the variance one has:

$$\mathrm{E}\left[(\widehat{p}_y^{>})^2\right] = p_{\text{pois}}^2\, p_{\text{pois}}^{-1/N}\prod_{d\in D_y}\frac{\Delta_d^2}{N}\left(N-1+\frac{1}{\Delta_d}\right)$$

$$\mathrm{E}\left[(\widehat{p}_y^{>})^2\right] = p_y^2\, p_y^{-1/N}\prod_{d\in D_y}\frac{\Delta_d(N-1)+1}{N\Delta_d^{1-1/N}}$$

which gives the result. □

On the one hand we have been able to define minimal variance unbiased estimators for both the strict and non-strict random walks. They become equal when the random variable is continuous and in this case one finds back the estimator defined in Section 3.3.1. Especially the variance increase due to discontinuities in the distribution of Y is clearly visible in Eq. (3.40) as $\nu(\Delta, N) > 1$ if $\Delta < 1$. On the other hand their distributions are not easy to characterise; in particular we could not derive any practical literal expression of the variance in the non-strict case. In this context we suggest to consider an auxiliary continuous random variable which involves an independent uniform random variable. This transformation is general and does not require any other knowledge on the problem as discussed below.

Pure Poisson estimator

Indeed when generating a Geometric random variable with *iid.* trials $(B_n)_n$ with the Bernoulli distribution $\mathcal{B}(p)$ with probability of success p, one can also consider the random variable $\widetilde{Y} = B + U$ with $B \sim \mathcal{B}(p)$ and $U \sim \mathcal{U}[0,1]$ an independent Uniform random variable on the interval $[0,1]$. Figure 3.4 plots the *cdf* of B and $\widetilde{Y}$. Furthermore, $\forall y \in [0,2], \{\widetilde{Y} > y\} = \{B \geq \lfloor y \rfloor\} \cap \{U > y - \lfloor y \rfloor\}$. Practically speaking, this means that the generation of the geometric random variable can be seen as a basic Acceptance-Rejection scheme used to generate the increasing random walk on $\widetilde{Y}$ until state 1: for each generated B, sample also $U \sim \mathcal{U}[0,1]$ and accept the transition for $\widetilde{Y}$ if $U > y - \lfloor y \rfloor$. Eventually, considering the fact that $p = \mathrm{P}[B = 1] = \mathrm{P}\left[\widetilde{Y} \geq 1\right] = \mathrm{P}\left[\widetilde{Y} > 1\right]$, p can also be estimated using the increasing random walk on the continuous random variable $\widetilde{Y}$.

To conclude, in addition to the MVUE of p defined in Lemma 3.3 and at the cost of the generation of an independent uniform random variable, one also produces an estimator of the form of Eq. (3.7) with the same statistical properties. Embedding this in the generation of the non-strict inequality random walk gives then an estimator with the same properties as the ones in the continuous case. Algorithm 5, Theorem 3.2 and Corollary 3.5 precise this point. In the sequel, we will refer to this estimator as the *pure Poisson* estimator.

Theorem 3.2. *In Algorithm 5, the random variable M_y follows a Poisson law with parameter $-\log \mathrm{P}[Y > y]$.*

Proof. The difference between the generation of the non-strict random walk and Algorithm 5 stands in the addition of the *while* loop from line 6 to line 12. This loop is entered when a Geometric scheme is started: two consecutive events being equal is a non-zero probability event only when $Y_n \in D$. In this context, the condition $\{U_{n+1} > U_n\}$ lets generate the counting random variable of the continuous random variable associated with the Geometric scheme as explained in Section 3.5.2. Therefore it follows a Poisson distribution with parameter $-\log \Delta_{Y_n}$. The renewal property of the Poisson process lets conclude the proof. □

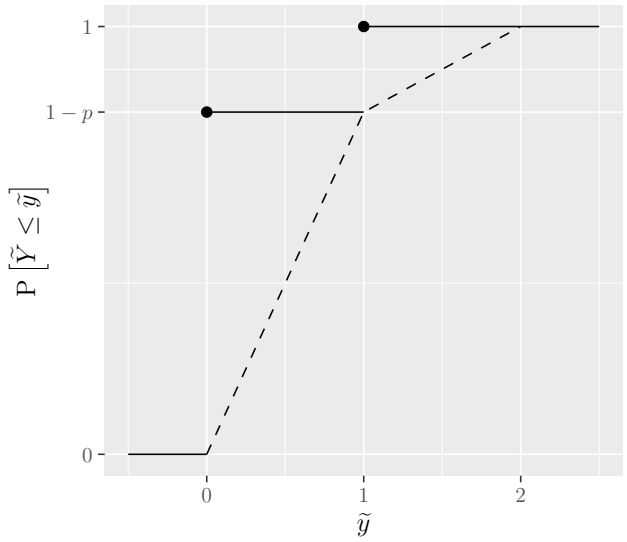


Figure 3.4: *cdf* of a Bernoulli random variable $B \sim \mathcal{B}(p)$ (solid dark line) and its associated continuous random variable $\widetilde{Y} = B + U$ (dark dashed line) with $U \sim \mathcal{U}[0,1]$.

Algorithm 5 Pseudo-code for the non-strict inequality random walk and the pure Poisson estimator

Require: y

$M_y = 0$

Draw $Y_1 \sim \mu^Y$ and $U_1 \sim U[0,1]$; $n = 1$

while $Y_n \leq x$ **do**

$M_y = M_y + 1$

Draw $Y_{n+1} \sim \mu^Y(\cdot \mid Y \geq Y_n)$ and $U_{n+1} \sim U[0,1]$

while $Y_{n+1} = Y_n$ **do**

if $U_{n+1} > U_n$ **then**

$M_y = M_y + 1$

end if

$n = n + 1$

Draw $Y_{n+1} \sim \mu^Y(\cdot \mid Y \geq Y_n)$ and $U_{n+1} \sim U[0,1]$

end while

$n = n + 1$

end while

return $M_y,\ (Y_n)_n$

Corollary 3.5. *Let $N \geq 2$ and $(M_y^i)_{i=1}^N$ be N iid. realisations of Algorithm 5, the estimator*

$$\widehat{p}_y = \left(1 - \frac{1}{N}\right)^{\sum_{i=1}^{N} M_y^i} \tag{3.41}$$

has the same properties as in Proposition 3.2.

Remark 3.8 (Notations)**.** *Throughout this section, we have clearly stipulated when we were considering the strict or the non-strict random walk. Here we use the same notation as in the continuous case because this estimator, even if it based on the non-strict random walk, is especially designed to have the same properties with or without discontinuities in the cdf of Y, precisely the properties of the continuous case.*

Finally the distinction between the strict and the non-strict random walks lets define two different estimators for the probability of exceeding a threshold. Both are unbiased and become the same when Y is indeed continuous. However their distributions are not well-characterised.

In this scope we have introduced a third estimator based on the non-strict random walk. With the addition of a *while* loop and an independent Uniform random variable, we have been able to produce an estimator which has always the same statistical properties, Y being continuous or not. This estimator is not optimal in terms of variance when Y is actually discontinuous but remains close to the optimal one when the jumps $(\Delta_d)_d$ remain close to 1: the MVUE of Lemma 3.3 has a squared coefficient of variation approximately equal to $(1-\Delta)/N$ while it is $-\log(\Delta)/N$ for the *pure Poisson* estimator. Furthermore this sub-optimal estimator is only a by-product of the MVUE and so both results can be considered at the same time: one for the *best* estimated value and the other one for building conservative confidence intervals. These results are illustrated in the following section.

3.6 Numerical examples

In this section we focus on the illustration of the theoretical properties of the point process based estimators, especially their ability to handle seamlessly discontinuous random variables. More usual test cases for reliability engineering are handled in Section A.4.

3.6.1 Discretised random path

Problem setting

We consider here the example used by Simonnet [2016] to illustrate his results. It is a numerical study with a Euler scheme of a diffusive process satisfying:

$$d\mathbf{X}_t = -\nabla V dt + \sqrt{\frac{2}{\beta}} d\mathbf{W}_t,\ \mathbf{X}_0 = \mathbf{x}_0 \tag{3.42}$$

where $\mathbf{W}_t$ is a Wiener process, β^{-1} is the temperature, and V is a potential defined by:

$$V(x_1, x_2) = -\left(\frac{x_1^2}{2} - \frac{x_1^4}{4}\right) - b\left(\frac{x_2^2}{2} - \frac{x_2^4}{4}\right) + \frac{a}{2}x_1^2 x_2^2,$$

for some a and b. The goal is to estimate the probability that the process enters a given set B before another set A from an initial state $\mathbf{x}_0$: if τ_C is the stopping time defined by $\tau_C := \inf\{t \geq 0 \mid \mathbf{X}_t \in C\}$, then one seeks for estimating:

$$p = P_{\mathbf{x}_0}[\tau_B < \tau_A],$$

where $P_{\mathbf{x}_0}$ is the distribution of $(\mathbf{X}_t)_t$ starting from $\mathbf{X}_0 = \mathbf{x}_0$. As a function of $\mathbf{x}_0$ this quantity is known as the *Committor*. From a practical point of view, it is often intractable and a *reaction coordinate* Φ is introduced to measure *how far* a trajectory is escaping from A before returning to it: $\Phi : \mathbb{R}^d \to \mathbb{R}$ such that $A = \Phi^{-1}((-\infty, 0])$ and $B = \Phi^{-1}((1, +\infty))$. With these notations, a trajectory enters B before returning to A if and only if:

$$Y := \sup_{t \in [0, \tau_A)} \Phi(\mathbf{X}_t) > 1.$$

Y is then the real-valued random variable of interest and the problem is indeed to estimate $\mathrm{P}[Y > 1]$. With this notation, the theoretical results of Section 3.5 can be used directly.

Conditional sampling

To avoid possible issues due to imperfect conditional sampling with methods such as the one described in Section 1.3.3, Simonnet [2016] makes use of an Acceptance-Rejection sampling to generate samples above a given threshold. While not relevant in practice, this method lets focus on the consistency between theoretical and practical results. A scheme of such sampling is given in Algorithm 6.

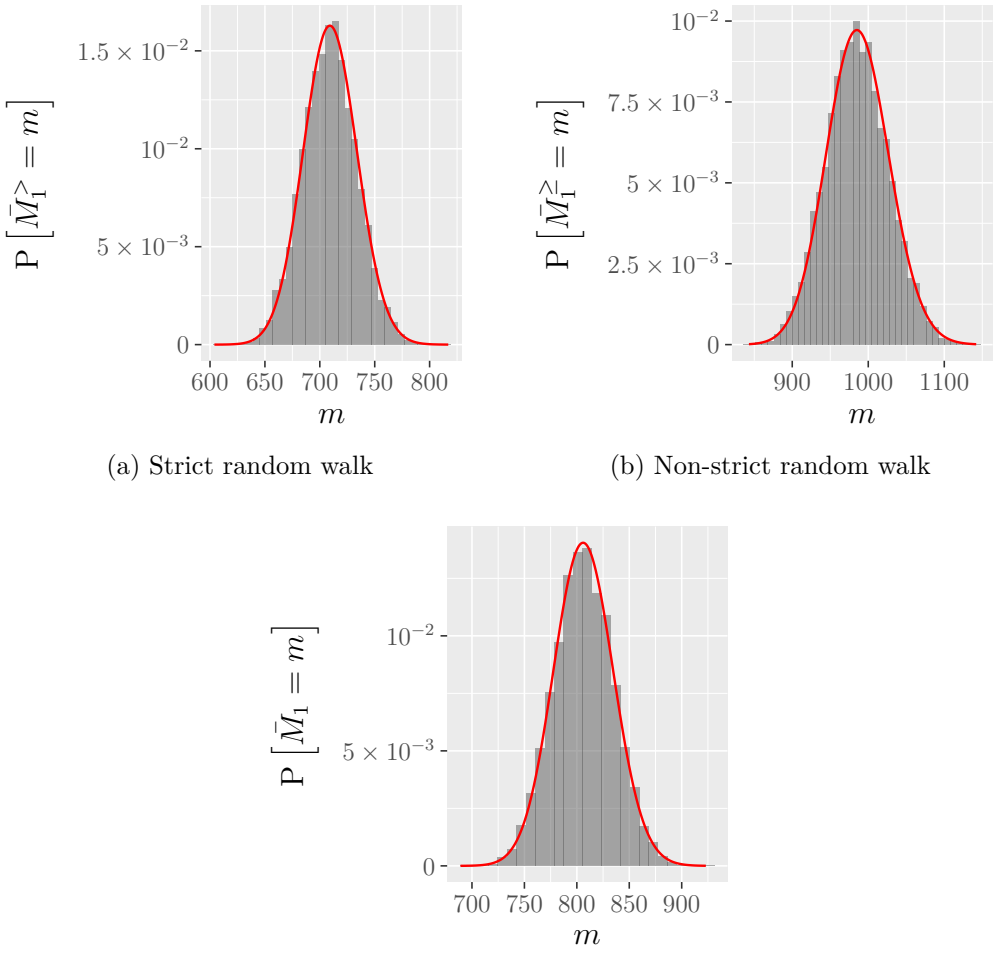
Algorithm 6 Perfect sampling of $Y \sim \mu^Y(\cdot \mid Y \geq y)$ for the diffusive process

Require: $y \in \mathbb{R}$ ▷ the current threshold one seeks to sample above
  $Y^* = -\infty$
  while $Y^* < y$ **do**
    Generate a new trajectory $\mathbf{X}_t$ starting from $\mathbf{x}_0$
    $Y^* = \sup_{t \in [0, \tau_{A \cup B}]} \Phi(X_t)$
  end while

Numerical results

Here we set $\Phi(\mathbf{x}) = 0.5(1 + x_1)$, $a = 0.6$, $b = 0.3$, $\mathbf{x}_0 = (-0.9, 0)$, $\beta = 10$ and $dt = 1$ as in [Simonnet, 2016]. $\Phi(\mathbf{x}) = 0.5(1 + x_1)$ and so $\Phi(\mathbf{x}_0) = 5 \times 10^{-2}$: with a large time-step some trajectories will go directly into A, producing a discontinuity in the *cdf* of Y. We computed reference values using a crude Monte Carlo estimator with $N = 10^6$ and found $p = 6.8 \times 10^{-2}$ and $\Delta = 0.4$. We then set $N = 300$ to get a coefficient of variation below 10% because: $\mathrm{CV}[\widehat{p}]^2 \approx -\log p / N \Rightarrow N \approx 268$.

We first focus on the distribution of the number of iterations described in Propositions 3.9 and 3.10 and on the *corrected* number of iterations to get a pure Poisson distribution (see Theorem 3.2).



(a) Strict random walk

(b) Non-strict random walk

(c) Pure Poisson correction

Figure 3.5: Histogram over 10^4 realisations of the sum of $N = 300$ *iid.* counting random variables for the strict random walk (Eq. 3.33), the non-strict one (Eq. 3.32) and the pure Poisson *correction* (Theorem 3.2). Y has one discontinuity at $y = 5 \times 10^{-2}$ and $\mathrm{P}\left[Y > y\right] / \mathrm{P}\left[Y \geq y\right] \approx 0.4$. The curves show the theoretical distributions.

Figure 3.5 shows the histograms of the sum of N *iid.* counting random variables for the strict random walk, the non-strict one and the pure Poisson *correction*. They are in good agreement with the theoretical distributions presented in Eqs. (3.33) and (3.32) and Theorem 3.2 respectively. Especially we can see that for a given N, the costs of the estimators are different. Indeed, if one considers that the cost is the number of calls to a conditional simulator, then it is equal to the final number of iterations and Figure 3.5a and 3.5b present a clear shift: on average the discontinuity will produces $(1/\Delta - 1)N$ iterations for the non-strict random walk and only $(1 - \Delta)N$ for the strict random walk; with $\Delta = 0.4$, this gives approximately 276 more iterations. On the histograms a shift of 250 to 300 is clearly visible in the x axis.

We now check the accuracy of the probability estimators. Firstly, they should be all unbiased. Secondly, for a given N one should see a variance increase from the MVUE of the non-strict random walk of Eq. (3.36) to the pure Poisson estimator (Eq. 3.41) and to the MVUE of the strict random walk (Eq. 3.39).

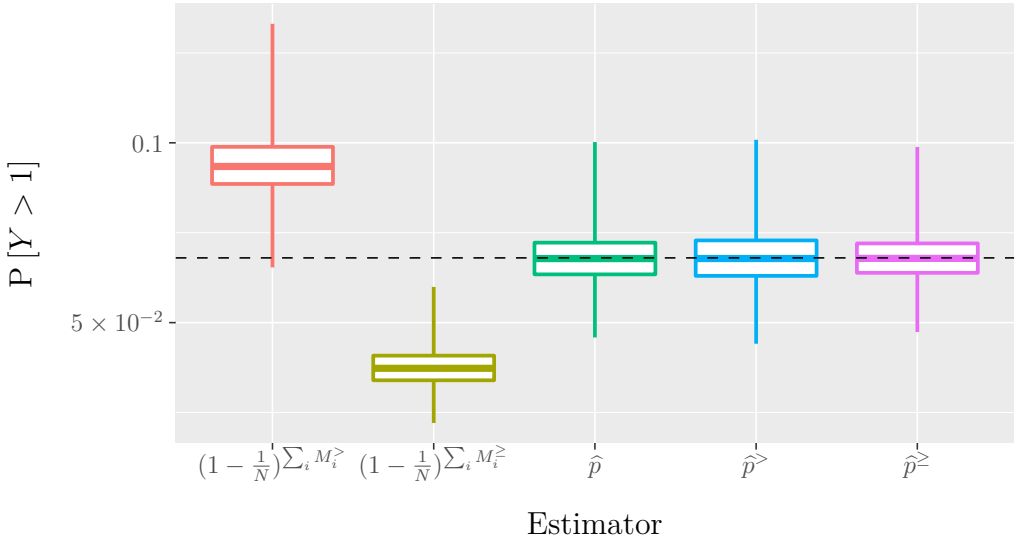


Figure 3.6: Boxplots of the estimation of $p \approx 6.8 \times 10^{-2}$ over 10^4 simulations with $N = 300$, whiskers extending to the extreme values. $\widehat{p}^{>}$: Strict random walk estimator (Eq. 3.39), $\widehat{p}^{\geq}$: Non-strict random walk estimator (Eq. 3.36), $\widehat{p}$: Pure Poisson estimator (Eq. 3.41); $(1 - 1/N)^{\sum M_i}$: estimators if the discontinuity is not taken into account with strict ($^{>}$) and non-strict equality ($^{\geq}$) random walks.

Figure 3.6 shows a boxplot of the three estimators over 10^4 simulations. As an illustration, the estimators computed directly as if Y were continuous are also added to the plot. The horizontal line stands for the reference value calculated with a crude Monte Carlo. The estimated means are 6.81×10^{-2} for the strict random walk estimator, 6.81×10^{-2} for the non-strict one and 6.81×10^{-2} for the pure Poisson one. This is in good agreement with the estimated reference value $p = 6.8 \times 10^{-2}$. Furthermore, the empirical variances are 5.18×10^{-5} for the strict inequality random walk and 4.21×10^{-5} for the pure Poisson estimator while the theoretical values given by Eqs. (3.40) and (3.9) are 5.09×10^{-5} and 4.16×10^{-5}. On the other hand, the probability estimators are clearly not consistent when the discontinuity is not handled properly, *i.e.* when the estimator is computed with the formula valid only in the continuous case (Eq. 3.7).

All together, these numerical results are in good agreement with the theoretical ones.

3.6.2 Discrete random variable

Counting problems are typical cases where the random variable of interest is known to be integer-valued. Indeed, it is shown that many of these problems can be put into the setting of estimating extreme probability [Mitzenmacher and Upfal, 2005, Bezáková et al., 2008,

Botev and Kroese, 2008, Motwani and Raghavan, 2010]. Among others, we focus here on the *Boolean SATisfiability Problem* (*SAT* problem). We do not pretend being competitive against specific *SAT* solvers. Instead, we use this test case because the random variable will have several discontinuity points not only at the origin.

The SAT problem

A SAT problem comprises

1. a binary vector of n *literals* which can be either `TRUE` (=1) of `FALSE` (=0): $\mathbf{x} = (x_1, \cdots, x_n)$ is called a *truth assignment*, *e.g.* $\mathbf{x} = (\texttt{TRUE}, \texttt{TRUE}, \cdots, \texttt{FALSE}) = (1, 1, \cdots, 0)$ and

2. a set of m clauses $\{g_1, \cdots, g_m\}$ expressed as `OR` logical operators (also denoted by $\vee$) on the *literals*, *e.g.*: $g_i = x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_k}$.

The SAT problem in itself is then defined as follows: find an assignment $\mathbf{x}$ such that all clauses are true (*SAT assignment problem*) or count the number of different assignments which satisfy all the clauses (*Sharp-SAT*). The conjunctive normal form (CNF) of a SAT problem is then the product (logical operator `AND` or $\wedge$) of all clauses $F = g_1 \wedge \cdots \wedge g_m$ and both problems can be rewritten:

SAT assignment problem is there at least one $\mathbf{x} \in \{0,1\}^n$ such that $F(\mathbf{x}) = \texttt{TRUE}$?

sharp-SAT find $card(\mathcal{S}) = |\mathcal{S}|$ with $\mathcal{S} = \{\mathbf{x} \in \{0,1\}^n \mid F(\mathbf{x}) = \texttt{TRUE}\}$.

If one considers $\mathbf{X}$ a Uniform random vector on $\{0,1\}^n$, then it is known [Rubinstein and Kroese, 2011] that:

$$p_m = \mathrm{P}\left[\mathbf{X} \in \mathcal{S}\right] = \frac{|\mathcal{S}|}{2^n}.$$

Hence one can build an estimator of $|\mathcal{S}|$ by estimating the (extreme) probability p_m. In order to make use of the results of Section 3.5.2, one can consider the discrete random variable $Y = g(\mathbf{X})$ of the number of clauses satisfied by the assignment $\mathbf{X}$:

$$Y = g(\mathbf{X}) = \sum_{i=1}^{m} g_i(\mathbf{X}). \tag{3.43}$$

Therefore $Y \in [\![0, m]\!]$ and:

$$\mathrm{P}\left[\mathbf{X} \in \mathcal{S}\right] = \mathrm{P}\left[F(\mathbf{X}) = \texttt{TRUE}\right] = \mathrm{P}\left[Y = m\right] = \mathrm{P}\left[Y \geq m\right].$$

Conditional simulations

In order to perform the conditional simulations $\mu^Y(\cdot \mid Y \geq i)$ needed by the random walks, we propose to use the Gibbs sampler described in Section 1.3.3: starting from a sample

$\mathbf{X}^*$ such that $g(\mathbf{X}^*) \geq i$, we re-sample each coordinate sequentially conditionally to the other ones to stay in the right domain.

Practically speaking, to avoid local maxima and improve the convergence of the Markov chain, we do not start from the current $\mathbf{X}$ such that $i = g(\mathbf{X})$. Instead, we pick at random a starting point $\mathbf{X}^*$ in a population already following the target distribution. This population is built *on-the-fly* with all the generated samples $\mathbf{X} \sim \mu^X(\cdot \mid g(\mathbf{X}) \geq j)$ with $j \leq i$ such that $g(\mathbf{X}) \geq i$. Especially each *fail* of a Geometric law will increase its size instead of replacing the previous vector as it is the case in usual Multilevel Splitting method (see also Section A.2).

Finally, we also generate both the strict and the non-strict random walks in the same run. This is to focus on the statistical properties of the number of iterations and of the estimators. Here some $\Delta_d = \mathrm{P}\left[Y > d\right] / \mathrm{P}\left[Y \geq d\right]$ are very small, so that the size of the population for the conditional sampling may become very *small* if one only keeps those starting points strictly above the current threshold.

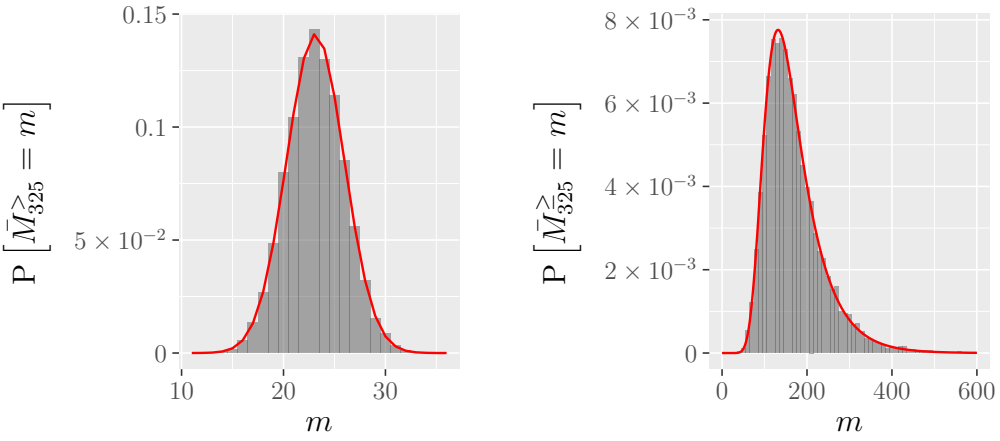
Numerical results

We consider here the *SAT* problem referred to as `uf75-01` on `satlib.org`, also used by Botev and Kroese [2012], who provide a reference value $p = 5.98 \times 10^{-20}$ with a relative error of 0.03%. It has $m = 325$ clauses in dimension $n = 75$. Hence Y is a discrete random variable with up to 325 jump points. We first focus on the number of iterations of the random walks. To do so, we simulate $N = 10^4$ random walks as well as the *pure Poisson correction*. Figure 3.7 plots the histograms of the random number of iterations for each case and theoretical curves with the Δ_d estimated using an other simulation with $N = 50000$.

These plots show a good consistency between numerical results and theoretical formulae from Eqs. (3.32) and (3.33) and Theorem 3.2. Especially with a lot of jump points, the number of iterations are very different from Figure 3.7a to Figure 3.7b (almost hundred times bigger).

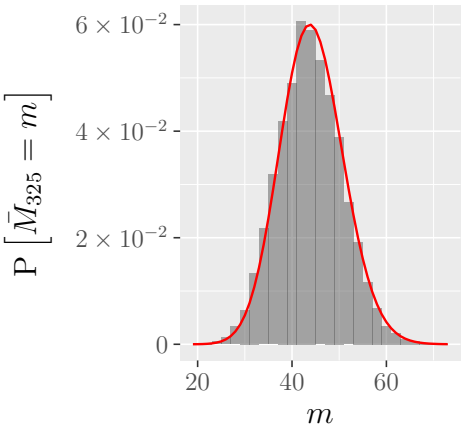
We now focus on the probability estimators. In this scope we also consider the *Smoothed Splitting Method (SSM)* [Cérou et al., 2011], which uses a case-specific continuous auxiliary random variable. The aim of this benchmark is to assess the relevance of using such transformations instead of considering the original random variable with the MVUE we have proposed in Section 3.5.2. We refer the reader to [Cérou et al., 2011] for further details on this transformation. The algorithm is then a usual Multilevel Splitting method with $p_0 = 0.2$. We set N_{SSM} such that the total number of simulated samples for the non-strict random walk and for the SSM are of the same order of magnitude. The non-strict random walk generates on average 170 samples while an AMS with $p_0 = 0.2$ typically generates $(1 - p_0)N \log p / \log p_0$ samples. We set $N = 10^3$ for the random walks, which gives $N_{\mathrm{SSM}} \approx 7712$. Here we set $N_{\mathrm{SSM}} = 8000$.

With a lot of discontinuity points, the differences between the variances of the three estimators is clearer, especially between the non-strict random walk estimator and the



(a) Strict random walk

(b) Non-strict random walk



(c) Pure Poisson correction

Figure 3.7: Histogram over 10^4 realisations of the number of iterations for the strict random walk (Eq. 3.33), the non-strict one (Eq. 3.32) and the pure Poisson *correction* (Theorem 3.2). Y is an integer-valued random variable with up to 325 jump points. The curves show the theoretical distributions with estimated Geometric parameters Δ_d with $N = 50000$.

pure Poisson one. Also the strict inequality estimator has a much bigger variance. Indeed, for some discontinuity points $d \in D_y$, $\Delta_d \approx 10^{-2}$ while $N = 10^3$, which gives coefficients of variations around $1/\sqrt{Np} \approx 32\%$. This is not an issue in the non-strict case as the coefficient of variation of the MVUE of Lemma 3.3 typically scales like $\sqrt{(1-p)/N}$. On the other hand, over the 10^2 simulations, we have an estimation of $\mathrm{card}(D_y) \approx 66$, which gives an upper bound for the squared coefficient of variation in the non-strict case (see Eq. 3.38): 6.83×10^{-2}, while the estimated squared coefficient of variation is 3.21×10^{-2}. Concerning the *SSM* estimator, we have found a coefficient of variation of 0.33 while the theoretical value should be 0.12. As already noticed by Cérou et al. [2011] this is due to a non-perfect implementation of the Multilevel Splitting. This limitation is less visible when keeping the discrete random variable because we save all generated samples and so improve the approximation of the target distribution at each iteration.

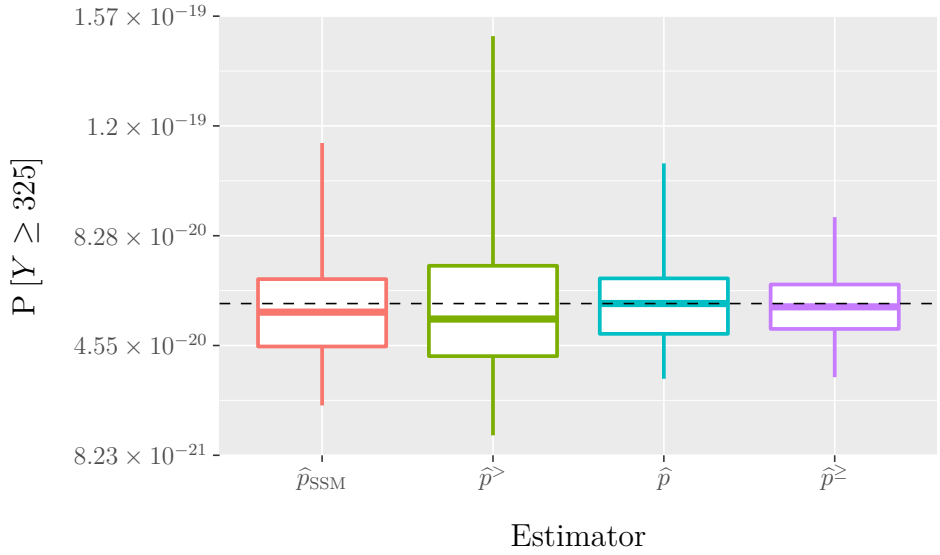


Figure 3.8: Boxplots of the estimation of $p \approx 5.98 \times 10^{-20}$ over 100 simulations with $N = 10^3$, whiskers extending to the extreme values. $\widehat{p}^{>}$: Strict random walk estimator (Eq. 3.39), $\widehat{p}^{\geq}$: Non-strict random walk estimator (Eq. 3.36), $\widehat{p}$: Pure Poisson estimator (Eq. 3.41), $\widehat{p}_{\mathrm{SSM}}$: Smoothed Splitting Method [Cérou et al., 2011] with $N_{\mathrm{SSM}} = 8 \times 10^3$.

Finally, these numerical results with a discrete random variable show a good consistency with the theoretical ones. Also it appears that it may not be relevant to transform the problem to consider a Multilevel Splitting method on a continuous random variable because this can make the conditional simulations harder to approximate. The pure Poisson *correction* is in this context a good trade-off between accuracy and knowledge of the distribution. Furthermore, it is *just* a by-product of the non-strict random walk and so the MVUE can also be computed in the same run. Concerning the strict inequality random walk, it may suffer from two limitations if some Δ_d are very small (typically if $1/\Delta_d$ becomes of the order of magnitude of the total population size N) because 1) the coefficient of variation of the MVUE of Lemma 3.4 is $\approx \sqrt{1/(N\Delta_d)}$, and 2) Markov chain drawing may be *poor* because only few samples will be available in the right domain: at each iteration, only the samples strictly above the current threshold are kept, so on average only $N\Delta_d$ samples will be in the right domain. If Markov chain drawing is used to approximate conditional sampling, then the diversity of the population may decay strongly iterations after iterations, or eventually becomes null (no sample above the threshold). Since Bréhier et al. [2015c] algorithm is based on strict inequality, it suffers from the same limitations and can eventually fail to estimate the sought probability. Thus it seems less robust than the non-strict random walk framework.

3.7 Conclusion

In this chapter, we defined the point process framework for rare event simulation. Indeed, focusing on a random walk defined on the real-valued random variable $Y = g(\mathbf{X})$ we

have been able to show that the optimal (minimal variance) splitting estimator, the Last Particle Algorithm, was indeed a particular non-parallel implementation of a more general estimator, it is the MVUE of the exponential of a Poisson parameter. Thus the optimal splitting estimator writes indeed with a sum of *iid.* realisations of Poisson random variables.

It is noticeable that in this framework, the optimal adaptive splitting does not make any subset any more. In this spirit we showed a Glivenko-Cantelli like theorem, that is the almost sure uniform convergence of the estimated *cdf*: the point process framework lets show that LPA not only produced an estimator of the sought probability $\mathrm{P}\left[g(\mathbf{X}) > q\right]$ but of the *cdf* of Y over $(-\infty, q]$. It also made us slightly modify the quantile estimator proposed in [Guyader et al., 2011] which brings improvement on its bias.

Then the point process framework allowed us to properly handle possible discontinuities in the *cdf* of Y. We showed that this leads to consider either strict or non-strict inequalities in the definition of the Markov chain and were able to recover the distribution of the counting random variables as well as the MVUE in both case. This does not require any previous knowledge on these potential discontinuities.

From a practical point of view, it is also interesting to have an algorithm giving always an estimator with the same distribution, with or without discontinuities. In this scope we proposed also a sub-optimal estimator as a by-product of the non-strict case which has the same distribution with or without discontinuities in the *cdf* of Y.

In Appendix A we will suggest some parallel implementations of the estimators seen in this chapter using the fact that all one requires is indeed to generate *iid.* copies of the increasing random walk. Practically speaking, the strict inequality implementation may be less efficient if conditional sampling has to be done with Markov chain drawing and can even return 0 if some jump points $d \in D_y$ are such that $\Delta_d^{-1} = \mathrm{P}\left[X \geq d\right] / \mathrm{P}\left[X > d\right]$ is of the order of magnitude of the population size N: this is due to the estimation of the Δ_d with crude Monte Carlo. The non-strict implementation prevents from such issues because it generates Geometric random variables.

Finally, considering the global cost of an estimator against its variance, some optimisation may be done to start some random walks only from a given point and/or stop them before the targeted value because all the jumps are not of equal size. This has not been studied here and is let for further research.

Chapter 4

Nested sampling and rare event simulation

This chapter addresses the issue of estimating the expectation of a real-valued random variable of the form $Y = g(\mathbf{X})$ where g is a deterministic function and $\mathbf{X}$ is a random finite- or infinite-dimensional vector. Using results on rare event simulation from Chapter 3, we come up with an other insight on the nested sampling algorithm [Skilling, 2006], precisely that it can be seen as an application of Campbell's theorem on sums over a Poisson process. Especially, it extends its use as follows: first the random variable Y does not need to be bounded any more: it gives the principle of an ideal estimator with an infinite number of terms that is unbiased and always better than a classical Monte Carlo estimator – in particular it has a finite variance as soon as there exists $k \in \mathbb{R} > 1$ such that $\mathrm{E}\left[|Y|^k\right] < \infty$. Moreover we address the issue of nested sampling termination and show that a random truncation of the sum can preserve unbiasedness while increasing the variance only by a factor up to 2 compared to the ideal case. We also build an unbiased estimator with fixed computational budget which supports a Central Limit Theorem and discuss parallel implementation of nested sampling, which can dramatically reduce its running time. Finally we extensively study the case where Y is heavy-tailed.

4.1 Introduction

Nested sampling was introduced in the Bayesian framework by Skilling [2006] as a method for "estimating directly how the likelihood function relates to prior mass". Formally, it builds an approximation for the evidence:

$$Z = \int_\Theta L(\theta)\pi(\theta)\mathrm{d}\theta,$$

where π is the prior distribution, L the likelihood, and $\Theta \subset \mathbb{R}^d$. It is somehow a quadrature formula but in the $[0, 1]$ interval rather than in the original multidimensional space Θ:

$$Z = \int_0^1 Q(P)\mathrm{d}P,$$

where Q is the quantile function which is the generalised inverse of:

$$P(\lambda) = \int_{L(\theta)>\lambda} \pi(\theta)\mathrm{d}\theta.$$

Hence the name *nested sampling* because the initial input space is divided into nested subsets $\{\theta \in \Theta \mid L(\theta) > \lambda\}$. Convergence of the approximation error toward a Gaussian distribution has been proved [Chopin and Robert, 2010] assuming that Q is twice continuously differentiable with its two first derivatives bounded over $[\varepsilon, 1]$ for some $\varepsilon > 0$.

On the other hand estimating a quantity such as $P(\lambda)$ for a given λ is a typical problem arising in rare event probability estimation. In this context, L represents a complex computer code (denoted by g throughout this thesis, not necessarily positive valued nor continuous nor bounded), θ is a vector of parameters ($\mathbf{X}$ in this thesis), and $F_\lambda = \{\theta \in \Theta \mid L(\theta) > \lambda\}$ is the so-called failure domain. As presented in Section 1.3, the idea of writing F_λ as a finite intersection of nested subsets $F_{\lambda_0} \supset \cdots \supset F_{\lambda_n},\ -\infty = \lambda_0 < \cdots < \lambda_n = \lambda$ goes back to Kahn and Harris [1951] and is now referred to as Multilevel Splitting [Garvels, 2000, Cérou and Guyader, 2007] or Subset Simulation [Au and Beck, 2001]. Statistical properties and convergence results have been derived by interpreting the Splitting algorithm in terms of an Interacting Particles System [Cérou et al., 2009, 2012]. Furthermore a particular implementation, sometimes called the *Last Particle Algorithm* (LPA), has gained a lot of attention and Huber and Schott [2011], Huber et al. [2014], Guyader et al. [2011] and Simonnet [2016] have independently proved its link with a Poisson process (see Section 3.2). This algorithm is indeed somehow the one proposed by Skilling [2006, Section 6] but the connection between nested sampling and rare event simulation remained unclear (see Guyader et al. [2011] and the discussion following Huber and Schott [2011] in Bernardo et al. [2011]).

The goal of this chapter is to fill this gap by recovering the nested sampling method from the point process framework defined in Chapter 3. This lets us extend the nested sampling to the estimation of the mean of any real-valued random variable (bounded or not) and brings new theoretical results: 1) the ideal estimator with an infinite number of terms (non truncated nested sampling) is unbiased; 2) the ideal nested sampling estimator is always better than the classical Monte Carlo estimator in terms of variance; and 3) it has a finite variance as soon as a moment of order $k \in (1, \infty)$ exists.

Moreover we address the issue of the nested sampling termination [see Skilling, 2006, Section 7]. Using results on Multilevel Monte Carlo [Giles, 2008, McLeish, 2011, Rhee and Glynn, 2015], we show that one can get an unbiased estimator with a random but a.s. finite number of terms whose variance is only twice the one of the ideal estimator. Note that the recent work on Generalised Adaptive Multilevel Splitting methods by Bréhier et al. [2015c] does not address this issue. It only stands that one can stop the algorithm at any time and still output an unbiased estimator by using the last particles as an approximation of the truncated distribution in a Monte Carlo estimator. We also build an unbiased estimator

with a fixed computational budget which supports a Central Limit Theorem.

All these theoretical results are derived assuming that it is possible to generate samples according to conditional laws when it is required. This is indeed a tough requirement but this problem is well identified and not particular to these randomised estimators [see Roberts, 2011]; especially Skilling [2006], Huber and Schott [2011], Guyader et al. [2011] already acknowledge it and make use of Markov Chain Monte Carlo sampling (see Section 1.3.3). While a lot of ongoing work on nested sampling focus on improving these conditional simulations [*e.g.* Brewer et al., 2011], we focus in this chapter on theoretical statistical properties and suggest a possible solution to the issue of choosing a *bad* stopping criterion.

In section 4.2 we briefly recall the point process framework and derive a new *ideal* (not practically implementable) estimator of $m = \mathrm{E}[Y] = \mathrm{E}[g(\mathbf{X})]$. It is closely related to nested sampling with an infinite number of terms and is compared to the usual Monte Carlo estimator. Section 4.3 proposes two possible estimators based on the *ideal* one. Section 4.4 studies the specific case where $Y = g(\mathbf{X})$ is heavy-tailed and Section 4.5 gives information on practical implementation and numerical results.

4.2 Ideal estimator

From now on we consider a real-valued random variable Y, which can be for instance the output of a mapping $Y = g(\mathbf{X})$, as discussed in the Introduction.

Furthermore for any integrable real-valued random variable Y, one can write $Y = Y_+ - Y_-$ with Y_+ and Y_- non-negative random variables. Then, $\mathrm{E}[Y] = \mathrm{E}[Y_+] - \mathrm{E}[Y_-]$. Thus in the sequel and without loss of generality we assume that Y is a non-negative random variable with law μ^Y. We also assume that Y has a continuous *cdf* F_Y. While we have shown in Section 3.5 how to handle possible discontinuities in the *cdf* of Y we stick to the continuous case for the sake of simplicity and to obtain closed-form formulas. As a matter of fact the following developments could be conducted without this hypothesis. Finally we write p_y instead of $\mathrm{P}[Y > y] = 1 - F_Y(y)$, for any $y \in \mathbb{R}^+$.

4.2.1 Extreme event simulation

In this section we briefly recall the Poisson process framework (see Chapter 3) and show how it can be used to define a mean estimator for a real-valued random variable.

Definition 4.1 (Increasing random walk)**.** *Let $Y_0 = 0$ and define recursively the Markov sequence $(Y_n)_n$ such that*

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

In other words $(Y_n)_n$ is a strictly increasing sequence where each element is generated

conditionally greater than the previous one: $Y_{n+1} \sim \mu^Y(\cdot \mid Y > Y_n)$. Furthermore we showed that is it a Poisson process with mean measure:

$$\forall y \in \mathbb{R},\ \lambda((-\infty, y]) = -\log \mathrm{P}\left[Y > y\right] = -\log\left(1 - \mu^Y((-\infty, y])\right).$$

Thus, the counting random variable of the number of events before $y \in \mathbb{R}$: $M_y = \mathrm{card}\{n \geq 1 \mid Y_n \leq y\}$ follows a Poisson law with parameter $t_y = -\log p_y$.

With *iid.* realisations of Poisson random variables with parameter t_y, the Lehmann-Scheffé theorem states that the minimum-variance unbiased estimator (MVUE) of $p_y = e^{-t_y}$ is

$$\widehat{p_y} = \left(1 - \frac{1}{N}\right)^{\sum_{i=1}^N M_y^i} \tag{4.1}$$

with $(M_y^i)_{i=1}^N$ *iid.* realisations of M_y. It has the following properties:

Proposition 4.1 (Statistical properties of $\widehat{p_x}$)**.**

$$\sup_{y_0 \leq y} |F_N(y_0) - F_Y(y_0)| \xrightarrow[N\to\infty]{a.s.} 0$$
$$\mathrm{var}\left[\widehat{p}_y\right] = p_x^2\left(p_y^{-1/N} - 1\right).$$

with $\forall y_0 \leq y,\ F_N(y_0) = 1 - \widehat{p}_{y_0}$.

Remark 4.1. *The MVUE* $\tilde{t}_y$ *of* $t_y = -\log p_y$ *is* $\sum_{i=1}^N M_y^i/N$*. From this relation one could consider the suboptimal estimator for* p_y*:*

$$\widetilde{p}_y = \left(e^{-\frac{1}{N}}\right)^{\sum_{i=1}^N M_y^i}. \tag{4.2}$$

From the moment-generating function of a Poisson random variable with parameter Nt_y *we get the mean and variance of* $\widetilde{p}_y$*:*

$$\begin{aligned}
\mathrm{E}\left[\widetilde{p}_y\right] &= p_y^{N(1-e^{-1/N})} = p_y + \frac{-p_y \log p_y}{2N} + o\left(\frac{1}{N}\right) \\
\mathrm{var}\left[\widetilde{p}_y\right] &= p_y^{N(1-e^{-2/N})} - p_y^{2N(1-e^{-1/N})} \\
&= \frac{-p_y^2 \log p_y}{N} + \frac{p_y^2 \log p_y}{N^2}\left(\log p_y + 1\right) + o\left(\frac{1}{N^2}\right).
\end{aligned}$$

Hence this suboptimal estimator has a positive bias of order $1/N$*. The variances* $\mathrm{var}\left[\widehat{p}_y\right]$ *and* $\mathrm{var}\left[\widetilde{p}_y\right]$ *differ only from order* $1/N^2$ *and* $\mathrm{var}\left[\widehat{p}_y\right] < \mathrm{var}\left[\widetilde{p}_y\right]$ *as soon as* $p_y < e^{-1}$*.*

4.2.2 Definition of the moment estimator

For now on, we assume that Y is integrable, *i.e.* $\mathrm{E}[Y] < \infty$. Noticing that for a non-negative real-valued random variable with mean $m = \mathrm{E}[Y] = \mathrm{E}[g(\mathbf{X})]$ one has:

$$m = \int_0^\infty p_y \mathrm{d}y, \tag{4.3}$$

the idea is to use the optimal estimator of p_y (see Eq. 4.1) to build an estimator for m.

Remark 4.2. *For any continuous real-valued random variable, one has indeed:*

$$m = \int_0^\infty \mathrm{P}[Y > y]\,\mathrm{d}y - \int_{-\infty}^0 \mathrm{P}[Y < y]\,\mathrm{d}y.$$

In this chapter we focus on the right-hand tail of Y. Eventually all the results are directly adaptable for the general case $Y \in \mathbb{R}$.

From now on we will assume that $N \geq 2$ point processes have been simulated and denote by $(\bar{M}_y)_y$ the counting random variables associated with the marked Poisson process: $\forall y > 0, \bar{M}_y \sim \mathcal{P}(-N \log p_y)$. The sequence $(Y_n)_{n\geq 1}$ is the cumulated one, *i.e.* the combination of the states of the N Markov chains sorted in increasing order. We set $Y_0 = 0$ and then consider the following estimator:

$$\begin{aligned} \widehat{m} &= \int_0^\infty \left(1 - \frac{1}{N}\right)^{\bar{M}_y} \mathrm{d}y \\ &= \sum_{i=0}^{\infty} (Y_{i+1} - Y_i) \left(1 - \frac{1}{N}\right)^i. \end{aligned} \tag{4.4}$$

The second equality comes from the fact that $y \mapsto \bar{M}_y$ is constant equal to i on each interval $[Y_i, Y_{i+1})$: there are 0 event before Y_1, then 1 event before Y_2, precisely at Y_1, etc.

While the first form is easier to analyse because the law of $(\bar{M}_y)_y$ is well determined, the second one paves the way for the practical implementation (see Section 4.3) and clarifies the link with Nested Sampling:

$$\widehat{m} = \sum_{i=1}^{\infty} Y_i \left[\left(1 - \frac{1}{N}\right)^{i-1} - \left(1 - \frac{1}{N}\right)^i\right]. \tag{4.5}$$

This estimator appears as the limit case (sum with an infinite number of terms) of the nested sampling estimator with a deterministic scheme [Skilling, 2006]:

$$\widetilde{m} = \sum_{i=1}^{\infty} Y_i \left(e^{\frac{1-i}{N}} - e^{\frac{-i}{N}}\right) \tag{4.6}$$

with slightly modified weights: $(1 - 1/N)$ instead of $e^{-1/N}$. This is a direct consequence of the fact that an optimal unbiased estimator for e^{-t_y} is not $e^{-\tilde{t}_y}$ (see Section 4.2.1, Remark 4.1).

Furthermore, Eq. (4.5) rewrites:

$$\widehat{m} = \sum_{i=1}^{\infty} \frac{Y_i}{N} \left(1 - \frac{1}{N}\right)^{i-1}. \tag{4.7}$$

Remember that $(Y_n)_n$ is a Poisson process with mean measure:

$$\lambda((-\infty, y]) = -N \log\left(1 - \mu^Y((-\infty, y])\right),$$

λ is absolutely continuous with respect to μ^Y and so according to the Radon–Nikodym theorem one has:

$$\mathrm{d}\lambda(y) = N \frac{\mathrm{d}\mu^Y(y)}{\mathrm{P}\left[Y > y\right]} = N \frac{\mathrm{d}\mu^Y(y)}{p_y}.$$

Then the expectation of Y can be rewritten:

$$\mathrm{E}\left[Y\right] = \int_0^{\infty} y \mathrm{d}\mu^Y(y) = \int_0^{\infty} \frac{y}{N} p_y \mathrm{d}\lambda(y).$$

Here the Campbell's theorem [see for example Kingman, 1992, p. 28] can be applied to the estimation of the mean: let $\widehat{m}_C$ be defined by:

$$\widehat{m}_C = \sum_{i=1}^{\infty} \frac{Y_i}{N} \mathrm{P}\left[Y > Y_i\right] = \sum_{i=1}^{\infty} h(Y_i) \tag{4.8}$$

with $h(y) = y p_y / N$, then it insures that $\widehat{m}_C$ is absolutely convergent with probability one *iff.*:

$$\int_0^{\infty} \min\left(h(y), 1\right) \mathrm{d}\lambda(y) = \frac{1}{N} \int_0^{\infty} \min\left(y p_y, 1\right) \frac{1}{p_y} \mathrm{d}\mu^Y < \infty.$$

If this condition holds, then:

$$\mathrm{E}\left[e^{\theta \widehat{m}_C}\right] = \exp\left(\int_0^{\infty} (e^{\theta h(y)} - 1) \mathrm{d}\lambda(y)\right). \tag{4.9}$$

This condition holds since one has:

$$\int_0^{\infty} \min\left(h(y), 1\right) \mathrm{d}\lambda(y) \leq \int_0^{\infty} h(y) \mathrm{d}\lambda(y) = \mathrm{E}\left[Y\right] < \infty$$

and one obtains:

$$\begin{aligned} \mathrm{E}\left[\widehat{m}_C\right] &= \int_0^{\infty} h(y) \mathrm{d}\lambda(y) = \int_0^{\infty} y \mathrm{d}\mu^Y(y) = \mathrm{E}\left[Y\right] \\ \mathrm{var}\left[\widehat{m}_C\right] &= \int_0^{\infty} h(y)^2 \mathrm{d}\lambda(y) = \frac{1}{N} \int_0^{\infty} y^2 p_y \mathrm{d}\mu^Y(y). \end{aligned} \tag{4.10}$$

Then Eq. (4.7) can be seen as an approximation of Eq. (4.8) where the terms $\mathrm{P}\left[Y > Y_i\right]$ are estimated with the Poisson process itself. Furthermore this estimator does not require the finiteness of $\mathrm{var}\left[Y\right]$ to have a finite variance. Indeed, the Markov inequality gives

$\forall y \in \mathbb{R}$, $\mathrm{E}[Y] \geq y p_y$ such that the variance can be bounded from above:

$$\mathrm{var}[\widehat{m}_C] = \frac{1}{N}\int_0^\infty y^2 p_y \mathrm{d}\mu^Y(y) \leq \frac{1}{N}\int_0^\infty y\,\mathrm{E}[Y]\,\mathrm{d}\mu^Y(y) = \frac{\mathrm{E}[Y]^2}{N}.$$

Hence the finiteness of $\mathrm{E}[Y]$ is a sufficient condition for the finiteness of $\mathrm{var}[m_C]$. In Corollary 4.1 we will show that $\widehat{m}$ requires the finiteness of a moment of order $1+\varepsilon$, $\varepsilon > 0$, for Y, to have a finite variance.

Proposition 4.2 (Statistical properties of $\widehat{m}$)**.**

$$\mathrm{E}[\widehat{m}] = m \tag{4.11}$$

$$\begin{aligned}\mathrm{var}[\widehat{m}] &= 2\int_0^\infty \int_0^y p_y p_{y'}^{1-1/N} \mathrm{d}y'\mathrm{d}y - m^2 \qquad (4.12)\\ &= \sum_{k=1}^\infty \frac{2}{N^k}\int_0^\infty \int_0^y p_y p_{y'} \frac{|\log p_{y'}|^k}{k!}\mathrm{d}y\mathrm{d}y'\\ &= \frac{2}{N}\int_0^\infty \int_0^y p_y p_{y'} |\log p_{y'}| \mathrm{d}y\mathrm{d}y' + o\left(\frac{1}{N}\right). \qquad (4.13)\end{aligned}$$

Proof. One has:

$$\mathrm{E}[\widehat{m}] = \int_0^\infty \mathrm{E}\left[\left(1-\frac{1}{N}\right)^{\bar{M}_y}\right]\mathrm{d}x = \int_0^\infty p_y \mathrm{d}y.$$

For the variance, one uses the fact that, for $y > y'$, $\bar{M}_y - \bar{M}_{y'}$ and $\bar{M}_{y'}$ are independent to expand $\mathrm{E}[\widehat{m}^2]$:

$$\begin{aligned}\mathrm{E}\left[\widehat{m}^2\right] &= 2\int_0^\infty \int_0^y \mathrm{E}\left[\left(1-\frac{1}{N}\right)^{\bar{M}_y+\bar{M}_{y'}}\right]\mathrm{d}y'\mathrm{d}y\\ &= \int_0^\infty \int_0^y \mathrm{E}\left[\left(1-\frac{1}{N}\right)^{\bar{M}_y-\bar{M}_{y'}}\left(1-\frac{1}{N}\right)^{2\bar{M}_{y'}}\right]\mathrm{d}y'\mathrm{d}y.\end{aligned}$$

Furthermore the renewal property of a Poisson process gives $\bar{M}_y - \bar{M}_{y'} \sim \mathcal{P}(-N\log(p_y/p_{y'}))$ and is independent of $\bar{M}_y$. Eventually one can conclude using the results of Proposition 4.1. □

Remark 4.3. *As a matter of comparison, $\widetilde{m}$ the original ideal nested sampling estimator can also be written $\widetilde{m} = \int_0^\infty \tilde{p}_y \mathrm{d}y$. Then Remark 4.1 allows us to conclude that $\widetilde{m}$ has a positive bias of order 1/N.*

Proposition 4.3 (Finiteness of $\mathrm{var}[\widehat{m}]$)**.**

$$\forall N \geq 2,\ \mathrm{var}[\widehat{m}] \leq \frac{2}{1+1/N}\,\mathrm{E}\left[Y^{1+1/N}\right]^{2/(1+1/N)}.$$

Proof. Starting from the expression of the variance found in Proposition 4.2:

$$\mathrm{var}[\widehat{m}] = 2\int_0^\infty p_y \int_0^y p_{y'}^{1-1/N}\mathrm{d}y'\mathrm{d}y - \mathrm{E}[Y]^2,$$

we make use of Hölder's inequality:

$$\int_0^y p_{y'}^{1-1/N} \mathrm{d}y' \leq \left(\int_0^y \mathrm{d}y'\right)^{1/N} \left(\int_0^y p_{y'} \mathrm{d}y'\right)^{1-1/N} \leq y^{1/N} \left(\int_0^\infty p_{y'} \mathrm{d}y'\right)^{1-1/N}$$
$$\leq y^{1/N} \,\mathrm{E}\left[Y\right]^{1-1/N}.$$

And therefore:

$$\mathrm{var}\left[\widehat{m}\right] \leq \frac{2}{1+1/N} \,\mathrm{E}\left[Y\right]^{1-1/N} \mathrm{E}\left[Y^{1+1/N}\right].$$

Using Hölder's inequality again, one gets:

$$\mathrm{var}\left[\widehat{m}\right] \leq \frac{2}{1+1/N} \,\mathrm{E}\left[Y^{1+1/N}\right]^{\frac{2}{1+1/N}}.$$

□

Corollary 4.1 (Value of N)**.** *Let $\varepsilon > 0$, if $\mathrm{E}\left[Y^{1+\varepsilon}\right] < \infty$ then for any $N \geq 1/\varepsilon$, $\widehat{m}$ has a finite variance.*

While the usual Monte Carlo estimator requires the finiteness of $\mathrm{E}\left[Y^2\right]$ to have a finite variance, this estimator only requires the finiteness of a moment of order $1+\varepsilon$. This is especially interesting when Y is heavy-tailed and this case is further investigated in Section 4.4.

4.2.3 Comparison with classical Monte Carlo

As the finiteness condition of the variance of $\widehat{m}$ is much weaker than for a naive Monte Carlo estimator, one can expect a globally lower variance. This result is shown in Proposition 4.4. We first recall the crude Monte Carlo estimator:

$$\widehat{m}_{MC} \overset{\mathrm{def}}{=} \frac{1}{N} \sum_{i=1}^{N} Y_i \tag{4.14}$$

with $(Y_i)_{i=1}^N$ *iid.* random variables with law μ^Y.

Proposition 4.4. *For any $N \geq 2$,* $\mathrm{var}\left[\widehat{m}\right] \leq \mathrm{var}\left[\widehat{m}_{MC}\right]$.

Proof. On the one hand one has:

$$N \,\mathrm{var}\left[\widehat{m}_{MC}\right] + m^2 = 2\int_0^\infty y p_y \mathrm{d}y,$$

and on the other hand one can write:

$$N \,\mathrm{var}\left[\widehat{m}\right] + m^2 = 2\int_0^\infty p_y \int_0^y p_{y'} \left[N(p_{y'}^{-1/N} - 1) + 1\right] dy' dy.$$

Considering $f : p \mapsto p\left[N(p^{-1/N} - 1) + 1\right]$, we have $f(1) = 1$ and:

$$f'(p) = (N-1)(p^{-1/N} - 1) \geq 0,\ \forall p \in [0,1].$$

Thus: $\forall p \in [0,1], f(p) \leq 1$. Therefore:

$$N \operatorname{var}[\widehat{m}] + m^2 \leq 2\int_0^\infty y p_y \mathrm{d}y$$

which shows that $\operatorname{var}[\widehat{m}] \leq \operatorname{var}[\widehat{m}_{MC}]$. □

Thus the *ideal* nested sampling estimator with corrected weights Eq. (4.5) is always better than classical Monte Carlo in terms of variance and especially does not require the finiteness of the second-order moment of Y to have a finite variance.

4.3 Randomised unbiased estimator

The ideal estimator Eq. (4.4) defined in Section 4.2 is not directly usable as it requires to simulate an infinite number of terms in sum (4.4). While the usual nested sampling implementations propose to stop the algorithm either after a given number of iterations, or according to some criterion estimated at each iteration, we propose a randomised unbiased estimator using recent results on paths simulation.

4.3.1 Definition

We are facing the issue of estimating $\mathrm{E}[\widehat{m}]$ while it is not possible to generate such a $\widehat{m}$ in a finite computer time. This problem is well identified in the field of Stochastic Differential Equations (SDE) where one often intends to compute the expectation of a path functional while only discrete-time approximations are available. Recently there have been two major breakthroughs that address this issue: first the Multilevel Monte Carlo (MLMC) method [Giles, 2008] has introduced the idea of combining *intelligently* different biased estimators (levels of approximations) to speed up the convergence and reduce the bias; then McLeish [2011] and Rhee and Glynn [2015] have introduced a general approach to constructing unbiased estimator based on a family of biased ones. Basically in our context it randomises the number of simulated steps of the Markov chain, and slightly modifies the weights of the nested sampling to *remove the bias* of the final estimator.

More precisely let us consider the truncated estimators $(\widehat{m}_n)_{n\geq 1}$:

$$\widehat{m}_n = \int_0^{Y_n} \left(1 - \frac{1}{N}\right)^{\bar{M}_y} \mathrm{d}y = \sum_{i=0}^{n-1} (Y_{i+1} - Y_i)\left(1 - \frac{1}{N}\right)^i$$

and T a non-negative integer-valued random variable independent of $(Y_n)_{n\in\mathbb{N}}$ such that

$\forall i \in \mathbb{N}, \mathrm{P}\left[T \geq i\right] \overset{\text{def}}{=} \beta_i > 0$; one builds the following estimator (with $\widehat{m}_0 = 0$):

$$\begin{aligned}\widehat{Z} &= \sum_{n=0}^{\infty} \frac{\widehat{m}_{n+1} - \widehat{m}_n}{\mathrm{P}\left[T \geq n\right]} \mathbb{1}_{T \geq n} = \sum_{n=0}^{T} \frac{\widehat{m}_{n+1} - \widehat{m}_n}{\mathrm{P}\left[T \geq n\right]} \\ &= \sum_{n=0}^{\infty} \left(Y_{n+1} - Y_n\right) \left(1 - \frac{1}{N}\right)^n \frac{\mathbb{1}_{T \geq n}}{\mathrm{P}\left[T \geq n\right]}. \end{aligned} \tag{4.15}$$

Remark 4.4. *The notation $\widehat{Z}$ might seem a bit confusing since Z is used in the Introduction for the evidence as in [Skilling, 2006]. This is to keep consistency with Rhee and Glynn [2015] notations where the randomising procedure comes from.*

Proposition 4.5 (Statistical properties of $\widehat{Z}$)**.**

$$\begin{aligned}\mathrm{E}\left[\widehat{Z}\right] &= m \\ \mathrm{var}\left[\widehat{Z}\right] &= \sum_{i=0}^{\infty} q_{i,N} \beta_i^{-1} - m^2\end{aligned}$$

with:

$$q_{i,N} = 2\left(1 - \frac{1}{N}\right)^{2i} \int_0^{\infty} \int_{y'}^{\infty} p_y p_{y'}^{N-1} \frac{\left[-N \log p_{y'}\right]^i}{i!} \mathrm{d}y \mathrm{d}y'. \tag{4.16}$$

Proof. For the unbiasedness we start from last formulation in Eq. (4.15) for $\widehat{Z}$. Then one uses the fact that T and $(Y_i)_i$ are independent. Finally, Eq. (4.4) and Proposition 4.2 let conclude: $\mathrm{E}\left[\widehat{Z}\right] = m$.

For the second-order moment, we use the fact that $\widehat{Z}$, like $\widehat{m}$, can be written with an integral:

$$\widehat{Z} = \int_0^{\infty} \left(1 - \frac{1}{N}\right)^{\bar{M}_y} \frac{\mathbb{1}_{T \geq \bar{M}_y}}{\mathrm{P}\left[T \geq \bar{M}_y\right]} \mathrm{d}y$$

and apply the same reasoning as for $\mathrm{E}\left[\widehat{m}^2\right]$: given $y > y'$, the random variables $\bar{M}_y - \bar{M}_{y'}$, $\bar{M}_{y'}$ and T are independent, which brings:

$$\begin{aligned}&\mathrm{E}\left[\left(1 - \frac{1}{N}\right)^{\bar{M}_y + \bar{M}_{y'}} \frac{\mathbb{1}_{T \geq \bar{M}_y}}{\mathrm{P}\left[T \geq \bar{M}_y\right]} \frac{\mathbb{1}_{T \geq \bar{M}_{y'}}}{\mathrm{P}\left[T \geq \bar{M}_{y'}\right]}\right] \\ &\qquad = \mathrm{E}\left[\left(1 - \frac{1}{N}\right)^{\bar{M}_y - \bar{M}_{y'}} \left(1 - \frac{1}{N}\right)^{2\bar{M}_{y'}} \beta_{\bar{M}_{y'}}^{-1} \frac{\mathbb{1}_{T \geq \bar{M}_y}}{\mathrm{P}\left[T \geq \bar{M}_y\right]}\right] \\ &\qquad = \mathrm{E}\left[\left(1 - \frac{1}{N}\right)^{\bar{M}_y - \bar{M}_{y'}} \left(1 - \frac{1}{N}\right)^{2\bar{M}_{y'}} \beta_{\bar{M}_{y'}}^{-1}\right] \\ &\qquad = \frac{p_y}{p_{y'}} \sum_{i=0}^{\infty} e^{N \log p_{y'}} \frac{\left[-N \log p_{y'} (1 - 1/N)^2\right]^i}{i!} \beta_i^{-1} \\ &\qquad = \sum_{i=0}^{\infty} p_y p_{y'}^{N-1} \frac{\left[-N \log p_{y'} (1 - 1/N)^2\right]^i}{i!} \beta_i^{-1}.\end{aligned}$$

Then using this equality in $\mathrm{E}\left[\widehat{Z}^2\right]$ gives the solution. □

The asymptotic behaviour of the sequence $(q_{i,N})_i$ will drive the possible choices for the randomising distribution $(\beta_i)_i$: $\mathrm{var}\left[\widehat{Z}\right]$ to remain finite implies that $q_{i,N}\beta_i^{-1} \to 0$ when $i \to \infty$.

Lemma 4.1. *The sequence $(q_{i,N})_i$ goes to 0 at least at exponential rate. Furthermore, if Y has density f_Y such that $\|f_Y\|_\infty < \infty$, it is also bounded from below by an exponentially decreasing sequence.*

Proof. Let $\varepsilon > 0$ be such that $\mathrm{E}\left[Y^{1+\varepsilon}\right] < \infty$, $N \in \mathbb{N} \mid N > 1/\varepsilon$ and $i \geq 0$. We further extend the definition of $\mathrm{var}\left[\widehat{m}\right]$ given in Proposition 4.1, Eq. (4.12) for any $N \in \mathbb{R}$. Proof of Proposition 4.3 is based on Hölder's inequality and still holds in this case, and so for Corollary 4.1. Hence, according to Corollary 4.1: $\exists N' \in \mathbb{R}$ such that $N' < N$ and $\mathrm{var}\left[\widehat{m}\right](N') < \infty$. Furthermore, given y and y' one can write:

$$p_y p_{y'}^{N-1}(-\log p_{y'})^i = p_y p_{y'}^{1-1/N'} p_{y'}^{N+1/N'-2}(-\log p_{y'})^i.$$

Moreover the function $p : (0,1) \mapsto p^{N+1/N'-2}(-\log p)^i$ is bounded above by $e^{-i}i^i(N + 1/N' - 2)^{-i}$. Using the Stirling lower bound $i! \geq i^i e^{-i}\sqrt{2\pi i}$ we can write:

$$p_y p_{y'}^{N-1}(-\log p_{y'})^i \leq p_y p_{y'}^{1-1/N'} \frac{i!}{\sqrt{2\pi i}(N + 1/N' - 2)^i}.$$

Finally, this inequality brings:

$$q_{i,N} \leq \mathrm{var}\left[\widehat{m}\right](N') \left(\frac{N(1-1/N)^2}{N + 1/N' - 2}\right)^i \frac{1}{\sqrt{2\pi i}}$$

and $(N + 1/N - 2)/(N + 1/N' - 2) < 1$, which concludes the first part of the proof.

Let us now assume that Y has a bounded density f_Y. One has:

$$q_{i,N} = 2\int_0^\infty \int_0^y p_y p_{y'}^{N-1} \frac{[-N \log p_{y'}(1-1/N)^2]^i}{i!} \mathrm{d}y' \mathrm{d}y.$$

Denote y_L the left end point of Y (remember that Y is non-negative valued so that $y_L \geq 0$). Then:

$$q_{i,N} \geq 2\int_{y_L}^\infty \int_{y_L}^y p_y p_{y'}^{N-1} \frac{[-N \log p_{y'}(1-1/N)^2]^i}{i!} \mathrm{d}y' \mathrm{d}y.$$

We then consider the change of variable $u = -\log p_y$ and $u' = -\log p_{y'}$; for all $i \geq 1$

one has:

$$\begin{aligned} q_{i,N} &\geq \frac{2}{\|f_Y\|_\infty^2}\left(1-\frac{1}{N}\right)^{2i}\int_0^\infty e^{-2u}\int_0^u \frac{e^{-Nu'}(Nu')^i}{i!}\mathrm{d}u'\mathrm{d}u \\ &\geq \frac{2}{\|f_Y\|_\infty^2}\left(1-\frac{1}{N}\right)^{2i}\int_0^\infty e^{-2u}\frac{1}{N}\sum_{k=i+1}^{\infty}\frac{e^{-Nu}(Nu)^k}{k!}\mathrm{d}u \\ &\geq \frac{2}{\|f_Y\|_\infty^2}\frac{1}{N(N+2)}\left(1-\frac{1}{N}\right)^{2i}\sum_{k=i+1}^{\infty}\left(\frac{N}{N+2}\right)^k \\ q_{i,N} &\geq \frac{1}{(N+2)\|f_Y\|_\infty^2}\left[\frac{N}{N+2}\left(1-\frac{1}{N}\right)^2\right]^i. \end{aligned}$$

□

Then it appears that the Geometric distribution plays a key role, as already noted by McLeish [2011]. Hence we provide some theoretical results assuming that T is a geometric random variable.

Proposition 4.6. *If* $\mathrm{P}\left[T \geq n\right] = e^{-\beta n}$, $\beta > 0$, *then:*

$$\mathrm{var}\left[\widehat{Z}\right] = 2\int_0^\infty\int_0^y p_y p_{y'}^{1-\gamma(\beta,N)^{-1}}\mathrm{d}y'\mathrm{d}y - m^2 \tag{4.17}$$

with $\gamma(\beta,N) = N/(1+(e^\beta-1)(N-1)^2)$.

Proof. Let $\alpha > 0$ be such that $(1-1/N) = e^{-\alpha}$. The argument is the same as the one in Proposition 4.5. One has:

$$\mathrm{E}\left[\widehat{Z}^2\right] = 2\int_0^\infty\int_0^y \mathrm{E}\left[e^{-\alpha(\bar{M}_y-\bar{M}_{y'})}e^{(\beta-2\alpha)\bar{M}'_y}\right]\mathrm{d}y'\mathrm{d}y = 2\int_0^\infty\int_0^y p_y p_{y'}^{1-\gamma(\beta,N)^{-1}}\mathrm{d}y'\mathrm{d}y$$

with:

$$\frac{N}{\gamma(\beta,N)} = 2N - N^2 + e^\beta(N-1)^2 = 1+(N-1)^2(e^\beta-1).$$

□

This expression is indeed the same as the one of Proposition 4.2 with the function $\gamma(\beta,N)$ instead of N. Hence the greater γ the smaller $\mathrm{var}\left[\widehat{Z}\right]$. Furthermore one has directly all the results from Section 4.2.2, especially the finiteness conditions for the variance given in Proposition 4.3 and Corollary 4.1, replacing N by $\gamma(\beta,N)$.

While there is no value of β minimising $\mathrm{var}\left[\widehat{Z}\right]$ at a given N (the smaller β the smaller the variance of the randomised estimator $\widehat{Z}$), there is an optimal value of N for a given β, *i.e.* for an estimator with an almost surely finite number: $N = \sqrt{1+\mathrm{E}\left[T\right]}$. One can reverse this relation, which gives:

$$\beta_{\mathrm{app}} \stackrel{\mathrm{def}}{=} \log\left(1+1/(N^2-1)\right). \tag{4.18}$$

Corollary 4.2. *Let $N \geq 2$ and $\mathrm{P}[T \geq n] = e^{-n\,\beta_{app}(N)}$, then:*

$$\mathrm{var}\left[\widehat{Z}\right](N) = \mathrm{var}\left[\widehat{m}\right]\left(\tfrac{N+1}{2}\right) \approx 2\,\mathrm{var}\left[\widehat{m}\right](N). \tag{4.19}$$

Proof. Noticing that for any $N \geq 2$, one has $\gamma(\beta_{\mathrm{app}}(N), N) = (N+1)/2$ gives the first equality. Then the fact $\mathrm{var}\left[\widehat{m}\right]$ typically scales with $1/N$ (see Eq. 4.13) gives the approximation. □

This means that instead of choosing an arbitrary stopping criterion for nested sampling, randomising the number of iterations and computing $\widehat{Z}$ allows for keeping an unbiased estimator without increasing drastically the variance (factor up to 2, reached with suboptimal implementation of Corollary 4.2). This result will be illustrated in the examples of Section 4.5.

4.3.2 Convergence rate

Throughout this thesis we consider that the computational cost for generating an estimator is the number of simulated samples. Since we have assumed that it is possible to generate directly according to any conditional distribution, it is here the number of calls to a simulator of a conditional law. Later for practical implementation, the conditional simulations will be performed using MCMC algorithms (see Section 1.3.3), *i.e.* one sample will require several calls to a generator of μ^Y. The cost will then be modified accordingly.

Proposition 4.7. *Let τ be the random variable of the number of samples required to generate $\widehat{Z}$. One has $\tau = N + T$.*

Proof. If $T = 0$ then no other simulation is done other than the first element of each Markov chain, *i.e.* that N simulations are done. Then each step requires the simulation of the next stopping time, *i.e.* one simulation. Finally, this brings $\tau = N + T$. □

Corollary 4.3 (Convergence rate of $\widehat{Z}$)**.** *For any non-negative integer-valued randomising variable T such that $\mathrm{E}[T] < \infty$ and $\forall i \in \mathbb{N},\ \mathrm{P}[T \geq i] > 0$, one has:*

$$\mathrm{E}[\tau] \cdot \mathrm{var}\left[\widehat{Z}\right] \geq 2q_{1,2} + O\left(\frac{1}{N}\right),\ N \to \infty. \tag{4.20}$$

Proof. Note that $\mathrm{var}\left[\widehat{m}\right] = \sum_{i=0}^{\infty} q_{i,N} - m^2$. Hence, one has $\mathrm{var}\left[\widehat{Z}\right] > \mathrm{var}\left[\widehat{m}\right]$ because $\mathrm{var}\left[\widehat{Z}\right] = \mathrm{var}\left[\widehat{m}\right] \Leftrightarrow \forall i \in \mathbb{N},\ \beta_i = \mathrm{P}[T \geq i] = 1$ and $\mathrm{E}[\tau] > N$ because $\mathrm{E}[\tau] = N \Leftrightarrow \mathrm{E}[T] = 0$ while $\forall i \in \mathbb{N},\ \mathrm{P}[T \geq i] > 0$. Furthermore, the power series expansion of the exponential function and the dominated convergence theorem let us rewrite $\mathrm{var}\left[\widehat{m}\right]$:

$$\mathrm{var}\left[\widehat{m}\right] = \sum_{i=1}^{\infty} 2 \int_0^{\infty} \int_{y'}^{\infty} p_y p_{y'} \frac{(-\log p_{y'})^i}{N^i i!} \mathrm{d}y \mathrm{d}y'$$

$$\mathrm{var}\left[\widehat{m}\right] = \sum_{i=1}^{\infty} q_{i,2} \left(\frac{2}{N}\right)^i$$

which gives: $\mathrm{var}\left[\widehat{m}\right] = 2q_{1,2}/N + O\left(1/N^2\right)$. All together, these inequalities complete the proof. □

If the inequality (4.20) is close to an equality then $\widehat{Z}$ has a canonical square-root convergence rate (as a function of the computational cost). However there is no guarantee on this rate of convergence. Especially Corollary 4.4 below shows that it is not the case when T has a geometric distribution.

Corollary 4.4. *If T is a Geometric random variable such that $\forall n \in \mathbb{N},\ \mathrm{P}\left[T \geq n\right] = e^{-\beta n}$ with $\beta = \Theta(1/N^{1+\varepsilon})$, $\varepsilon \geq 0$, then:*

$$\begin{cases} \mathrm{E}\left[\tau\right] \cdot \mathrm{var}\left[\widehat{Z}\right] = \Theta\left(N\right) & \varepsilon \in [0,1] \\ \mathrm{E}\left[\tau\right] \cdot \mathrm{var}\left[\widehat{Z}\right] = \Theta\left(N^{\varepsilon}\right) & \varepsilon > 1 \end{cases}$$

with $\beta = \Theta(N^{-1-\varepsilon})$ meaning:

$$\exists k_1 > 0,\ k_2 > 0,\ N_0 \geq 1 \mid \forall N \geq N_0,\ k_1 N^{-1-\varepsilon} \leq \beta \leq k_2 N^{-1-\varepsilon}.$$

Proof. Denote $B = 1/(e^{\beta} - 1)$; one has:

$$\frac{N+B}{\gamma(B,N)} = N + \frac{B}{N} + \frac{N^2}{B} - 1 - \frac{2N}{B} + \frac{1}{B} + \frac{1}{N}.$$

With $\beta = \Theta(1/N^{1+\varepsilon})$, $\varepsilon \geq 0$, one has $B \sim 1/\beta \sim N^{1+\varepsilon}$. Finally, this gives:

$$\frac{N+B}{\gamma(B,N)} \sim N + N^{\varepsilon} + N^{1-\varepsilon} + O(1),$$

which concludes the proof. □

Hence the unbiased randomised estimator of Corollary 4.2 with $\beta = \beta_{\mathrm{app}} = \Theta(1/N^2)$ does not have a canonical square-root convergence rate. Furthermore, even though the realisation of the geometric random variable gives a small number of iterations, one may want to run the algorithm longer to probe the tail of the random variable Y to make sure that no important part is missing [Skilling, 2006]. This is why the idea behind randomised estimators is to average several replicas of $\widehat{Z}$ because it will somehow average the quantities $\mathbb{1}_{T \geq n} / \mathrm{P}\left[T \geq n\right]$ in Eq. (4.15). More precisely, let $G(c)$ be the random variable of the number of simulations of $\widehat{Z}$ one can afford with a computational budget c:

$$G(c) = \max\{n \geq 0 \mid \sum_{i=1}^{n} \tau_i \leq c\}$$

where τ_i is the computational effort required to generate the i^{th}-sample $\widehat{Z}_i$, one considers the following estimator:

$$\zeta(c) = \frac{1}{G(c)} \sum_{i=1}^{G(c)} \widehat{Z}_i. \tag{4.21}$$

In this setting Glynn and Whitt [1992] showed a CLT-like result:

$$c^{1/2}(\zeta(c) - \mathrm{E}\left[\widehat{Z}\right]) \xrightarrow[c\to\infty]{\mathcal{L}} (\mathrm{E}\left[\tau\right] \cdot \mathrm{var}\left[\widehat{Z}\right])^{1/2}\mathcal{N}(0,1). \tag{4.22}$$

Hence in our context one has to tune $(\beta_i)_i$ and N to minimise the product $\mathrm{E}\left[\tau\right] \cdot \mathrm{var}\left[\widehat{Z}\right]$.

4.3.3 Optimal randomisation

Since T is a non-negative random variable one has $\mathrm{P}\left[T \geq 0\right] = \beta_0 = 1$. Let $\mathcal{C} = \{(\beta_i)_i \in (0,1]^{\mathbb{N}} \mid \beta_0 = 1 \text{ and } \forall i \in \mathbb{N}, \beta_{i+1} \leq \beta_i\}$; we intend to solve the optimisation problem:

$$\underset{\substack{(\beta_i)_i\in\mathcal{C}\\ N\in[\![2,\infty)}}{\operatorname{argmin}} \mathrm{E}\left[\tau\right] \cdot \mathrm{var}\left[\widehat{Z}\right] = \underset{\substack{(\beta_i)_i\in\mathcal{C}\\ N\in[\![2,\infty)}}{\operatorname{argmin}} \left(N - 1 + \sum_{i=0}^{\infty}\beta_i\right)\left(\sum_{i=0}^{\infty} q_{i,N}\beta_i^{-1} - m^2\right) \tag{4.23}$$

where the $(q_{i,N})_i$ are given by Eq. (4.16). Furthermore, one can rewrite the $(q_{i,N})_i$ assuming that Y has a density $f_Y > 0$. Indeed in this context Y_n has a density f_n such that:

$$\forall n \geq 1,\ f_n(y) = N\frac{p_y^{N-1}(-N\log p_y)^{n-1}}{(n-1)!}f_Y(y).$$

This gives:

$$\forall i \in \mathbb{N},\ q_{i,N} = 2\left(1 - \frac{1}{N}\right)^{2i}\frac{\mathrm{E}\left[\mathcal{R}(Y_{i+1})\right]}{N}$$

with $\mathcal{R}(y) = \int_y^\infty p_u \mathrm{d}u / f_Y(y)$. Hence we further assume that $(q_{i,N})_i$ is decreasing, which is the case for a Pareto random variable (see Section 4.4.1) and at least for any distribution for which $\mathcal{R}$ is non-increasing like exponential and uniform distributions. In this context Proposition 4.8 gives the optimal distribution for T for a given N.

Proposition 4.8 (Optimal distribution for T)**.** *If $(q_{i,N})_{i\geq 1}$ is decreasing then the optimal distribution $(\beta_i^*)_i$ for T is given by:*

$$\forall i \in [\![0, i_0]\!], \beta_i^* = 1$$
$$\forall i > i_0, \beta_i^* = \sqrt{\frac{N+i_0}{S_0}}\sqrt{q_{i,N}}$$

with $i_0 = \min\{i \in \mathbb{N} \mid \sum_{j=0}^{i} q_{j,N} - m^2 > (N+i)q_{(i+1),N}\}$ and $S_0 = \sum_{j=0}^{i_0} q_{j,N} - m^2$.

Proof. First one shows that i_0 is well determined. The sequence $(\Delta_i)_i$ defined by:

$$\forall i \in \mathbb{N}, \Delta_i = \sum_{j=0}^{i} q_{j,N} - m^2 - (N+i)q_{(i+1),N}$$

is increasing:

$$\begin{aligned}\Delta_{i+1} - \Delta_i &= q_{(i+1),N} - (N+i+1)q_{(i+2),N} + (N+i)q_{(i+1),N} \\ &= (N+i+1)(q_{(i+1),N} - q_{(i+2),N}) > 0.\end{aligned}$$

Furthermore $q_{0,N} - m^2 = 2\int_0^\infty \int_{y'}^\infty p_y p_{y'} \left(p_{y'}^{N-2} - 1\right) \mathrm{d}y\mathrm{d}y' \leq 0 < Nq_{1,N}$, so $\Delta_0 < 0$, and $\Delta_i \to \mathrm{var}\,[\widehat{m}]$ when $i \to \infty$ because $(q_{i,N})_i$ decreases at exponential rate. So there exists $i_0 \in \mathbb{N} \mid \Delta_{i_0-1} \leq 0$ and $\Delta_{i_0} > 0$.

Let us now consider the auxiliary problem:

$$\underset{\substack{(\beta_i)_{i\geq 1} \\ \beta_i > 0}}{\mathrm{argmin}} \left(\beta + \sum_{i=1}^\infty \beta_i\right)\left(q + \sum_{i=1}^\infty q_i \beta_i^{-1}\right)$$

with $\beta > 0$ and $q \in \mathbb{R}$. We show that it has a solution if and only if $q > 0$. Let $i \geq 1$, cancelling the partial derivatives brings:

$$\forall i \geq 1,\ 0 = \left(q + \sum_{j=1}^\infty q_j \beta_j^{-1}\right) + \left(\beta + \sum_{j=1}^\infty \beta_j\right)\frac{-q_i}{\beta_i^2}.$$

Then the solution should be of the form: $\forall i \in [\![1, \infty)\,, \beta_i = c_0\sqrt{q_i}$ for some $c_0 > 0$. Solving now the problem with c_0, the derivative writes $q - \beta/c_0^2$. If $q \leq 0$ then it is strictly decreasing and there is no global minimiser. On the contrary, $q > 0$ brings $c_0 = \sqrt{\beta/q}$ and $\forall i \geq 1\,, \beta_i = c_0\sqrt{q_i}$.

Thus, in our context with the constraint $\forall i \in \mathbb{N}\,, \beta_i \leq 1$, this means that solving the optimisation problem will set iteratively $\beta_i = 1$ until the minimiser is feasible, *i.e.* until $i_0 \overset{\text{def}}{=} \min\{i \in \mathbb{N} \mid \sum_{j=0}^{i} q_{j,N} - m^2 > (N+i)q_{(i+1),N}\}$. Then the solution will be given by:

$$\begin{aligned}&\forall i \in [\![1, i_0]\!],\ \beta_i = 1 \\ &\forall i > i_0,\ \beta_i = \frac{\sqrt{q_{i,N}}}{\sqrt{\dfrac{1}{N+i_0}\sum_{j=0}^{i_0}(q_{j,N} - m^2)}}.\end{aligned}$$

□

It is part of the proof that i_0 is well defined and so it appears that the optimal distribution enforces the estimator to go at least until the i_0^{th} event. Recalling that $(Y_n)_n$ is the superposed Poisson process, this can be understood in the sense that at least N events are necessary to *use* at least one time each process. Even if the link between i_0 and N is not that straightforward, one can then conjecture that $\liminf_{N\to\infty} i_0 = \infty$. As a matter of fact, the exact resolution for Pareto random variables in Proposition 4.12 gives $i_0 \sim (N \log N)$.

Corollary 4.5 (Bounds on β_i^*)**.** *For all $i > i_0$, one has:*

$$\sqrt{\frac{q_{i,N}}{q_{i_0+1,N}}} > \beta_i^* \geq \sqrt{\frac{q_{i,N}}{q_{i_0,N}}}. \tag{4.24}$$

Proof. By definition of i_0, one has:

$$(N + i_0)q_{i_0+1,N} < \sum_{j=0}^{i_0} q_{j,N} - m^2 \leq (N + i_0 - 1)q_{i_0,N} + q_{i_0,N}$$

which concludes the proof. □

Thus the tail of the optimal distribution $(\beta_i^*)_i$ is exponentially decreasing by Lemma 4.1. From these bounds on the $(\beta_i)_i$ one can also derive bounds on the variance:

$$q_{i_0+1,N} \operatorname{E}[\tau]^2 < \operatorname{E}[\tau] \cdot \operatorname{var}\left[\widehat{Z}\right] \leq q_{i_0,N} \operatorname{E}[\tau]^2 .$$

Assuming $\liminf_{N\to\infty} i_0 = \infty$ and using the lower bound on $q_{i,N}$ from Lemma 4.1, one can show that $\liminf_{N\to\infty} \operatorname{E}[\tau] \cdot \operatorname{var}\left[\widehat{Z}\right] = \infty$, which implies the existence of an optimal N. Section 4.4.1 presents an exact resolution of this optimisation problem for a Pareto random variable.

Finally, we have presented in this section the framework for an optimal resolution of Problem (4.23) and proven existence of a solution under reasonable assumptions: $(q_{i,N})_i$ is decreasing and $\lim_{N\to\infty} i_0 = \infty$. Furthermore the comprehensive resolution in the case of a Pareto distribution in Section 4.4.1 legitimises them. Generally speaking, if $(q_{i,N})_{i\geq 1}$ is not decreasing the optimisation has to be performed over all the decreasing sub-sequences of $(q_{i,N})_i$, which turns it into a combinatorial problem [see Rhee and Glynn, 2015, Theorem 3].

4.3.4 Geometric randomisation

On the one hand the computation of the optimal distribution for T can be quite demanding in computing time; and on the other hand the geometric law plays a key role as for any distribution p_y, the sequence $(q_{i,N})_i$ decreases at exponential rate and the optimal randomising distribution (when $(q_{i,N})_i$ is decreasing) is somehow a shifted geometric law. Therefore we study the parametric case where $\operatorname{P}[T \geq n] = e^{-\beta n}$, $\beta > 0$ and tune β and N to minimise $\operatorname{E}[\tau] \cdot \operatorname{var}\left[\widehat{Z}\right]$.

Using the exponential power series in $\operatorname{var}\left[\widehat{Z}\right]$ (cf Eq. 4.17), the optimisation problem (4.23) becomes:

$$\min_{\substack{\beta>0 \\ N\in[\![2,\infty)}} \left(N + \frac{1}{e^\beta - 1}\right) \left(\sum_{i=0}^{\infty} q_{i,2} \left(\frac{2}{\gamma(\beta, N)}\right)^i - m^2\right). \tag{4.25}$$

Proposition 4.9. *There exists a global minimiser* (β_{opt}, N_{opt}) *to Problem* (4.25)*. Furthermore, for the extended minimisation problem with* $N \in [2, \infty)$*,* (β_{opt}, N_{opt}) *satisfies the relationship:*

$$\beta_{opt} = \log\left(1 + \frac{2}{N_{opt}^2 - 1 + (N_{opt} - 1)\sqrt{N_{opt}^2 + 6N_{opt} + 1}}\right). \tag{4.26}$$

Proof. Denote $Q_N(\beta)$ the quantity one seeks to minimise in Eq. (4.25). First, we show that for any fixed N, there exists a global minimiser of $Q_N(\beta)$. One has $Q_N(\beta) \to \infty$ when $\beta \to 0$ and $\gamma(\beta, N) \to 0$ when $\beta \to \infty$. Hence, either $\exists \beta_\infty \in (0, \infty]$ such that:

$$\begin{cases} Q_N(\beta) \xrightarrow[\beta \nearrow \beta_\infty]{} \infty \\ Q_N(\beta) < \infty \qquad \forall \beta < \beta_\infty. \end{cases}$$

Then Q_N is continuous on $(0, \beta_\infty)$ with infinite limits on 0 and β_∞, so it reaches its minimum on $(0, \beta_\infty)$; or $\exists \beta_\infty \in (0, \infty)$ such that:

$$\begin{cases} Q_N(\beta) < \infty & \forall \beta \in (0, \beta_\infty] \\ Q_N(\beta) = \infty & \forall \beta > \beta_\infty. \end{cases}$$

Since Q_N is continuous on β_∞^- by Monotone Convergence Theorem, Q_N reaches its minimum on $(0, \beta_\infty]$.

Let $\beta_{\text{opt}}(N) > 0$ be such that $\inf_\beta Q_N(\beta) = Q_N(\beta_{\text{opt}})$. We now show that there exists an optimal N. It is sufficient to show $Q_N(\beta_{\text{opt}}) \to \infty$ when $N \to \infty$. Denote $B = 1/(e^\beta - 1)$; one has:

$$\frac{1}{\gamma(B, N)} = \frac{1}{N} + \frac{N}{B} - \frac{2}{B} + \frac{1}{NB}.$$

Hence, depending on the growth rate of B when $N \to \infty$, one would have:

$$\begin{aligned} &B = O(N), \frac{1}{\gamma} \sim \frac{N}{B} \Rightarrow \inf_\beta Q_N(\beta) \xrightarrow[N \to \infty]{} \infty \\ &N = o(B), \frac{1}{\gamma} \sim \frac{1}{N} \text{ or } \frac{N}{B} \Rightarrow \inf_\beta Q_N(\beta) \sim \frac{B}{N} \text{ or } N \Rightarrow \inf_\beta Q_N(\beta) \xrightarrow[N \to \infty]{} \infty. \end{aligned}$$

Then in any cases $Q_N(\beta_{\text{opt}}) \to \infty$ when $N \to \infty$, which means that there exists $N_{\text{opt}} \in \mathbb{N} \mid Q_{N_{\text{opt}}}(\beta_{\text{opt}}) = \inf_N Q_N(\beta_{\text{opt}})$.

We now show the relationship between β_{opt} and N_{opt}. Let us consider $N \in [2, \infty)$, the partial derivatives of $\mathrm{E}[\tau] \cdot \mathrm{var}\left[\widehat{Z}\right]$ against B and N write:

$$\begin{cases} \dfrac{\partial\left(\mathrm{E}[\tau] \cdot \mathrm{var}\left[\widehat{Z}\right]\right)}{\partial B} = \mathrm{var}\left[\widehat{Z}\right] + \mathrm{E}[\tau] \dfrac{\partial \mathrm{var}\left[\widehat{Z}\right]}{\partial \gamma} \dfrac{\partial \gamma}{\partial B} \\ \dfrac{\partial\left(\mathrm{E}[\tau] \cdot \mathrm{var}\left[\widehat{Z}\right]\right)}{\partial N} = \mathrm{var}\left[\widehat{Z}\right] + \mathrm{E}[\tau] \dfrac{\partial \mathrm{var}\left[\widehat{Z}\right]}{\partial \gamma} \dfrac{\partial \gamma}{\partial N}. \end{cases}$$

At point $(\beta_{\text{opt}}, N_{\text{opt}})$, both equations are cancelled, which gives:

$$\frac{\partial \gamma}{\partial N}(B_{\text{opt}}, N_{\text{opt}}) = \frac{\partial \gamma}{\partial B}(B_{\text{opt}}, N_{\text{opt}}).$$

Recalling that $\gamma(B, N) = NB/(B + (N-1)^2)$, this gives the equation: $B_{\text{opt}}^2 - (N_{\text{opt}}^2 - 1)B_{\text{opt}} - N_{\text{opt}}(N_{\text{opt}} - 1)^2 = 0$. One can solve it in B_{opt} and keep the positive root, which gives the solution. □

Hence there is always an optimal solution to Problem (4.25), meaning this parametrisation is meaningful.

To summarise we have shown that by randomising the finite number of iterations and slightly modifying the weights of the original nested sampling, it is possible to define an unbiased estimator for the mean of any real-valued random variable with continuous *cdf*, resolving the issue of choosing an *appropriate* stopping criterion. With a suboptimal geometric randomisation as in Corollary 4.2, the variance is at most twice the one of the ideal case (estimator of Eq. 4.4). However it is not usable with a fixed predetermined computational budget and its convergence rate is slower than the canonical square-root one. To circumvent this limitation, the idea is to average several replicas of the randomised unbiased estimator (see Eq. 4.21). This new estimator remains unbiased and also supports a Central Limit Theorem.

All these theoretical results assume that it is possible to generate conditional random variables when required, as for the original nested sampling algorithm [see Skilling, 2006, Section 9]. Efficient conditional simulation can be carried out in different ways, from perfect simulation [see for example Propp and Wilson, 1996] to approximation using random walks (see the Metropolis-Hastings algorithm in Section 1.3.3). The aim of this chapter and the spirit of this thesis is not to challenge this hypothesis in a general manner but only to provide a new insight on the theoretical definitions of the estimators; especially here on the risk of choosing a *bad* stopping criterion in nested sampling, and to propose an other tool to deal with this issue. Since nested sampling has been applied successfully to a great number of problems so far, these results are expected to hold in these situations. Also the examples of Section 4.5 are in good agreement with these theoretical results.

In the next section, we discuss the different stopping criteria usually recommended for nested sampling and parallel implementation of the estimators.

4.3.5 Parallel implementation

Skilling [2006, Section 7] presents two possible termination rules based on criteria evaluated on-the-fly:

- stop when the greatest expected increment (current weight and biggest found likelihood value) is smaller than a given fraction of the current estimate;

- stop when the number of iterations *significantly* exceeds NH with H the information, estimated on-the-fly.

Chopin and Robert [2010] use an other stopping criterion, close to the first one above, it is: "stop when the new increment is smaller than a given fraction of the current estimate". An other option is to do a predetermined number of iterations [Brewer et al., 2011]. Unfortunately these criteria give no guarantee on the convergence of the estimator to the sought value and may lead to biased estimation.

A first difference between the three first criteria and the last one stands in the fact that this latter uses a known computational budget while the other ones will run until the criterion is satisfied; hence there is no way to estimate the (random) final number of iteration in advance. This difference is also to be found between $\widehat{Z}$ (see Eq. 4.15) and ζ (Eq. 4.21): the first one will use a random number of simulated samples (the draw of the randomising variable) while the second one is defined with a fixed computational budget. Hence it is not straightforward to compare these two categories of estimators because the setting is not the same.

An other main difference between these estimators is whether they enable parallel computation or not. The three first stopping criteria need to be evaluated at each iteration and are based on quantities estimated with the full process with parameter N. Hence they do not allow for parallel computation. On the other hand, with a predetermined total number of iterations, parallel computation as for the quantile estimator (see Appendix A) can be carried out. The randomised estimator $\widehat{Z}$ also enables this feature as the random number of iterations is drawn before the algorithm starts. Considering ζ, each replica can be computed in parallel, and further the computation of each replica also allows for parallel implementation. Hence ζ allows for a *double* parallelisation, which is worth noticing as it may require a substantial computational budget to become effectively Gaussian (see numerical examples of Section 4.5.4).

To conclude, one stresses out the fact that among estimators with random computational budget, $\widehat{Z}$ is the only one allowing for parallel computation; furthermore it is also the only one unbiased and its variance is at worst twice the one of the ideal estimator (upper bound reached with suboptimal implementation of $\widehat{Z}$ as in Corollary 4.2). Both fixed-budget estimators enable parallel implementation; however nested sampling with a predetermined number of iterations has no reason to be close to the sought value. On the other hand, ζ is unbiased and supports a CLT. All these considerations are illustrated in Section 4.5.

4.4 Application to heavy-tailed random variables

In this section we give insights on the properties of the estimators defined in Sections 4.2.2 and 4.3 when $Y = g(\mathbf{X})$ is heavy-tailed. Mean estimation for heavy-tailed random variables is a well identified problem often addressed by some parametric assumptions on the *cdf* of Y; see Beirlant et al. [2012] for a comprehensive overview of tail index estimation,

and Peng [2001], Johansson [2003], Necir et al. [2010] or Hill [2013] for references on mean estimation for heavy-tailed random variables.

In the sequel we then give explicit results for the Pareto distribution: $p_y = \mathrm{P}[Y > y] = 1 \wedge y^{-a},\ a > 1$. Note that this constraint $a > 1$ is the integrability condition for Y and thus a sufficient condition for the convergence of the sum in the Campbell's theorem, see Eq. (4.8).

4.4.1 Exact resolution for a Pareto distribution

With an analytic form for the *cdf* of Y, we can derive explicit formulae for the variance of the ideal estimator (infinite number of terms, see Eq. 4.12) and the optimisation problem of Eq. (4.23).

First we compare the variance of the ideal estimator $\widehat{m}$ against usual Monte Carlo and Importance Sampling estimators. In this latter case the importance density is chosen to be a Pareto distribution with parameter $b > 0$.

Proposition 4.10 (Variance comparison)**.** *For a Pareto distribution, one has $m = a/(a-1)$ and the variances write:*

$$\begin{aligned} a > 2,\ \mathrm{var}\left[\widehat{m}_{MC}\right] &= \frac{m(m-1)^2}{2N - mN} \\ a > 1\,;\mathrm{var}\left[\widehat{m}_{C}\right] &= \frac{m}{2N} \\ a > \frac{2N}{2N-1},\ \mathrm{var}\left[\widehat{m}\right] &= \frac{m(m-1)^2}{2N - m} \\ a > 1 + \frac{b}{2},\ \mathrm{var}\left[\widehat{m}_{IS}\right] &= \frac{m^2(B-1)^2}{N(2B-1)} \end{aligned}$$

with $B = (a-1)/b \in (1/2, \infty)$.

Proof. For the first equality:

$$\begin{aligned} \mathrm{E}[Y] &= \int_0^\infty p_y \mathrm{d}y = \frac{a}{a-1} \\ \mathrm{var}\left[\widehat{m}_{MC}\right] &= \frac{1}{N}\left(\mathrm{E}\left[Y^2\right] - \mathrm{E}[Y]^2\right) = \frac{a}{N(a-2)(a-1)^2} = \frac{m(m-1)^2}{(2-m)N}; \end{aligned}$$

for the second one:

$$\mathrm{var}\left[\widehat{m}_{C}\right] = \frac{1}{N}\int_0^\infty y^2 p_y \mathrm{d}\mu^Y(y) = \frac{1}{N}\int_1^\infty y^2 y^{-a} a y^{-a-1} \mathrm{d}y = \frac{a}{2N(a-1)};$$

for the third one:

$$\begin{aligned}\mathrm{E}\left[\widehat{m}^2\right] &= 2\int_0^\infty \int_0^y p_y p_{y'}^{1-1/N} \mathrm{d}y'\mathrm{d}y = 2\int_0^1 \int_0^y \cdots + 2\int_1^\infty \int_0^1 \cdots + 2\int_1^\infty \int_1^y \cdots \\ &= 1 + \frac{2}{a-1} + \frac{2}{(a-1)(2(a-1)-a/N)} \\ \mathrm{var}\left[\widehat{m}\right] &= \frac{a}{N(a-1)^2(2(a-1)-a/N)};\end{aligned}$$

and for the last one:

$$\begin{aligned}\mathrm{var}\left[\widehat{m}_{IS}\right] &= \frac{1}{N}\left[\int_1^\infty y^2 \frac{a^2}{b} y^{-2a+b-1}\mathrm{d}y - \frac{a^2}{(a-1)^2}\right] \\ \mathrm{var}\left[\widehat{m}_{IS}\right] &= \frac{a^2}{N(a-1)^2}\left(\frac{1}{B(2-B)} - 1\right)\end{aligned}$$

with $B = b/(a-1)$. □

It is clearly visible that the classical Monte Carlo estimator needs a second-order moment while $\widehat{m}_C$ only requires the finiteness of $\mathrm{E}[Y]$; $\widehat{m}$ requires $a > 2N/(2N-1) \approx 1 + 1/2N$ and $\widehat{m}_{IS}$ requires $a > 1 + b/2$. It also illustrates the result of Proposition 4.4: $\mathrm{var}[\widehat{m}] < \mathrm{var}[\widehat{m}_{MC}]$. The optimal value $b = a - 1$ cancels out $\mathrm{var}[\widehat{m}_{IS}]$. It is well known that there is an optimal density q for IS that cancels out the variance of the IS estimator but it is case-specific: here a Pareto density with parameter $a-1$ (see also Section 1.2).

Remark 4.5 (Limit distribution of classical Monte Carlo estimator)**.** *In the case of Pareto distribution, when $a > 2$ the Central Limit Theorem gives the limit law of the estimator while for $1 < a < 2$ the Generalised Central Limit Theorem [see for example Embrechts et al., 1997] states that $\sum_{i=1}^N Y_i$ the sum of N iid. realisations of Y is in the domain of attraction of a stable law with parameter a:*

$$N^{1-1/a}\left(\frac{1}{N}\sum_{i=1}^N Y_i - m\right)\frac{1}{C_a} \xrightarrow[N\to\infty]{\mathcal{L}} Y_a$$

with the characteristic function of Y_a, ϕ_{Y_a}, writing:

$$\phi_{Y_a}(t) = \exp\left[-|t|^a\left(1 - i\left(\tan\left(\pi a/2\right)\right) sgn(t)\right)\right]$$

and C_a the normalising constant $C_a = \pi^{1/a}\left(2\Gamma(a)\sin \pi a/2\right)^{-1/a}$.

We now detail the resolution of optimisation problems (4.23) and (4.25). Especially we first explicit the form of the sequence $(q_{i,N})_i$ defined in Eq. (4.16).

Proposition 4.11. *If Y is a Pareto random variable with parameter $a > 1$, then:*

$$\forall i \in \mathbb{N},\ q_{i,N} = \frac{2}{(a-1)(aN-2)}\left[\frac{a(N-1)^2}{N(aN-2)}\right]^i + \mathbb{1}_{i=0}\frac{(a+1)}{2(a-1)}.$$

Proof. Let $i \geq 0$, one has:

$$\begin{aligned}
\int_1^\infty \int_{y'}^\infty p_y p_{y'}^{N-1} \frac{\left[-N \log p_{y'}(1-1/N)^2\right]^i}{i!} \mathrm{d}y\mathrm{d}y' & \\
&= \frac{\left[aN(1-1/N)^2\right]^i}{i!} \int_1^\infty \int_{y'}^\infty y^{-a} y'^{-a(N-1)} (\log y')^i \mathrm{d}y\mathrm{d}y' \\
&= \frac{\left[aN(1-1/N)^2\right]^i}{(a-1)i!} \int_1^\infty y'^{1-aN} (\log y')^i \mathrm{d}y' \\
&= \frac{\left[aN(1-1/N)^2\right]^i}{(a-1)i!} \frac{\Gamma(i+1)}{(aN-2)^{i+1}} \\
&= \frac{1}{(a-1)(aN-2)} \left[\frac{aN}{aN-2}\left(1-\frac{1}{N}\right)^2\right]^i
\end{aligned}$$

with Γ standing here for the Gamma function. Furthermore:

$$\int_0^1 \int_{y'}^\infty p_y p_{y'}^{N-1} \frac{\left[-N \log p_{y'}(1-1/N)^2\right]^i}{i!} \mathrm{d}y\mathrm{d}y' = \mathbb{1}_{i=0} \frac{(a+1)}{2(a-1)}.$$

$(q_{i,N})_i$ is decreasing *iff.*:

$$\frac{aN}{aN-2}\left(1-\frac{1}{N}\right)^2 < 1 \text{ which rewrites } \; 1 < a\left(1-\frac{1}{2N}\right),$$

which is indeed the condition for the finiteness of var $[\widehat{m}]$ already stated in Proposition 4.10. □

Hence for a Pareto distribution $(q_{i,N})_i$ is decreasing. One can then look for i_0, the solution of the problem $i_0 = \min\{i \in \mathbb{N} \mid \sum_{j=0}^i q_{j,N} - m^2 > (N+i) q_{(i+1),N}\}$. Let W_{-1} be the lower branch of the Lambert W function [see for example Corless et al., 1996], *i.e.* the function W_{-1} defined over $(-e^{-1}, 0)$ such that:

$$x = W_{-1}(x) e^{W_{-1}(x)}.$$

Let $k > 0$ and $K > 0$ be such that $\forall i \geq 1$, $q_{i,N} = kK^{N+i}$; let $i \geq 0$, one has:

$$\sum_{j=0}^i q_{j,N} = \frac{a+1}{2(a-1)} + \sum_{j=0}^i kK^{N+j} = \frac{a+1}{2(a-1)} + kK^N \frac{1-K^{i+1}}{1-K}.$$

Using this equality, one can look for $i \in \mathbb{R}$ such that:

$$\begin{aligned}
\frac{a+1}{2(a-1)} - m^2 + kK^N \frac{1-K^{i+1}}{1-K} &= (N+i) kK^{N+i+1} \\
\frac{a+1}{2(a-1)} - m^2 + kK^N \frac{1}{1-K} &= K^{N+i+1}\left((N+i)k + \frac{k}{1-K}\right)
\end{aligned}$$

$$K' = \log K \left(N + i + \frac{1}{1-K} \right) K^{N+i+1/(1-K)}$$

with:

$$\begin{aligned} K' &= \log K \left(\frac{1}{k} \left(\frac{a+1}{2(a-1)} - m^2 \right) + \frac{K^N}{1-K} \right) K^{K/(1-K)} \\ &= \log K \left(-\frac{a+1}{2k(a-1)} + \frac{K^N}{1-K} \right) K^{K/(1-K)}. \end{aligned}$$

This latter equality is solved using W_{-1}:

$$\begin{aligned} \log K \left(N + i + \frac{1}{K-1} \right) &= W_{-1}(K') \\ i &= \frac{W_{-1}(K')}{\log K} - N - \frac{1}{1-K}, \end{aligned}$$

such that i_0 finally writes:

$$i_0 = \lceil \frac{W_{-1}(K')}{\log K} - N - \frac{1}{1-K} \rceil. \tag{4.27}$$

The following proposition gives an asymptotic approximation when $N \to \infty$ to precise the growth rate of i_0.

Proposition 4.12. *If Y is a Pareto random variable, then:*

$$i_0 = \frac{Nm}{2} \left(\log N + \log\log N - \log(\frac{m}{2}) \right) + o(N),\ N \to \infty.$$

Proof. The problem can be rewritten:

$$\min \left\{ i \geq 1 \mid \frac{1}{1-\beta} - \frac{aN-2}{2(a-1)} > \beta^{i+1} \left(N + i + \frac{1}{1-\beta} \right) \right\}.$$

Furthermore one has:

$$\frac{1}{1-\beta} = \frac{Nm}{2} + \frac{(a-2)^2}{4(a-1)^2} + o(1)$$

which brings that the left hand term is equal to $(m/2)^2 + o(1)$. Writing $i = N(k_0 + k_1 \log N + k_2 \log\log N)$ brings:

$$\beta^{i+1} = e^{-\frac{2k_0}{m}} N^{-\frac{2k_1}{m}} (\log N)^{-\frac{2k_2}{m}} (1 + o(1)).$$

Hence one has to choose k_0, k_1 and k_2 such that the right hand term also equals $(m/2)^2 + o(1)$, which gives the solution. □

Corollary 4.6 (Order of magnitude of $\mathrm{E}[\tau] \cdot \mathrm{var}\left[\widehat{Z}\right]$)**.**

$$\mathrm{E}[\tau] \cdot \mathrm{var}\left[\widehat{Z}\right] \underset{N\to\infty}{\sim} \left(\frac{m(m-1)}{2}\right)^2 \log N.$$

Proof. Using the asymptotic expansion of i_0 one finds $q_{i_0} \sim (N^2 \log N)^{-1}(m-1)^2$. Furthermore, one has $\mathrm{E}[\tau] \sim i_0$. Finally, the use of $\mathrm{E}[\tau] \cdot \mathrm{var}\left[\widehat{Z}\right] \sim q_{i_0} \mathrm{E}[\tau]^2$ gives the result. □

Corollary 4.6 shows that $\mathrm{E}[\tau] \cdot \mathrm{var}\left[\widehat{Z}\right] \to \infty$ when $N \to \infty$ so there is an optimal value for N that minimises $\mathrm{E}[\tau] \cdot \mathrm{var}\left[\widehat{Z}\right]$; a numerical resolution for several values of a from 1 to 3 was performed and the result is displayed in Figure 4.1. We also present in Figure 4.2 a comparison between the minimal variance estimate (estimator of Eq. (4.21) with the optimal distribution $(\beta_i^*)_i$ and optimal N) and the classical Monte Carlo one. There we can see that for $a \lesssim 2.5$ the new estimator performs better in terms of variance; especially for $a < 2$ it remains finite while $\mathrm{var}[\widehat{m}_{MC}] = \infty$.

As explained in Section 4.3.4 we consider now a Geometric random variable T with parameter β for the random truncation.

Proposition 4.13. *If Y is a Pareto random variable with parameter $a > 1$ and $\forall n \in \mathbb{N}$, $\mathrm{P}[T \geq n] = e^{-\beta n}$ then:*

$$\mathrm{var}\left[\widehat{Z}\right] = \frac{m(m-1)^2}{2\gamma(\beta, N) - m}$$

and

$$\beta_{opt} = \log\left(\frac{1}{B_+} + 1\right) \tag{4.28}$$

where B_+ is the positive root of the quadratic polynomial $P(B)$:

$$P(B) = \frac{2N_{opt} - m}{(N_{opt}-1)^2} B^2 - 2mB - \left(m(N_{opt}-1)^2 + 2N_{opt}^2\right).$$

Proof. One gets the expression of the variance directly from Section 4.2.2 with $\gamma(N, \beta)$ instead of N. Then, denoting $B = 1/(e^\beta - 1)$, one solves the problem:

$$\frac{\partial}{\partial B}\left((N+B)\left(\frac{a}{2(a-1)\gamma - a}\right)\right) = 0.$$

□

With this relation and the one of Eq. (4.26) one can derive the optimal parameters $(\beta_{\mathrm{opt}}, N_{\mathrm{opt}})$. Figure 4.1 shows a numerical resolution of this problem for several values of $a \in (1, 3]$.

Furthermore, if one considers the approximation of the optimisation problem (4.25)

with relation (4.18) instead of (4.26), one has to minimise:

$$N \mapsto \frac{N^2 + N - 1}{N + 1 - m} m(m-1)^2.$$

The derivative against $N > m - 1$ writes:

$$1 - \frac{1}{(N + 1 - m)^2} m(m-1).$$

Denoting $N_{\text{app}} \in \mathbb{R}$ the minimiser, one has:

$$N_{\text{app}} = \max\left(m - 1 + \sqrt{m^2 - m - 1}, 2\right). \tag{4.29}$$

This approximation is the red dotted-dashed line of Figure 4.1. As we can see, it is in good agreement with the optimal values, both for the parameter N and for the global variance (see further Section 4.4.2 and Figure 4.2). For numerical simulation, we will choose $\lfloor N_{\text{app}} \rfloor$.

4.4.2 Comparison of the estimators

We have seen in Sections 4.3.3 and 4.3.4 two ways of implementing the *ideal* estimator $\widehat{m}$ defined in Section 4.2.2 with a fixed computational budget. Then we have presented their exact behaviour in a case of a Pareto random variable. These two ways involve a truncation of the infinite sum (4.4) by an integer-valued random variable T. In the first implementation the distribution of T and the number N of point processes are optimised in order to minimise the estimator variance. In the second implementation, the distribution of T is enforced to be geometric and its parameter as well as N are optimised.

While the first implementation is optimal in terms of variance, it requires to solve a combinatorial problem, which can turn it into a poorer algorithm in terms of computational time. In this scope, the parametric algorithm constraining the randomising variable T to be geometric with parameter β is much simpler to implement. The aim of this section is to benchmark these two implementations and to challenge the optimal parameters against the fixed ones we will suggest.

More precisely, while both optimisations ended up with optimal parameters depending on the distribution of Y, we also consider the parametric algorithm with parameter β_{app} given by Eq. (4.18) and $N = N_{\text{app}}$, 2, 5 or 10.

Figure 4.2 shows the relative increase of the standard deviations due to the suboptimal implementations for a given computational budget, *i.e.* for a given number of generated samples. It also shows the standard deviation ratios between the optimal implementation, the classical Monte Carlo estimator (see Eq. 4.14) and $\widehat{m}$ given by Eq. (4.4). For this latter, it is assumed that its computational cost is N, *i.e.* that it costs 1 to simulate an increasing random walk (see Definition 4.1) while it requires an infinite number of simulated samples. This calls for certain comments:

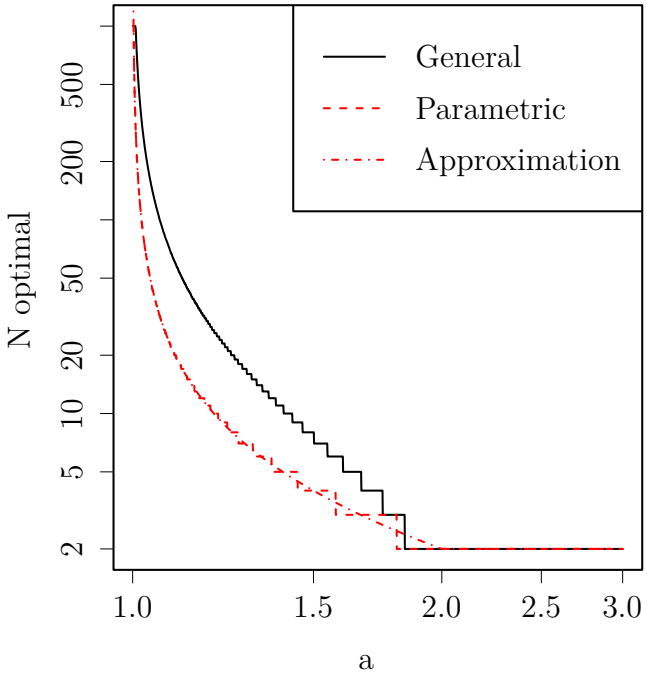


Figure 4.1: Optimal values for N in the general (cf Section 4.3.3) and in the parametric (cf Section 4.3.4) cases with the approximation of Eq. (4.29). a is the parameter of the Pareto distribution.

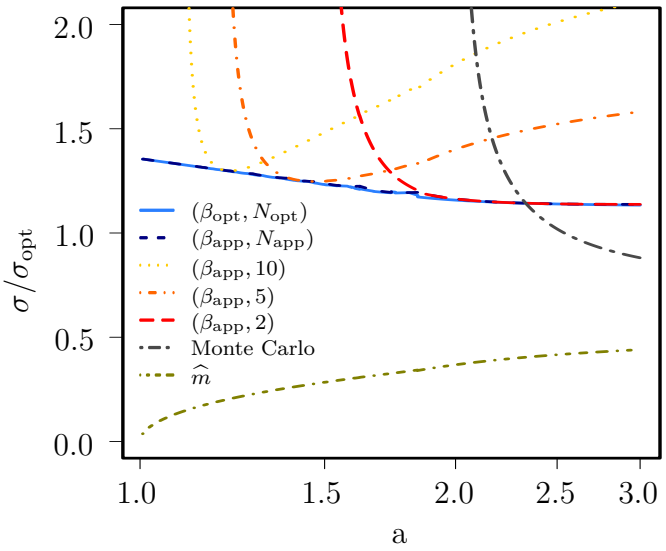


Figure 4.2: Ratios of the standard deviations of different estimators over the standard deviation of the optimal estimator ζ of Section 4.3.3. The classical Monte Carlo estimator is defined in Eq. (4.14); $\widehat{m}$ is the ideal estimator (4.4); the other estimators are randomised estimators of Eq. (4.21) with enforced geometric distribution for T with parameter β and N as follows: $(\beta_{\text{opt}}, N_{\text{opt}})$: optimal parameters of Proposition 4.9; $(\beta_{\text{app}}, N_{\text{app}})$: approximated optimal parameters of Eqs. (4.18) and (4.29). a is the parameter of the Pareto distribution.

- the parametric implementation with optimised parameters $(\beta_{\text{opt}}, N_{\text{opt}})$ remains competitive against the optimal implementation (solid black line going from ≈ 1.3 to ≈ 1.1);

- the parametric implementation with parameters β_{app} and N_{app} is almost not distinguishable from the parametric implementation with optimal parameters β_{opt} and N_{opt}. This means that it is not necessary to strive to estimate the parameters $(\beta_{\text{opt}}, N_{\text{opt}})$;

- the classical Monte Carlo estimator is better than the optimal implementation as soon as $a \gtrsim 2.5$ and better than the parametric implementation as soon as $a \gtrsim 2.3$; this confirms that nested sampling is especially convenient for heavy-tailed random variables;

- the standard deviation of $\widehat{m}$ illustrates the efficiency of the ideal estimator compared to the classical Monte Carlo one (cf. Proposition 4.10), with a standard deviation at least twice as small;

- generally speaking and without any knowledge on the distribution of Y, N should not be set too small as the variance increases much faster when it is smaller than the optimal value; especially with $\beta = \beta_{\text{app}}$ the finiteness condition of the variance writes $a > 1 + 1/N$.

Given these results we can consider that the parametric implementation is a good trade-off between minimal variance estimation and complexity, especially when no information on the distribution of Y is provided.

4.5 Examples

In the previous sections, we have demonstrated how the point process point of view lets improve the original nested sampling in three ways: 1) we demonstrated a bias or order $1/N$ for the original ideal nested sampling scheme and proposed corrected weights to get an ideal unbiased estimator; by ideal we mean a sum with an infinite number of terms. 2) We proposed to resolve the issue of choosing a termination rule by using a randomised sum. This new estimator remains unbiased and we further suggested a general implementation which does not depend on the random variable of interest and only doubles the variance of the ideal estimator. 3) We defined a nested sampling based estimator with a finite predetermined computational budget, which is unbiased and supports a Central Limit Theorem. The aim of this section is to illustrate these results and potential issues with the termination rule on a usual test case for nested sampling.

4.5.1 Practical implementation

All what the nested sampling really requires is the generations of several increasing random walks. Appendix A is devoted to the definition of practical guidelines on possible parallel implementations and computational times of the increasing random walk. Hence we

consider in this section that one has a generator of a random walk and present directly the adaptation for the computation of $\widehat{Z}$ and ζ.

As explained above, we do not intend to solve the combinatorial optimisation problem in the general case and so we present here a pseudo-code for the parametric case. Reader interested in the optimal resolution is referred to [Rhee and Glynn, 2015]. We then present in Algorithm 7 how to compute $\widehat{Z}$ and in Algorithm 8 how to compute $\zeta(c)$. In this latter case we assume that N and β are given, being optimised (with previous knowledge or simulations) or not.

Algorithm 7 Pseudo-code for $\widehat{Z}$

Require: N, β
Generate $\mathtt{T}$ according to $\mathrm{P}\left[T \geq n\right] = e^{-\beta n}$
2: Get $(Y_n)_{n=1}^{\mathtt{T}}$ an increasing random walk with parameter N until the $\mathtt{T}^{\text{th}}$ event
Set $Y_0 = 0$
4: $\widehat{Z} = \sum_{i=0}^{T} \left(Y_{i+1} - Y_i\right) \left(1 - \frac{1}{N}\right)^i e^{\beta i}$

Algorithm 8 Pseudo-code for $\zeta(c)$

Require: c, N, β
$G \leftarrow 0$; $\zeta \leftarrow 0$;
while $c > 0$ **do**
Generate T^* according to $\mathrm{P}\left[T \geq n\right] = e^{-\beta n}$
$c = c - (N + T^*)$; $G = G + 1$; $T[G] = T^*$
end while
if $c < 0$ **then** ▷ discard the last replica if it exceeds the budget
$G = G - 1$; $T = T[1 : G]$
end if
foreach g in 1:G **do**
Start Algorithm 7 from step 2 with $\mathtt{T} = T[g]$
$\zeta = \zeta + \widehat{Z}$
end foreach
$\zeta = \zeta / G$

Basically, Algorithm 8 is just a wrap-up of Algorithm 7 with an update of the remaining computational budget.

Note that these practical algorithms make use of Markov chain drawing to approximate the conditional distributions and for this purpose generate indeed several samples for only one state of the increasing random walk. This is referred to as a *burn-in* and modify the cost of an estimator defined in Proposition 4.7 as follows: $\tau = N + bT$, with b this *burn-in* parameter. Now the cost is the number of calls to the generator of Y (which amounts to generate $\mathbf{X}$ and to call g). Since this increase is common to all algorithms considered here, we will not mention it any more.

4.5.2 Variance increase

In this section, we intend to check the unbiasedness of the ideal estimator $\widehat{m}$ of Section 4.2.2 as opposed to the bias of order $1/N$ of the original nested sampling (see Eq. 4.6). They differ only in the weights used: $\exp -1/N$ instead of $1 - 1/N$; thus they are computed in the same run. Also we check the variance increase between $\widehat{m}$ and the suboptimal randomised estimator of Corollary 4.2. To do so, we use an example from Skilling [2006] where it is known that 100 iterations of the increasing random walk on average are enough. The aim is to estimate the evidence of a likelihood with uniform prior over a $d-$dimensional unit cube: $m = \mathrm{E}\left[g(\mathbf{X})\right] = \mathrm{E}\left[Y\right]$ with:

$$g(\mathbf{x}) = 100 \prod_{i=1}^{d} \frac{e^{-x_i^2 \big/ 2u^2}}{\sqrt{2\pi}u} + \prod_{i=1}^{d} \frac{e^{-x_i^2 \big/ 2v^2}}{\sqrt{2\pi}v}, \tag{4.30}$$

$\mathbf{X} \sim \mathcal{U}\left(-[\frac{1}{2}, \frac{1}{2}]^d\right)$, $d = 20$, $u = 0.01$ and $v = 0.1$. This represents a Gaussian "spike" of width 0.01 superimposed on a Gaussian "plateau" of width 0.1. Figure 4.3 plots the log-likelihood $\log y$ against the log-tail distribution $\log p_y$.

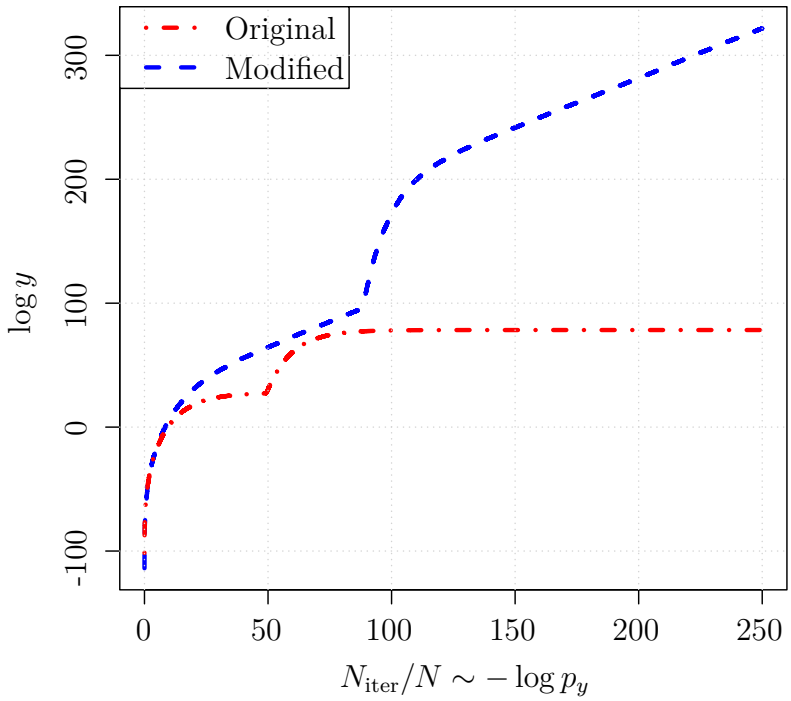


Figure 4.3: Log-Likelihood against probability for the original example of Skilling [2006, Section 18] defined in Eq. (4.30) and the modified version of Eq. (4.31). Both lines are got from a single run of nested sampling with $N = 300$ and stopping criterion $250N$ iterations.

We then run nested sampling with stopping criterion "number of iterations = $100N$" as well as $\widehat{Z}$ for several values of N from 100 to 500. Figure 4.4 shows the boxplots of the estimators and Table 4.1 summarises these numerical results over 500 simulations: both $\widehat{Z}$ and $\widehat{m}$ are unbiased while (NS) has a bias of order $1/N$ (cf Remark 4.3). The

variance increase between $\widehat{m}$ and $\widetilde{Z}$ is in good agreement with the theoretical relationship of Corollary 4.2, it is $\operatorname{var}\left[\widehat{Z}\right](N) = \operatorname{var}[\widehat{m}]((N+1)/2) \approx 2\operatorname{var}[\widehat{m}]$. Also the ratio $\operatorname{var}[\mathrm{NS}] / \operatorname{var}[\widehat{m}]$ goes from 1.14 to 1.9. This variance increase appears to be of order $1/N^2$, which is consistent with the variance increase between $\widehat{p}_y$ and $\widetilde{p}_y$ (see Remark 4.1).

Hence, the optimal choice of the nested sampling weights leads to significant variance reduction and removes the bias of the original nested sampling when it goes *far enough*. Unbiasedness can be maintained at the cost of at most doubling the variance of the estimator and even less compared to the currently used nested sampling weights. Furthermore, there is no need to choose (and justify) a stopping criterion for nested sampling any more.

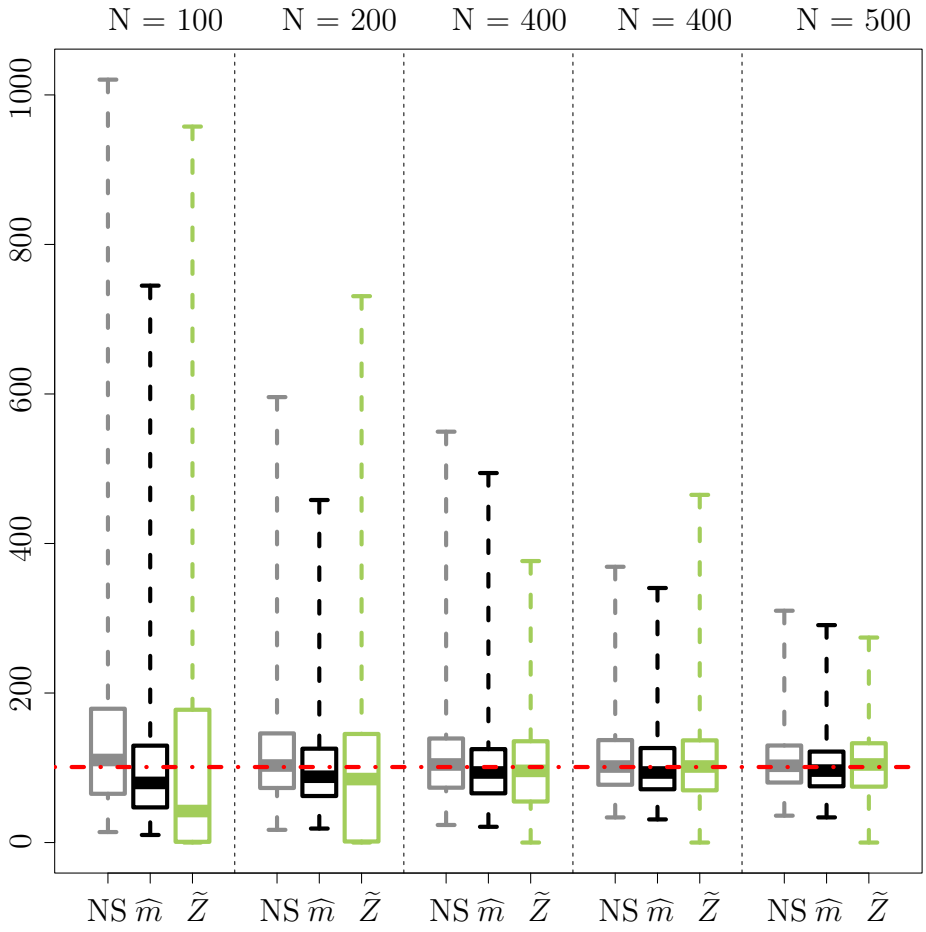


Figure 4.4: Boxplots of ideal *infinite* nested sampling $\widehat{m}$ of Eq. (4.4) and (NS) of Eq. (4.5) and randomly truncated $\widehat{Z}$ (Corollary 4.2) for the estimation of $\mathrm{E}[g(\mathbf{X})]$ with g as in Eq. (4.30) and $\mathbf{X} \sim \mathcal{U}\left(-[\frac{1}{2}, \frac{1}{2}]^d\right)$, $d = 20$. (NS) Ideal nested sampling is got with $N_{\text{iter}} = 100N$ as this is known to be enough in this case. (NS) and $\widehat{m}$ are obtained from the same runs. The (red) dot-dashed line is the theoretical value of m.

4.5.3 Adaptive stopping criteria

As we stated in the Introduction, one of the main concerns of this work was to point out the potential risk of using nested sampling with a *bad* stopping criterion. In this context we run nested sampling on the previous example with the adaptive stopping

N	100	200	300	400	500
E [NS]	142.3	117.7	114.6	111.5	109.5
E $[\widehat{m}]$	103.0	100.8	102.8	102.8	102.6
E $\left[\widehat{Z}\right]$	111.9	97.4	100.4	103.7	102.4
var $\left[\widehat{Z}\right]$ / var $[\widehat{m}]$	3.23	2.49	1.90	2.20	1.70
var [NS] / var $[\widehat{m}]$	1.90	1.33	1.24	1.17	1.14
var $\left[\widehat{Z}\right]$ / var [NS]	1.71	1.87	1.54	1.87	1.5

Table 4.1: Variance increase between the randomised unbiased nested sampling estimator $\widehat{Z}$, the original biased nested sampling (NS) and the ideal unbiased estimator $\widehat{m}$.

criteria mentioned in Section 4.3.5. The first one is directly picked out from [Chopin and Robert, 2010], it is "stop when the current increment is less than 10^{-8} times the current estimate". The second one is based on the estimation of the information H and is the one described in the Appendix of [Skilling, 2006]; it is "stop when the number of iterations is greater than $2NH$". Figure 4.4 shows that for $N = 500$ the estimators should be well converged and so we set $N = 500$.

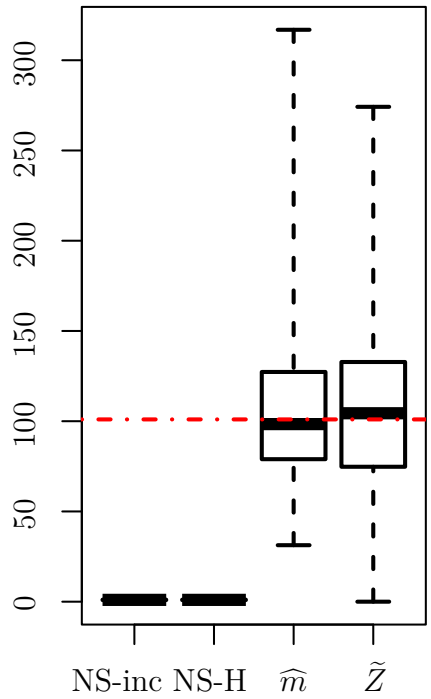
Figure 4.5 shows that nested sampling estimator can be not consistent if the termination rule is not well-chosen. Here both implementations miss the spike. In this context, the random truncation of $\widehat{Z}$ appears as a conservative practice. However, even though $\widehat{Z}$ allows for parallel computing (see Appendix A), $\widehat{Z}$ as well as the adaptive stopping criteria do not let work with a fixed computational budget. Yet one may have to work with fixed computational resources.

4.5.4 Nested sampling with fixed computational budget

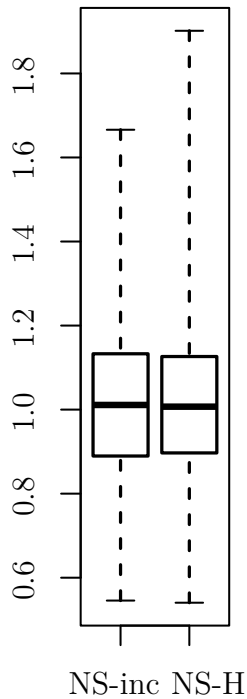
There is only one nested sampling implementation which allows for fixing the total computational budget in advance. It is the one which stops after a given number of iterations. Following Rhee and Glynn [2015] we have proposed in Sections 4.3.3 and 4.3.4 a randomised estimator which also works with a predetermined computational budget. It is still unbiased and supports a Central Limit Theorem. The goal of this section is to compare these two estimators. We slightly modify the previous example (see Eq. 4.30) to narrow the spike: $u = 0.001$ instead of $u = 0.01$, and to make the random variable heavy-tailed:

$$g_{\text{ht}}(\mathbf{x}) = g(\mathbf{x}) / \left(\sum_{i=1}^{d} x_i^2 \right)^{0.4d} . \tag{4.31}$$

Figure 4.3 compares this modified example with the original one. The heavy-tailed behaviour with tail index $1/0.8 = 1.25$ is clearly visible (limit slope of log-likelihood is 0.8), *i.e.* that $\mathrm{P}[Y > y] \sim k y^{-1.25}$ for some $k > 0$ as $y \to \infty$, as well as the effect of the narrower spike (shift of the mass from $-\log p \approx 50$ to $-\log p \approx 90$). With Inv-χ^2



(a) Nested sampling estimators with adaptive stopping criteria, $\widehat{m}$ and $\widehat{\widetilde{Z}}$



(b) Zoom on the nested sampling estimators with adaptive stopping criteria

Figure 4.5: Effect of the choice of a stopping criterion for nested sampling estimator when estimating $m = \mathrm{E}\left[g(\mathbf{X})\right]$ with g as in Eq. (4.30) and $\mathbf{X} \sim \mathcal{U}\left(-[\frac{1}{2}, \frac{1}{2}]^d\right)$, $d = 20$. (NS-inc): nested sampling stopped when current increment is less than 10^{-8} times the current estimator; (NS-H): nested sampling stopped when the number of iterations exceeds $2NH$; $\widehat{m}$ and $\widehat{\widetilde{Z}}$ as in Figure 4.4. The (red) dot-dashed line is the theoretical value of m.

approximation of $1/\sum X_i^2$, the sought value is $\mathrm{E}\left[g_{\mathrm{ht}}(\mathbf{X})\right] \approx 1.08 \times 10^{42}$.

Nested sampling is run with $N = 1000$ and $N = 10000$. We stop it after $100N$ iterations as in [Brewer et al., 2011]. This makes a total computational budget $c = 10^5$ (resp. 10^6). ζ is implemented with a suboptimal geometric randomising variable with parameter β_{app} (see Eq. 4.18) and $N = 20$. $N = d$ because it is both the theoretical parameter of ζ and the population size for conditional sampling. Hence it should not be set too small according to the dimension of the input space (see Appendix A). Considering the heavy-tail behaviour of $Y = g(\mathbf{X})$, the estimator has a finite variance as soon as $a > 1 + 1/N = 1.05$. One the one hand we know here that the tail index of Y is equal to $1/0.8 = 1.25$; on the other hand it is easy to check this condition afterwards by estimating the slope on the plot $\log Y$ against N_{iter}/N as in Figure 4.3.

It is visible on Figure 4.6 that nested sampling did not go far enough and misses an important part of the mass: $\mathrm{E}\left[\mathrm{NS}(10^5)\right] = 5.32 \times 10^{29}$ and $\mathrm{E}\left[\mathrm{NS}(10^6)\right] = 2.41 \times 10^{29}$ while the reference value is 1.08×10^{42}. On the one hand, ζ is unbiased (estimated means are 6.43×10^{41} and 1.52×10^{42}). On the other hand it does not seem to be approximately Gaussian yet. Indeed $\widehat{Z}$ can be relatively heavy-tailed [McLeish, 2011] and a consequent

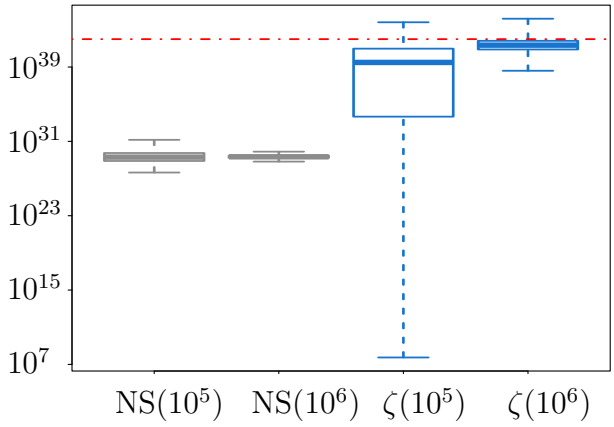


Figure 4.6: Estimation of $m = \mathrm{E}\left[g_{\mathrm{ht}}(\mathbf{X})\right]$ with $\mathbf{X} \sim \mathcal{U}\left(-[\frac{1}{2}, \frac{1}{2}]^d\right)$, $d = 20$. (NS): nested sampling stopped after $100N$ iterations; ζ: estimator of Section 4.3.4 with β_{app} (Eq. 4.18) and $N = 20$. 10^5 and 10^6 are the computational budgets used. The (red) dot-dashed line is the theoretical value of m.

computational budget may be required for ζ to effectively become normally distributed.

4.6 Conclusion

Nested Sampling has been proposed as a method for estimating the evidence in a Bayesian framework and applied with success in a great variety of areas like astronomy and cosmology. Since its introduction, a lot of work has been done to clarify its convergence properties [*e.g.* Evans, 2007, Chopin and Robert, 2010, Keeton, 2011] and to handle the issue of conditional sampling [*e.g.* Mukherjee et al., 2006, Brewer et al., 2011, Martiniani et al., 2014]. However nested sampling termination remains an open issue and a matter of user judgement [Skilling, 2006, Section 7].

Linking nested sampling with the results on rare event simulation, we have extended it to the estimation of the mean of any real-valued random variable (being bounded or not) and derived the optimal nested sampling weights. Furthermore we proved that 1) an idealised nested sampling with slightly modified weights and an infinite number of iterations is unbiased; 2) its variance is always lower than the classical Monte Carlo estimator one's; and 3) the random variable of interest does not need to have a finite second-order moment to produce an estimator with finite variance. This latter property makes nested sampling especially relevant for heavy-tailed random variables as developed Section 4.4.

Furthermore, we also present two ways of implementing a practical unbiased estimator with an a.s. finite number of terms, resolving the issue of choosing an arbitrary stopping criterion. The first estimator can be used exactly as usual nested sampling and preserves unbiasedness while only doubling the variance of the ideal estimator (infinite number of terms). The second one can be used with a predetermined fixed computational budget and supports a Central Limit Theorem. Practically speaking, they both enable parallel

implementation (unlike usual adaptive nested sampling strategies) and do not depend on the random variable of interest. Finally, it is worth mentioning that even if these randomising procedures can seem too burdened, at least the use of the corrected nested sampling weights ($1 - 1/N$ instead of $\exp -1/N$) in the usual nested sampling implementations removes the bias (if it goes *far enough*) and decreases the variance of the estimator.

Chapter 5

Rare event simulation with random processes

We have described in Chapter 1 advanced statistics for the estimation of extreme quantile and probability of the form $p = \mathrm{P}\left[g(\mathbf{X}) > q\right]$ with $\mathbf{X} \in \mathbb{X}$ a finite- or infinite-dimensional random vector with known distribution μ^X and g a deterministic function $g : \mathbb{X} \to \mathbb{R}$ *quantifying* the state of a *system*. We now focus on the static case, *i.e.* $\mathbb{X} \subset \mathbb{R}^d$ for some $d \geq 1$: $\mathbf{X}$ is a set of parameters of a computer code modelling an industrial system. Precisely at the end of this chapter we will focus on the simulation of a spherical tank under internal pressure and try to estimate the probability that the cumulated equivalent plastic strain of this tank be greater than a given security threshold q (failure mode).

In Chapter 3 we have presented the point process framework for extreme event simulation and have shown that it enables parallel implementation of what was known as the Last Particle Algorithm estimator, the minimal variance Multilevel Splitting estimator. Indeed we proved that this estimator was simply the Minimal Variance Unbiased Estimator (MVUE) of the exponential of a Poisson parameter. This lets us define parallel practical algorithms for estimating probability, quantile and moment of $g(\mathbf{X})$. This can dramatically reduce their computational time without lowering their good statistical properties (see Appendix A).

However in some settings only few hundred samples are available and it is still too much computational time. The above mentioned method typically requires $-bN_{\mathrm{batch}} \log p$ sequential simulations with N_{batch} the number of point processes per sequential algorithm and b a *burn-in* parameter for conditional sampling (see Sections 1.3.3 and A.2). With standard values of $N_{\mathrm{batch}} = 20$, $p = 10^{-7}$ and $b = 20$, this means that it would require ≈ 6447 sequential calls to the model g. In order to set these ideas down, our numerical code for the spherical tank has a random computational time depending on the plastic state in the vessel of about 5 to 10 minutes. It would eventually require on average 3.22×10^4 to 6.45×10^4 minutes or 23 to 45 days to compute the probability estimator. This can be even worse with more complex industrial problem for which computational time is about 5 hours.

In this context a common idea is to use a cheap-to-evaluate surrogate model ξ instead of the true model g. This surrogate model is trained with the computational budget and used either as a plug-in estimator in a crude Monte Carlo method or in more advanced statistics (see Chapter 2). Especially we have seen that when p is *small* the required

number of samples for a Monte Carlo estimator is very high and the failure domain may be very difficult to approach. That is why some algorithms now combine both methods, *i.e.* make use of Multilevel Splitting to both estimate the sought probability and improve the learning of the failure domain [Bourinet et al., 2011, Bourinet, 2016, Li et al., 2012, Bect et al., 2016].

In this chapter we propose to integrate the results on Poisson processes related to rare event simulation developed in Chapter 3 into these algorithms. Especially we show that the point process framework is well suited for the case where ξ is a random process with known distribution. Essentially it *adds a dimension* to the original problem defined with the deterministic code g but all the results can be applied directly on this *augmented problem*.

Furthermore, the point process framework allows for a cheap and natural computation of the SUR criteria presented in Section 2.2.3. Combined with the parallel algorithms presented in Appendix A it means that SUR strategies can be used even for extreme probability estimation at a very reasonable cost.

We also define new SUR criteria which aim at getting a global precision of the metamodel over the safety domain $\bar{F} = \{\mathbf{x} \in \mathbb{X} \mid g(\mathbf{x}) \leq q\}$. These criteria are also straight away estimated in the point process framework and allow for a more robust approach to the failure domain when the probability is very low. Especially, and unlike in the above mentioned splitting based learning approaches, it makes a clear distinction between the enrichment step and the estimation of the probability. In this mood the number of samples for the enrichment of the model can remain low. Furthermore it can output an estimation of the probability at any time: there is no need to wait until the algorithm reaches the final threshold.

Finally, we present a numerical study of these new criteria on usual test cases as well as on an industrial problem from CEA.

5.1 Rare event simulation

5.1.1 Augmented problem

We now consider that g is a realisation of a random process ξ with known distribution defined over a given probability space $(\Omega_0, \mathcal{F}_0, \mathrm{P}_0)$ indexed by $\mathbf{x} \in \mathbb{X}$:

$$\forall \omega_0 \in \Omega_0,\ \mathbf{x} \in \mathbb{X} \mapsto \xi(\mathbf{x}, \omega_0) \in \mathbb{R} \tag{5.1}$$

is a measurable function. Formally, one can consider that this assumption only *adds a dimension* to the original problem. Recall that $\mathbf{X} \in \mathbb{X}$ is a random variable defined on the probability space $(\Omega, \mathcal{F}, \mathrm{P})$, one can consider the random variable on the product space $\Omega_Y = \Omega \times \Omega_0$:

$$Y : (\omega, \omega_0) \in \Omega \times \Omega_0 \mapsto \xi(\mathbf{X}(\omega), \omega_0) \in \mathbb{R}.$$

Eventually the real-valued random variable is not $Y = g(\mathbf{X})$ any more, but $Y = \xi(\mathbf{X}(\omega), \omega_0)$, defined on the probability space $(\Omega_Y, \mathcal{F}_Y, \mathrm{P}_Y)$. In essence the uncertainty on the code is treated exactly the same way as the uncertainty on the inputs: as soon as the computer code g is regarded as a black-box, it means that only some *point-wise measurements* can be done, exactly like, for example, the thickness of the tank. In this latter case this uncertainty is modelled with a random variable with a known distribution, and so it is for the computer code. When there is no risk of confusion, we will drop in the sequel the dependence on the variables ω and ω_0 and write only $Y = \xi(\mathbf{X})$.

We will refer to the original setting with the deterministic code g as the deterministic case, and to the following setting with the random process ξ as the random case. While in the deterministic case, Y given $\mathbf{X}$ was a deterministic quantity, we now have that Y given $\mathbf{X}$ follows the distribution $\mu^{\xi(\mathbf{X})}$ of the random process at point $\mathbf{X}$. Still the problem writes:

$$\mathrm{P}_Y\left[Y > q\right] = \int_{\mathbb{R}} \mathbb{1}_{y>q} \mathrm{d}\mu^Y(y) = \int_{\mathbb{X}\times\mathbb{R}} \mathbb{1}_{y>q} \mathrm{d}\mu^{\xi(\mathbf{x})}(y) \mathrm{d}\mu^X(\mathbf{x}). \tag{5.2}$$

Eventually all the results on the point process framework for rare event simulation are valid because they make no assumption on the real-valued random variable Y: probability, quantile and moments of Y can be estimated using the methods described in Chapters 3 and 4. Especially the random process does not need to be Gaussian.

5.1.2 Link with other metamodel based algorithms

We recall here some well-used methods for probability estimation based on random processes and presented in Section 2.3.

AKMCS The Active learning using Kriging and Monte Carlo Simulation [AKMCS, Echard et al., 2011] method uses the mean function of the random process: $m : \mathbf{x} \in \mathbb{X} \mapsto \mathrm{E}_0\left[\xi(\mathbf{x})\right] = m(\mathbf{x})$ to define a surrogate classifier: $\mathbb{1}_{g(\mathbf{x})>q} \leftarrow \mathbb{1}_{m(\mathbf{x})>q}$. This mean function is supposed to be cheap-to-evaluate for any $\mathbf{x} \in \mathbb{X}$ and then AKMCS proposes to make a crude Monte Carlo estimation of the quantity $\mathrm{P}\left[m(\mathbf{X}) > q\right] = \mathrm{P}\left[\mathrm{E}_0\left[\xi(\mathbf{X})\right] > q\right]$. In the light of the point process framework, this calls for few comments:

1. when the sought probability is extreme, the quantity $\mathrm{P}\left[m(\mathbf{X}) > q\right]$ will be hard to estimate with a crude Monte Carlo sampling even though m is *cheap*. Here the point process estimator could be used.

2. it neglects the uncertainty due to the non-perfect knowledge of g and to consider that the point-wise mean of the process is *close* to the unknown function g. In other words, it means that the error due to the random process is now measured in terms of area in $\mathbb{X}$. Since the true function g is not accessible, this quantity cannot be known: the error due to the process is not taken into account any more.

MetaIS The Metamodel-based Importance Sampling [MetaIS, Dubourg et al., 2013] method uses the *cdf* of $\mu^{\xi(\mathbf{x})}$ to define an importance distribution for the estimation of the probability $\mathrm{P}\left[g(\mathbf{X}) > q\right]$. Indeed, recall that the optimal importance distribution is defined by (see Section 1.2):

$$\mathrm{d}\mu^{\tilde{X}}(\mathbf{x}) = \frac{\mathbb{1}_{g(\mathbf{x})>q}}{p}\mathrm{d}\mu^{X}(\mathbf{x})$$

it proposes to approximate the indicator function $\mathbf{x} \mapsto \mathbb{1}_{g(\mathbf{x})>q}$ with the complementary *cdf* of $\xi(\mathbf{x})$: $\mathbb{1}_{g(\mathbf{x})>q} \leftarrow \mathrm{P}_0\left[\xi(\mathbf{x}) > q\right] = \mu^{\xi(\mathbf{x})}((q,\infty))$. In this context, the normalising constant of the importance distribution becomes:

$$\widetilde{p}_{\mathrm{aug}} = \int_{\mathbb{X}} \mu^{\xi(\mathbf{x})}((q,\infty))\mathrm{d}\mu^{X}(\mathbf{x}) = \int_{\mathbb{X}}\int_{\mathbb{R}} \mathbb{1}_{y>q}\mathrm{d}\mu^{\xi(\mathbf{x})}(y)\mathrm{d}\mu^{X}(\mathbf{x}). \tag{5.3}$$

This latter quantity is indeed the one found in Eq. (5.2) and is referred to as the *augmented failure probability* by Dubourg et al. [2013]. This means that the point process estimator can directly be used in the MetaIS method as an other tool to estimate it. Instead Dubourg et al. [2013] suggested the use of a crude Monte Carlo. Here the point process estimator is especially interesting because the points got from this crude Monte Carlo sampling are also used as initial states for the sampling of the importance density with a Metropolis-Hastings method. As explained in Chapter 4, the point process can be seen as both a mean to estimate the normalising constant of the distribution and to generate samples according to it. In this context, it would output both a more precise estimation of the augmented failure probability and samples for the Importance Sampling scheme.

Bayesian approach The Bayesian decision-theoric framework has been applied to extreme probability estimation first by Bect et al. [2012]. In this context, it considers the random variable $\alpha = \mathrm{P}\left[\xi(\mathbf{X}) > q\right]$ instead of the deterministic quantity $p = \mathrm{P}\left[g(\mathbf{X}) > q\right]$ and uses the (conditional) expectation of α as an estimator for p (see Section 2.2.3). This expectation writes:

$$\mathrm{E}_0\left[\alpha\right] = \mathrm{E}_0\left[\int_{\mathbb{X}} \mathbb{1}_{\xi(\mathbf{x})>q}\mathrm{d}\mu^{X}(\mathbf{x})\right] = \int_{\mathbb{X}} \mathrm{E}_0\left[\mathbb{1}_{\xi(\mathbf{x})>q}\right]\mathrm{d}\mu^{X}(\mathbf{x}) = \int_{\mathbb{X}\times\Omega_0} \mathbb{1}_{\xi(\mathbf{x},\omega)>q}\mathrm{d}\mu^{X}(\mathbf{x})\mathrm{d}\,\mathrm{P}_0(\omega). \tag{5.4}$$

This latter quantity is the one found in Eq. (5.2) and the point process framework can also be used in this algorithm as the tool to estimate it.

Therefore the point process estimator can be used directly in already-existing metamodel-based algorithms instead of crude Monte Carlo to estimate the probability of failure.

As presented in Chapter 2, such strategies are the combination of a statistical method (crude Monte Carlo, Importance Sampling or Splitting for instance) and a learning strategy. In this thesis we argue that a clear distinction has to be made between these two aspects

of metamodel based algorithms. For instance the heuristic criteria of AKMCS or MetaIS (see also Section 2.2.2) have proven good practical results and should not be underrated. In the next section, we show how the point process associated with Y can be used to quantify the uncertainty due to the use of the random process ξ. We also show how it can be used to ease the use of SUR strategies presented in Section 2.2.3.

5.1.3 Uncertainty reduction

From an initial prior on the distribution of the random process, the goal is to evaluate iteratively the deterministic code g at some locations to reduce the uncertainty carried out by the introduction of the random process. In Section 2.2.3 we presented the Stepwise Uncertainty Reduction (SUR) framework, which aims at sequentially choosing the next sample(s) to be evaluated in order to minimise the *remaining error*.

We briefly recall here the quantities and the formalism described in this section. We focus on the random variable α defined in Eq. (2.38), it is:

$$\alpha = \mathrm{P}\left[\xi(\mathbf{X}) > q\right] = \int_{\mathbb{X}} \mathbb{1}_{\xi(\mathbf{x})>q} \mathrm{d}\mu^X(\mathbf{x}) \tag{5.5}$$

because it embeds the uncertainty on the probability of failure $\mathrm{P}_Y\left[Y > q\right]$ due to the random process ξ.

Recall that ξ is defined on the probability space $(\Omega_0, \mathcal{F}_0, \mathrm{P}_0)$, we also define for any $n \geq 1$ the $\sigma-$algebra $\mathcal{F}_n$ generated by the random vector $(\xi(\mathbf{x}_i))_{i=1}^n$ for any set of points $(\mathbf{x}_1, \cdots, \mathbf{x}_n) \in \mathbb{X}^n$. We further assume that ξ is a stationary process with finite variance (see Section 2.1.4).

We denote by $\mu_n^{\xi(\mathbf{x})}$ the conditional distribution of $\xi(\mathbf{x}) \mid \mathcal{F}_n$, $\xi_n(\mathbf{x})$ the conditional mean $\mathrm{E}_0\left[\xi(\mathbf{x}) \mid \mathcal{F}_n\right]$ and $\sigma_n^2(\mathbf{x})$ the conditional variance; P_n is the conditional distribution $\mathrm{P}_0\left[\cdot \mid \mathcal{F}_n\right]$, $\mathrm{E}_n\left[\cdot\right] = \mathrm{E}_0\left[\cdot \mid \mathcal{F}_n\right]$ the conditional expectation and $\mathrm{cov}_n\left[\cdot\right] = \mathrm{cov}_0\left[\cdot \mid \mathcal{F}_n\right]$ the conditional covariance.

In the Bayesian decision-theoretic framework, an algorithm is seen as a sequence of decisions made from an increasing amount of information in order to reduce the *risk* carried out with the final approximation α_n of α. Here a decision corresponds to the evaluation of the computer code at one or more locations and n is a total number of calls to g. By selecting the quadratic loss function, the risk is:

$$\mathrm{E}_0\left[\left(\alpha - \alpha_n\right)^2\right].$$

By definition of the conditional expectation, this quantity is minimised amongst all $\mathcal{F}_n$-measurable functions by:

$$\alpha_n = \mathrm{E}_n\left[\alpha\right] = \int_{\mathbb{X}} p_n^q(\mathbf{x}) \mathrm{d}\mu^X(\mathbf{x}) = \mathrm{P}_Y\left[Y > q \mid \mathcal{F}_n\right] \tag{5.6}$$

with $p_n^q(\mathbf{x}) = \mathrm{P}_n\left[\xi(\mathbf{x}) > q\right]$. Conditionally to $\{\xi(\mathbf{x}_i) = g(\mathbf{x}_i)\}$, Eq. (5.6) is deterministic

and can be estimated with a point process as described previously. The conditional variance:

$$\nu_n = \mathrm{E}_n\left[(\alpha - \alpha_n)^2\right]$$

is a measure of the remaining error, *i.e.* of the randomness of α not captured by the filtration $\mathcal{F}_n$. We can also derive an upper bound of ν_n as in Section 2.2.3:

$$\Gamma_n = \int_{\mathbb{X}} \mathrm{var}_n\left[\mathbb{1}_{\xi(\mathbf{x})>q}\right] \mathrm{d}\mu^X(\mathbf{x}) = \int_{\mathbb{X}} p_n^q(\mathbf{x})\left(1 - p_n^q(\mathbf{x})\right) \mathrm{d}\mu^X(\mathbf{x}). \tag{5.7}$$

This upper bound is interesting because it reduces the integral over $\mathbb{X} \times \mathbb{X}$ of ν_n to an integral over $\mathbb{X}$.

A direct application of the SUR concept in this case leads to find at a given iteration n the $r \geq 1$ next samples $(\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r}) \in \mathbb{X}^r$ minimising the *forthcoming* conditional variance:

$$J^{\alpha}_{n,n+r}(\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r}) = \mathrm{E}_n\left[\nu_{n+r}(\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r})\right] \tag{5.8}$$

or its upper bound:

$$J^{\Gamma}_{n,n+r}(\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r}) = \mathrm{E}_n\left[\Gamma_{n+r}(\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r})\right] \tag{5.9}$$

where:

$$\begin{aligned}
\nu_{n+r}(\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r}) &= \mathrm{var}\left[\alpha \mid \mathcal{F}_n, \xi(\mathbf{x}_{n+1}), \cdots, \xi(\mathbf{x}_{n+r})\right] \\
\Gamma_{n+r}(\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r}) &= \int_{\mathbb{X}} \mathrm{var}\left[\mathbb{1}_{\xi(\mathbf{x})>q} \mid \mathcal{F}_n, \xi(\mathbf{x}_{n+1}), \cdots, \xi(\mathbf{x}_{n+r})\right] \mathrm{d}\mu^X(\mathbf{x})
\end{aligned}$$

Note that with these criteria, the selected sample(s) depend(s) measurably on $\mathcal{F}_n$ [Bect et al., 2012].

We further assume that ξ is a Gaussian process. With standard covariance kernels described in Section 2.1.4, Y is then a continuous random variable and the increasing random walk associated to Y is a Poisson process. In Section 2.2.3 we showed that these quantities can be rewritten assuming that ξ is a Gaussian process. Using Eq. (2.41), Eq. (5.8) becomes:

$$\begin{aligned}
J^{\alpha}_{n,n+r}(\mathbf{x}^*) = \mathrm{E}_n\left[\alpha^2\right] - \\
\int_{\mathbb{X}\times\mathbb{X}} \mathrm{P}_n\left[U^{(2)}_{n+r} > q, U^{(1)}_{n+r} > q \mid \mathbf{X}_1 = \mathbf{x}_1, \mathbf{X}_2 = \mathbf{x}_2\right] \mathrm{d}\mu^X(\mathbf{x}_1)\mathrm{d}\mu^X(\mathbf{x}_2)
\end{aligned} \tag{5.10}$$

with $\mathbf{x}^* = (\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r}) \in \mathbb{X}^r$ and $(U^{(1)}_{n+r}, U^{(2)}_{n+r})^\top$ a random vector whose conditional distribution given $(\mathbf{X}_1, \mathbf{X}_2)^\top$ and $\mathcal{F}_n$ is:

$$\begin{pmatrix} U^{(1)}_{n+r} \\ U^{(2)}_{n+r} \end{pmatrix} \mid \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{pmatrix}, \mathcal{F}_n \sim \mathcal{N}\left(\begin{pmatrix} \xi_n(\mathbf{X}_1) \\ \xi_n(\mathbf{X}_2) \end{pmatrix}, \begin{pmatrix} \sigma^2_n(\mathbf{X}_1) & \mathrm{cov}_n\left[\xi_{n+r}(\mathbf{X}_1), \xi_{n+r}(\mathbf{X}_2)\right] \\ \mathrm{cov}_n\left[\xi_{n+r}(\mathbf{X}_1), \xi_{n+r}(\mathbf{X}_2)\right] & \sigma^2_n(\mathbf{X}_2) \end{pmatrix}\right). \tag{5.11}$$

Eq. (5.10) can be rewritten:

$$J^{\alpha}_{n,n+r}(\mathbf{x}^*) = \mathrm{E}_n\left[\alpha^2\right] - \alpha_n \int_{\mathbb{X}\times\mathbb{X}\times\mathbb{R}} \mathrm{P}_n\left[U^{(2)}_{n+r} > q \mid U^{(1)}_{n+r} = y, \mathbf{X}_1 = \mathbf{x}_1, \mathbf{X}_2 = \mathbf{x}_2\right] \frac{\mathbb{1}_{y>q}\mathrm{d}\mu_n^{\xi(\mathbf{x}_1)}(y)\mathrm{d}\mu^X(\mathbf{x}_1)}{\alpha_n}\mathrm{d}\mu^X(\mathbf{x}_2).$$

This latter expression shows that this criterion can be estimated using the point process on Y: as for the MetaIS algorithm, it can be used to both estimate α_n and to get *iid.* samples according to:

$$\frac{\mathbb{1}_{y>q}\mathrm{d}\mu_n^{\xi(\mathbf{x}_1)}(y)\mathrm{d}\mu^X(\mathbf{x}_1)}{\alpha_n}.$$

These samples can be used together with *iid.* samples drawn according to μ^X to get an *iid.* population well suited to the estimation of the second part of the equality (see Section 5.2.1 below). As mentioned by Chevalier et al. [2014], the first one $\mathrm{E}_n\left[\alpha^2\right]$ does not depend on $\mathbf{x}^*$ and it is thus pointless to strive to estimate it for the SUR strategy. Note that when $r = 0$, $U^{(1)}_n$ and $U^{(2)}_n$ are independent conditionally to $\mathcal{F}_n$ and one finds back:

$$J^{\alpha}_{n,n} = \mathrm{E}_n\left[\alpha^2\right] - \alpha_n^2 = \nu_n.$$

This development can also be conducted with the upper bound Γ_n. Applying the same reasoning to Eq. (2.48), one finds:

$$\begin{aligned} J^{\Gamma}_{n,n+r}(\mathbf{x}^*) &= \int_{\mathbb{X}} \mathrm{P}_n\left[U^{(2)}_{n+r} < q, U^{(1)}_{n+r} > q \mid \mathbf{X} = \mathbf{x}\right] \mathrm{d}\mu^X(\mathbf{x}) \\ &= \alpha_n \int_{\mathbb{X}\times\mathbb{R}} \mathrm{P}_n\left[U^{(2)}_{n+r} < q \mid U^{(1)}_{n+r} = y, \mathbf{X} = \mathbf{x}\right] \frac{\mathbb{1}_{y>q}\mathrm{d}\mu_n^{\xi(\mathbf{x})}(y)\mathrm{d}\mu^X(\mathbf{x})}{\alpha_n}. \end{aligned} \tag{5.12}$$

Note that for this criterion, the distribution of the couple $U^{(1)}_{n+r}$ and $U^{(2)}_{n+r}$ is conditional on only one sample in $\mathbf{X} \in \mathbb{X}$ since Γ_n reduces the integral over $\mathbb{X}\times\mathbb{X}$ to an integral over $\mathbb{X}$:

$$\begin{pmatrix} U^{(1)}_{n+r} \\ U^{(2)}_{n+r} \end{pmatrix} \mid \mathbf{X}, \mathcal{F}_n \sim \mathcal{N}\left(\begin{pmatrix} \xi_n(\mathbf{X}) \\ \xi_n(\mathbf{X}) \end{pmatrix}, \begin{pmatrix} \sigma_n^2(\mathbf{X}) & \mathrm{cov}_n\left[\xi_{n+r}(\mathbf{X}), \xi_{n+r}(\mathbf{X})\right] \\ \mathrm{cov}_n\left[\xi_{n+r}(\mathbf{X}), \xi_{n+r}(\mathbf{X})\right] & \sigma_n^2(\mathbf{X}) \end{pmatrix}\right). \tag{5.13}$$

For $r = 0$, $U^{(1)}_n$ and $U^{(2)}_n$ are decorrelated conditional on $\mathcal{F}_n$ and $J^{\Gamma}_{n,n} = \Gamma_n$ appears indeed as the average probability of *misclassification* of the last samples of the point process.

Furthermore, the Mean Squared Error (MSE) in the estimation of $p = \mathrm{P}\left[g(\mathbf{X}) > q\right]$ by $\widehat{\alpha_n}$ writes:

$$\begin{aligned} \mathrm{E}\left[(\widehat{\alpha_n} - p)^2\right] &= \mathrm{E}\left[\mathrm{E}\left[(\widehat{\alpha_n} - p)^2 \mid \mathcal{F}_n\right]\right] = \mathrm{E}\left[\alpha_n^{2-1/N} + p^2 - 2\alpha_n p\right] \\ &= \mathrm{E}\left[\alpha_n^2\left(\alpha_n^{-1/N} - 1\right) + (\alpha_n - p)^2\right]. \end{aligned} \tag{5.14}$$

In this latter equality, the first term is due to the estimation of α_n while the second one is related to the conditional variance of α, *i.e.* the remaining randomness due to ξ:

conditionally to $\{\xi(\mathbf{x}) = g(\mathbf{x}), \forall \mathbf{x} \in \mathbb{X}\}$, one has $\alpha = p$.

Numerical results of Section 5.3.2 will illustrate this trade-off between model error and statistical error. While it appears that the number N of random processes used to estimate α_n should be as large as possible to reduce the statistical error below (or at the same level as) the model error, a relatively moderate number of Poisson processes N_{SUR} can be sufficient to drive the learning of the metamodel. This can be useful to accelerate the minimisation of the SUR criterion and eventually the whole *learning step*.

5.1.4 Integrated SUR criteria

Let us denote by $\mathbf{X}_{\text{aug}} = (\mathbf{X}, Y) \in \mathbb{X} \times \mathbb{R}$ the random vector with distribution conditional on $\mathcal{F}_n$, $\mu_{n,q}^{\mathbf{X}_{\text{aug}}}$, such that:

$$\mathrm{d}\mu_{n,q}^{\mathbf{X}_{\text{aug}}}(\mathbf{x}, y) = \frac{\mathbb{1}_{y>q} \mathrm{d}\mu_n^{\xi(\mathbf{x})}(y) \mathrm{d}\mu^X(\mathbf{x})}{\alpha_n}. \tag{5.15}$$

$\mathrm{E}_n[\alpha^2]$ can be rewritten:

$$\begin{aligned} \mathrm{E}_n\left[\alpha^2\right] &= \int_{\mathbb{X}^2} \mathrm{P}_n\left[\xi(\mathbf{x}_1) > q, \xi(\mathbf{x}_2) > q\right] \mathrm{d}\mu^X(\mathbf{x}_1) \mathrm{d}\mu^X(\mathbf{x}_2) \\ &= \alpha_n \int_{\mathbb{X}^2 \times \mathbb{R}} \mathrm{P}_n\left[\xi(\mathbf{x}_2) > q \mid \xi(\mathbf{x}_1) = y\right] \mathrm{d}\mu_{n,q}^{\mathbf{X}_{\text{aug}}}(\mathbf{x}_1, y) \mathrm{d}\mu^X(\mathbf{x}_2). \end{aligned} \tag{5.16}$$

Eventually the criterion $J_{n,n+r}^{\alpha}$ can be rewritten:

$$J_{n,n+r}^{\alpha}(\mathbf{x}^*) = \alpha_n \int_{\mathbb{X}^2 \times \mathbb{R}} \left(p_n^q(\mathbf{x}_2 \mid \mathbf{x}_1, y) - p_{U_{n+r}}^q(\mathbf{x}_2 \mid \mathbf{x}_1, y) \right) \mathrm{d}\mu_{n,q}^{\mathbf{X}_{\text{aug}}}(\mathbf{x}_1, y) \mathrm{d}\mu^X(\mathbf{x}_2) \tag{5.17}$$

with:

$$\begin{aligned} p_n^q(\mathbf{x}_2 \mid \mathbf{x}_1, y) &= \mathrm{P}_n\left[\xi(\mathbf{x}_2) > q \mid \xi(\mathbf{x}_1) = y\right] \\ p_{U_{n+r}}^q(\mathbf{x}_2 \mid \mathbf{x}_1, y) &= \mathrm{P}_n\left[U_{n+r}^{(2)} > q \mid \mathbf{X}_1 = \mathbf{x}_1, \mathbf{X}_2 = \mathbf{x}_2, U_{n+r}^{(1)} = y\right]. \end{aligned}$$

For $J_{n,n+r}^{\Gamma}$, and using Eq. (5.12), it reduces to:

$$J_{n,n+r}^{\Gamma}(\mathbf{x}^*) = \alpha_n \int_{\mathbb{X} \times \mathbb{R}} \left(1 - p_{U_{n+r}}^q(\mathbf{x} \mid \mathbf{x}, y)\right) \mathrm{d}\mu_{n,q}^{\mathbf{X}_{\text{aug}}}(\mathbf{x}, y). \tag{5.18}$$

On the one hand both Eqs. (5.17) and (5.18) are expressed as the product of α_n (the sought probability) and an integral which can be seen as an error term for the level q. Indeed in the estimation of the criterion, only the integral part depends on the proposed batch $\mathbf{x}^*$.

On the other hand methods such as ^{2}SMART [Deheeger, 2008, Bourinet et al., 2011] or BSS [Li et al., 2012, Bect et al., 2016] propose to use a Subset Simulation algorithm (see Section 1.3.2) to progressively learn the model on a sequence of increasing thresholds

such that the failure domain is iteratively approached but never corresponds to an *extreme* event (see Section 2.3.3). These algorithms have shown good practical results. However they suffer from the same limitations as the original Subset Simulation algorithm: choice of the thresholds, parallelisation, optimality... Eventually they *learn* a sequence of thresholds which are useless in the end for the quantification of the final estimator. Finally the question of the computational load for each intermediate threshold is an open issue.

The point process framework appears here as a good theoretical tool to integrate these empirical improvements into a well-defined algorithm. Indeed, let us denote by

$$h^{\alpha}_{n,n+r}(q,\mathbf{x}^*) = \frac{J^{\alpha}_{n,n+r}(\mathbf{x}^*)}{\alpha_n}(q) \tag{5.19}$$

$$h^{\Gamma}_{n,n+r}(q,\mathbf{x}^*) = \frac{J^{\Gamma}_{n,n+r}(\mathbf{x}^*)}{\alpha_n}(q) \tag{5.20}$$

the *error term* of the model according to each one of the two criteria.

In the Poisson process framework, a natural idea to assess the global precision of the model over the whole safety domain is to integrate[1] these quantities as function of q against the mean measure λ_n of the Poisson process associated with Y conditional on $\mathcal{F}_n$. If one considers that the metamodel should be *accurate* over the whole interval $(-\infty, q]$, one can define:

$$I^{\alpha}_{n,n+r}(\mathbf{x}^*) = \int_{\mathbb{R}} \mathbb{1}_{y\leq q} h^{\alpha}_{n,n+r}(y,\mathbf{x}^*)\mathrm{d}\lambda_n(y) \tag{5.21}$$

$$I^{\Gamma}_{n,n+r}(\mathbf{x}^*) = \int_{\mathbb{R}} \mathbb{1}_{y\leq q} h^{\Gamma}_{n,n+r}(y,\mathbf{x}^*)\mathrm{d}\lambda_n(y). \tag{5.22}$$

Then the Campbell's Theorem (see Section 4.2.2) can be applied to estimate Eqs. (5.21) and (5.22):

$$\widehat{I^{\alpha}_{n,n+r}(\mathbf{x}^*)} = \sum_{i\geq 1} h^{\alpha}_{n,n+r}(Y_i,\mathbf{x}^*)\mathbb{1}_{Y_i\leq q} = \sum_{i\geq 1} \frac{J^{\alpha}_{n,n+r}(\mathbf{x}^*)}{\alpha_n}(Y_i)\mathbb{1}_{Y_i\leq q} \tag{5.23}$$

$$\widehat{I^{\Gamma}_{n,n+r}(\mathbf{x}^*)} = \sum_{i\geq 1} h^{\Gamma}_{n,n+r}(Y_i,\mathbf{x}^*)\mathbb{1}_{Y_i\leq q} = \sum_{i\geq 1} \frac{J^{\Gamma}_{n,n+r}(\mathbf{x}^*)}{\alpha_n}(Y_i)\mathbb{1}_{Y_i\leq q} \tag{5.24}$$

where $(Y_i)_{i\geq 1}$ are the arrival times of a Poisson process associated with Y conditional on $\mathcal{F}_n$, *i.e.* with conditional distribution μ^Y_n such that:

$$\mathrm{d}\mu^Y_n(y) = \int_{\mathbb{X}} \mathrm{d}\mu^{\xi(\mathbf{x})}_n(y)\mathrm{d}\mu^X(\mathbf{x}).$$

In other words, these criteria write as the sum of the *point-wise* usual criteria over the states of the Poisson process. It means that it considers the precision of the metamodel for each level *used* for the conditional simulations. In the end, this quantity quantifies the variance increase in the generation of the counting random variables due to the use of a

[1] the original idea of integrating a SUR criterion comes from Julien Bect.

random process instead of a deterministic code g.

More precisely, using the expression of $\mathrm{d}\lambda_n$ defined in Section 4.2.2, one has:

$$\mathrm{d}\lambda_n(y) = \frac{\mathrm{d}\mu_n^Y(y)}{\alpha_n}$$

so that the criteria rewrite:

$$I^{\alpha}_{n,n+r}(\mathbf{x}^*) = \int_{\mathbb{R}} \mathbb{1}_{y\leq q} \frac{J^{\alpha}_{n,n+r}(\mathbf{x}^*)}{\alpha_n^2}(y)\mathrm{d}\mu_n^Y(y) \tag{5.25}$$

$$I^{\Gamma}_{n,n+r}(\mathbf{x}^*) = \int_{\mathbb{R}} \mathbb{1}_{y\leq q} \frac{J^{\Gamma}_{n,n+r}(\mathbf{x}^*)}{\alpha_n^2}(y)\mathrm{d}\mu_n^Y(y). \tag{5.26}$$

Especially, for $r = 0$, $J^{\alpha}_{n,n+r} = \mathrm{var}_n[\alpha]$ and these integrated criteria appear as the average of the squared coefficient of variation (or an upper bound) of α over $(-\infty, q]$:

$$I^{\alpha}_{n,n} = \int_{\mathbb{R}} \mathbb{1}_{y\leq q} \frac{\mathrm{var}_n[\alpha]}{\mathrm{E}_n[\alpha]^2}(y)\mathrm{d}\mu_n^Y(y).$$

This is consistent with the observation that the point process framework not only produces an estimator of the sought probability $\mathrm{P}[Y > q]$ but of the whole *cdf* of Y over $(-\infty, q]$ (see Section 3.3.1). The numerical examples of Section 5.3 will illustrate the behaviour of these new SUR criteria. They appear to be more robust, especially for extreme events, but also more computationally demanding in computational time.

5.2 Algorithms

Algorithms presented in Appendix A for parallel Poisson process generation and probability estimation can be used directly. The only difference stands in the sampling of Y: in the deterministic case, one draws $\mathbf{X}$ to fully determine Y while in the random case, Y given $\mathbf{X}$ is a random variable with distribution $\mu^{\xi(\mathbf{X})}$ (which is Gaussian if ξ is a Gaussian process) and has to be sampled too.

We focus here on the estimation of the SUR criteria as well as on the workflow of an algorithm combining the point process framework and SUR strategies.

In all this section, we consider that one can generate *iid.* Poisson processes associated with Y. Appendix A Section A.2 gives all the details on practical parallel implementation for this generation. Furthermore, for the sake of completeness we recall the estimator $\widehat{\alpha_n}$ built from N *iid.* Poisson processes:

$$\widehat{\alpha_n} = \left(1 - \frac{1}{N}\right)^{\bar{M}_q}$$

with $\bar{M}_q = \sum_{i=1}^{N} M_q^i$ the sum of the N *iid.* counting random variables of the number of

events *before* q on each Poisson process.

5.2.1 SUR criteria estimation

We first address the issue of estimating $J^{\alpha}_{n,n+r}$ and $J^{\Gamma}_{n,n+r}$ for given $n \geq 0$ and $r \geq 0$. In the sequel we assume that $N_{\mathrm{SUR}} \geq 1$ *iid.* Poisson processes associated with a random variable with distribution μ^Y_n have been generated until state q.

Let $(\mathbf{X}_i, Y_i)_{i=1}^{N_{\mathrm{SUR}}}$ be the N_{SUR} *iid.* samples with distribution $\mu^{\mathbf{X}_{\mathrm{aug}}}_{n,q}$ (see Eq. 5.15). In other words $(Y_i)_{i=1}^{N_{\mathrm{SUR}}}$ are the N_{SUR} first states above q of the N_{SUR} Poisson processes and $(\mathbf{X}_i)_{i=1}^{N_{\mathrm{SUR}}}$ are the N_{SUR} samples in $\mathbb{X}$ such that for each $i \in [\![1, N_{\mathrm{SUR}}]\!]$, Y_i was sampled $\sim \mu^{\xi(\mathbf{X}_i)}_n$.

Even though $J^{\alpha}_{n,n+r}$ depends on $\mathrm{E}_n[\alpha^2]$, this quantity is not required to solve the minimisation problem:

$$\underset{(\mathbf{x}_n,\cdots,\mathbf{x}_{n+r})\in\mathbb{X}^r}{\operatorname{argmin}} J^{\alpha}_{n,n+r}(\mathbf{x}_n,\cdots,\mathbf{x}_{n+r}). \tag{5.27}$$

However we first give an algorithm for estimating $\mathrm{E}_n[\alpha^2]$, which can be useful to estimate the conditional variance at a given iteration.

Algorithm 9 Point process based estimation of $\mathrm{E}_n[\alpha^2]$ (see Eq. 5.16).

Require: N_{SUR} *iid.* Poisson process associated with Y

Require: $\widehat{\alpha_n}$ an estimator of α_n

Get $(\mathbf{X}^i, Y_i)_{i=1}^{N_{\mathrm{SUR}}}$ the N_{SUR} *iid.* samples $\sim \mu^{\mathbf{X}_{\mathrm{aug}}}_{n,q}$ ▷ first states of the Poisson processes above q

Generate $(\mathbf{X}^i_2)_{i=1}^{N_{\mathrm{SUR}}}$ *iid.* samples $\sim \mu^X$

for $i = 1..N_{\mathrm{SUR}}$ **do**

get m_i and σ_i^2 the mean and variance of a Gaussian rv with distribution $\xi(\mathbf{X}^i_2) \mid \mathcal{F}_n, \xi(\mathbf{X}^i) = Y_i$

end for

return $s^{\alpha}_n(q) = \dfrac{1}{N_{\mathrm{SUR}}} \times \sum_{i=1}^{N_{\mathrm{SUR}}} (1 - \Phi(q|m_i, \sigma_i^2))$ ▷ $\Phi(\cdot|m, \sigma^2)$ is the *cdf* of a rv $\mathcal{N}(m, \sigma^2)$

return $\widehat{\alpha_n} \times s^{\alpha}_n(q)$

We present in Algorithm 10 how to estimate $J^{\alpha}_{n,n+r}$ assuming that estimators of α_n and $\mathrm{E}_n[\alpha^2]$ are available. Note however that these quantities are not necessary to solve the optimisation problem (5.27).

We also give the algorithm for estimating $J^{\Gamma}_{n,n+r}$. As for $J^{\alpha}_{n,n+r}$, it depends on α_n but this quantity is not necessary to solve the optimisation problem (5.28):

$$\underset{(\mathbf{x}_n,\cdots,\mathbf{x}_{n+r})\in\mathbb{X}^r}{\operatorname{argmin}} J^{\Gamma}_{n,n+r}(\mathbf{x}_n,\cdots,\mathbf{x}_{n+r}). \tag{5.28}$$

Algorithm 10 Estimation of $J^{\alpha}_{n,n+r}(\mathbf{x}_n, \cdots, \mathbf{x}_{n+r})$ (see Eq. 5.17).

Require: N_{SUR} *iid.* Poisson process associated with Y

Require: $\widehat{\alpha_n}$ an estimator of α_n

Require: $\widehat{\mathrm{E}_n[\alpha^2]}$ an estimator of $\mathrm{E}_n[\alpha^2]$ ▷ got from Algorithm 9

Get $(\mathbf{X}^i, Y_i)_{i=1}^{N_{\mathrm{SUR}}}$ the N_{SUR} *iid.* samples $\sim \mu_{n,q}^{\mathbf{X}_{\mathrm{aug}}}$ ▷ first states of the Poisson processes above q

Generate $(\mathbf{X}_2^i)_{i=1}^{N_{\mathrm{SUR}}}$ *iid.* samples $\sim \mu^X$

for $i = 1..N_{\mathrm{SUR}}$ **do** ▷ see Eq. (5.11)

get m_i and σ_i^2 the mean and variance of a Gaussian rv with distribution $U_{n+r}^{(2)} \mid \mathcal{F}_n, \mathbf{X}_1 = \mathbf{X}^i, \mathbf{X}_2 = \mathbf{X}_2^i, U_{n+r}^{(1)} = Y_i$

end for

return $s^{\alpha}_{n,n+r}(q) = \dfrac{1}{N_{\mathrm{SUR}}} \sum_{i=1}^{N_{\mathrm{SUR}}} (1 - \Phi(q|m_i, \sigma_i^2))$ ▷ $\Phi(\cdot|m, \sigma^2)$ is the *cdf* of a rv $\mathcal{N}(m, \sigma^2)$

return $\widehat{\mathrm{E}_n[\alpha^2]} - \widehat{\alpha_n} \times s^{\alpha}_{n,n+r}(q)$

Algorithm 11 Estimation of $J^{\Gamma}_{n,n+r}(\mathbf{x}_n, \cdots, \mathbf{x}_{n+r})$ (see Eq. 5.18).

Require: N_{SUR} *iid.* Poisson process associated with Y

Require: $\widehat{\alpha_n}$ an estimator of α_n

Get $(\mathbf{X}^i, Y_i)_{i=1}^{N_{\mathrm{SUR}}}$ the N_{SUR} *iid.* samples $\sim \mu_{n,q}^{\mathbf{X}_{\mathrm{aug}}}$ ▷ first states of the Poisson processes above q

for $i = 1..N_{\mathrm{SUR}}$ **do** ▷ see Eq. (5.13)

get m_i and σ_i^2 the mean and variance a of Gaussian rv with distribution $U_{n+r}^{(2)} \mid \mathcal{F}_n, \mathbf{X} = \mathbf{X}^i, U_{n+r}^{(1)} = Y_i$

end for

return $s^{\Gamma}_{n,n+r}(q) = \dfrac{1}{N_{\mathrm{SUR}}} \sum_{i=1}^{N_{\mathrm{SUR}}} \Phi(q|m_i, \sigma_i^2)$ ▷ $\Phi(\cdot|m, \sigma^2)$ is the *cdf* of a rv $\mathcal{N}(m, \sigma^2)$

return $\widehat{\alpha_n} \times s^{\Gamma}_{n,n+r}(q)$

5.2.2 Integrated SUR criteria estimation

Thanks to the Campbell's theorem, the integrated SUR criteria are estimated with a discrete almost surely finite sum of standard SUR criteria over the states of the Poisson process (see Eqs. (5.23) and (5.24)). Furthermore the renewal property of the Poisson process insures that for all $y \in \mathbb{R}$, the first event of the Poisson process after y is an *iid.* sample with distribution $\mu_n^Y(\cdot \mid Y > y)$.

Eventually, it all comes down to applying to each state of the superposed Poisson process with parameter N_{SUR} Algorithms 9, 10 or 11 with q replaced with the states of the Poisson process and the *iid.* population of size N_{SUR} rebuilt as explained above. Then all these criteria are summed.

Comparing to Multilevel Splitting based learning approaches, there is no arbitrary choice of the thresholds and the criterion favours no level. Furthermore it considers all the levels simultaneously and not sequentially, which allows for going back to more probable regions if it appears that they are not *well known*. For the sake of clarity, we detail these operations in Algorithms 12 and 13.

Algorithm 12 Estimation of $I^{\alpha}_{n,n+r}(\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r})$ (see Eq. 5.21).

Require: N_{SUR} *iid.* Poisson process associated with Y

Get $(Y_i)_{i=1}^{\bar{M}_q}$ the superposed Poisson process ▷ $Y_{\bar{M}_q}$ is the last state before q

foreach $i \in [\![1, \bar{M}_q]\!]$ **do** ▷ this can be done in parallel

Get the N_{SUR} *iid.* population with distribution $\mu_n^Y(\cdot \mid Y > Y_i)$

Get $s_n^{\alpha}(Y_i)$ with Algorithm 9

Get $s^{\alpha}_{n,n+r}(Y_i)$ with Algorithm 10

end foreach

return $\sum_{i=1}^{\bar{M}_q} \left(s_n^{\alpha}(Y_i) - s^{\alpha}_{n,n+r}(Y_i) \right)$

Algorithm 13 Estimation of $I^{\Gamma}_{n,n+r}(\mathbf{x}_{n+1}, \cdots, \mathbf{x}_{n+r})$ (see Eq. 5.22).

Require: N_{SUR} *iid.* Poisson process associated with Y

Get $(Y_i)_{i=1}^{\bar{M}_q}$ the superposed Poisson process ▷ $Y_{\bar{M}_q}$ is the last state before q

foreach $i \in [\![1, \bar{M}_q]\!]$ **do** ▷ this can be done in parallel

Get the N_{SUR} *iid.* population with distribution $\mu_n^Y(\cdot \mid Y > Y_i)$

Get $s^{\Gamma}_{n,n+r}(Y_i)$ with Algorithm 11

end foreach

return $\sum_{i=1}^{\bar{M}_q} s^{\Gamma}_{n,n+r}(Y_i)$

We insist on the fact that these algorithms may seem a little bit complex at first glance. This is because we manipulate *iid.* Poisson processes and have to specify each time the indices. In any practical implementation, the random walks would be stored as matrices and vectors and direct manipulation of all states of all Poisson processes be done with matrix computation.

5.2.3 Bayesian Moving Particles

We are now in position to derive a standard framework for what we call a Bayesian Moving Particles algorithm, in reference to [Moving Particles, Walter, 2015a] where we studied the parallelisation of the Last Particle Algorithm presented in Appendix A.

The basic idea is to iteratively generate $N_{\text{SUR}} \geq 1$ *iid.* Poisson processes, to estimate one of the above mentioned SUR criteria and to train the model with the r selected samples.

Algorithm 14 Bayesian Moving Particles

Require: g ▷ the numerical code

Require: $q \in \mathbb{R} \mid \mathrm{P}\left[g(\mathbf{X}) > q\right] > 0$ ▷ the threshold defining safety/failure domain

Require: N_{DoE} ▷ the size of the first DoE (see Section 2.2.1)

Require: N_{iter} ▷ a total number of iterations

Require: r ▷ the number of points added at each iterations

Require: $N_{\text{SUR}} \geq 1$ ▷ the number of Poisson processes for the SUR criterion

Require: δ ▷ the target c.o.v of the estimator

Get a first DoE of size N_{DoE}

$n \leftarrow N_{\text{DoE}}$

Get $\xi \mid \mathcal{F}_n$ the conditional distribution

while $n < N_{\text{iter}}$ **do**

$r \leftarrow \min(r, N_{\text{iter}} - n)$

Generate N_{SUR} *iid.* Poisson processes conditional on $\mathcal{F}_n$

Get $\widehat{\alpha_n}$

Minimise the SUR criterion ▷ see Algorithms 10, 11, 12 or 13

Evaluate the model g onto the SUR minimiser(s)

Get $\xi \mid \mathcal{F}_{n+r}$ the conditional distribution

$n \leftarrow n + r$

end while

$N_{\text{BMP}} \leftarrow \dfrac{-\log \widehat{\alpha_n}}{\delta^2}$

Generate N_{BMP} *iid.* Poisson processes conditional on $\mathcal{F}_n$

return $\widehat{\alpha_n}$ ▷ the target probability estimator

return $(Y_i)_{i=1}^{N_{\text{BMP}}}$ ▷ the N_{BMP} Poisson processes

The few parameters used are:

$N_{\mathbf{DoE}}$ the number of points for the first Design of Experiments (see Section 2.2.1). Usual values are between 5 and 10 times the dimension of the input space $\mathbb{X}$. Setting a proper value for the first DoE is a common problem to all metamodel based algorithms.

$N_{\mathbf{iter}}$ this is the number of points added to the Design of Experiments during the learning step such that the total number of calls to the model g amounts to $N_{\mathrm{DoE}} + N_{\mathrm{iter}}$. This number is machine rather than algorithm dependent. In a practical setting, one knows in advance how many evaluations of g will be possible.

r this is the number of points added to the training set at each iterations. According to Remark 2.1 this is more a machine dependent parameter, precisely the number of parallel calls to g one can afford.

$N_{\mathbf{SUR}}$ this is the number of Poisson processes used for the refinement step, *i.e.* the number of Poisson processes used for the SUR criterion estimation. This does not require any call to the computer code g.

δ this is the targeted coefficient of variation of the final estimator. Note that if the model error is still dominant (see Section 5.1.4 and numerical results of Section 5.3.2) then the relative Root Mean Squared Error (rRMSE) of the estimator will be greater.

Another tough problem not yet addressed here is the minimisation of the SUR criterion. Indeed, if the point process framework allows for an easy point-wise evaluation of the criterion, solving the minimisation problem when the dimension gets relatively high ($d \approx 10$) can become tedious. In this context, and following similar strategies from [Bect et al., 2012, Chevalier et al., 2014, Bect et al., 2016] we suggest to perform a discrete search over the states of the superposed Poisson process with parameter N_{SUR}. Hence this parameter N_{SUR} will serve to determine the precision of the SUR criterion estimation and is the size of the population onto which the discrete search is performed. The complexity of the SUR criterion estimation then typically scales like ${N_{\mathrm{SUR}}}^2 |\log p|$ for the usual SUR criterion and ${N_{\mathrm{SUR}}}^3 |\log p|^2$ for the integrated ones:

- there are on average $-N_{\mathrm{SUR}} \log p$ states of the superposed Poisson process before q;
- the estimation of a usual SUR criterion makes an average over N_{SUR} *iid.* samples;
- the estimation of an integrated SUR criterion makes on average $-N_{\mathrm{SUR}} \log p$ estimations of a usual SUR criterion.

On our numerical experiments, it appears that the computational time of the SUR criterion minimisation is driven by the computational time of the numerical approximation of the *cdf* of a standard Gaussian random variable. As a matter of fact with `R` base function `pnorm`, we have:

```
u = runif(1e7)
(t <- system.time(pnorm(u)))

##    user  system elapsed
##   0.698   0.039   0.776
```

On the other hand, with a complexity in ${N_{\text{SUR}}}^3 |\log p|^2$, the integrated SUR criterion minimisation typically requires $10^2 {N_{\text{SUR}}}^3$ calls to `pnorm`, *i.e.* 7.76×10^3 seconds, or ≈ 129 minutes with $N_{\text{SUR}} = 1000$. This computational time has to be compared with the one of the numerical code g. In the end, this latter should be several order of magnitudes greater to justify such a complexity for the SUR criterion estimation. For moderatly long computational codes such as the one we will use in Section 5.3.3 this is not the case.

To circumvent this limitation, we suggest to either fix the number of Poisson processes used for the estimation to a given constant $N_{\text{SUR, PPP}}$ or to use approximated criteria which involve less calls to `pnorm`. In the first case the complexities hence become $(-N_{\text{SUR}} \log p) \times N_{\text{SUR, PPP}}$ for the usual SUR criterion and $(-N_{\text{SUR}} \log p) \times (-N_{\text{SUR, PPP}}^2 \log p)$ for the integrated one. These values increase linearly with N_{SUR} and are well-suited for parallelisation: the wall-clock time of the algorithm can remain the same by increasing the computational resources as much as N_{SUR} the parameter driving the minimisation of the SUR criterion. The other alternative is to use a criterion which does not need to call `pnorm`. For instance, similarly to the Integrated Mean Squared Error (see Section 2.2.2) we propose to use the average of the updated kriging variances; or to make a direct average of the normalised random variables involved in Algorithms 11 and 10:

$$\begin{aligned} \tilde{s}^{\alpha}_{n,n+r}(q) &= \frac{1}{N_{\text{SUR}}} \sum_{i=1}^{N_{\text{SUR}}} \frac{m_i - q}{\sigma_i} \\ \tilde{s}^{\Gamma}_{n,n+r}(q) &= \frac{1}{N_{\text{SUR}}} \sum_{i=1}^{N_{\text{SUR}}} \frac{q - m_i}{\sigma_i} \end{aligned}.$$

Eventually the *true* quantities $J^{\Gamma}_{n,n}$ and $I^{\Gamma}_{n,n}$ can still be evaluated at each iteration and give an insight on the learning of the model.

5.3 Numerical results

Our goal is to apply the algorithms described in Section 5.2 to the problem of the estimation of the reliability of a containment vessel subject to dynamic pressure loading. We first show the behaviour of the new SUR criterion on academic test cases. Then we illustrate the trade-off between model error and statistical error presented in Section 5.1.4. Finally, we handle the real problem of estimating the probability of failure of the containment vessel.

In all the following numerical examples, we consider the corresponding problem in the standard space, *i.e.* with standard Gaussian input random variables. This lets us use the direct transition kernel for conditional sampling presented in Section A.2.1. Especially when the input parameters are not Gaussian, an iso-probabilistic transformation is done.

Furthermore, we start each algorithm with a first uniform Design of Experiments of size $5d$ in a hypersphere as in [Dubourg, 2011]. The radius of the sphere is set 6.5 because this corresponds to a probability of $\approx 10^{-6}$ for $d = 8$, the dimension of our numerical

code. Finally, the mean of the random process is enforced to be equal to the failure threshold during the refinement step: this is to ensure that *non-visited* regions of the input space will not be classified safe or failing while the random process has never *visited* them. Concerning the hyperparameters of the Gaussian process, a plug-in approach is used: they are estimated with Maximum Likelihood at each iteration.

5.3.1 SUR criteria

This section aims at illustrating the behaviour of the new SUR criterion proposed in Section 5.1.4. We focus directly on the criterion based on the upper bound of the conditional variance (see Eqs. (5.12) and (5.26)). As explained by Chevalier et al. [2014], it leads to similar performances as the criterion based on the conditional variance and is much easier to estimate. Especially with the Poisson process framework its computation reduces to an average of Gaussian one-dimensional *cdf* taken at some locations depending on the Poisson process.

Non linear oscillator This test case in dimension $d = 6$ is presented in [Echard et al., 2013, Bect et al., 2016]. The input variables are independent Gaussian random variables with parameters described in Table 5.1. Since we want to work with $\mathbf{X}$ in the standard Gaussian input space, appropriate rescaling is done before each call to the limit-state function. This limit-state function is:

$$g : \mathbf{x} \in \mathbb{R}^6 \mapsto g(\mathbf{x}) = 3x_4 - |\frac{2x_5}{x_2 + x_3} \sin\left(\frac{\omega_0 x_6}{2}\right)| \tag{5.29}$$

with $\omega_0 = \sqrt{(x_2 + x_3)/x_1}$.

Variable	x_1	x_2	x_3	x_4	x_5	x_6
Mean	1	1	0.1	0.5	0.45	1
Standard deviation	0.05	0.1	0.01	0.05	0.075	0.2

Table 5.1: Distribution parameters of the Gaussian vector $\mathbf{X} = (x_1, \cdots, x_6) \in \mathbb{R}^6$ for the non linear oscillator given in Eq. (5.29). The coordinates are independent.

The failure domain is defined as $\{\mathbf{x} \in \mathbb{R}^6 \mid g(\mathbf{x}) < 0\}$. We get a reference value for the probability of failure $\mathrm{P}[g(\mathbf{X}) < 0]$ with a run of the Subset Simulation algorithm from `R` package `mistral` with $N = 5 \times 10^3$. We found: $p \approx 1.54 \times 10^{-8}$. As a matter of comparison we also consider the non-linear oscillator in dimension $d = 8$ presented in Section A.4.1 with reference value $p \approx 3.75 \times 10^{-7}$.

Four branches serial system We also present a 2 dimensional test case for illustrative purposes. This example comes from [Waarts, 2000] and is defined in the two dimensional

standard Gaussian input space $\mathbf{X} \sim \mathcal{N}(0, \mathbf{I}_2)$ with a limit-state function g as follows:

$$g : \mathbf{x} \in \mathbb{R}^2 \mapsto \min \begin{cases} 3 + \dfrac{(x_1 - x_2)^2}{10} - \dfrac{|x_1 + x_2|}{\sqrt{2}} \\ -|x_1 - x_2| + \dfrac{6}{\sqrt{2}} \end{cases} . \tag{5.30}$$

The reference value for $\mathrm{P}\left[g(\mathbf{X}) < 0\right]$ and $\mathrm{P}\left[g(\mathbf{X}) < -4\right]$ are computed with `R` package `mistral` using a single run of the function `MP` implementing the algorithms described in Appendix A, with $N = 5 \times 10^3$. We got: $\mathrm{P}\left[g(\mathbf{X}) < 0\right] \approx 4.59 \times 10^{-3}$ and $\mathrm{P}\left[g(\mathbf{X}) < -4\right] \approx 5.98 \times 10^{-9}$.

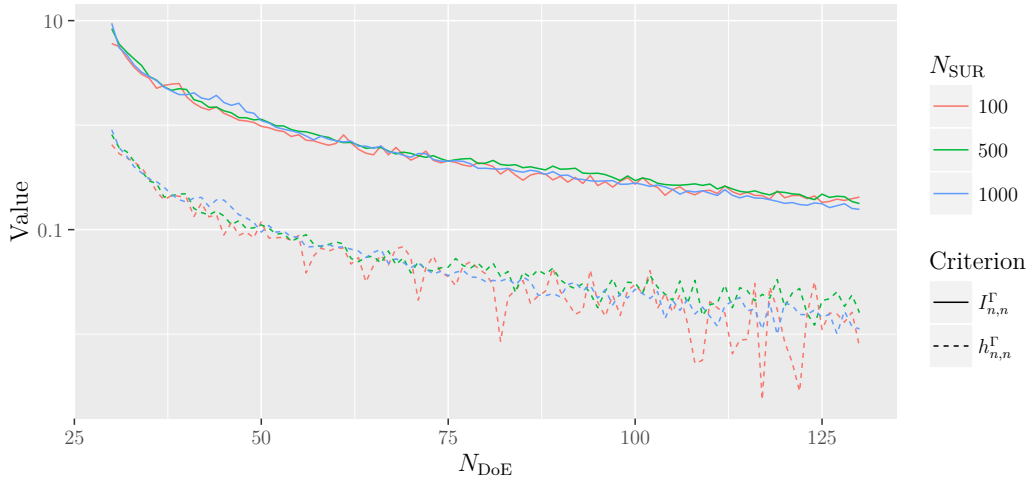
We run the Bayesian Moving Particle algorithm with $N_{\mathrm{SUR}} \in \{100, 500, 1000\}$ and $N_{\mathrm{SUR,\ PPP}} = 100$ to assess the impact of N_{SUR} on the learning curve of the model. While the integrated SUR criterion $I^{\Gamma}_{n,n+1}$ is used to select the next sample at each iteration, we present in Figure 5.1 the evolution of both $h^{\Gamma}_{n,n} = J^{\Gamma}_n / \alpha_n = \Gamma_n \alpha_n$ and $I^{\Gamma}_{n,n}$ quantifying the error of the model at a given iteration.

On the one hand the learning curves of both $I^{\Gamma}_{n,n}$ and $h^{\Gamma}_{n,n}$ seem to have the same behaviour, which does not depend on the size N_{SUR} of the number of Poisson processes used for the refinement step. While this value N_{SUR} should depend on the dimension of the problem as well as the complexity of the limit-state function, this means that it can eventually remain low to save computational time for the SUR step. On the other hand, the speeds of convergence are very different in the three test cases: the non-linear oscillator in dimension 6 is learnt much faster than the two other ones. Especially, the non-linear oscillator in dimension 8 shows a very slow convergence of the criteria and requires hundreds of calls. As a matter of fact, the MetaIS method (see Section 5.1.2 and Dubourg [2011]) requires ≈ 700 samples to produce an estimator with a estimated coefficient of variation of 5% [see Dubourg, 2011, Table 5.4].

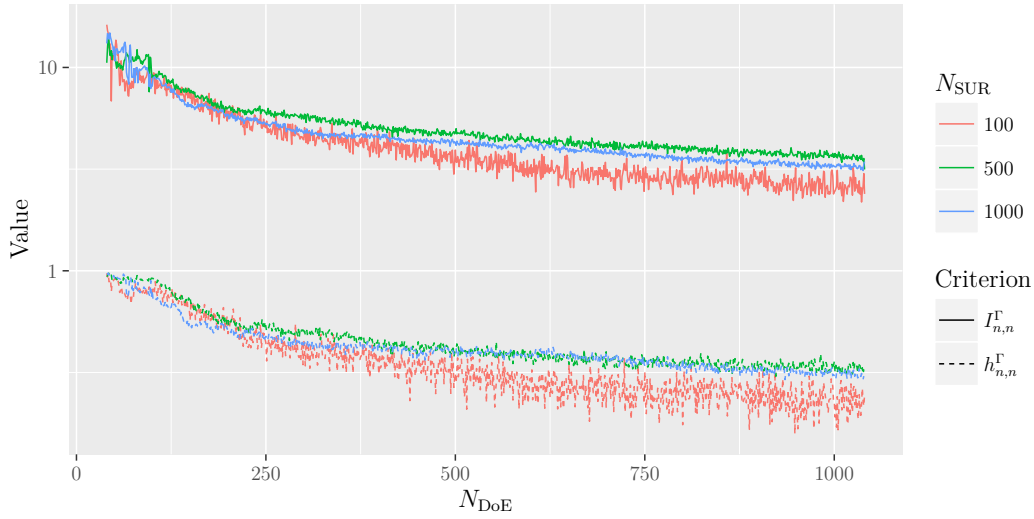
One can also notice that the estimation of the relative usual SUR criterion $J^{\Gamma}_{n,n}/\alpha_n = h^{\Gamma}_{n,n}$ is less robust than the one of $I^{\Gamma}_{n,n}$. Indeed it has lower values and with a small N_{SUR} the variance of its estimation becomes important. This is especially visible in Figure 5.1a. Furthermore in each one of the test cases, $I^{\Gamma}_{n,n}$ seems to be approximately 10 times bigger than $h^{\Gamma}_{n,n}$. Indeed, recall that $I^{\Gamma}_{n,n}$ is estimated with a sum of several $h^{\Gamma}_{n,n}$ over the states of the Poisson process. Moreover, there are on average $-\log p$ states before q. With $p \approx 10^{-5}$, we have $-\log p \approx 11.51$, which is consistent with this empirical difference in the orders of magnitude of $I^{\Gamma}_{n,n}$ and $h^{\Gamma}_{n,n}$.

As a matter of illustration, we present in Figure 5.1 the models obtained after $N_{\mathrm{iter}} = 50$ iterations of BMP with $N_{\mathrm{SUR}} = 1000$ for the 2−dimensional test case. Figures 5.1a and 5.1c are obtained using the usual SUR criterion $J^{\Gamma}_{n,n+1}$ focusing only on the boundary $\{\mathbf{x} \in \mathbb{R}^2 \mid g(\mathbf{x}) = q\}$. Figures 5.1b and 5.1d used the integrated criterion $I^{\Gamma}_{n,n+1}$.

While the usual SUR criterion performs well when the probability is moderately low (see Figure 5.1a) because it precisely samples on the boundary $\{\mathbf{x} \in \mathbb{R}^2 \mid g(\mathbf{x}) = 0\}$, it totally misses a part of the failure domain in the second case (Figure 5.1c). Here the first



(a) Non-linear oscillator in dimension 6 defined in Eq. (5.29).

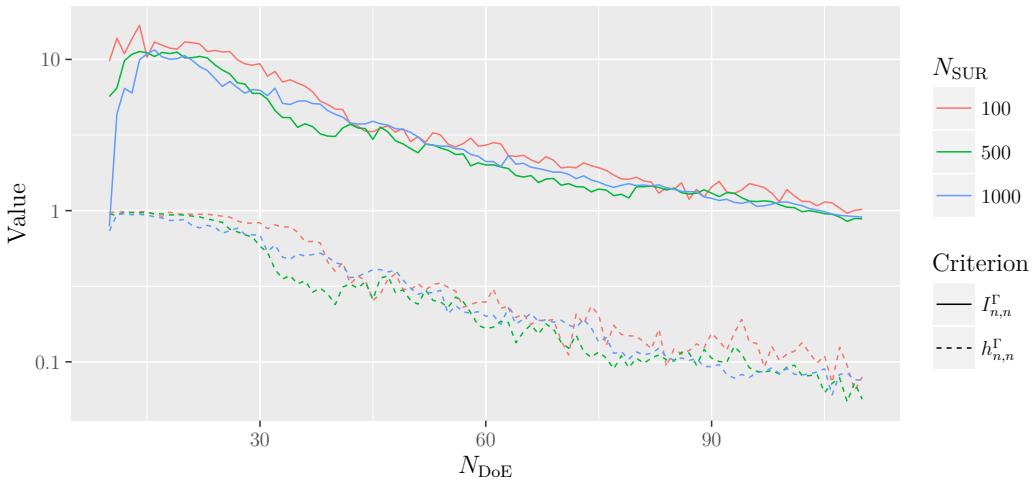


(b) Non-linear oscillator in dimension 8 presented in Section A.4.1.

Figure 5.1: Evolution of the criteria $h^{\Gamma}_{n,n} = J^{\Gamma}_{n,n}/\alpha_n$ and $I^{\Gamma}_{n,n}$ presented in Eqs. (5.20) and (5.22) against the size of the Design of Experiments (DoE) for one single run of the BMP algorithm. The DoE is built with an initial uniform sampling of size $5d$ in a hypersphere of radius 6.5. It is then iteratively updated with a SUR strategy: the criterion $I^{\Gamma}_{n,n+1}$ is minimised at each iteration in a discrete population generated by the N_{SUR} Poisson processes. For each tried sample, the value of $I^{\Gamma}_{n,n+1}$ is estimated with only $N_{\text{SUR,PPP}} = 100$ Poisson processes to reduce the computational time.

DoE does not find the failure domain and the criterion $J^{\Gamma}_{n,n+1}$ is very unlikely to recover it in totality because if focuses on the already *known* boundary.

On the other hand the integrated SUR criterion $I^{\Gamma}_{n,n+1}$ covers the whole safety domain $\{\mathbf{x} \in \mathbb{R}^2 \mid g(\mathbf{x}) > q\}$ in both cases and is able to recover the four branches of the boundary (Figure 5.1d) in the extreme case as well as on the moderate one. In this spirit we present in Figure 5.2 the empirical *cdf* estimated with both runs of BMP with the integrated SUR criterion, *i.e.* the one with $q = 0$ and the one with $q = -4$. We compare these



(c) 2 dimensional four-branches serial system defined in Eq. (5.30).

Figure 5.1 (continued): Evolution of the criteria $h^{\Gamma}_{n,n} = J^{\Gamma}_{n,n}/\alpha_n$ and $I^{\Gamma}_{n,n}$ presented in Eqs. (5.20) and (5.22) against the size of the Design of Experiments (DoE) for one single run of the BMP algorithm. The DoE is built with an initial uniform sampling of size $5d$ in a hypersphere of radius 6.5. It is then iteratively updated with a SUR strategy: the criterion $I^{\Gamma}_{n,n+1}$ is minimised at each iteration in a discrete population generated by the N_{SUR} Poisson processes. For each tried sample, the value of $I^{\Gamma}_{n,n+1}$ is estimated with only $N_{\text{SUR,PPP}} = 100$ Poisson processes to reduce the computational time.

empirical *cdf* to the one got from a single run of the Poisson process estimator on the true model g with $N = 5000$, which stands for a reference *cdf*. These empirical *cdf* are in good agreement with the reference one. Especially we found the following Kolmogorov–Smirnov statistics:

$$\sup_{y \geq 0} |F_{N_{\text{iter}}=50,q=0}(y) - F(y)| = 4.42 \times 10^{-2}$$

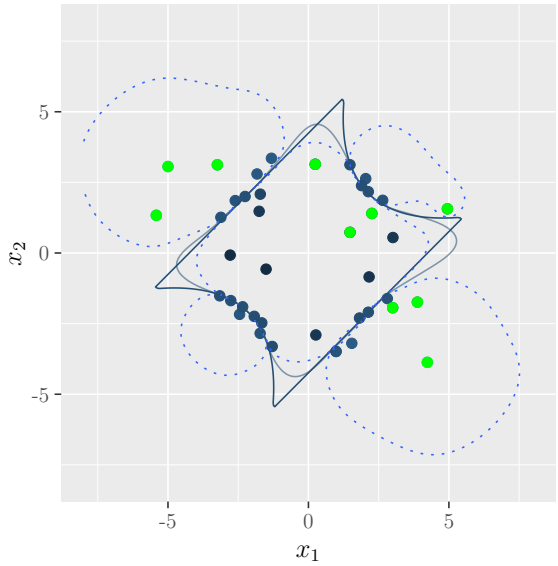
$$\sup_{y \geq -4} |F_{N_{\text{iter}}=50,q=-4}(y) - F(y)| = 0.13$$

$$\sup_{0 \geq y \geq -4} |F_{N_{\text{iter}}=50,q=-4}(y) - F(y)| = 2.83 \times 10^{-4}$$

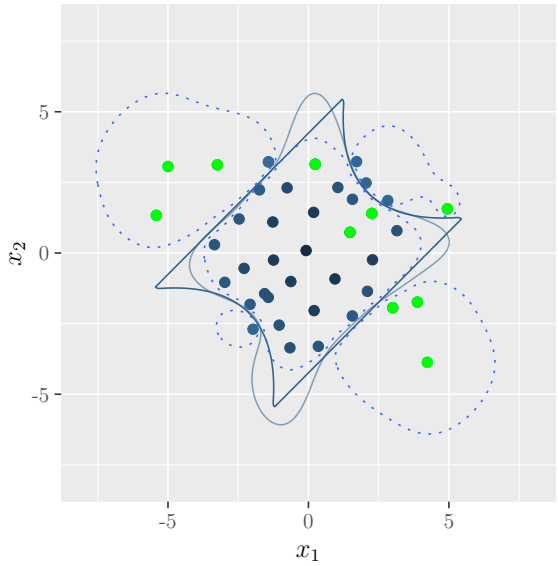
$$\sup_{-2 \geq y \geq -4} |F_{N_{\text{iter}}=50,q=-4}(y) - F(y)| = 1.6 \times 10^{-6}$$

with $F_{N_{\text{iter}},q}$ the empirical *cdf* built with the BMP algorithm run with N_{iter} iterations with target probability $\mathrm{P}\left[g(\mathbf{X}) < q\right]$. The empirical *cdf* over $[0, \infty)$ is more precise because the N_{iter} samples are used to learn the true model g onto a smaller interval, precisely $[0, \infty)$ instead of $[-4, \infty)$.

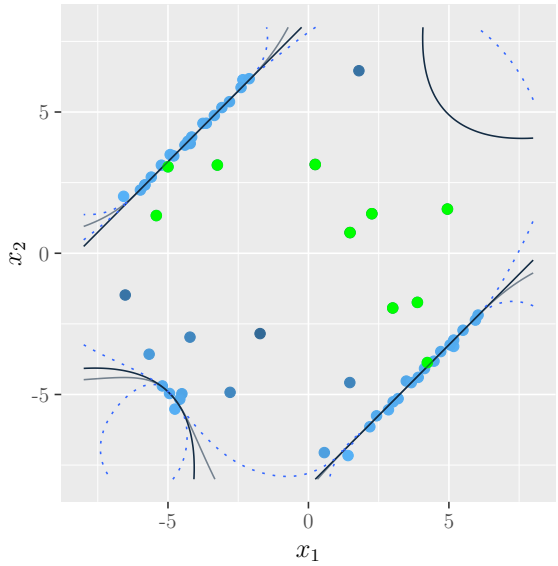
All together, these results suggest that the integrated relative SUR criterion $I^{\Gamma}_{n,n+1}$ described in Eq. (5.22) is a robust criterion to drive the *learning* of the random process. Especially it lets address extreme and non-extreme events in the same manner and is easily estimated with the Poisson process framework.



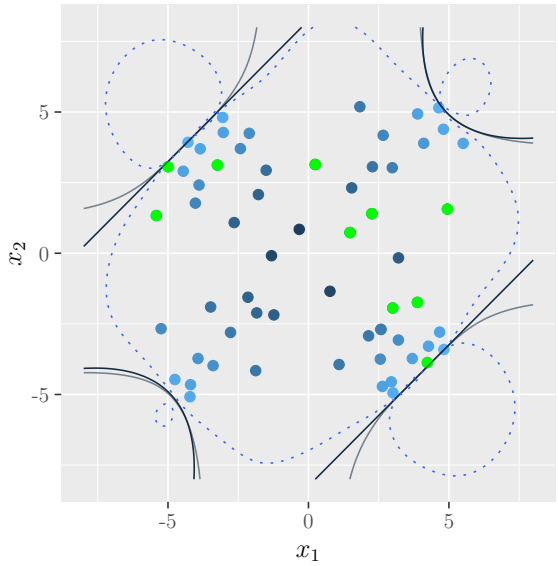
(a) Usual SUR criterion $J^{\Gamma}_{n,n+I}$



(b) Integrated SUR criterion $I^{\Gamma}_{n,n+1}$



(c) Usual SUR criterion $J^{\Gamma}_{n,n+I}$

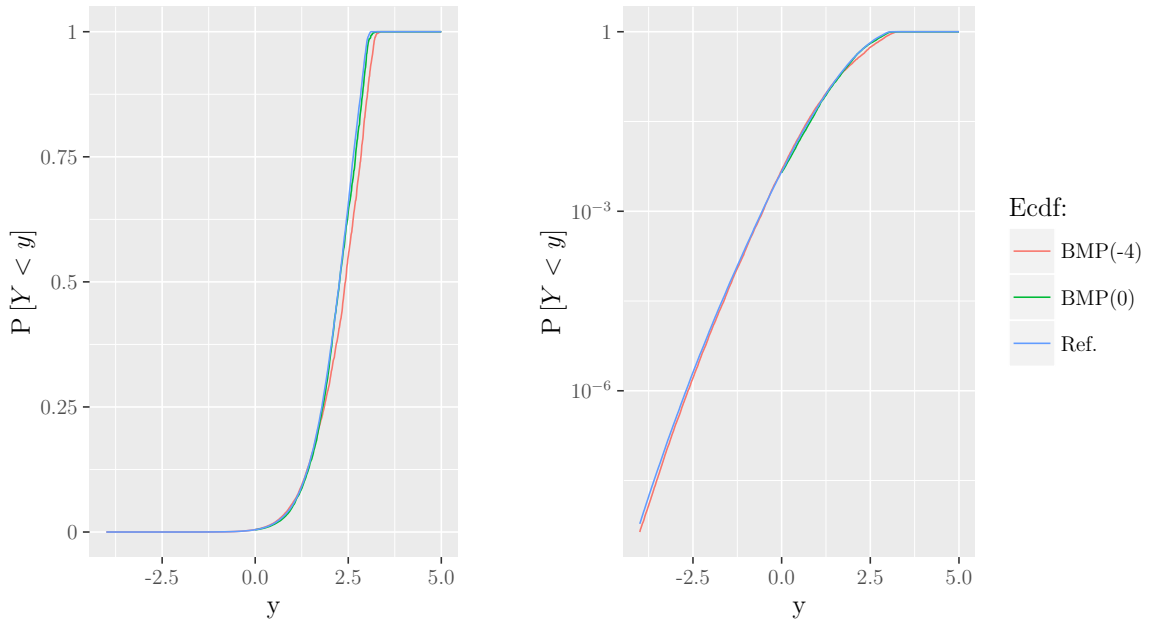


(d) Integrated SUR criterion $I^{\Gamma}_{n,n+1}$

Figure 5.1: 50 iterations of the SUR criteria $J^{\Gamma}_{n,n+1}$ and $I^{\Gamma}_{n,n+1}$ with $N_{\mathrm{SUR}} = 1000$. This academic test case is given in Eq. (5.30). The first DoE is the same in both case, represented with the green samples. For Figures 5.1a and 5.1b the target probability is moderately low: $\mathrm{P}\left[g(\mathbf{X}) < 0\right] \approx 4.59 \times 10^{-3}$. For Figures 5.1c and 5.1d it is extreme: $p = \mathrm{P}\left[g(\mathbf{X}) < -4\right] \approx 5.98 \times 10^{-9}$.

5.3.2 Statistical error

We now focus on the probability estimation problem: instead of looking at the values of the SUR criteria, we look at the evolution of the estimated probability $(\widehat{\alpha_n})_{n=N_{\mathrm{DoE}}}^{N_{\mathrm{DoE}}+N_{\mathrm{iter}}}$. We run 20 simulations of the BMP algorithm on the non-linear oscillator test case in dimension $d = 6$ (see Eq. 5.29). We try it with $N_{\mathrm{SUR}} \in \{100, 1500, 3100, 6300, 9500\}$ and $N_{\mathrm{iter}} = 120$, so that the total number of calls to the limit-state function amounts to $N_{\mathrm{DoE}} + N_{\mathrm{iter}} = 30 + 120 = 150$.



(a) Normal scale

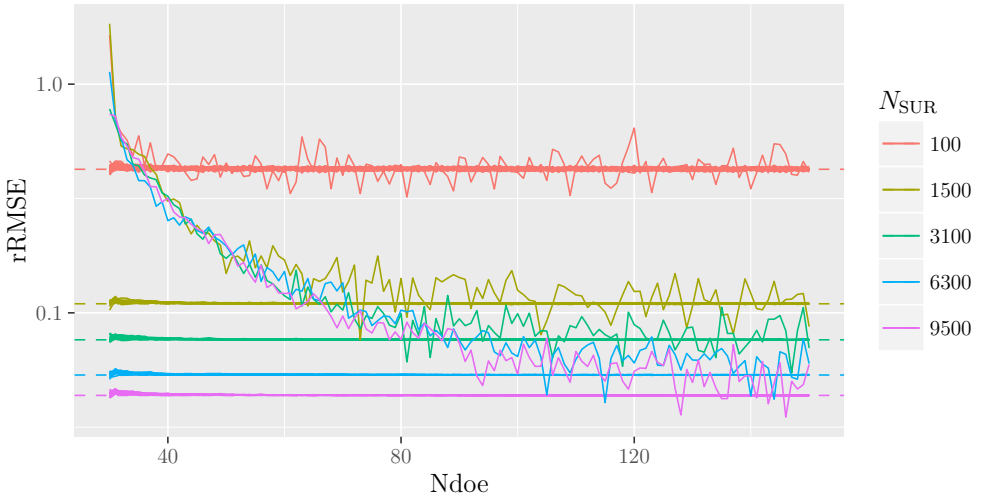
(b) Log scale

Figure 5.2: Empirical *cdf* of $Y = g(\mathbf{X})$ with $\mathbf{X} \sim \mathcal{N}(0, \mathbf{I}_2)$ and g as in Eq. (5.30). Both curves are built with a single run of BMP with $q \in \{0, -4\}$, $N_{\text{SUR}} = 1000$ and $N_{\text{iter}} = 50$, *i.e.* a total number of calls to g equals to $N_{\text{iter}} + 5 \times 2 = 60$. The reference *cdf* is got from a single run of the Poisson process estimator on the true model g with $N = 5000$.

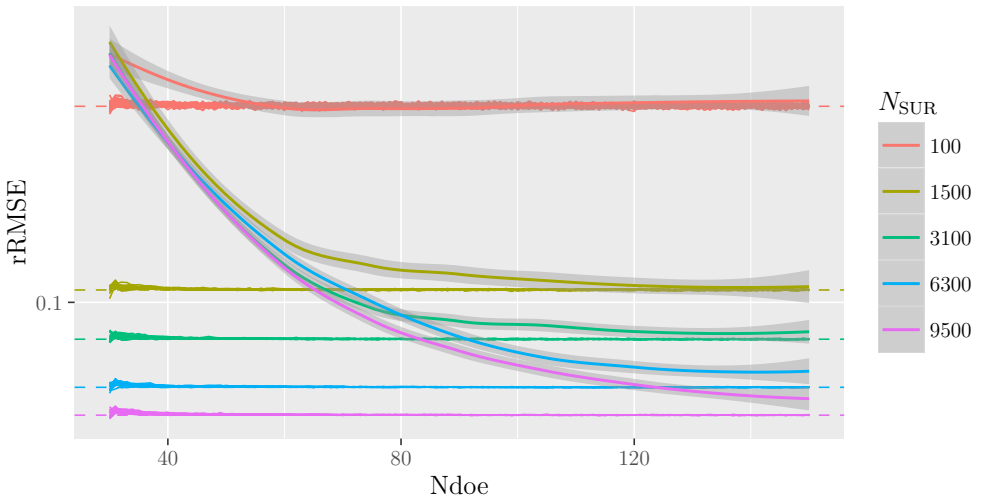
Figure 5.3 plots the relative Root Mean Square Error (rRMSE) of the estimator at each iteration. It shows that the model error tends to *vanish* behind the statistical error: when the DoE is small the model error is predominant. From one moment on, the statistical error due to the estimation of the conditional expectation $\mathrm{E}_n[\alpha] = \mathrm{P}_n[Y < 0]$ becomes the more important. Eventually with a constant N_{SUR} this limitation cannot be overpassed: even though the model is perfectly known (as in the deterministic case), the statistical error remains (see Eq. 5.14).

We compare these results to the one found in [Bect et al., 2016] reported in Table 5.2 with the BSS algorithm. Indeed BSS applies the Bayesian framework presented in Section 2.2.3 and used here to the Subset Simulation algorithm. The relation between the Poisson process framework and the Subset Simulation methods lets assume that the results should be similar.

Note that the BSS is implemented with an automatic stopping criterion and the comparison thus cannot be done with constant number of calls to g. However, we can still look at the precision reached when it stopped and compare it to the one we had after the same number of iterations. Table 5.2 shows that even when looking at the worst case for BMP, *i.e.* comparing the results of BSS to the ones with N_{SUR} smaller for BMP, the rRMSE are significantly lower for BMP. There is indeed an average increase of 40%, which is in good agreement with the theoretical increase found between Subset Simulation with $p_0 = 0.1$ and the Poisson process estimator (see Sections 1.3.2 and A.3.1).



(a) Raw results.



(b) Smoothed results using local polynomial regression fitting (loess) [Chambers and Hastie, 1992].

Figure 5.3: relative Root Mean Squared Error (rRMSE) in the estimation of $\mathrm{P}\left[g(\mathbf{X}) < 0\right]$ with g defined in Eq. (5.29) and $\mathbf{X}$ a random vector with independent Normal coordinates with parameters given in Table 5.1. The dotted horizontal lines stand for the *true* achievable coefficients of variation, *i.e.* $\sqrt{-\log p / N_{\mathrm{SUR}}}$. The plain lines *oscillating* around them are the estimated coefficient of variations of the probability at each iteration, *i.e.* $\sqrt{-\log \widehat{\alpha_n} / N_{\mathrm{SUR}}}$.

Furthermore Figure 5.3 suggests that the estimated coefficient of variation at each iteration does not depart much from its theoretical value with a perfect knowledge of the model. This means that instead of choosing an arbitrary value for N_{BMP} the number of Poisson processes used for the final estimator, one can instead select a target value of the squared coefficient of variation $\delta^2_{\mathrm{target}}$. Then the final N_{BMP} for probability estimation will be given by: $N_{\mathrm{BMP}} = -\log \widehat{\alpha_{N_{\mathrm{DoE}}+N_{\mathrm{iter}}}} / \delta^2_{\mathrm{target}}$.

N (N_{SUR})	# of calls	rRMSE (BSS)	rRMSE (BMP)
100	48		0.49
500	48	0.55	
1000	51	0.35	
1500	$51 \rightarrow 55$		$0.17 \rightarrow 0.16$
2000	55	0.23	
3100	$55 \rightarrow 63$		$0.14 \rightarrow 0.11$
4000	63	0.17	
6300	$63 \rightarrow 75$		$0.12 \rightarrow 0.09$
8000	75	0.13	
9500	75		0.08

Table 5.2: Relative Root Mean Squared Error (rRMSE) in the estimation of P $[g(\mathbf{X}) < 0]$ with g given in Eq. (5.29) and $\mathbf{X}$ in Table 5.1. Results for BSS are got from [Bect et al., 2016]. In this latter case, the algorithm stops randomly according to a stopping criterion. Thus the presented number of calls is an average result. Results for BMP are the same as the ones in Figure 5.3.

5.3.3 Industrial problem

We now address the issue of estimating the reliability of a spherical containment vessel subject to internal blast. This vessel is a spherical tank as presented in Figure 5.4. Further technical details onto this problem can be found in [Defaux and Evrard, 2014].

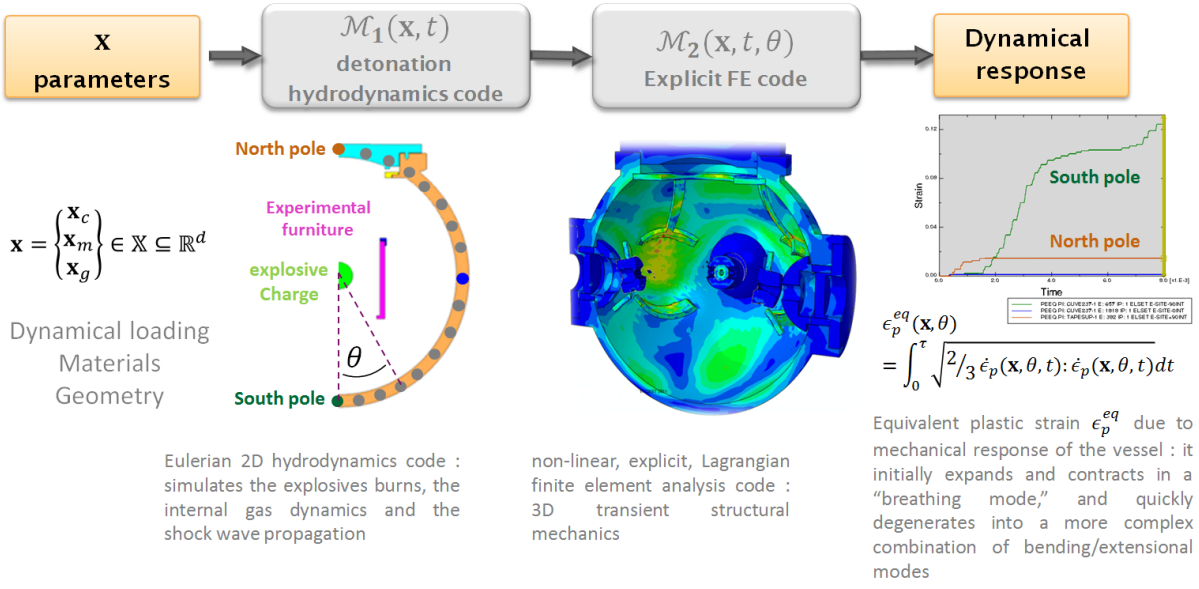


Figure 5.4: Workflow of the simulation of the response of the spherical tank [Defaux and Evrard, 2014].

Presentation of the problem The mechanical response (displacement, stress, strain) of the tank is modelled with a coupling of two independent numerical codes: first a hydrodynamic code $\mathcal{M}_1$ lets simulate the explosion of a bursting charge placed in the center of the tank. This model uses a 2-dimensional Eulerian scheme. Then it outputs the time-dependent distribution of the pressure at several spots on the tank localised by their angle θ with respect to the vertical axis and for a given input vector $\mathbf{x}$ that characterizes the dynamical loading, some material properties and the geometry (see Figure 5.4). These distributions are then used as input of a structure code $\mathcal{M}_2$. This second code simulates the vibrations of the tank under a dynamic excitation and evaluates the displacement, strain and stress tensors as functions of the time at each one of the above mentioned locations. Figure 5.5 shows the dynamic load simulated by the first model $\mathcal{M}_1$ while 5.6 represents the dynamical response $t \mapsto \mathcal{M}_2 \circ \mathcal{M}_1(\mathbf{x}, t, \theta)$ for given $\mathbf{x}$ and $\theta \in [0, \pi]$.

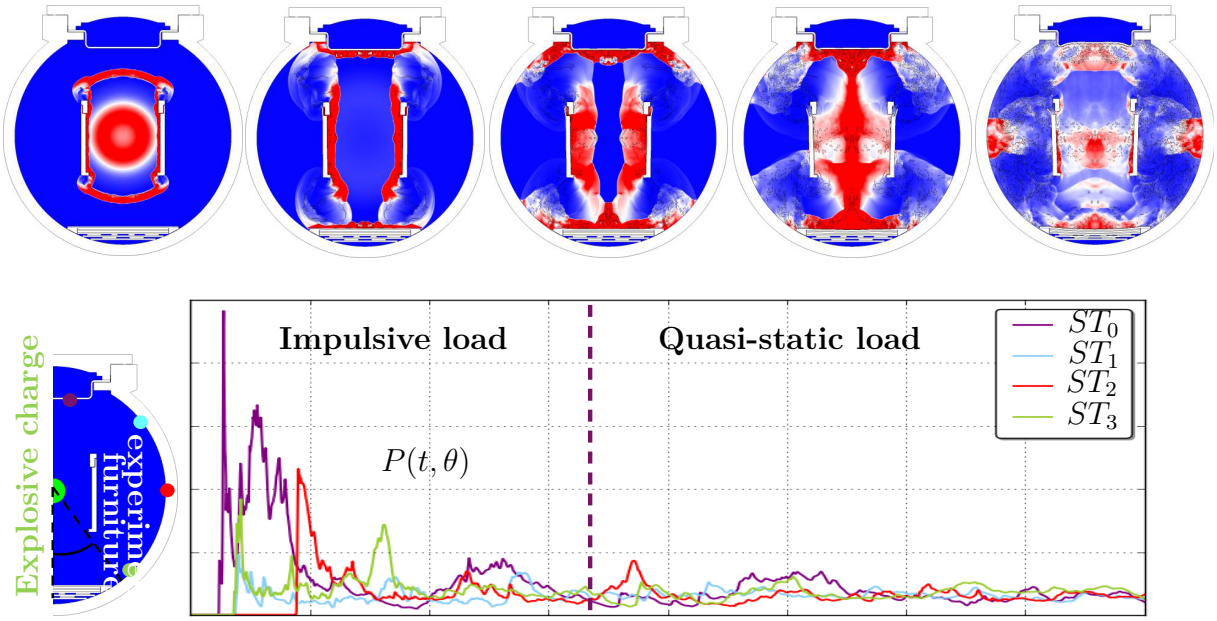


Figure 5.5: Dynamic pressure at several locations of the tank got from the model $\mathcal{M}_1$.

Finally we look at the cumulated equivalent plastic strain ϵ_p^{eq} over a given time range $[0, \tau]$ using perfect elastoplastic strain stress curve to model the plastic behaviour of the involved materials:

$$\epsilon_p^{\text{eq}}(\mathbf{x}, \theta) = \int_0^\tau \sqrt{\frac{2}{3}\, \dot{\epsilon}\,(\mathbf{x}, \theta, t) : \dot{\epsilon}\,(\mathbf{x}, \theta, t)}\mathrm{d}t \tag{5.31}$$

with $\dot{\epsilon}\,(\mathbf{x}, \theta, t)$ the derivative of the strain tensor against time and ":" standing for the double dot product for tensors.

By introducing uncertainties in the input parameters $\mathbf{x}$, the variability of $\epsilon_p^{\text{eq}}(\mathbf{x}, \theta)$ can be quantified in terms of probability. The stochastic model used in the following of this section is given in Table 5.3.

Especially here, we focus on the plasticity level at angle $\theta = 0$ and the question is to estimate the probability that it exceeds $2.5\%, 5\%$ or 10%. $Y = \epsilon_p^{\text{eq}}(\mathbf{X}, 0)$ is therefore the

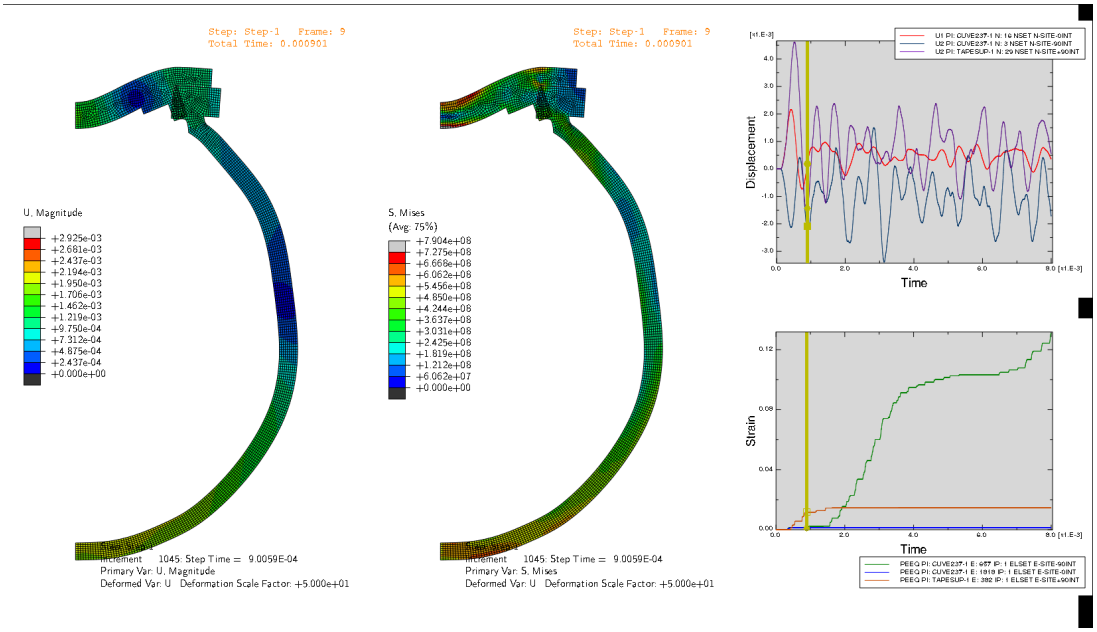


Figure 5.6: Illustration of the dynamic response of the tank under internal pressure. Animated figure online.

Variable	Phys. Meaning	Distribution	P_1	P_2
x_1	Internal radius of the tank (m)	Normal	0.720	0.005
x_2	Thickness (m)	Log normal	0.073	0.0015
x_3	Scaling factor on pressure	Weibull	24.95	1.022
x_4	Scaling factor on time	Weibull	24.95	1.022
x_5	Young modulus of the tank (Pa)	Log normal	2.1×10^{11}	2.1×10^{10}
x_6	Elastic limit of the tank (Pa)	Normal	7×10^{8}	3×10^{7}
x_7	Young modulus of the tap (Pa)	Log normal	2.1×10^{11}	2.1×10^{10}
x_8	Elastic limit of the tap (Pa)	Normal	8.60×10^{8}	3×10^{7}

Table 5.3: Stochastic model of the tank. For Normal and Log normal distributions, P_1 is the mean and P_2 the standard deviation. For the Weibull distribution, P_1 is the shape parameter and P_2 the scale parameter. The scaling factors x_3 and x_4 define the pressure time history acting on the inner radius of the vessel (see the curve in Figure 5.5) . The coordinates are independent.

real-valued random variable of interest and one indeed wants to estimate $\mathrm{P}\left[Y > 0.025\right]$, $\mathrm{P}\left[Y > 0.05\right]$ and $\mathrm{P}\left[Y > 0.1\right]$. The point process framework developed in Chapter 3 is especially interesting in this configuration because it outputs an estimation of the *cdf* of Y over $(-\infty, y]$ at the same cost as a punctual estimation of $\mathrm{P}\left[Y > y\right]$. Hence running a point process estimator until threshold $q = 0.1$ will directly output an estimator of the three quantities, each one with known distribution.

Experimental setting The number of iterations is determined by the machine used. The algorithm is run on a cluster with up to 2048 MPI threads and a time limit of 24 hours per job. On the one hand numerical benchmarks give that the minimal wall-clock time for a call to the chained simulator is got when it is spread in parallel over 4 threads. With this setting, it takes up to 5 minutes and requires 2 cores of 4Gb each per thread for memory reasons. We noted however that this gain is not significant compared to the computational time of the code using only 1 thread. On the other hand we can request more threads for the computation of the Poisson process and the SUR criterion at each iteration since these algorithms are massively parallel. In order to reduce to a minimum this *side* computational load and according to the dimension of the problem ($d = 8$) we generate the Poisson process by batches of size 20 (see the numerical study of Section A.4.2).

Finally, since the 40 first samples of the DoE can be computed totally in parallel and the gain in the computational time of the code with 4 threads is not very important, we will use 40 MPI threads for our algorithm. This implies $N_{\text{SUR}} = 40 \times 20 = 800$. We also set $N_{\text{SUR, PPP}} = 100$ as for the theoretical test cases. All together, this gives a complexity of $N_{\text{SUR}} \times (N_{\text{SUR, PPP}} \log p)^2$ for the estimation of the SUR criteiron. In addition to the random time of the code, we then expect a duration of ≈ 5 to 10 minutes per iteration and a total number of iterations ≈ 250. In any case, we set $N_{\text{iter}} = \infty$ and wait for the job to be killed by the resource management utility of Airain because of the time limit.

The BMP algorithm is implemented in `R` using the parallel processing facility through the packages `doMPI`, `doParallel` and `foreach` [Weston, 2015a,b,c].

Results Recall that the problem is to estimate in the same run the three quantities $\mathrm{P}[Y > 0.025]$, $\mathrm{P}[Y > 0.05]$ and $\mathrm{P}[Y > 0.1]$ we first present in Figure 5.7 the evolution of the criteria $I^{\Gamma}_{n,n}$ and $h^{\Gamma}_{n,n}$ representing the model error. The convergence of both criteria is rather slow comparing to the one of the test cases (5.29) and (5.30) but of the same order of magnitude as the one of the non-linear oscillator in dimension $d = 8$ (see Figure 5.1b). We also show the evolution of the estimated standard deviation of $\widehat{\alpha_n}$. As for the previous test cases, it quickly stabilises around what should be its theoretical value (unknown for this real computer experiment).

Figure 5.8 gives the boxplots of the wall-clock time of the main steps of the BMP algorithm. We can see that most of the time is spent in the evaluation of the limit-state function, which is the goal of this kind of algorithms. Especially the wall-clock time required by the generation of the Poisson processes is negligible, which means that this step can easily be added to any other metamodel based approach at almost no cost to estimate efficiently $I^{\Gamma}_{n,n}$ or $h^{\Gamma}_{n,n}$ for instance. As a matter of fact, we present in Table 5.4 the mean wall-clock time of these steps. Also we were finally able to make 219 iterations.

Finally we run a BMP algorithm without sampling but gathering all the different numerical experiments done so far. Indeed in addition to the $N_{\text{iter}} = 219$ iterations done

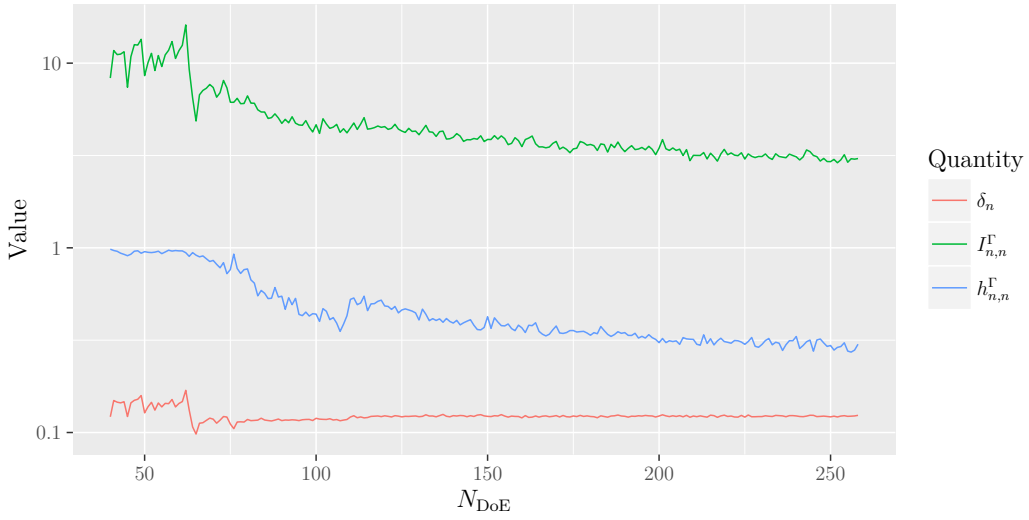


Figure 5.7: Evolution of the SUR criteria $I^{\Gamma}_{n,n}$ and $h^{\Gamma}_{n,n} = J^{\Gamma}_{n,n}/\alpha_n$ against the number of points added to the Design of Experiments by minimising at each iteration $I^{\Gamma}_{n,n+1}$. δ_n is the estimated coefficient of variation such that $\delta_n^2 = -\log \widehat{\alpha_n}/N_{\text{SUR}}$. The first 40 samples of the DoE are drawn in a uniform hypersphere of radius 6.5. $N_{\text{SUR}} = 800$ and $N_{\text{SUR, PPP}} = 100$.

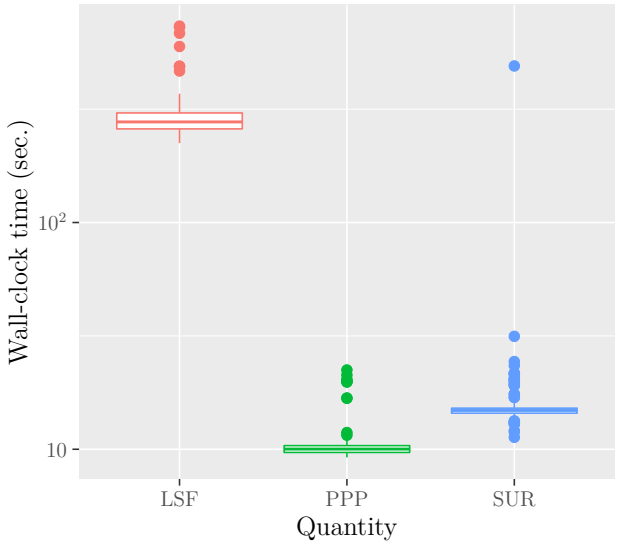


Figure 5.8: Wall-clock time of the three main steps of the BMP algorithm run with 40 MPI threads: generation of the Poisson process (PPP), evaluation of the SUR criterion (SUR) and call to the limit-state function (LSF).

we had available some previous computations and so a total database of 1375 data. In other words we train the Gaussian process with the 1375 data on the tank and then only run a Poisson process associated with Y given these data. While the first experiment served the estimation of the learning curve of the tank, this run can give a sort of reference value because it gathers much more information than only running the algorithm from scratch.

Table 5.5 gives the estimated values of the three sought probabilities as well as their coefficients of variation. Note that this latter value could be as small as desired since it

Operation	Mean wall-clock time (s)
Generation of the Poisson process	10.37
Discrete minimisation of the SUR criterion	17.36
Evaluation of the limit-state function	287.867

Table 5.4: Mean wall-clock time of the three main steps of the BMP algorithm during the estimation of the reliability of a spherical tank modelled with two independent numerical codes (see Figure 5.4). The algorithm used 40 MPI threads.

only involves the simulation of Poisson processes associated with the *augmented* random variable $Y = \xi(\mathbf{X})$. On the other hand we have no possible estimation of the relative Root Mean Squared Error of the estimators.

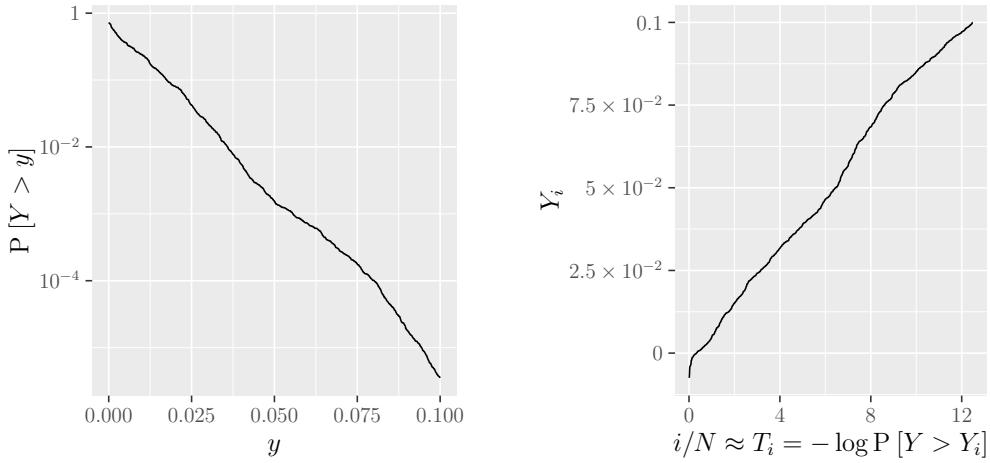
Probability	Estimation	Coef. of variation
$\mathrm{P}\left[Y > 0.025\right]$	4.22×10^{-2}	0.18
$\mathrm{P}\left[Y > 0.05\right]$	1.5×10^{-3}	0.26
$\mathrm{P}\left[Y > 0.1\right]$	3.53×10^{-6}	0.35

Table 5.5: Estimated sought probabilities with $N_{\mathrm{BMP}} = 10^2$ Poisson processes with a metamodel trained with $N_{\mathrm{DoE}} = 1375$ data.

We also present in Figure 5.9a the empirical *cdf* got from this run. In log-scale the complementary *cdf* seems to be linear. Indeed, one can also look at the values of the states of the superposed Poisson process against the number of iterations. This is shown in Figure 5.9b. Eventually, one could conjecture that Y follows an exponential distribution and estimate its parameter with a linear regression:

```
T = seq(Y)/N; lin.mod <- lm(Y~T); summary(lin.mod)

##
## Call:
## lm(formula = Y ~ T)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -4.77e-03 -1.29e-03  7.30e-05  1.47e-03  4.05e-03
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.79e-03   1.16e-04     -24   <2e-16
## T            8.61e-03   1.61e-05     534   <2e-16
##
```



(a) Empirical complementary *cdf*

(b) States $(Y_i)_i$ of the superposed Poisson process

Figure 5.9: Superposed Poisson process with parameter $N_{\mathrm{BMP}} = 10^2$ until threshold $q = 0.1$ associated with the cumulated equivalent plastic strain of the containment vessel. The Gaussian process is trained with a legacy database of $N_{\mathrm{DoE}} = 1375$ samples.

```
## (Intercept) ***
## T           ***
## ---
## Signif. codes:
## 0 '***' 1e-03 '**' 1e-02 '*' 5e-02 '.' 0.1 ' ' 1
##
## Residual standard error: 2.1e-03 on 1247 degrees of freedom
## Multiple R-squared:  0.996,Adjusted R-squared:  0.996
## F-statistic: 2.85e+05 on 1 and 1247 DF,  p-value: <2e-16
```

In a first approximation it appears that the random variable could be approximated with an Exponential random variable with parameter $1/(8.61 \times 10^{-3}) = 116.188$.

5.4 Conclusion

In this chapter we applied the point process framework to the case where the computer code g is also modelled with a random process ξ. We show that compared with the usual setting where g is deterministic, it only *adds a dimension* to the problem. Eventually all the results derived in this deterministic setting apply directly on the *augmented* random variable $Y = \xi(\mathbf{X})$.

The point process framework hence appeared as an efficient tool to estimate $\mathrm{P}\,[Y > q]$ and could be directly plugged into already existing algorithms such as the MetaIS method or the AKMCS one. Especially when these methods have shown good results for the

learing step, it can be used in the end to produce advanced statistics, an estimation of the *cdf* or a quantile for instance.

Considering SUR strategies, we defined a new SUR criterion, it is the integration of the relative usual criterion for contour estimation against the distribution of the Poisson process. Formally, this criterion appears as the average with respect to the distribution of Y of the conditional variance of α as a function of q the sought level. This new criterion seems more robust and especially allows for targeting extreme probability directly without introducing an arbitrary sequence of thresholds as in multilevel splitting based algorithms.

Eventually on numerical examples it is visible that its convergence can be relatively slow. However the generation of the Poisson process associated with Y not only serves the estimation of the SUR criterion but can also be used to estimate the two criteria $h^{\Gamma}_{n,n}$ (or $J^{\Gamma}_{n,n}$) and $I^{\Gamma}_{n,n}$ giving indications on the current *accuracy* of the random process. In other words even though one chooses another criterion for driving the *learning* of the metamodel, $h^{\Gamma}_{n,n}$ and I^{Γ}_{n} appear as interesting quantities to quantify the precision of the model.

At the cost of the generation of independent random variables with distribution μ^X, the Poisson process can also be used to estimate the conditional variance of α. However if the sought probability is extreme this estimation may have a huge variance and further simulations should be done. For instance it is possible to start over a new Poisson process estimator to estimate the remaining conditional probability (see Eq. 5.16). This is let for further developments. Also not addressed here is the definition of a stopping criterion for the learning step. Indeed while we focused on the learning and considered that the number of steps was given, in some setting one can simply look for a estimator with a given precision. To this end the learning step should be stopped when the model error becomes non significant compared to the targeted statistical error.

Conclusion and perspectives

This thesis addresses the issue of rare event simulation, more specifically the problem of estimating extreme probability and quantile of the form $p = \mathrm{P}[Y > q]$ with Y a real-valued random variable $Y = g(\mathbf{X})$ (the output of a computer code for instance) with $\mathbf{X}$ a random finite- or infinite dimensional random vector $\mathbf{X} \in \mathbb{X}$ with known distribution and g a deterministic *black-box* function. Also considered here are the problems of estimating the quantile corresponding to a given probability p and the (conditional) mean $\mathrm{E}[Y \mid Y > q]$. These problems arise in many branches of the industry like in civil engineering, telecommunication or finance.

The common setting of these applications is the use of complex numerical codes to define the problem. Indeed the sought quantity corresponds to the determination of the *risk* of a failure mode of the system modelled by the code: either the probability of failure for a given security threshold q or the threshold q for a given acceptable risk p. These codes are very time consuming. Yet they are required to simulate the random variable of interest Y. Hence the considered statistics used for probability estimation should be very *greedy* in the sense that they should minimise the variance of the estimator with respect to N a total number of generated samples, *i.e.* a total number of calls to g.

In the rare event setting and when nothing is assumed about the behaviour of g, a usual tool is the so-called Multilevel Splitting method. On the other hand when this method still requires too much computational time, one often relies on surrogate models of g. These models are cheap to evaluate but have to be *trained* to approximate *well* (in a sense depending on the quantity of interest) the true model g. A specific class of algorithms is the one using Kriging, and especially Gaussian Process regression. Recently, both methods (advanced statistics and metamodeling) started being combined into the same algorithm, producing promising practical results.

In this thesis we strived to develop a common theoretical framework to answer these questions while embracing the more advanced developments on specific estimators. Instead of considering the properties of one given algorithm, we started by defining a point process associated with any real-valued random variable. Then we developed formal estimators for probability, quantile and moment using independent and identically distributed (*iid.*) replicas of such a process. We showed that these estimators were optimal (in terms of variance against number of generated samples). We insisted on the fact that these estimators appeared as a true counterpart of the usual Monte Carlo ones, but based on *iid.* replicas of the point process defined in this thesis. In this spirit the statistical properties of the probability and quantile estimators (bias, confidence intervals, limit distribution) are truly similar to the ones of the crude Monte Carlo estimators; they can be obtained

directly by substituting $1/p$ for $\log 1/p$ in all the considered formulas. This is especially interesting when $p \to 0$. Indeed, we showed that the point process framework not only lets estimate probabilities or quantiles but gives an estimation of the whole cumulative distribution function (*cdf*) of Y.

Since these estimators are based on *iid.* replicas of a point process, they are intrinsically parallel in the sense that they allow for an almost direct use of parallel processing to be generated faster. Indeed manufacturers stopped building more powerful CPU about 10 years ago. Instead, they have been focusing on creating multi-core processors. This paradigm shift is a main concern for defining new practical algorithms. For instance theoretical optimal estimators which do not allow for the use of parallel computers may become useless compared to more naive versions using parallel processing. In this thesis we argued that a *parallel estimator* is an estimator which allows for the use of parallel computing but has the same statistical properties with or without parallel computing. In this spirit averaging several sequential estimators allows for using parallel facilities but is not what we can call a *parallel estimator*, although it may turn out to be a good strategy provided the estimators are unbiased.

We then developed the links between the theoretical parallel estimators and the ones of well-known efficient algorithms. Especially we showed that the Last Particle Algorithm is a particular non parallel implementation of our probability estimator and that our moment estimator is an optimised (minimal variance and unbiased) version of the nested sampling method. For this latter we were also able to address the issue of the termination rule while keeping a parallel unbiased estimator. Concerning the quantile estimator, it is as far as we know the only alternative to the crude Monte Carlo estimator for parallel quantile estimation, thus allowing for a great saving in computational time for extreme quantile estimation.

When the computational time of the computer is still too important for using the above mentioned statistics, a common practice is to model it with a random process with known distribution. We showed that the point process framework is well suited to handle such a case. Especially probability, quantile and moment of the corresponding *augmented* random variable (the random variable embedding the uncertainty on the computer code in addition to the uncertainty on the input parameters) can be estimated exactly the same way. Furthermore we defined new criteria for quantifying the quality of the metamodel over an interval. While metamodel based algorithms for pointwise estimation of a probability were already available, it is as far as we know the first framework for estimating quantile and the *cdf* of the real-valued random variable integrating the uncertainty on the computer code itself.

We argued that the estimators are well-defined, parallel and optimal in the sense above mentioned. On the other hand, an optimal implementation (*i.e.* an algorithm giving as output the sought quantity in the minimal wall-clock time) is a somehow different question. While it is possible to show that a given estimator is optimal regarding its

variance against the number of generated samples, a practical implementation involves a lot a *side parameters and operations* which can indeed dramatically improve or reduce the performance of an estimator. Without underrating this question, we think that a lot of machine-dependent parameters have to be taken into account to address this issue. Depending on the machine used (CPU, GPU), the parallel framework (fork, socket cluster, MPI), the linear algebra library, the programming language... some algorithms may exhibit very strong or weak performances. As an illustration we found in our experiments significant variations (up to 10 times) in the computational time of call to the numerical approximation of the *cdf* of a standard Gaussian random variable depending on the machine used (different laptops, university cluster, Airain supercomputer).

In this thesis we strived to cover the whole spectrum from the full theoretical definition of the estimator to the effective implementation on supercomputers and development of *industrial* `R` code. However it appears obvious that the finally adopted implementations are very dependent to our setting. In this context it may be relevant to integrate in the theoretical analysis of the estimator parameters representing the performances of the used machine for the corresponding operations. This would let define a more practical oriented notion of optimality.

In this context a setting not studied in this thesis is the use of not only one but several random processes instead of the true heavy code g. This so-called multi-fidelity framework allows for building an approximation of the computer code with several cheap approximations. Each level of code has its own computational complexity and a trade-off has to be found between *learning* of the model and computational time. For instance results of Le Gratiet [2013] on sequential learning for multifidelity Gaussian process regression could be applied to the rare event setting.

Finally we did not derive any stopping criterion for the enrichment step of an algorithm combining Poisson processes and metamodeling. While this question can be of great practical use, we emphasised instead the interest of using Poisson processes in addition to metamodel based algorithms because they let estimate easily quantities already used in other methods, especially the conditional expectation and SUR criteria. The question of defining an optimal *ready-to-use* method with tuned parameters is let for further research.

Part III

Appendix

Appendix A

Parallel computation of the estimators

A.1 Introduction

In Chapter 3 we have defined the point process framework for any real-valued random variable Y. This framework lets obtain estimators of a probability $p = \mathrm{P}[Y > q]$ for a given $q \in \mathbb{R}$, (see Sections 3.3 and 3.5.2), a quantile q such that $\mathrm{P}[Y > q] = p$ for a given $p \in (0, 1)$ (Section 3.4) and of the expectation $\mathrm{E}[Y]$ (Chapter 4).

Especially some of these estimators (the probability and the moment estimators) had already been proposed as estimators given by specific sequential algorithms, the Last Particle Algorithm [Guyader et al., 2011] and the nested sampling method [Skilling, 2006] respectively. Instead we insisted on the fact that these estimators were, indeed, not outputs of specific algorithms but rather that the algorithms were particular implementations of general well-defined estimators based on a point process associated to Y. We have argued that this framework allows for parallel implementation of the above mentioned estimators because they appear to be defined with *iid.* replicas of the increasing random walk (see Section 3.2).

However, generating an increasing random walk requires to be able to perform conditional simulations $\mu^Y(\cdot \mid Y > y)$ for any $y \in \mathbb{R}$. While some specific tools may be used depending on the setting of the problem [see for instance Coupling from the past, Propp and Wilson, 1996] we focus here on general methods using tools defined in Section 1.3.3: the Metropolis-Hastings method or the Gibbs sampler.

These conditional simulation tools use the convergence property of a Markov chain to its unique invariant distribution to generate samples according to the conditional distributions. Hence several transitions of such Markov chains are done to output only one sample: this is referred to as a *burn-in*. These transitions serve both to reach the stationarity of the Markov chain and to decorrelate the initial sample from the kept output. General values for this parameter are between 10 and 20.

So these methods require an initial feasible state of the target distribution: if one intends to generate a sample according to $\mu^Y(\cdot \mid Y > y)$ for a given $y \in \mathbb{R}$, then one requires to initiate the Markov chain with a sample $Y^* > y$. If one generates the increasing random walk with non-strict inequality (see Section 3.5.1) the current state Y_n can be used as a *seed* to generate $Y_{n+1} \sim \mu^Y(\cdot \mid Y \geq Y_n)$.

Despite this possibility, it is often preferable to use a starting point already following the target distribution as the *burn-in* will then serve only the independence purpose and eventually remain low. Remember that the renewal property of a Poisson process insures that $\forall y \in \mathbb{R}$, $Y_{M_y+1} \sim \mu^Y(\cdot \mid Y > y)$, this can be done for instance by manipulating several increasing random walks simultaneously. Basically at a given iteration, one can use the states of the *more advanced* random walks as *seeds* for the conditional generation of the other ones. Hence there is a trade-off between the parallel implementation of the increasing random walk based estimators (*i.e.* generating the *iid.* random walks *separately*) and a *proper* simulation of such random walks.

Finally, in our context $Y = g(\mathbf{X})$ with $\mathbf{X} \in \mathbb{X}$ a random finite- or infinite-dimensional vector with known distribution μ^X and $g : \mathbb{X} \to \mathbb{R}$ a *performance* function standing for the computer code for instance. One does not have a generator of μ^Y but instead one needs to simulate $\mathbf{X} \sim \mu^X$ and then to compute $Y = g(\mathbf{X})$. While the usual Splitting framework focuses directly on the input $\mathbf{X}$, the point process framework considers it only as a tool to generate Y. Especially in this framework there is no reason that the number of *particles* $(\mathbf{X}_i)_{i=1}^n$ carried out by an algorithm be equal to the number N of generated random walks.

In Section A.2 we present different algorithms for generating several increasing random walks in parallel, *i.e.* algorithms which can use parallel computing. Then we study in Section A.3 the computing times of the proposed algorithms. Finally Section A.4 presents a numerical study of the impact of the parallelisation of the probability and quantile estimators on usual test cases. There it appears that no decay on the quality of the estimators is found, which means that parallel computing can be applied and save a lot of computational time.

A.2 Parallel algorithms

For defining our algorithms, we focus only on the number of simulated samples, *i.e.* the number of calls to g. All other operations (comparisons, sorting, ...) are considered as *free* (not time consuming) compared to the computing time of the computer code.

Furthermore, we present all algorithms in the non-strict case, *i.e.* with non-strict inequalities. Algorithms for the strict case can be obtained by replacing all the $\geq$ with $>$.

A.2.1 Sampling conditional distributions

This is the main and only requirement of the increasing random walk. We present here practical algorithms used to performed such sampling for both the *static* case, *i.e.* $\mathbf{X} \in \mathbb{X} \subset \mathbb{R}^d$ and the *dynamic* one: $\mathbf{X} = (\mathbf{X}_t)_t \in \mathbb{X} \subset (\mathbb{R}^d)^{\mathbb{R}}$. Both cases require an initial state $\mathbf{X}$ feasible for the target distribution.

Dynamic case Recall that in this context, $\mathbf{X} \in \mathbb{X} \subset (\mathbb{R}^d)^{\mathbb{R}}$ is a random path (the solution of a Stochastic Differential Equation for instance) and one seeks for estimating

the probability that it enters a given set B before another set A. These sets are defined with a measurable *performance function* $\Phi : \mathbb{R}^d \to \mathbb{R}$ such that: $A = \{\mathbf{x} \in \mathbb{R}^d \mid \Phi(\mathbf{x}) \leq 0\}$ and $B = \{\mathbf{x} \in \mathbb{R}^d \mid \Phi(\mathbf{x}) > 1\}$. Then a given trajectory $\mathbf{X}$ enters B before A if $\tau_B < \tau_A$ with τ_C a stopping time defined for any set $C \subset \mathbb{R}^d$ by:

$$\tau_C = \inf\{t \geq 0 \mid \mathbf{X}_t \in C\}.$$

Now let $g : \mathbb{X} \to \mathbb{R}$ be such that $g(\mathbf{X}) = \sup_{t \in [0, \tau_A)} \Phi(\mathbf{X}_t)$ and $Y = g(\mathbf{X})$, the sought probability writes $\mathrm{P}[Y > 1]$. With this setting the idea is to start from a trajectory going at least *as high as* the current level, to replicate it until the first time it overpasses this level, and then to simulate the dynamic of the Stochastic Differential Equation. This is summarised in Algorithm 15.

Algorithm 15 Conditional sampling for Markov process: $\mathbf{X} \in \mathbb{X} \subset (\mathbb{R}^d)^{\mathbb{R}}$

Require: $y \in \mathbb{R}$ ▷ the current level

Require: $\mathbf{X} \mid g(\mathbf{X}) \geq y$ ▷ initial state such that $Y \geq y$

Find $\tau = \inf\{t > 0 \mid \Phi(\mathbf{X}_t) \geq y\}$

Generate a new trajectory $\mathbf{X}^*$ starting from $\mathbf{X}_\tau$

$y^* = \sup_{t \in [0, \tau_{A \cup B}]} \Phi(\mathbf{X}_t^*)$

return $(\mathbf{X}^*)$ and y^*

Static case Here $\mathbf{X} \in \mathbb{X} \subset \mathbb{R}^d$ and g is a complex computer code. We further assume that $\mathbf{X}$ has a density π with respect to the Lebesgue measure: $\mathrm{d}\mu^X(\mathbf{x}) = \pi(\mathbf{x})\mathrm{d}\mathbf{x}$. As stated in Section 1.3.3, the conditional sampling can then be conducted in two steps: first generate a sample according to μ^X, then evaluate g and accept the transition if it lies in the right domain. In this context, the use of a transition kernel targeting μ^X instead of the available generator of μ^X is made to benefit from the *knowledge* of a sample $\mathbf{X}$ already in the right domain: if g has some regularity then it is expected that other *good* samples can be found close to a first one. We give in Algorithm 16 a practical implementation of the Metropolis-Hastings kernel with a symmetric proposal.

We recall also that for a standard Gaussian input space, a direct transition kernel is available [Cérou et al., 2012], which is described in Algorithm 17.

Then let us denote by K a transition kernel with μ^X as stationary distribution, the conditional sampling is done according to Algorithm 18.

Note that in Algorithm 18, using $K(\mathbf{x}, \mathrm{d}\mathbf{x}') = \mu^X(\mathrm{d}\mathbf{x}')$ as a transition kernel amounts to making a basic acceptance-rejection scheme for simulating above the given threshold y.

While the initial state can be the current state of the increasing random walk, it is better to select it in a population already following the target distribution. This will make the *burn-in* only serve the independence purpose and eventually approximate the

Algorithm 16 Metropolis-Hastings transition kernel

Require: $\mathbf{X}$ ▷ an initial state

Require: $\sigma > 0$ ▷ an *exploration* parameter

Generate $\mathbf{U} \sim \mathcal{N}(0, \mathbf{I})$ or $\mathbf{U} \sim \mathcal{U}\left([-1,1]^d\right)$ ▷ $\mathbf{I}$ is the identity matrix in dimension d

$\mathbf{X}^* \leftarrow \mathbf{X} + \sigma\mathbf{U}$

Generate $\rho \sim \mathcal{U}([0,1])$

if $\rho > \min\left(1, \pi(\mathbf{X}^*)/\pi(\mathbf{X})\right)$ **then**

$\mathbf{X}^* \leftarrow \mathbf{X}$

end if

return $\mathbf{X}^*$

Algorithm 17 Transition kernel for standard Gaussian input space

Require: $\mathbf{X}$ ▷ an initial state

Require: $\sigma > 0$ ▷ an *exploration* parameter

Generate $\mathbf{U} \sim \mathcal{N}(0, \mathbf{I})$ ▷ $\mathbf{I}$ is the identity matrix in dimension d

$\mathbf{X}^* \leftarrow \dfrac{\mathbf{X} + \sigma\mathbf{U}}{\sqrt{1+\sigma^2}}$

return $\mathbf{X}^*$

conditional sampling *better*. In other words, this means that one requires a sort of a *database* containing such samples, available at each iteration of Algorithm 5. There could be several ways of getting such a database, from previous sampling of Y to the use of a *pilot run* of any algorithm mentioned so far (see Chapter 1). This database should contain:

$(\mathbf{X}_i)_{i=1}^n$ n samples in the input space $\mathbb{X}$;

$(Y_i)_{i=1}^n$ the response of the code on these samples: $Y_i = g(\mathbf{X}_i)$; and

$(Y_i^{\mathbf{prev}})_{i=1}^n$ the level against which it has been generated.

Algorithm 18 Conditional sampling in the static case: $\mathbf{X} \in \mathbb{X} \subset \mathbb{R}^d$

Require: $K(\cdot,\cdot)$ ▷ a transition kernel for μ^X

Require: $y \in \mathbb{R}$ ▷ the current level

Require: $\mathbf{X}_0$ and $y_0 = g(\mathbf{X}_0)$ ▷ an initial state such that $y_0 \geq y$

Generate $\mathbf{X}^* \sim K(\mathbf{X}_0, \cdot)$

$Y^* \leftarrow g(\mathbf{X}^*)$

if $Y^* < y$ **then**

$\mathbf{X}^* \leftarrow \mathbf{X}_0$ and $Y^* \leftarrow y_0$

end if

return $\mathbf{X}^*$ and Y^*

This last point is important because having $Y_i > y$ with y the current level one seeks to sample above is not sufficient to insure that Y_i follows the target distribution. Indeed if one wants to start from sample following the target distribution, Y_i is a possible starting point *iff.* $Y_i^{\text{prev}} \leq y \leq Y_i$.

Furthermore a database can be augmented at each iteration by adding the points generated by the algorithm itself. Especially all accepted transitions of the *burn-in* step can be saved.

A.2.2 Batch simulation of random walks

When no pilot run nor database is available, there is an easy way to generate such a database on-the-fly, it is by generating not only one random walk but a batch of $N_{\text{batch}} \geq 1$ random walks simultaneously. Then the *less advanced* point processes can use the *more advanced* ones to pick a starting point into them. This raises several questions:

1. how does this deplete the parallel implementation of the estimators?
2. what should be the size of this batch?
3. how many increasing random walks are *moved* at each iteration? *i.e.* for how many of the N_{batch} random walks simulated in the same algorithm is the next state simulated?
4. how to select the initial state for conditional sampling?

Question 1 is very legitimate but there is no general answer to it. Indeed this is mainly machine dependent: if the number of available cores $n_c \geq 1$ is such that $n_c < N$ the total number of desired random walks, then even with a perfect conditional sampler there will be sequentially computed. Also some parallel computing into each batch can be done (see Algorithm 19). In this context $N_{\text{batch}} = \lceil N/n_c \rceil$ will certainly not decay the computing performance. In many practical situations, one has $N \geq 10^3$ while n_c is of order 10 to 100. So the response would be "as soon as $N_{\text{batch}} < N/n_c$ there is no loss in computing time", and even if this is not the case, intra-batch parallelisation can be done to maximise the benefit of multi-core computers. Moreover the problem is generally defined the opposite way: given a number of cores n_c and minimal batch size N_{batch}, how many N can be simulated?

This makes us come to question 2. Indeed the batch size will be somehow the size of the database for conditional sampling (depending on point 3 and if all the generated samples are saved). While in some settings the limit-state function may be very smooth, the dimension of the input space will certainly be a lower bound for N_{batch}: this parameter can be seen as the size of a discretised version of the conditional truncated distributions. The question can then be turned the other way around: what is the minimal sample size required to pick a starting point into it for conditional simulation?

In this context, question 3 comes in mind. The number of random walks whose next step is generated should be as large as possible given that the starting point is chosen amongst a population *large enough* to approximate well the target distribution. Eventually there is no reason to set this value greater than the number of available cores n_c.

Concerning the last point, the starting point is often picked at random in the database, but this can be restrained to the points not directly related to the current state of the increasing random walk in order to limit the correlation between samples.

We give now in Algorithm 19 a general approach for batch generation of increasing random walks taking into account these specifications. Then we will show how the Last Particle Algorithm is only one possible easily implementable solution which falls into this framework.

Line 9 of Algorithm 19 defines which and how many random walks will be updated at a given iteration. This should depend on the number of samples available as starting points in the database. This number should not be too low regarding the dimension of the input space, otherwise the *diversity* of the population may decay quickly. Here one can also include some other conditions mentioned above: $\text{card}(\texttt{ind}) = n_c$ for instance (see line 9 of Algorithm 19).

Then for each random walk selected in `ind`, one performs the conditional simulation with b transitions of the Markov chain with the reversible kernel K. The selection of the starting point at line 11 should be done according to the distribution of the sample. Some other criteria can be taken into account here. For instance one may want to avoid to pick a sample $\mathbf{X}^*$ belonging to the same Markov chain used for the last generation of the random walk.

Finally, the stopping condition of the algorithm is not defined. Depending on the estimator used, the increasing random walks have to be generated either until a given common threshold $q \in \mathbb{R}$, or such that the total number of events of the superposed process be equal to some prescribed value $m \in \mathbb{N}^*$ as described in before.

A.2.3 Fixed threshold

For the estimation of $p = \mathrm{P}[Y > q]$ for a given $q \in \mathbb{R}$, one requires to generate N *iid.* increasing random walks until the first time after q, *i.e.* until they all overpass q. Then the superposed process (*i.e.* the merged and sorted sequence of all the states of all the random walks) will be complete until q.

In this case, the condition at line 8 of Algorithm 19 simply writes:

$$\texttt{cond} = \min_i Y_{n_i}^i \leq q.$$

Algorithm 20 for probability estimation is a direct wrap-up of Algorithm 19 with this stopping condition. Note that it is defined with non-strict random walks. If the strict

Algorithm 19 Batch generation of increasing random walks

Require: $N_{\text{batch}} \geq 1$ ▷ the number of simulated random walks

Require: $K(\cdot,\cdot)$ ▷ a transition kernel

Require: $b \geq 1$ ▷ a *burn-in* parameter

Require: `cond` ▷ a boolean checking a stopping condition

$\mathbf{n} = (n_i)_{i=1}^{N_{\text{batch}}} = (1, \cdots, 1)$ ▷ the number of simulated states of each random walk

$\mathbf{M} = (M_i)_{i=1}^{N_{\text{batch}}} = (0, \cdots, 0)$ ▷ the counting rv for the Poisson correction

Generate *iid.* $(\mathbf{X}_i)_{i=1}^{N_{\text{batch}}}$ according to μ^X

$\forall i \leq N_{\text{batch}}$, $Y_{n_i}^i = g(\mathbf{X}_i)$ ▷ first state of the N_{batch} random walks

Generate *iid.* $(U_{n_i}^i)_{i=1}^{N_{\text{batch}}}$ according to $\mathcal{U}([0,1])$ ▷ for the Poisson correction

$Y^{\text{prev}} = (-\infty, \cdots, -\infty)$ ▷ the first sampling is $\sim \mu^Y(\cdot \mid Y > -\infty)$

`db` $= [(\mathbf{X}_i, Y_{n_i}^i, Y_i^{\text{prev}})_{i=1}^{N_{\text{batch}}}]$ ▷ initialisation of the database

while `cond` **do**

  Get `ind` $\subset [\![1, N_{\text{batch}}]\!]$ ▷ the index of the random walks updated at this iteration

  foreach $j \in$ `ind` **do** ▷ this can be done in parallel

    Get $(\mathbf{X}^*, Y^*) \in$ `db` a starting point

    $Y_{n_j+1}^j \leftarrow Y^*$

    do b **times** ▷ conditional simulation

      $\mathbf{X}_{\text{tmp}}^* \sim K(\mathbf{X}^*, \cdot)$; $Y^* = g(\mathbf{X}_{\text{tmp}}^*)$

      if $Y^* \geq Y_{n_j}^j$ **then**

        $Y_{n_j+1}^j \leftarrow Y^*$

        add $(\mathbf{X}_{\text{tmp}}^*, Y^*, Y_{n_j}^j)$ to `db`

        $\mathbf{X}^* \leftarrow \mathbf{X}_{\text{tmp}}^*$

      end if

    end do

    $U_{n_j+1}^j \sim \mathcal{U}([0,1])$ ▷ for the Poisson correction

    $n_j \leftarrow n_j + 1$; $M_j \leftarrow M_j + 1$

    if $Y_{n_j}^j = Y_{n_j-1}^j$ & $U_{n_j}^j < U_{n_j-1}^j$ **then**

      $M_j \leftarrow M_j - 1$ ▷ transition refused for the Poisson correction

    end if

  end foreach

  update `cond`

end while

return $(Y^i)_{i=1}^{N_{\text{batch}}}$ ▷ the N_{batch} increasing random walks

return $(M_i)_{i=1}^{N_{\text{batch}}}$ ▷ the N_{batch} Poisson random variables

random walk is considered instead, line 10 should be modified accordingly:

$$\widehat{p}^{>} = \prod_i \left(1 - \frac{r_i}{N}\right).$$

Algorithm 20 Probability estimator

Require: $q \in \mathbb{R} \mid \mathrm{P}\left[Y > q\right] > 0$ ▷ the common threshold

Require: $N_{\mathrm{batch}} \geq 1$ ▷ the number of random walks per batch

Require: $k \geq 1$ ▷ the number of batches

$N \leftarrow kN_{\mathrm{batch}}$ ▷ the total number of simulated increasing random walks

do k **times** ▷ this can done in parallel

3: Run Algorithm 19 with N_{batch} and $\texttt{cond} = \min Y_{n_i}^i \leq q$

end do

$(Y^i)_{i=1}^N \leftarrow ((Y^i)_{i=1}^{N_{\mathrm{batch}}}, \cdots, (Y^i)_{i=(k-1)N_{\mathrm{batch}}+1}^{kN_{\mathrm{batch}}})$ ▷ sequence of random walks

6: $\bar{M}_q^{\geq} = \sum_{i=1}^{N} (n_i - 1)$ ▷ the counting random variable of the superposed process

$(M^i)_{i=1}^N \leftarrow ((M^i)_{i=1}^{N_{\mathrm{batch}}}, \cdots, (M^i)_{i=(k-1)N_{\mathrm{batch}}+1}^{kN_{\mathrm{batch}}})$ ▷ sequence of Poisson rv

Get $(Y_n)_{n=1}^{\bar{M}_q^{\geq}}$ the superposed process

9: $r \leftarrow \mathrm{RLE}((Y_n)_{n=1}^{\bar{M}_q^{\geq}})$ ▷ the Run-Length encoding, see Definition 3.4

return $\widehat{p}^{\geq} = \prod_i (N-1)/(N-1+r_i)$ ▷ the MVUE of the probability

return $\widehat{p} = (1 - 1/N)^{\sum M^i}$ ▷ the *pure Poisson* estimator

A.2.4 Fixed number of terms

For the quantile estimator as well as for the mean estimator (nested sampling) one requires indeed to simulate the superposed process until a given iteration $m \in \mathbb{N}^*$. However recover the full process with parameter N (the superposed process of N *iid.* increasing random walks) until event number m requires to make sure that all the random walks have overpassed a given state $y \in \mathbb{R}$ and that the number of events before that time y is greater than m. In this context, the stopping criterion for Algorithm 19 becomes:

$$\texttt{cond} = \sum_{i=1}^{N_{\mathrm{batch}}} \sum_{j=1}^{n_i} \mathbb{1}_{Y_j^i < y_{\min}} < m$$

with $y_{\min} = \min_{i \in [\![1, N_{\mathrm{batch}}]\!]} Y_{n_i}^i$ the smallest farthest state of the N_{batch} random walks.

Since Algorithm 19 allows for parallel computation at each iteration (see line 10) it is a direct possible parallel algorithm for quantile estimation and for nested sampling ($\widehat{Z}$, see Section 4.3.1). However it is only *sequentially* parallel (as any Multilevel Splitting methods) in the sense that it can only distribute the computing load of each iteration. Apart from making an intensive use of inter-processes communication, which is more

complicated and time consuming, it makes the algorithm wait for all the calculations to be done before going to the next step. If the computing time of g is not deterministic, then it means that the wall-clock time of each iteration is driven by the law of the maximum of k realisations of the random time of g.

To circumvent this limitation we suggest a 2-passes algorithm: we run a first time in parallel several Algorithms 19 with a fixed target number of events m_0, then get the farthest state reached and relaunch all the random walks until they reach it too. The choice of this parameter m_0 will be discussed in Section A.3.2. As a matter of fact, if the target number of events is m and one runs k algorithms in parallel, then $m_0 = \lceil m/k \rceil$ insures that the final number of events will be greater than m.

Algorithm 21 Parallel generation of a superposed point process for a given number of iterations

Require: m ▷ the target number of events of the superposed process

Require: m_0 ▷ the fixed number of events per Algorithm 19

Require: $N_{\text{batch}} \geq 1$ ▷ the number of random walks per batch

Require: $k \geq 1$ ▷ the number of batches

do k **times**

Run Algorithm 19 with N_{batch} and $\texttt{cond} = \sum_{i=1}^{N_{\text{batch}}} \sum_{j=1}^{n_i} \mathbb{1}_{Y_j^i < y_{\min}} < m_0$

3: **end do**

$q_{\max} = \max_{l \in [\![1,k]\!]} \min_{i \in [\![(l-1)N_{\text{batch}}+1, lN_{\text{batch}}]\!]} Y_{n_i}^i$ ▷ the farthest state of the k point processes

do k-1 **times** ▷ no need to relaunch the farthest point process

6: Restart Algorithm 19 with $\texttt{cond} = \min Y_{n_i}^i \leq q_{\max}$

end do

$(Y^i)_{i=1}^N \leftarrow ((Y^i)_{i=1}^{N_{\text{batch}}}, \cdots, (Y^i)_{i=(k-1)N_{\text{batch}}+1}^{kN_{\text{batch}}})$ ▷ sequence of random walks

9: **return** $(Y^i)_{i=1}^N$

A.2.5 Last Particle Algorithm

The Last Particle Algorithm is a specific implementation of the increasing random walk for continuous random variables and so falls into the framework of Algorithm 19.

More precisely, it gives the following answers to the above mentioned questions:

batch size it is the total number of point processes wanted for the statistic considered, *i.e.* $N_{\text{batch}} = N$;

***moves* per iteration** only the less advanced point process is considered, so the name *Last Particle*. Line 9 is then replaced by $\texttt{ind} = \text{argmin}_i Y_{n_i}^i$.

database it is composed by the last state of the $N-1$ other point processes.

While in its original formulation it was proposed to use the Last Particle Algorithm directly without parallel computing, it is definitely possible to use it as the batch generating algorithm process in Algorithms 20 and 21.

The main interest of this algorithm comes from the fact that at each iteration, it generates only one sample, precisely the next state of the superposed process with parameter N_{batch}. Hence at any iteration one has directly the superposed process until this given state. Furthermore the random number of iterations of the algorithm is driven by the law of the counting random variable of the increasing random walk with parameter N.

Finally we stress out the fact that this property is to be found with Algorithm 19 as soon as line 9 is replaced by:

$$\texttt{ind} = \operatorname{argmin} Y_{n_i}^i.$$

Algorithm 19 still carries much information: precisely it allows for handling potential discontinuities in the *cdf* of Y and returns not only the states and the counting random variable of the superposed process with parameter N but separately the states of the N_{batch} *iid.* (in the ideal case where the conditional sampling is *perfect*) point processes and the corrected counting random variables (those ones following a Poisson distribution with or without discontinuities). In other words it outputs N_{batch} *iid.* random variables with distribution described in Proposition 3.9 or 3.10 and N_{batch} *iid.* Poisson random variables. Usual statistical tests such as the χ^2 tests (goodness-of-fit or independence) can be used to verify this property.

For the sake of completeness, we give in Algorithm 22 the original Last Particle Algorithm.

A.3 Wall-clock time

In this section we focus on a *last particle* implementation of Algorithm 19. We further assume that the *cdf* of Y is continuous. In the following we consider the setting where $n_c \geq 1$ *cores* are available and are used to generate in parallel n_c Algorithms 19, each with N_{batch} point processes. This makes a total of $N = n_c N_{\text{batch}}$ generated point processes.

In all this section, the number n_c of cores is supposed to be large.

A.3.1 Fixed threshold algorithm

We first focus on the wall-clock time of the probability estimator. We know that the random number of iterations N_{iter} of Algorithm 19 before it stops follows a Poisson law with parameter $-N_{\text{batch}} \log p$. Note that considering a parallelisation at line 9 with k generations at each iteration only changes the parameter of N_{iter}: $-(N_{\text{batch}}/k) \log p$.

The wall-clock time is defined as the duration of the algorithm in real time (seconds, minutes, etc.), it is the time spent by the user to get the result of the algorithm. In our context where the main source of running time comes from the call to the computer code

Algorithm 22 Last Particle Algorithm

Require: $N \geq 1$ ▷ the number of increasing random walks
Require: $b \geq 1$ ▷ the *burn-in* parameter
Require: $K(\cdot, \cdot)$ ▷ a transition kernel for μ^X
$(q \in \mathbb{R}$ and $N_{\text{iter}} = \infty)$ or $(q = \infty$ and $N_{\text{iter}} \in \mathbb{N})$
$M \leftarrow 0$ ▷ the number of event of the superposed process
Generate $(\mathbf{X}_i)_{i=1}^N$ *iid.* replicas of $\mathbf{X} \sim \mu^X$
$\mathbf{y} \leftarrow (g(\mathbf{X}_1), \cdots, g(\mathbf{X}_N))$
while $\min \mathbf{y} < q$ & $M < N_{\text{iter}}$ **do**
  $M \leftarrow M + 1$
  $i = \operatorname{argmin} \mathbf{y}$
  Get $J \sim \mathcal{U}([\![1, N]\!] \setminus \{i\})$
  $\mathbf{X}_i = \mathbf{X}_J$; $y_i = y_J$
  do b **times**
    $\mathbf{X}^* = K(\mathbf{X}_i, \cdot)$
    $y^* = g(\mathbf{X}^*)$
    if $y^* > y_i$ **then**
      $\mathbf{X}_i \leftarrow \mathbf{X}^*$; $y_i \leftarrow y^*$
    end if
  end do
end while

g, we simplify the analysis by counting only the number of times g is called sequentially in an algorithm and refer to this quantity as *effective computing time*.

The following proposition aims at giving the expected duration time of Algorithm 20.

Proposition A.1. *Let t_{par} be the random variable of the effective computing time of the probability estimator (Algorithm 20) with N_{batch} point processes per algorithm and n_c cores, one has:*

$$\mathrm{E}\left[t_{par}\right] = \frac{b(\log p)^2}{n_c \delta^2} \left(1 + \sqrt{\frac{n_c \delta^2}{(\log p)^2}} \sqrt{2 \log n_c} + \frac{1}{b \log 1/p}\right) \tag{A.1}$$

with b the burn-in parameter and δ the coefficient of variation of the estimator.

Proof. Let $N = n_c N_{\text{batch}}$ be the total number of generated point processes and λ be the parameter of the Poisson laws: $\lambda = -N_{\text{batch}} \log p = -(N/n_c) \log p$. The effective computing time of Algorithm 20 will be the maximum of n_c *iid.* Poisson random variables with parameter λ.

In the extreme value theory framework, we are interested in the so called location parameter b_{n_c} which drives the mean of the maximum of n_c *iid.* random variables with

cdf F. It is the solution of the equation:

$$b_{n_c} = F^{-1}\left(1 - \frac{1}{n_c}\right).$$

For the Normal distribution, we have:

$$b_{n_c} = \sqrt{2\log n_c} + \frac{\log\log n_c + \log 4\pi}{2\sqrt{2\log n_c}} \sim \sqrt{2\log n_c}.$$

Furthermore, we know that Normal approximation of a Poisson distribution with parameter λ is valid in the range $(-\sqrt{\lambda}, \sqrt{\lambda})$. Here, this means that as soon as $\sqrt{2\log n_c} < \sqrt{\lambda}$, we can use the approximation of the Poisson law by a Normal one's to calculate the constant. The practical values of $N_{\text{batch}} = N/n_c \approx 10^1$, $-\log p \approx 10^1$ and $n_c \approx 10^2$ allow us to make the approximation:

$$\mathcal{P}(\lambda) \sim \mathcal{N}(\lambda, \lambda).$$

Let $N_{\max}$ be the random variable of the maximum of n_c *iid.* standard Gaussian variables. The total number of calls is the sum of the N_{batch} initial calls to the limit-state function and the number of iterations. The former are ready made from the simulator of μ^X while the latter require the Markov chain drawing; thus we have:

$$\mathrm{E}\left[t_{\text{par}}\right] = \mathrm{E}\left[b(N_{\max}\sqrt{\lambda} + \lambda) + N/n_c\right] \approx b\left(\frac{N}{n_c}\log 1/p + \sqrt{\frac{N}{n_c}\log 1/p}\sqrt{2\log n_c}\right) + \frac{N}{n_c}$$

and so in terms of coefficient of variation:

$$\mathrm{E}\left[t_{\text{par}}\right] = b\frac{(\log p)^2}{n_c\delta^2} + b\frac{|\log p|}{\delta}\sqrt{\frac{2\log n_c}{n_c}} + \frac{-\log p}{n_c\delta^2}.$$

□

Hence we can see that with an embarrassingly parallel implementation, the expected *duration*, *i.e.* here the expected longest sequence of calls made by a computational unit (a core, a thread, a node, ...) is the sum of the mean of the Poisson random variable of the number of iterations per Algorithm 19 (the number of events before the considered threshold) and a term due to the extreme value theory: the algorithm stops when the longest sequence is done.

This additional term due to the full parallelisation could drop with a dynamic allocation of the computational resources. Indeed, considering that Poisson dsitributions of the random number of events are symmetrically distributed, there will be as many *shorter* than *longer* algorithms comparing to the reference value (the mean) $-N/n_c \log p$, so that the computational resources liberated by the first ones could be allocated to the second ones, and finally approximately compensate each other.

In comparison, a classical Subset Simulation method does not allow for embarrassingly parallel implementation. Rather it sequentially distributes the load of each iteration. Then

the running time is driven by the law of the maximum of N/n_c calls to the code g, times the number of steps, *i.e.* almost surely $\log p / \log p_0$. In a *last particle* setting, this law of the maximum of the N/n_c calls to g *disappears* since calls are made sequentially and so in expectation one can retain only the expected computational time of g.

Finally for a first comparison with classical Multilevel Splitting methods one can retain:

$$t_{\text{par}} = \frac{b(\log p)^2}{n_c \delta^2}. \tag{A.2}$$

In Multilevel Splitting algorithms, there are N samples generated initially and then $N(1-p_0)$ regenerated at each iteration. If one considers that the running time of g is constant, the computational time writes as follows:

$$\begin{aligned} t_{\text{MS}} &= \frac{N}{n_c} + \lfloor \frac{\log p}{\log p_0} \rfloor b \left[\frac{N(1-p_0)}{n_c} \vee 1 \right] \\ t_{\text{MS}} &\approx \frac{\log p}{\log p_0} \frac{1-p_0}{n_c \delta^2 p_0} + \frac{\log p}{\log p_0} b \left[\frac{\log p}{\log p_0} \frac{(1-p_0)^2}{n_c \delta^2 p_0} \vee 1 \right]. \end{aligned}$$

Depending on the parametrisation of the algorithm (choice of N and p_0, number of *cores* n_c) we will have either $N(1-p_0) \geq n_c$ and so:

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

or $N(1-p_0) \leq n_c$ and so:

$$t_{\text{MS}} = \frac{\log p}{\log p_0} b \tag{A.4}$$

with b the *burn-in* parameter.

Formula (A.3) is strictly decreasing in p_0 while Eq. (A.4) is strictly increasing. This can eventually suggest an optimal value p_0^* for p_0 depending on N and n_c, it is the solution of:

$$\frac{\log p}{\log p_0^*} \frac{(1-p_0^*)^2}{n_c \delta^2 p_0^*} = \frac{N(1-p_0^*)}{n_c} = 1. \tag{A.5}$$

This optimal value of p_0 means indeed that one should resample at each iteration only n_c samples. Finally we can write:

$$t_{\text{MS}}(p_0) \geq t_{\text{MS}}(p_0^*) = \frac{b(\log p)^2}{n_c \delta^2} \frac{(1-p_0^*)^2}{p_0^* (\log p_0^*)^2} > \frac{b(\log p)^2}{n_c \delta^2} = t_{\text{par}}. \tag{A.6}$$

These calculations show that also when looking at the effective implementation of the statistics defined in Section 1.3 and Chapter 3 the point process lets obtain the best estimator in terms of variance against effective computational time. Especially, the usual value $p_0 = 0.1$ results in an effective speed up of:

$$\frac{(1-p_0^*)^2}{p_0^* (\log p_0^*)^2} \approx 1.53.$$

A.3.2 Fixed number of terms algorithm

We now focus on Algorithm 21, which is used for both the quantile and the moment estimator. Especially we first address the issue of choosing the parameter m_0 of the deterministic number of iterations of the first pass.

Let us denote by $(q_i)_{i=1}^{n_c}$ the sequence of the n_c last states of the after the first pass (see Algorithm 21 line 2):

$$\forall i \in [\![1, n_c]\!],\ q_i = \min_{j \in [\![1, N_{\text{batch}}]\!]} Y_{n_j}^j$$

and $(T_i)_{i=1}^{n_c}$ the sequence of the corresponding times of the homogeneous Poisson processes:

$$\forall i \in [\![1, n_c]\!],\ T_i = -\log \mathrm{P}\left[Y > q_i\right].$$

Since $(T_i)_i$ are the times of homogeneous Poisson processes with parameter N_{batch}, one knows that they are *iid.* Gamma random variables $\Gamma_{m_0}/N_{\text{batch}}$.

Let $T_{\max} = \max_i T_i$. Since $y \mapsto -\log \mathrm{P}\left[Y > y\right]$ is an increasing function, one has:

$$T_{\max} = -\log \mathrm{P}\left[Y > q_{\max}\right].$$

On the one hand $T_{\max}$ is then the maximum of n_c *iid.* random variables with distribution $\Gamma_{m_0}/N_{\text{batch}}$. On the other hand the random number of events before $q_{\max}$ is the same as the random number of events before $T_{\max}$ for the corresponding homogeneous Poisson processes.

Proposition A.2. *Let t_{par} be the effective computing time of algorithm 21, we have:*

$$\mathrm{E}\left[t_{par}\right] = m_0 + \sqrt{2 m_0 \log n_c}. \tag{A.7}$$

Proof. We apply here the same line of argumentation as in the proof of Proposition A.1, which lets us approximate the Gamma distribution by a Gaussian one:

$$\frac{\Gamma_{m_0}}{N_{\text{batch}}} \sim \mathcal{N}\left(\frac{m_0}{N_{\text{batch}}}, \frac{m_0}{{N_{\text{batch}}}^2}\right).$$

Let $N_{\max}$ be the random variable of the maximum of n_c standard Gaussian variables, we have:

$$\mathrm{E}\left[T_{\max}\right] = \mathrm{E}\left[N_{\max}\frac{\sqrt{m_0}}{N_{\text{batch}}} + \frac{m_0}{N_{\text{batch}}}\right] \approx \frac{m_0}{N_{\text{batch}}} + \frac{\sqrt{m_0}}{N_{\text{batch}}}\sqrt{2\log n_c}.$$

Then

$$\mathrm{E}\left[t_{\text{par}}\right] = \mathrm{E}\left[\mathrm{E}\left[t_{\text{par}} \mid T_{\max}\right]\right] = N_{\text{batch}}\,\mathrm{E}\left[T_{\max}\right] = m_0 + \sqrt{2 m_0 \log n_c}.$$

□

We now provide a criterion for choosing m_0. Indeed, we consider that one can accept

to take a *risk* α that the Poisson Process does not go far enough.

Let M_t be the counting random variable of the marked Poisson Process a time t and $m = \lceil -n_c N_{\text{batch}} \log p \rceil$ the targeted number of events, the criterion writes as follows:

$$\mathrm{P}\left[M_{T_{\max}} \leq m\right] \leq \alpha. \tag{A.8}$$

Proposition A.3 (Choice of m_0)**.** *With the previous notations and $t = -\log p$, we have:*

$$m_0 = \lceil N_{batch}\, t + \beta^2/2 - \beta\sqrt{\Delta}/2 \rceil \tag{A.9}$$

with:

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

and b_{n_c} the localisation parameter of the Gaussian law in the framework of Extreme Value Theory: $b_{n_c} = \sqrt{2\log n_c} - \left(\log\log n_c + \log 4\pi\right)/2\sqrt{2\log n_c}$ and $\Delta = \beta^2 + 4N_{batch}\, t$.

Proof. We have $\mathrm{P}\left[M_{T_{\max}} \leq m\right] = \mathrm{P}\left[T_{\max} \leq T_m\right]$. Furthermore, $T_m = \Gamma_m/N \sim \mathcal{N}(t, t/N)$, which gives: $\mathrm{P}\left[T_{\max} \leq T_m\right] \approx \mathrm{P}\left[T_{\max} \leq t\right]$.

Finally, we seek for m_0 such that:

$$\mathrm{P}\left[T_{\max} \leq t\right] \leq \alpha.$$

Approximating once again Gamma laws with Gaussian distributions and using the extreme value theory, we get:

$$\alpha = \exp\left(-\exp\left(-\sqrt{2\log n_c}\left(\frac{N_{\text{batch}}\, t}{\sqrt{m_0}} - \sqrt{m_0} - b_{n_c}\right)\right)\right)$$

which concludes the proof. □

Remark A.1. *The targeted value α should not be set too small as the approximation of Gamma laws with Gaussian distributions is not correct for rare events. However we know that taking $m_0 = \lceil -N_{batch} \log p \rceil$ ensures a sufficient number of iterations because $n_c \lceil -N_{batch} \log p \rceil \geq \lceil -n_c N_{batch} \log p \rceil$. Furthermore, if after Algorithm 21 the number of events is not sufficient, one can restart Algorithm 19 with the whole point process with parameter N for the number of missing events.*

Eventually this risk *is only a mean to allow for embarrassingly parallel computation of the quantile and moment estimators and the worst that can happen is that it ends up by doing a sequential parallelisation on the form of Algorithm 19.*

Corollary A.1 (Expected effective computing time of the quantile estimator (see Algorithm 21))**.** *With this value of m_0, we have:*

$$\mathrm{E}\left[t_{par}\right] \approx -N_{batch} \log p + \sqrt{2N_{batch} \log 1/p \log n_c}. \tag{A.10}$$

As for the probability estimator, we now give the effective computing time of the quantile estimator as a function of its coefficient of variation δ:

$$\mathrm{E}\left[t_{\mathrm{par}}\right] \approx \frac{b}{n_c}\left(\frac{p\log p}{\delta q f_Y(q)}\right)^2\left(1+\frac{1}{b\log 1/p}+\delta\gamma(q)\sqrt{2n_c\log n_c}\right) \tag{A.11}$$

with f_Y the *pdf* of Y, $\gamma(q)=\dfrac{qf_Y(q)}{-p\log p}$ and b the *burn-in* parameter.

Remark A.2 (Order of magnitude of $\gamma(q)$)**.** *While there is no general result on the order of magnitude of $\gamma(q)$, it can be shown that in a lot of cases it remains small. For instance if one considers von Mises distributions, i.e. distributions such that the cdf F_Y of Y has the following representation:*

$$1-F_Y(q)=\bar{F_Y}(q)=c\exp\left(-\int_z^q\frac{1}{a(t)}\mathrm{d}t\right)$$

with c a given positive constant and $a(\cdot)$ the auxiliary function of F_Y: $a=\bar{F_Y}/f$ [see Embrechts et al., 1997, Definition 3.3.18], one obtains:

$$\frac{1}{\gamma(q)}=\frac{-p\log p}{qf_Y(q)}=-\frac{a(q)}{q}\log c+\frac{a(q)}{q}\int_z^q\frac{\mathrm{d}t}{a(t)}.$$

In this equality, the first term goes to 0 [see Embrechts et al., 1997, Proposition 3.3.24] and the second one can be bounded from below by $1-z/q$ for any $z\in\mathbb{R}$ such that $a'>0$ over $[z,\infty)$. This eventually means that $\gamma(q)\in[0,1]$. For instance, Exponential, Weibull or Erlang distributions all have a von Mises representation.

As for the probability estimator there is an extra term in Eq. (A.11) driven by $\sqrt{n_c\log n_c}$. It is a direct consequence of the embarrassingly parallel implementation. Eventually the conclusions remain the same.

A.4 Numerical benchmark of the parallelisation

In this section, we try to estimate probabilities and quantiles on usual test cases with different values for N_{batch} for a given total number of processes N. This aims at evaluating the effect on the parallelisation on numerical results. Especially while there should not be any difference in theory, the *quality* of conditional simulations depends on the size N_{batch}.

In the following, we use the Last Particle Algorithm as a particular implementation of Algorithm 19 and the results are got from `R` package `mistral` [Bousquet et al., 2015]. This is to underline the fact that the Last Particle Algorithm as proposed by Guyader et al. [2011] or Simonnet [2016] is only one possible implementation of the probability estimator, and to show how this estimator is altered or not with the use of parallel computing. Especially some authors [Bréhier et al., 2015a] have proposed to average

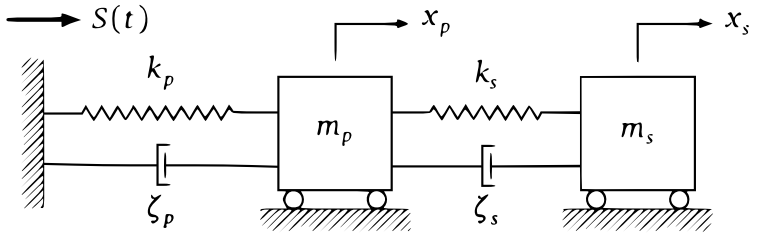


Figure A.1: A 2 degrees of freedom damped oscillator [illustration from Dubourg et al., 2011].

several *iid.* realisations of the Last Particle Algorithm to benefit from parallel computers. This is clearly under optimal because:

$$\forall (n_c, N_{\text{batch}}) \in (\mathbb{N}^*)^2,\ \frac{1}{n_c}\left(p^{-1/N_{\text{batch}}} - 1\right) \geq p^{-1/(n_c N_{\text{batch}})} - 1.$$

Furthermore, the law of the averaged estimator is not the same as the one of the original MVUE estimator (see Section 3.3). Eventually it depends on the implementation (number of *iid.* replicas).

A.4.1 Presentation of the examples

Watermarking detection This example is the one used by Cérou et al. [2012] and Guyader et al. [2011] to show the properties of their algorithms. 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]:

$$F = \{\mathbf{x} \in \mathbb{R}^d \mid g(\mathbf{x}) := \frac{\mid \mathbf{x}^T \mathbf{u} \mid}{\parallel \mathbf{x} \parallel} > q\}. \tag{A.12}$$

The analytic relation between p and q writes as follows:

$$p = \mathbb{P}(g(\mathbf{X}) > q) = 1 - F_Y(q) = 1 - G\left(\frac{(d-1)q^2}{1-q^2}\right)$$

with F_Y the *cdf* of $g(\mathbf{X})$ and G the *cdf* of a Fisher variable with $(1, d-1)$ degrees of freedom [see Guyader et al., 2011].

A two-degrees-of-freedom damped oscillator This example sketched in Figure A.1 was first proposed by Kiureghian and Stefano [1991] and then used by Bourinet et al. [2011] and Dubourg et al. [2011].

It is a two degrees of freedom damped oscillator characterised by masses m_p and m_s, spring stiffnesses k_p and k_s, natural frequencies $\omega_p^2 = k_p/m_p$ and $\omega_s^2 = k_s/m_s$ and damping ratios ζ_p and ζ_s. 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 as follows:

$$\mathbb{E}[x_s^2] = \pi \frac{S_0}{4\zeta_s\omega_s^2} \frac{\zeta_a\zeta_s}{\zeta_p\zeta_s(4\zeta_a^2+\theta^2)+\gamma\zeta_a^2} \frac{(\zeta_p\omega_p^3+\zeta_s\omega_s^3)\omega_p}{4\zeta_a\omega_a^4}$$

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

Finally, Kiureghian and Stefano [1991] showed that the limit-state function could write under reasonable approximation as follows:

$$g(\mathbf{x}) = F_s - p\, k_s \sqrt{\mathbb{E}[x_s^2]} \tag{A.13}$$

with F_s the force capacity of the secondary spring and $p = 3$ a peak factor [Dubourg et al., 2011].

Table A.1 presents the probabilistic model used. As it uses lognormal distributions

Variable	Mean	CV (%)
m_p	1.5	10
m_s	0.01	10
k_p	1	20
k_s	0.01	20
ζ_p	0.05	40
ζ_s	0.02	50
F_s	27.5	10
S_0	100	10

Table A.1: Stochastic model of the oscillator. All random variables are lognormally distributed.

and reversible kernel is defined in the standard space an iso-probabilistic transformation is done before each call to the limit-state function.

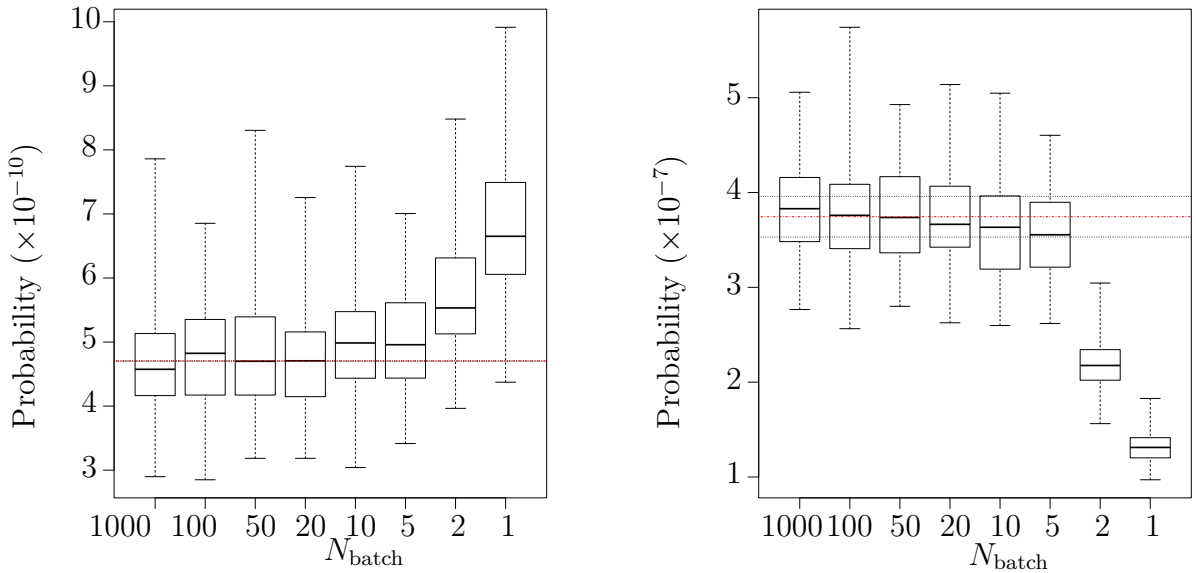
A.4.2 Estimation of failure probability

Watermarking detection Here we set $d = 20$, $q = 0.95$ and we try to estimate $p = 1 - G\left(\frac{(d-1)q^2}{1-q^2}\right) = 4.704\,10^{-11}$ where G is the *cdf* of a Fisher random variable with $(1, d-1)$ degrees of freedom.

A two-degrees-of-freedom damped oscillator Failure is defined as $g(\mathbf{x}) < 0$. There is no analytical expression available. However a reference value was calculated using the usual Subset Simulation algorithm with $N = 4 \times 10^6$, and one found: $p \approx 3.75 \times 10^{-7}$ with a coefficient of variation lower than 3%.

The purpose of this part is to evaluate the behaviour of the estimator depending on the batch size of Algorithm 20. Especially the smaller N_{batch} the faster the algorithm in wall-clock time. The following configurations have been tested, expressed as "$n_c \times N_{\text{batch}}$":

"1x1000", "10x100", "20x50", "50x20", "100x10", "200x5", "500x2" and "1000x1". Results are shown in Figure A.2 as boxplots over 100 simulations, whiskers extending to the extreme values. The reference value is displayed with the red dashed line, and in the case of the 2 d-o-f oscillator a 95% confidence interval is displayed as well with the black dashed lines. In both examples it appears that from $N_{\text{batch}} = 10$ the estimators are almost the



(a) Watermarking detection

(b) 2 d-o-f oscillator with $\mathrm{E}[F_s] = 27.5$

Figure A.2: Boxplots of the probability estimator given by Algorithm 20 with a total of $N = 1000$ random walks generated by batches of size N_{batch}. Results over 100 simulations, whiskers extending to the extreme values. The dashed lines stand for the references values.

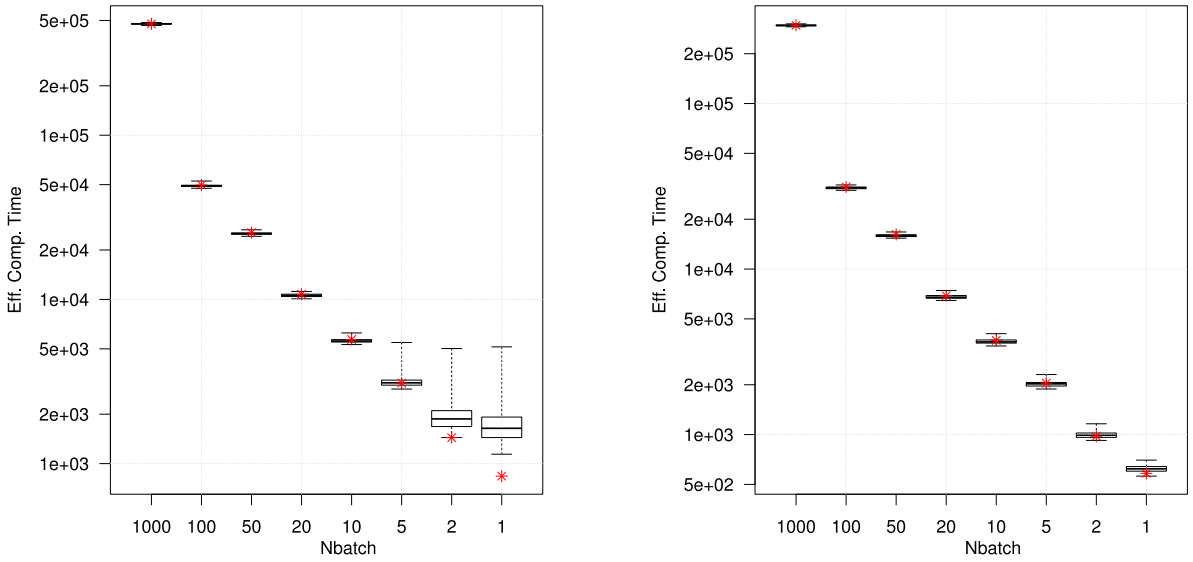
same.

We now look at the effective computing time of the estimators, *i.e.* the maximum number of calls to the limit-state function made by each one of the n_c algorithms for a given configuration. Especially we intend to check the consistency of formula (A.1).

As for the probability, results are displayed as boxplots over 100 simulations in Figure A.3. The red dots show the theoretical value given by Eq. (A.1). One can see that apart from the two last configurations of the watermarking detection example, these values are in good agreement with the empirical results.

To conclude, the point process point of view allows to define new parallel algorithms for extreme probability estimation. The numerical results for a particular implementation (the Last Particle Algorithm as the batch generator of point processes) show that parallel computation can efficiently reduce the wall-clock time of this algorithm without altering its statistical properties. Since it is linked with Multilevel Splitting methods, we hence have obtained the most efficient Multilevel Splitting strategy in terms of variance of the estimator against time. Especially the gain is approximately 50% comparing to the original *Subset Simulation* as described by Au and Beck [2001] (see Eq. A.6).

Practically speaking, the minimal number of particles to be considered in each algorithm



(a) Watermarking detection

(b) 2 d-o-f oscillator with $\mathrm{E}\left[F_s\right] = 27.5$

Figure A.3: Effective computing time of the probability estimator given by Algorithm 20 with a total of $N = 1000$ random walks generated by batches of size N_{batch}. Results over 100 simulations, whiskers extending to the extreme values. The starred dots stand for the theoretical values given by Eq. (A.1).

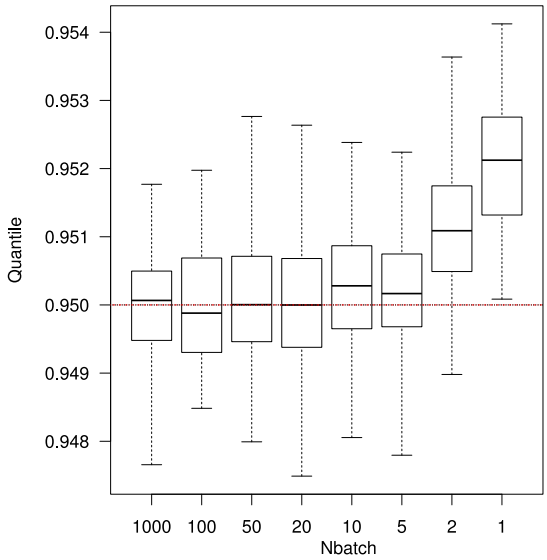
19 seems to depend on the limit-state function as well as on the dimension on the input space and should be set accordingly.

A.4.3 Estimation of quantile

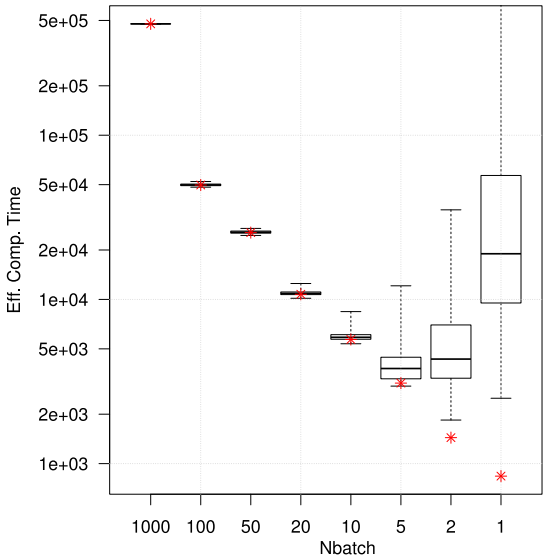
We use the watermarking detection example presented Section A.4.1. We now set $p = 4.704\,10^{-11}$ and try to find back $q = 0.95$. Results are displayed in Figure A.4 as boxplots over 100 simulations, whiskers extending to the extreme values and reference value is added to the plot with a dashed line. For $N_{\text{batch}} \geq 10$ the estimators seem to be almost the same. Indeed one could expect to get the same type of results as for the probability estimators because the algorithms are intrinsically the same, *i.e.* a wrap-up of Algorithm 19, in other words the practical generation of *iid.* point processes. Thus a higher estimation of the probability means a too low number of iterations, *i.e.* that point processes are moving *too fast*, which will directly produce an overestimation of the quantile.

As for the probability estimator we now intend to validate formula (A.11) on the effective computing time. The results are presented in Figure A.4b and show a good agreement with the formula apart from the *extreme* cases ($N_{\text{batch}} \leq 5$). This is because the smaller the population, the higher the intermediate failure level (cf. Figure A.4d) and so the greater the number of transitions to stop the algorithm.

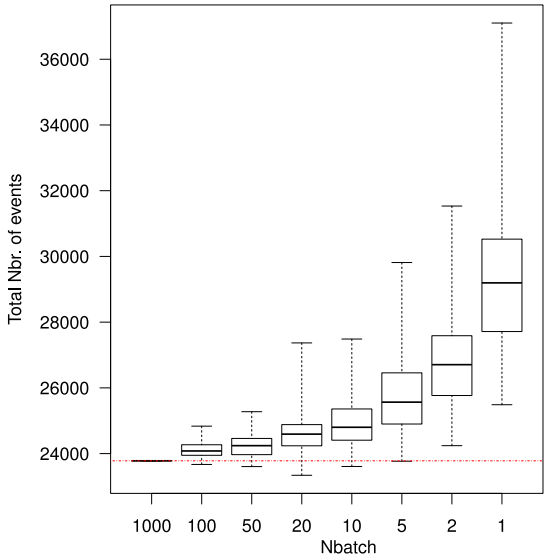
We also check the total number of events of the superposed process as this should be ideally equal to the targeted one's $m = \lceil -N \log p \rceil$, and in practice as close as possible. Especially we have accepted here to take a risk $\alpha = 5\%$ (see Eq. A.3) not to have enough



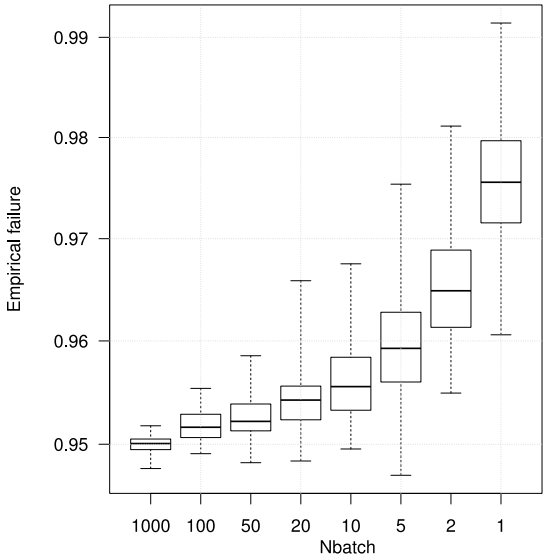
(a) Quantile estimates. The dashed line stands for the theoretical value.



(b) Effective computing time. The stars give the theoretical values calculated with Eq. (A.10).



(c) Total number of events generated by the algorithm. The dashed line stands for the targeted number of events.



(d) Distribution of $q_{\max}$ over the 100 simulations. $q_{\max}$ is the farthest state reached after the first pass of Algorithm 21.

Figure A.4: Statistics on quantile estimator with Algorithm 21 over 100 simulations, whiskers extending to the extreme values.

events at the end of the algorithm. In Figure A.4c we can see that in some cases the algorithm actually did not produce enough events. The number of "too short" algorithms are 5, 4, 3, 3 and 1 for $N_{\text{batch}} = 100, 50, 20, 10$ and 5 respectively, and 0 elsewhere. On a total of 100 simulations this is in good agreement with the parameter α set to 5%.

A.5 Conclusion on parallel implementation

On the one hand we have found an optimal value for p_0 for classical Multilevel Splitting algorithms in terms of computational time against variance of the estimator (considering that the only important operation is a call to the limit-state function, see Eq. (A.5)). On the other hand we see that our approach gives always a better result than the Multilevel Splitting method with the optimal p_0. In other words, our approach allows for taking $p_0 \to 1$ while keeping the parallel computation. Thus this is the optimal way of computing Multilevel Splitting methods.

Furthermore, with standard values of $\log 1/p \approx 10^1$, $n_c \approx 10^2$ and $\delta^2 = 10^{-2}$, we get $(\log p)^2/(n_c\delta^2) \approx 10^2$, which means that $\mathrm{E}\left[t_{\mathrm{par}}\right]$ is multiplied by ≈ 1.1 if one considers only the embarrassingly parallel implementation (see Eqs. A.1 and A.10). As sequential parallelisation and dynamic allocation of the computational resources can indeed increase the computational time consequently, this result is of great interest because it shows that without losing too much time the implementation of Multilevel Splitting methods can be a lot easier. Finally, we point out the fact that these methods have been implemented in the `R` package `mistral` [Bousquet et al., 2015]. This allows for a direct use of parallel computing for probability and quantile estimation without specific knowledge on parallel computing.

Bibliography

Michael Amrein and Hans R Künsch. A variant of importance splitting for rare event estimation: Fixed number of successes. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 2011.

Barry C Arnold, Narayanaswamy Balakrishnan, and Haikady Navada Nagaraja. *A first course in order statistics*, volume 54. Siam, 1992.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Siu-Kui Au and James L Beck. Estimation of small failure probabilities in high dimensions by subset simulation. *Probabilistic Engineering Mechanics*, 16(4):263–277, 2001.

François Bachoc. *Parametric estimation of covariance function in Gaussian-process based Kriging models. Application to uncertainty quantification for computer experiments.* PhD thesis, Université Paris-Diderot-Paris VII, 2013.

Mathieu Balesdent, Jerome Morio, and Julien Marzat. Kriging-based adaptive importance sampling algorithms for rare event estimation. *Structural Safety*, 44:1–10, 2013.

Anirban Basudhar and Samy Missoum. An improved adaptive sampling scheme for the construction of explicit boundaries. *Structural and Multidisciplinary Optimization*, 42 (4):517–529, 2010.

Julien Bect, David Ginsbourger, Ling Li, Victor Picheny, and Emmanuel Vazquez. Sequential design of computer experiments for the estimation of a probability of failure. *Statistics and Computing*, 22(3):773–793, 2012.

Julien Bect, Roman Sueur, Alexis Gérossier, Loïc Mongellaz, Sébastien Petit, and Emmanuel Vazquez. Échantillonnage préférentiel et méta-modèles : méthodes bayésiennes optimale et défensive. In *47èmes Journées de Statistique de la SFdS - JdS 2015*, Lille, France, June 2015. URL `https://hal-supelec.archives-ouvertes.fr/hal-01163632`.

Julien Bect, Ling Li, and Emmanuel Vazquez. Bayesian subset simulation. *arXiv preprint arXiv:1601.02557*, 2016.

Jan Beirlant, Frederico Caeiro, and Ivette M Gomes. An overview and open research topics in statistics of univariate extremes. *REVSTAT-Statistical Journal*, 10(1):1–31, 2012.

José M Bernardo, M J Bayarri, James O Berger, Philip A Dawid, and David Heckerman. *Bayesian Statistics 9*. Oxford University Press, 2011.

Alexandros Beskos, Ajay Jasra, Nikolas Kantas, and Alexandre Thiery. On the convergence of adaptive sequential monte carlo methods. *Annals of Applied Probability*, 26(2), 2016.

Ivona Bezáková, Daniel Štefankovic, Vijay V Vazirani, and Eric Vigoda. Accelerating simulated annealing for the permanent and combinatorial counting problems. *SIAM Journal on Computing*, 37(5):1429–1454, 2008.

Barron J Bichon, Michael S Eldred, Laura Painton Swiler, Sandaran Mahadevan, and John M McFarland. Efficient global reliability analysis for nonlinear implicit performance functions. *AIAA journal*, 46(10):2459–2468, 2008.

Géraud Blatman. *Adaptive sparse polynomial chaos expansions for uncertainty propagation and sensitivity analysis*. PhD thesis, Clermont-Ferrand 2, 2009.

Géraud Blatman and Bruno Sudret. An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. *Probabilistic Engineering Mechanics*, 25(2):183–197, 2010.

Géraud Blatman and Bruno Sudret. Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of Computational Physics*, 230(6):2345–2367, 2011.

Zdravko I Botev and Dirk P Kroese. An efficient algorithm for rare-event probability estimation, combinatorial optimization, and counting. *Methodology and Computing in Applied Probability*, 10(4):471–505, 2008.

Zdravko I Botev and Dirk P Kroese. Efficient monte carlo simulation via the generalized splitting method. *Statistics and Computing*, 22(1):1–16, 2012.

Jean-Marc Bourinet. Rare-event probability estimation with adaptive support vector regression surrogates. *Reliability Engineering & System Safety*, 150:210–221, 2016.

Jean-Marc Bourinet, François Deheeger, and Maurice Lemaire. Assessing small failure probabilities by combined subset simulation and support vector machines. *Structural Safety*, 33(6):343–353, 2011.

Nicolas Bousquet, Gilles Defaux, Bertrand Iooss, Vincent Moutoussamy, and Clement Walter. *mistral: Methods in Structural Reliability*, 2015. URL `http://CRAN.R-project.org/package=mistral`. R package version 2.0.2.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Charles-Édouard Bréhier. Large deviations principle for the adaptive multilevel splitting algorithm in an idealized setting. *ALEA, Latin American Journal of Probability and Mathematical Statistics*, 12:717–742, 2015.

Charles-Édouard Bréhier, Maxime Gazeau, Ludovic Goudenege, Tony Lelièvre, and Mathias Rousset. Unbiasedness of some generalized Adaptive Multilevel Splitting algorithms. *Annals of Applied Probability*, To Appear, 2015a.

Charles-Édouard Bréhier, Ludovic Goudenege, and Loic Tudela. Central limit theorem for adaptative multilevel splitting estimators in an idealized setting. *Proceedings of the 14th MCQMC conference*, 2015b.

Charles-Édouard Bréhier, Tony Lelièvre, and Mathias Rousset. Analysis of adaptive multilevel splitting algorithms in an idealized case. *ESAIM: Probability and Statistics*, 19:361–394, 2015c.

Karl Breitung. Asymptotic approximations for multinormal integrals. *Journal of Engineering Mechanics*, 110(3):357–366, 1984.

Brendon J Brewer, Livia B Pártay, and Gábor Csányi. Diffusive nested sampling. *Statistics and Computing*, 21(4):649–656, 2011.

Francesco Cadini, Francisco Santos, and Enrico Zio. An improved adaptive kriging-based importance technique for sampling multiple failure regions of low probability. *Reliability Engineering & System Safety*, 131:109–117, 2014.

Frédéric Cérou and Arnaud Guyader. Adaptive multilevel splitting for rare event analysis. *Stochastic Analysis and Applications*, 25(2):417–443, 2007.

Frederic Cérou and Arnaud Guyader. Fluctuation analysis of adaptive multilevel splitting. *Annals of Applied Probability*, To Appear, 2016.

Frédéric Cérou, Pierre Del Moral, Teddy Furon, Arnaud Guyader, et al. Rare event simulation for a static distribution. 2009.

Frédéric Cérou, Arnaud Guyader, Reuven Rubinstein, and Radislav Vaisman. On the use of smoothing to improve the performance of the splitting method. *Stochastic Models*, 27 (4):629–650, 2011.

Frédéric Cérou, Pierre Del Moral, Teddy Furon, and Arnaud Guyader. Sequential Monte Carlo for rare event estimation. *Statistics and Computing*, 22(3):795–808, 2012.

John M Chambers and Trevor Hastie. *Statistical Models in S*. Wadsworth & Brooks/Cole computer science series. Wadsworth & Brooks/Cole Advanced Books & Software, 1992. ISBN 9780534167646. URL `https://books.google.fr/books?id=uyfvAAAAMAAJ`.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Pierre Chauvet. *Aide-mémoire de géostatistique linéaire*. Presses des MINES, 2008.

Clément Chevalier. *Fast uncertainty reduction strategies relying on Gaussian process models*. PhD thesis, University of Bern, 2013.

Clément Chevalier, Julien Bect, David Ginsbourger, Emmanuel Vazquez, Victor Picheny, and Yann Richet. Fast parallel kriging-based stepwise uncertainty reduction with application to the identification of an excursion set. *Technometrics*, 56(4):455–465, 2014. doi: 10.1080/00401706.2013.860918. URL `http://dx.doi.org/10.1080/00401706.2013.860918`.

Jean-Paul Chiles and Pierre Delfiner. *Geostatistics: modeling spatial uncertainty*, volume 497. John Wiley & Sons, 2009.

Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3): 539–552, 2002. doi: 10.1093/biomet/89.3.539. URL `http://biomet.oxfordjournals.org/content/89/3/539.abstract`.

Nicolas Chopin and Christian P. Robert. Properties of nested sampling. *Biometrika*, 2010. doi: 10.1093/biomet/asq021. URL `http://biomet.oxfordjournals.org/content/early/2010/06/01/biomet.asq021.abstract`.

Robert M Corless, Gaston H Gonnet, David EG Hare, David J Jeffrey, and Donald E Knuth. On the lambertw function. *Advances in Computational mathematics*, 5(1): 329–359, 1996.

Guillaume Damblin, Mathieu Couplet, and Bertrand Iooss. Numerical studies of space-filling designs: optimization of latin hypercube samples and subprojection properties. *Journal of Simulation*, 7(4):276–289, 2013.

Anirban DasGupta. *Probability for statistics and machine learning: fundamentals and advanced topics*. Springer Science & Business Media, 2011.

G Defaux and P Evrard. Probabilistic analysis of a containment vessel subjected to dynamic pressure loading using surrogate models. In *Safety, Reliability, Risk and Life-Cycle Performance of Structures and Infrastructures*, pages 3203–3210. CRC Press, jan 2014. ISBN 978-1-138-00086-5. doi: doi:10.1201/b16387-463. URL `http://dx.doi.org/10.1201/b16387-463`.

François Deheeger. *Couplage mécano-fiabiliste: 2 SMART-méthodologie d'apprentissage stochastique en fiabilité*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, 2008.

Pierre Del Moral. *Feynman-Kac Formulae*. Springer, 2004.

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

Jean-François Delmas and Benjamin Jourdain. *Modèles aléatoires: applications aux sciences de l'ingénieur et du vivant*, volume 57. Springer Science & Business Media, 2006.

George Deodatis, Bruce R Ellingwood, and Dan M Frangopol. *Safety, reliability, risk and life-cycle performance of structures and infrastructures*. CRC Press, 2014.

Armen Der Kiureghian and Taleen Dakessian. Multiple design points in first and second-order reliability. *Structural Safety*, 20(1):37–49, 1998.

Persi Diaconis and Susan Holmes. *Three examples of Monte-Carlo Markov chains: at the interface between statistical computing, computer science, and statistical mechanics*. Springer, 1995.

Ove Ditlevsen and Henrik O Madsen. *Structural reliability methods*, volume 178. Wiley New York, 1996.

Vincent Dubourg. *Adaptive surrogate models for reliability analysis and reliability-based design optimization*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, 2011.

Vincent Dubourg, François Deheeger, and Bruno Sudret. Metamodel-based importance sampling for the simulation of rare events. *Applications of Statistics and Probability in Civil Engineering*, 26:192, 2011.

Vincent Dubourg, B Sudret, and F Deheeger. Metamodel-based importance sampling for structural reliability analysis. *Probabilistic Engineering Mechanics*, 33:47–57, 2013.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

Delphine Dupuy, Jessica Franco, and Xavier Bay. Planification d'expériences numériques à partir du processus ponctuel de strauss. In *12ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2011)*, pages submission–462, 2011.

Delphine Dupuy, Céline Helbert, and Jessica Franco. Dicedesign and diceeval: Two r packages for design and analysis of computer experiments. *Journal of Statistical Software*, 65(11):1–38, 2015.

B Echard, N Gayton, and M Lemaire. Ak-mcs: An active learning reliability method combining kriging and monte carlo simulation. *Structural Safety*, 33(2):145–154, 2011.

B Echard, Nicolas Gayton, Maurice Lemaire, and N Relun. A combined importance sampling and kriging reliability method for small failure probabilities with time-demanding numerical models. *Reliability Engineering & System Safety*, 111:232–240, 2013.

P Embrechts, C Klüppelberg, and T Mikosch. *Modelling extremal events: for insurance and finance*, volume 33. Springer, 1997.

Oliver G Ernst, Antje Mugler, Hans-Jörg Starkloff, and Elisabeth Ullmann. On the convergence of generalized polynomial chaos expansions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(02):317–339, 2012.

Michael J Evans. Discussion of nested sampling for bayesian computations by john skilling. *Bayesian Statistics*, 8:491–524, 2007.

Kai-Tai Fang, Runze Li, and Agus Sudjianto. *Design and modeling for computer experiments*. CRC Press, 2005.

William Fauriat and Nicolas Gayton. Ak-sys: an adaptation of the ak-mcs method for system reliability. *Reliability Engineering & System Safety*, 123:137–144, 2014.

JC Ferreira and VA Menegatto. Eigenvalues of integral operators defined by smooth positive definite kernels. *Integral Equations and Operator Theory*, 64(1):61–81, 2009.

Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

Alexandre Froda. *Sur la Distribution des Propriétés de Voisinage des Fonctions de Variables Réelles*. PhD thesis, Université de Paris, 1929.

Marnix Garvels. *The splitting method in rare event simulation*. Universiteit Twente, 2000.

Roger Ghanem and Pol D Spanos. Polynomial chaos in stochastic finite elements. *Journal of Applied Mechanics*, 57(1):197–202, 1990.

Roger Ghanem, David Higdon, and Houman Owhadi. *Handbook of Uncertainty Quantification*. Springer International Publishing, 2017.

Dan M Ghiocel and Roger G Ghanem. Stochastic finite-element analysis of seismic soil-structure interaction. *Journal of Engineering Mechanics*, 128(1):66–77, 2002.

Michael B Giles. Multilevel monte carlo path simulation. *Operations Research*, 56(3): 607–617, 2008.

Paul Glasserman, Philip Heidelberger, Perwez Shahabuddin, and Tim Zajic. Splitting for rare event simulation: analysis of simple cases. In *Proceedings of the 28th conference on Winter simulation*, pages 302–308. IEEE Computer Society, 1996.

Paul Glasserman, Philip Heidelberger, Perwez Shahabuddin, and Tim Zajic. A large deviations perspective on the efficiency of multilevel splitting. *Automatic Control, IEEE Transactions on*, 43(12):1666–1679, 1998.

Paul Glasserman, Philip Heidelberger, Perwez Shahabuddin, and Tim Zajic. Multilevel splitting for estimating rare event probabilities. *Operations Research*, 47(4):585–600, 1999.

Peter W Glynn and Donald L Iglehart. Importance sampling for stochastic simulations. *Management Science*, 35(11):1367–1392, 1989.

Peter W Glynn and Ward Whitt. The asymptotic efficiency of simulation estimators. *Operations Research*, 40(3):505–520, 1992.

Arnaud Guyader, Nicolas Hengartner, and Eric Matzner-Løber. Simulation and estimation of extreme quantiles and extreme probabilities. *Applied Mathematics & Optimization*, 64(2):171–196, 2011.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

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

Terje Haukaas, editor. *Proceedings of the 12th International Conference on Applications of Statistics and Probability in Civil Engineering*, 2015. University of British Columbia. ISBN 978-0-88865-245-4.

Jonathan B Hill. Robust estimation for average treatment effects. *Available at SSRN 2260573*, 2013. URL `http://dx.doi.org/10.2139/ssrn.2260573`.

Xiaoxu Huang, Jianqiao Chen, and Hongping Zhu. Assessing small failure probabilities by ak–ss: An active learning method combining kriging and subset simulation. *Structural Safety*, 59:86–95, 2016.

Mark Huber and Sarah Schott. Using tpa for bayesian inference. *Bayesian Statistics 9*, 9: 257–282, 2011.

Mark Huber, Sarah Schott, et al. Random construction of interpolating sets for high-dimensional integration. *Journal of Applied Probability*, 51(1):92–105, 2014.

Jorge Eduardo Hurtado. *Structural reliability: statistical learning perspectives*, volume 17. Springer Science & Business Media, 2013.

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

Bertrand Iooss and Paul Lemaître. A review on global sensitivity analysis methods. In *Uncertainty Management in Simulation-Optimization of Complex Systems*, pages 101–122. Springer, 2015.

Pierre E Jacob, Alexandre H Thiery, et al. On nonnegative unbiased estimators. *The Annals of Statistics*, 43(2):769–784, 2015.

Joachim Johansson. Estimating the mean of heavy-tailed distributions. *Extremes*, 6(2): 91–109, 2003.

Mark E Johnson, Leslie M Moore, and Donald Ylvisaker. Minimax and maximin distance designs. *Journal of statistical planning and inference*, 26(2):131–148, 1990.

Sandeep Juneja and Perwez Shahabuddin. Rare-event simulation techniques: an introduction and recent advances. *Handbooks in operations research and management science*, 13:291–350, 2006.

H Kahn and T E Harris. Estimation of particle transmission by random sampling. *National Bureau of Standards applied mathematics series*, 12:27–30, 1951.

Andreas Keese and Hermann G Matthies. Hierarchical parallelisation for the solution of stochastic finite element equations. *Computers & Structures*, 83(14):1033–1047, 2005.

Charles R Keeton. On statistical uncertainty in nested sampling. *Monthly Notices of the Royal Astronomical Society*, 414(2):1418–1426, 2011.

John Frank Charles Kingman. *Poisson processes*, volume 3. Clarendon Press, 1992.

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

Jack PC Kleijnen. *Design and analysis of simulation experiments*, volume 20. Springer, 2008.

Hasan Ugur Koyluoglu and Soren RK Nielsen. New approximations for sorm integrals. *Structural Safety*, 13(4):235–246, 1994.

Daniel G Krige. *A statistical approach to some mine valuation and allied problems on the Witwatersrand*. PhD thesis, 1951.

François Le Gland. Combined use of importance weights and resampling weights in sequential monte carlo methods. In *ESAIM: Proceedings*, volume 19, pages 85–100. EDP Sciences, 2007.

Loic Le Gratiet. *Multi-fidelity Gaussian process regression for computer experiments*. PhD thesis, Université Paris-Diderot-Paris VII, 2013.

Olivier P Le Maître, Matthew T Reagan, Habib N Najm, Roger G Ghanem, and Omar M Knio. A stochastic projection method for fluid flow: Ii. random process. *Journal of computational Physics*, 181(1):9–44, 2002.

Pierre L'Ecuyer, Jose H Blanchet, Bruno Tuffin, and Peter W Glynn. Asymptotic robustness of estimators in rare-event simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 20(1):6, 2010.

Ling Li, Julien Bect, and Emmanuel Vazquez. Bayesian Subset Simulation: a kriging-based subset simulation algorithm for the estimation of small probabilities of failure. In *11th International Probabilistic Assessment and Management Conference (PSAM11) and The Annual European Safety and Reliability Conference (ESREL 2012)*, pages CD–ROM Proceedings (10 pages), Helsinki, Finland, June 2012. URL `https://hal-supelec.archives-ouvertes.fr/hal-00715316`.

Jason L Loeppky, Jerome Sacks, and William J Welch. Choosing the sample size of a computer experiment: A practical guide. *Technometrics*, 2012.

Stefano Martiniani, Jacob D Stevenson, David J Wales, and Daan Frenkel. Superposition enhanced nested sampling. *Physical Review X*, 4(3):031034, 2014.

Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963.

Georges Matheron. Le krigeage universel. 1969.

Dan McLeish. A general method for debiasing a monte carlo estimator. *Monte Carlo Methods and Applications*, 2011.

J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 209(441-458):415–446, 1909. ISSN 0264-3952. doi: 10.1098/rsta.1909.0016. URL `http://rsta.royalsocietypublishing.org/content/209/441-458/415`.

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

Nicholas Metropolis and Stanislaw Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2015. URL `http://CRAN.R-project.org/package=e1071`. R package version 1.6-7.

Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

Thomas Most. An adaptive response surface approach for structural reliability analyses based on support vector machines. In *Proceedings of the Eleventh International Conference on Civil, Structural and Environmental Engineering Computing, BHV Topping*, 2007.

Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

Pia Mukherjee, David Parkinson, and Andrew R Liddle. A nested sampling algorithm for cosmological model selection. *The Astrophysical Journal Letters*, 638(2):L51, 2006.

Abdelhakim Necir, Abdelaziz Rassoul, and Ričardas Zitikis. Estimating the conditional tail expectation in the case of heavy-tailed losses. *Journal of Probability and Statistics*, 2010, 2010. URL `http://dx.doi.org/10.1155/2010/596839`.

Art Owen and Yi Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143, 2000.

Liang Peng. Estimating the mean of a heavy tailed distribution. *Statistics & Probability Letters*, 52(3):255–264, 2001.

Victor Picheny, David Ginsbourger, Olivier Roustant, Raphael T Haftka, and Nam-Ho Kim. Adaptive designs of experiments for accurate approximation of a target region. *Journal of Mechanical Design*, 132(7):071008, 2010.

Luc Pronzato and Werner G Müller. Design of computer experiments: space filling and beyond. *Statistics and Computing*, 22(3):681–701, 2012.

James Gary Propp and David Bruce Wilson. Exact sampling with coupled markov chains and applications to statistical mechanics. *Random structures and Algorithms*, 9(1-2): 223–252, 1996.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL `https://www.R-project.org/`.

Carl Edward Rasmussen and Christopher KI Williams. Gaussian processes for machine learning. *the MIT Press*, 2006.

Sidney I Resnick. *Extreme values, regular variation and point processes*. Springer, 2013.

Chang-han Rhee and Peter W Glynn. Unbiased estimation with square root convergence for sde models. *Operations Research*, 63(5):1026–1043, 2015.

Christian P Robert and George Casella. *Monte Carlo statistical methods*. Springer, 2004.

Gareth Roberts. Comments on "Using TPA for Bayesian inference" by Huber, M. and Schott, S. In J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, Smith A. F. M., and M. West, editors, *Bayesian Statistics 9*, pages 257–282. Oxford University Press, 2011.

Claudio M Rocco and José Alı Moreno. Fast monte carlo reliability evaluation using support vector machine. *Reliability Engineering & System Safety*, 76(3):237–243, 2002.

Sheldon Ross. *Simulation*. Academic Press, fifth edition, 2013. ISBN 978-0-12-415825-2.

Olivier Roustant, David Ginsbourger, and Yves Deville. DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of Statistical Software*, 51(1):1–55, 2012. URL `http://www.jstatsoft.org/v51/i01/`.

Gerardo Rubino, Bruno Tuffin, et al. *Rare event simulation using Monte Carlo methods*, volume 73. Wiley Online Library, 2009.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

Reuven Rubinstein. Entropy and cloning methods for combinatorial optimization, sampling and counting using the gibbs sampler. In *Information Theory and Statistical Learning*, pages 385–434. Springer, 2009a.

Reuven Rubinstein. The gibbs cloner for combinatorial optimization, counting and sampling. *Methodology and Computing in Applied Probability*, 11(4):491–549, 2009b.

Reuven Rubinstein. Randomized algorithms with splitting: Why the classic randomized algorithms do not work and how to make them work. *Methodology and Computing in Applied Probability*, 12(1):1–50, 2010.

Reuven Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*, volume 707. John Wiley & Sons, 2011.

Reuven Rubinstein, Andrey Dolgin, and Radislav Vaisman. The splitting method for decision making. *Communications in Statistics-Simulation and Computation*, 41(6): 905–921, 2012.

Jerome Sacks, Susannah B Schiller, and William J Welch. Designs for computer experiments. *Technometrics*, 31(1):41–47, 1989.

Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

Thomas J Santner, Brian J Williams, and William I Notz. *The design and analysis of computer experiments*. Springer, 2003.

R Schöbi, B Sudret, and S Marelli. Rare event estimation using polynomial-chaos kriging. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, page D4016002, 2016.

Eric Simonnet. Combinatorial analysis of the adaptive last particle method. *Statistics and Computing*, 26(1-2):211–230, 2016.

John Skilling. Nested sampling for general bayesian computation. *Bayesian Analysis*, 1(4): 833–859, 2006.

Adrian Smith, Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2013.

Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

Christian Soize and Roger Ghanem. Physical systems with random uncertainties: chaos representations with arbitrary probability measure. *SIAM Journal on Scientific Computing*, 26(2):395–410, 2004.

Daniel Straub and Iason Papaioannou. Bayesian updating with structural reliability methods. *Journal of Engineering Mechanics*, 141(3):04014134, 2014.

Daniel Straub, Iason Papaioannou, and Wolfgang Betz. Bayesian analysis of rare events. *Journal of Computational Physics*, 314:538–556, 2016.

Bruno Sudret. Meta-models for structural reliability and uncertainty quantification. In K. Phoon, M. Beer, S Quek, and S Pang, editors, *Proc. 5th Asian-Pacific Symp. Struct. Reliab. (APSSRA 2012)*, pages 53–76, 2012.

Vladimir Vapnik. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963.

Vladimir Vapnik and Alexey Chervonenkis. A note on one class of perceptrons. *Automation and remote control*, 25(1), 1964.

Christelle Vergé, Cyrille Dubarry, Pierre Del Moral, and Eric Moulines. On parallel implementation of sequential monte carlo methods: the island particle model. *Statistics and Computing*, pages 1–18, 2013.

Manuel Villén-Altamirano and Jose Villén-Altamirano. Restart: A method for accelerating rare event simulations. *Analysis*, 3:3, 1991.

PH Waarts. *Structural reliability using finite element methods: an appraisal of directional adaptive response surface sampling (DARS)*. PhD thesis, Ph. D. Thesis, 2000.

Hans Wackernagel. *Multivariate geostatistics: an introduction with applications*. Springer Science & Business Media, 2013.

Grace Wahba et al. Support vector machines, reproducing kernel hilbert spaces and the randomized gacv. *Advances in Kernel Methods-Support Vector Learning*, 6:69–87, 1999.

Clément Walter. Moving particles: A parallel optimal multilevel splitting method with application in quantiles estimation and meta-model based algorithms. *Structural Safety*, 55 (0):10 – 25, 2015a. ISSN 0167-4730. doi: http://dx.doi.org/10.1016/j.strusafe.2015.02.002. URL `http://www.sciencedirect.com/science/article/pii/S0167473015000156`.

Clément Walter. Point process-based monte carlo estimation. *Statistics and Computing*, To Appear, 2015b. doi: 10.1007/s11222-015-9617-y. URL `http://dx.doi.org/10.1007/s11222-015-9617-y`.

Clément Walter. Rare event simulation and splitting for discontinuous random variables. *ESAIM: PS*, 19:794–811, 2015c. doi: 10.1051/ps/2015017. URL `http://dx.doi.org/10.1051/ps/2015017`.

Clément Walter and Gilles Defaux. Rare event simulation: a point process interpretation with application in probability and quantile estimation. *Proceedings of the 12th International Conference on Applications of Statistics and Probability*, 2015.

Steve Weston. *doMPI: Provides a parallel backend for the %dopar% function using the Rmpi package*, 2015a. URL `http://CRAN.R-project.org/package=doParallel`. R package version 1.0.10.

Steve Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, 2015b. URL `http://CRAN.R-project.org/package=doParallel`. R package version 1.0.10.

Steve Weston. *foreach: Provides Foreach Looping Construct for R*, 2015c. URL `http://CRAN.R-project.org/package=foreach`. R package version 1.4.3.

Dongbin Xiu. Fast numerical methods for stochastic computations: a review. *Communications in computational physics*, 5(2-4):242–272, 2009.

Yuan Xiukai, Lu Zhenzhou, and Lu Yuanbo. Support vector machine response surface method based on fast markov chain simulation. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 4, pages 279–282. IEEE, 2009.

Yanling Zhang and A Der Kiureghian. Two improved algorithms for reliability analysis. In *Reliability and optimization of structural systems*, pages 297–304. Springer, 1995.

Résumé

Cette thèse est une contribution à la problématique de la simulation d'événements rares. A partir de l'étude des méthodes de *Splitting*, un nouveau cadre théorique est développé, indépendant de tout algorithme. Ce cadre, basé sur la définition d'un processus ponctuel associé à toute variable aléatoire réelle, permet de définir des estimateurs de probabilités, quantiles et moments sans aucune hypothèse sur la variable aléatoire. Le caractère artificiel du *Splitting* (sélection de seuils) disparaît et l'estimateur de la probabilité de dépasser un seuil est en fait un estimateur de la fonction de répartition jusqu'au seuil considéré. De plus, les estimateurs sont basés sur des processus ponctuels *iid.* et permettent donc l'utilisation de machine de calcul massivement parallèle. Des algorithmes pratiques sont ainsi également proposés.

Enfin l'utilisation de *métamodèles* est parfois nécessaire à cause d'un temps de calcul toujours trop important. Le cas de la modélisation par processus aléatoire est abordé. L'approche par processus ponctuel permet une estimation simplifiée de l'espérance et de la variance conditionnelles de la variable aléaoire résultante et définit un nouveau critère d'enrichissement SUR adapté aux événements rares.

Mots clefs: Événements rares ∗ Splitting ∗ Subset Simulation ∗ Nested sampling ∗ Calcul parallèle ∗ Analyse de fiabilité ∗ Krigeage ∗ Stepwise Uncertainty Reduction

Abstract

This thesis address the issue of extreme event simulation. From a original understanding of the *Splitting* methods, a new theoretical framework is proposed, regardless of any algorithm. This framework is based on a point process associated with any real-valued random variable and lets defined probability, quantile and moment estimators without any hypothesis on this random variable. The *artificial* selection of threshold in *Splitting* vanishes and the estimator of the probability of exceeding a threshold is indeed an estimator of the whole cumulative distribution function until the given threshold. These estimators are based on the simulation of *iid.* replicas of the point process. So they allow for the use of massively parallel computer cluster. Suitable practical algorithms are thus proposed.

Finally it can happen that these advanced statistics still require too much samples. In this context the computer code is considered as a random process with known distribution. The point process framework lets handle this additional source of uncertainty and estimate easily the conditional expectation and variance of the resulting random variable. It also defines new SUR enrichment criteria designed for extreme event probability estimation.

Keywords: Rare events ∗ Splitting methods ∗ Subset Simulation ∗ Nested sampling ∗ Parallel algorithms ∗ Reliability analysis ∗ Kriging ∗ Stepwise Uncertainty Reduction