

Simulating a Random Vector Conditionally to a Subvariety: A Generic Dichotomous Approach

Frédéric Dambreville ^a

ONERA/DTIS, Université Paris Saclay, F-91123 Palaiseau Cedex, France

Keywords: Rare Event Simulation, Subvariety, Interval Analysis, Black-box Optimization.

Abstract: The problem of sampling a random vector conditionally to a subvariety within a box (actually, a small volume around the subvariety) is addressed. The approach is generic, in the sense that the subvariety may be defined by an isosurface related to any (computable) continuous function. Our approach is based on a dichotomous method. As a result, the sampling process is straightforward, accurate and avoids the use of Markov chain Monte Carlo. Our implementation relies on the evaluation of the matching with the subvariety at each dichotomy step. By using interval analysis techniques for evaluating the matching, our method has been applied up to dimension 11. Perspectives are evoked for improving the sampling efficiency on higher dimensions. An example of application of this simulation technique to black-box function optimization is detailed.

1 INTRODUCTION

The paper addresses the issue of sampling a given law defined on an n -dimension box conditionally to a subvariety of this box. Given a random vector \mathbf{X} of density $f_{\mathbf{X}}$ defined on \mathbb{R}^n , given box:

$$[\mathbf{b}] \triangleq \prod_{k=1}^n [b_k^-, b_k^+] \triangleq [b_1^-, b_1^+] \times \cdots \times [b_n^-, b_n^+] \quad (1)$$


of \mathbb{R}^n , small box $[\epsilon] \triangleq \prod_{k=1}^m [\epsilon_k^-, \epsilon_k^+]$ around $\mathbf{0} \in \mathbb{R}^m$ with $-1 \ll \epsilon_k^-$ and $\epsilon_k^+ \ll 1$, and map $g : [\mathbf{b}] \rightarrow \mathbb{R}^m$, our objective is to sample the conditional vector $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$ ¹. When space dimension n and constraint dimension m increase, event $[g(\mathbf{X}) \in [\epsilon]]$ has very low probability. We are dealing here with a particular case of rare event simulation. Moreover, as $[g(\mathbf{X}) \in [\epsilon]]$ approximates a subvariety, it is foreseeable that conditional random vector $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$ is essentially and “continuously” multimodal.

In survey (Morio et al., 2014), Morio, Balesdent et al. have evaluated the advantages and drawbacks of various rare event sampling methods. At this point, a comment may be done. In the literature, rare events are generally modelled by a function of risk being above an acceptable threshold: a rare event is of the form $[h(\mathbf{X}) > \gamma]$, where h maps to \mathbb{R} and $P(h(\mathbf{X}) > \gamma)$ is a very small probability to

be evaluated. This formalism is quite general, and it is easy to rewrite event $[g(\mathbf{X}) \in [\epsilon]]$ under this form. However, formalism $[h(\mathbf{X}) > \gamma]$ suggests that the simulation of a rare event is tightly related the maximization of a function (function h). In practice, the maximizing set of a function is unimodal or somewhat multimodal. It is uncommon that this maximizing set is a subvariety.

Not surprisingly, many methods evaluated in (Morio et al., 2014) are working well when the rare event is unimodal or moderately multimodal. In cross-entropy method (Rubinstein and Kroese, 2004) for example, the multimodality has to be managed by the sampling law family. It is then possible, for example by a combination of EM algorithms and cross-entropy minimization, to sample a moderately multimodal rare event (Dambreville, 2015). In the case of conditional vector $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$, these approaches do not work properly.

Good candidates for our sampling problem are some non parametric rare event simulation methods (Morio et al., 2014; Cérou et al., 2012). Adaptive splitting techniques, as described in (Morio et al., 2014), are well suited for high dimensions and nonlinear systems. Although the method is designed for events of the form $[h(\mathbf{X}) > \gamma]$, it seems applicable to “subvariety” events of the form $[g(\mathbf{X}) \in [\epsilon]]$. However, as mentioned in (Morio et al., 2014), the approach needs an important simulation budget. In particular, the Markov transition kernel must be

^a  <https://orcid.org/0000-0002-1460-4126>

¹ $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$ is used as an abbreviation for $[\mathbf{X} | \mathbf{X} \in [\mathbf{b}] \ \& \ g(\mathbf{X}) \in [\epsilon]]$, knowing that $g : [\mathbf{b}] \rightarrow \mathbb{R}^m$

iterated several times during the resampling steps in order to get independent samples; the choice of the Markov transition kernel is crucial in design of the sampler. In our case, moreover, additional studies to take into account the subvariety structure can be beneficial. These arguments make the splitting technique promising for large-dimensional problems, but not necessarily for medium-dimensional problems.

This article promotes an alternative approach based on a dichotomous exploration in order to sample conditionally to a subvariety characterized by a known function. By design, this approach produces independent samples and avoids the phenomenon of sample impoverishment. These properties make it particularly well suited for an accurate approximation of a law conditionally to the subvariety. However, the method must fight the curse of dimensionality and we will be faced with two orthogonal dimensional problems: the exponential increase in sampling exploration and the degeneracy of particle weights. This paper presents an approach and its perspectives, as well as a generic and weakly parameterized algorithm, allowing good sampling performance for moderate dimensions (up to 11 for now) with balanced management of dimensional issues.

In this introduction, we should also mention a work, which is connex to our concern. In (Bui Quang et al., 2016), Musso, Bui Quang and Le Gland proposed to combine Laplace method and particle filtering in order to estimate a state from partial measures with small observation noise. Actually, a partial measure with small observation noise implies a conditioning of the law by an approximated constraint. However, the approach proposed in (Bui Quang et al., 2016) is essentially unimodal, although it is possible to address some level of multimodality by a mixture of law (Musso et al., 2016).

The paper is divided in four more parts. Section 2 introduces concepts of interval analysis and derives some first ideas for a generic subvariety conditional sampler. Section 3 deepens the intuitions introduced in section 2 and presents the working generic algorithm. Few discussions about performance and perspectives follow. Section 4 presents tests and analyses. An example of application to black-box optimization is presented. Section 5 concludes.

2 A NAIVE DICHOTOMOUS APPROACH FOR SAMPLING

Interval analysis is a performing and accurate tool for dealing with constraints. Moreover, the approach provides a precise control on the approximation er-

rors. Another interesting point is that intervals and boxes are relatively simple domains when dealing with probability distributions. For example, it is rather easy to evaluate the probability of a box being given the multivariate cumulative distribution function, especially for independent random variables. Boxes and intervals are also a good way to cope with poorly mastered random parameters. Thereby, Abdallah, Gning and Bonnifait proposed the box particle filter, mixing boxes into a particle filter, which resulted in better performance when dealing with non-white and biased measurement (Abdallah et al., 2008). Our work addresses a quite different issue. Essentially, boxes are not part of our data model, but we use the interval analysis as a tool and a guide for a dichotomous sampling on the subvariety.

2.1 About Interval Analysis

It is not the purpose of this section to perform a good introduction on interval analysis (Alefeld and Mayer, 2000; Jaulin et al., 2001), and neither do we introduce properly the notion of paver for set inversion. Nevertheless, we refer to some key concepts of interval analysis, which are inspiring ideas for this paper. On the other hand, we absolutely do not introduce the techniques of contractors (Chabert and Jaulin, 2009), but mention it as a possibility for improvement.

Operators & Functions Applied on Intervals.

First at all, let us introduce some notations:

- \mathbb{R} is the set of reals and x, y, z are real variables.
- $[x] \triangleq [x^-, x^+]$, $[y]$, $[z]$ are notations for intervals. Bold notations $\mathbf{x} \triangleq \prod_{k=1}^n [x_k] = \prod_{k=1}^n [x_k^-, x_k^+]$ and $\mathbf{x}[\triangleq \prod_{k=1}^n [x_k^-, x_k^+]$ are used for boxes and half-open boxes respectively.
- $[\mathbb{R}] \triangleq \{[x^-, x^+] : [x^-, x^+] \subset \mathbb{R}\}$ is the set of interval subsets of \mathbb{R} . $[\mathbb{R}^n] \triangleq \{\prod_{k=1}^n [x_k] : \forall k, [x_k] \in [\mathbb{R}]\}$ is the set of box subsets of \mathbb{R}^n .
- $\rho([x]) = x^+ - x^-$ is the length of $[x] \in [\mathbb{R}]$. $\rho(\mathbf{x}) = \max_{1 \leq k \leq n} \rho([x_k])$ is the length of $\mathbf{x} \in [\mathbb{R}^n]$.
- $g : \mathbb{R} \rightarrow \mathbb{R}$, $g_j : \mathbb{R}^k \rightarrow \mathbb{R}$, $h : \mathbb{R}^k \rightarrow \mathbb{R}^j$ are notations for (multivariate) real functions,
- $[g] : [\mathbb{R}] \rightarrow [\mathbb{R}]$, $[g_j] : [\mathbb{R}^k] \rightarrow [\mathbb{R}]$, $[h] : [\mathbb{R}^k] \rightarrow [\mathbb{R}^j]$ are notations for interval functions,

At this point, one should not confuse \mathbf{x} and \mathbf{x} , nor should be confused g and $[g]$, nor should be confused the notations $[g](\mathbf{x})$ and $g(\mathbf{x})$:

- $\mathbf{x} \in \mathbf{x}$ is equivalent to $\mathbf{x} \in \prod_{k=1}^n [x_k^-, x_k^+]$.

- $g([\mathbf{x}]) = \{g(\mathbf{x}) : \mathbf{x} \in [\mathbf{x}]\}$ is not the same as $[g]([\mathbf{x}])$ and is not even necessarily a box.

Nevertheless, we assume in this paper, that g and $[g]$ are related by the following properties:

- $g([\mathbf{x}]) \subset [g]([\mathbf{x}])$ for any $[\mathbf{x}] \in [\mathbb{R}^n]$. In particular, $[g]$ is minimal when $[g]([\mathbf{x}])$ is the minimal box containing $g([\mathbf{x}])$ as subset for all $[\mathbf{x}] \in [\mathbb{R}^n]$.
- $[g]([\mathbf{x}]) \subset [g]([\mathbf{y}])$ when $[\mathbf{x}] \subset [\mathbf{y}]$.
- If $\rho([\mathbf{x}])$ vanishes, then $\rho([g]([\mathbf{x}]))$ vanishes:

$$\rho(g([\mathbf{x}])) \xrightarrow{\rho([\mathbf{x}]) \rightarrow 0} 0 \quad (2)$$

These properties expresses that $[g]$ implies a bound on the error propagated by g , and this bound has good convergence behavior in regards to the error.

Before going further, let us consider how to build the interval functions on some common examples:

1. Reference functions, $g \in \{\ln, \exp, \sin, \cos, \cdot^n, \dots\}$, are continuous onto \mathbb{R} , so that $g([x]) \in [\mathbb{R}]$ for all $[x] \in [\mathbb{R}]$. Then, it is optimal to set $[g]([x]) = g([x])$ for all $[x] \in [\mathbb{R}]$. As a consequence, the interval functions are easily optimally implementable for most reference functions. Here are some incomplete examples of definitions:

$$[\ln]([x]) \triangleq [\ln(x^-), \ln(x^+)], \quad (3)$$

$$[x]^n \triangleq [x]^n = \begin{cases} [x_-^n, x_+^n] & \text{for } n > 0, \\ [x_+^n, x_-^n] & \text{for } n < 0, \end{cases} \quad (4)$$

$$[\cos][x] \triangleq \begin{cases} [\cos(x^-), \cos(x^+)] & \text{for } [x] \subset [-\pi, 0] \\ [\cos(x^+), \cos(x^-)] & \text{for } [x] \subset [0, \pi] \\ [\min(\cos(x^-), \cos(x^+)), 1] & \text{for } 0 \in [x] \subset [-\pi, \pi] \\ \text{etc.} \end{cases} \quad (5)$$

2. Minimal interval functions for classical operators $+$, \cdot , $-$, $/$ are also easily defined. For example:

$$[x][+][y] \triangleq [x] + [y] = [x^- + y^-, x^+ + y^+] \quad (6)$$

3. Function g defined by $g(\theta) = (\cos(\theta), \sin(\theta))$, is an example where $g([\theta]) \notin [\mathbb{R}^2]$. One would rather define $[g]([\theta]) = [\cos]([\theta]) \times [\sin]([\theta])$ which is a strict supset of $g([\theta])$ in general. By the way, this construction is an illustration on how the interval functions of reference are used to define complex interval functions straightforwardly.

There is no unicity in the construction of $[g]$. Let us consider the case of function $g : \theta \mapsto \cos^2 \theta + \sin^2 \theta$. Then, there are two obvious definitions for $[g]$.

1. By using the reference functions \cos , \sin , \cdot^2 and $+$ one may derive:

$$[g]([\theta]) = ([\cos]([\theta]))^2 + ([\sin]([\theta]))^2. \quad (7)$$

For example, we compute: $[g]([0, \frac{\pi}{2}]) = [0, 2]$ which is a bad error bound on theoretical value 1. Now, we also compute: $[g]([-\frac{1}{10}, \frac{1}{10}]) \simeq [0.99, 1.01]$ which is a tight error bound on 1. This example holds confirmation that $[g]([\theta])$ has a good behavior for small boxes $[\theta]$.

2. By noticing that $\cos^2 \theta + \sin^2 \theta = 1$, it is optimal to define $[g]([\theta]) = [1, 1]$.

Although approach 2 gives the best solution, in practice approach 1 is preferred since it is generic, it is based on already implemented functions of reference, and it provides a way to construct automatically $[g]$ without any specific knowledge.

Subpaving. As $[g]$ implies a bound on the error propagated by f with good convergence behavior, it may be used combined with a dichotomous process to produce a subpaving which efficiently approximates a set inversion $g^{-1}([\mathbf{y}])$. The example of figure 1 is kindly given by professor Jaulin. It shows a resulting subpaving which approximates a set inversion.

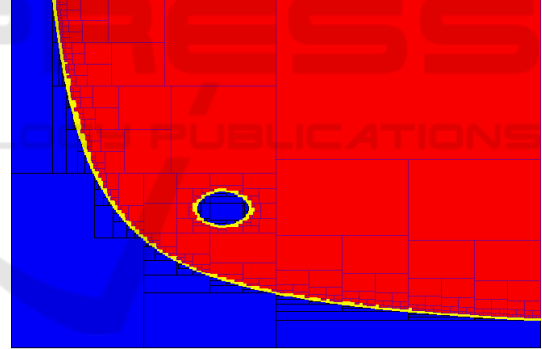


Figure 1: Example of set inversion.

The decomposition is clearly dichotomous. A bisection process is iterated starting from the main box; at each iteration, sub-boxes $[\mathbf{x}]$ are tested against the constraint $g([\mathbf{x}]) \subset [\mathbf{y}]$. Then three cases arise:

case a: $g([\mathbf{x}]) \subset [\mathbf{y}]$, then $[\mathbf{x}]$ is among *red* boxes, which constitute a *subpaving* of $g^{-1}([\mathbf{y}])$.

case b: $g([\mathbf{x}]) \cap [\mathbf{y}] = \emptyset$, then $[\mathbf{x}]$ is among *blue* boxes, which constitute a *subpaving* of $\mathbb{R}^n \setminus g^{-1}([\mathbf{y}])$.

case c: Otherwise, bisection has to be repeated on $[\mathbf{x}]$ until sufficient convergence (*yellow* color).

Property (2) plays a key role in the decomposition process, ensuring that cases **a** or **b** are finally achieved when sub-boxes $[\mathbf{x}]$ are sufficiently small and sufficiently far from the frontier of set $g^{-1}([\mathbf{y}])$.

2.2 Toward a Generic Sampling Method based on a Dichotomous Approach

Now, we settle the hypotheses mentioned in introduction. Let μ be Borel measure on \mathbb{R}^n . It is given:

- a random vector \mathbf{X} defined on \mathbb{R}^n characterized by a *bounded* density $f_{\mathbf{X}}$. Cumulative distribution function $F_{\mathbf{X}}$ of \mathbf{X} defined for all $\mathbf{x} \in \mathbb{R}^n$ by:

$$F_{\mathbf{X}}(\mathbf{x}) = P(\mathbf{X} \leq \mathbf{x}) = \int_{\mathbf{y} \leq \mathbf{x}} f_{\mathbf{X}}(\mathbf{y}) \mu(d\mathbf{y}), \quad (8)$$

is assumed to be easily computable,

- a box $[\mathbf{b}] \in [\mathbb{R}^n]$,
- a small box $[\epsilon] \in [\mathbb{R}^m]$,
- a continuous map $g : [\mathbf{b}] \rightarrow \mathbb{R}^m$ defined by composing functions and operators of reference,
- $[g]$ derived from g by composing the related interval functions and operators of reference.

Cumulative Functions and Boxes. We point out that it is easy to compute $P(\mathbf{X} \in [\mathbf{y}])$ or to sample $[\mathbf{X} | \mathbf{X} \in [\mathbf{y}]]$ when $F_{\mathbf{X}}$ is available. Thus, these features are taken for granted in this paper.

First, it is well known that $P(\mathbf{X} \in [\mathbf{y}])$ is literally computable from $F_{\mathbf{X}}$:

$$P(\mathbf{X} \in [\mathbf{y}]) = \sum_{\sigma \in \{-1, +1\}^n} F_{\mathbf{X}} \left(\left(y_k^{\text{sgn}(\sigma_k)} \right)_{1 \leq k \leq n} \right) \prod_{k=1}^n \sigma_k, \quad (9)$$

where $\text{sgn}(1) \triangleq +$ and $\text{sgn}(-1) \triangleq -$. When the components of \mathbf{X} are jointly independent, the computation is dramatically accelerated by factoring $F_{\mathbf{X}}$ and computing time becomes linear with dimension.

Sampling $[\mathbf{X} | \mathbf{X} \in [\mathbf{y}]]$ is easily done by:

```

1 for  $i \leftarrow 1$  to  $n$  do
2    $F_i \leftarrow F_{\mathbf{X}}(x_1, \dots, x_{i-1}, \cdot, \infty, \dots, \infty)$ 
3    $\theta \leftarrow \text{RandUniform}([F_i(y_i^-), F_i(y_i^+)])$ 
4    $x_i \leftarrow F_i^{-1}(\theta)$ 
5 end
6 return  $x_{1:n}$ 

```

The principle is to iteratively apply the *inverse transform sampling method* to F_i , the marginal in x_i of the cumulative function conditionally to the already sampled components x_1, \dots, x_{i-1} .

Subpaving based Sampling. Now assume that set $g^{-1}([\epsilon])$ has been approximated (by default) by a subpaving as shown in figure 1. In other words, there is $\mathbb{P} \subset [\mathbb{R}^n]$ such that:

- For all $[\mathbf{x}], [\mathbf{y}] \in \mathbb{P}$, boxes $[\mathbf{x}]$ and $[\mathbf{y}]$ are disjoint except possibly on the edges,
- $g^{-1}([\epsilon]) \simeq \sqcup_{\mathbb{P}} \subseteq g^{-1}([\epsilon])$, where $\sqcup_{\mathbb{P}} \triangleq \bigsqcup_{[\mathbf{x}] \in \mathbb{P}} [\mathbf{x}]$.

Considering Borel measure μ on \mathbb{R}^n , the quality of the approximation may be quantified by set measures:

$$\alpha_{\mathbb{P}} = \mu \left(g^{-1}([\epsilon]) \setminus \sqcup_{\mathbb{P}} \right) = \mu(g^{-1}([\epsilon])) - \sum_{[\mathbf{x}] \in \mathbb{P}} \mu([\mathbf{x}]). \quad (10)$$

Smaller is $\alpha_{\mathbb{P}}$, better is the approximation. Interval based set inversion algorithms are able to reach arbitrary precision for small dimensions (2 or 3 typically).

Now, for all $\mathbf{y} \in \mathbb{R}^n$, it happens that:

$$f_{\mathbf{X} | \mathbf{X} \in \sqcup_{\mathbb{P}}}(\mathbf{y}) = \frac{f_{\mathbf{X}}(\mathbf{y}) \delta_{\mathbf{y} \in \sqcup_{\mathbb{P}}}}{P(\mathbf{X} \in \sqcup_{\mathbb{P}})} \quad (11)$$

$$= \frac{f_{\mathbf{X}}(\mathbf{y}) \sum_{[\mathbf{x}] \in \mathbb{P}} \delta_{\mathbf{y} \in [\mathbf{x}]} P(\mathbf{X} \in [\mathbf{x}])}{\sum_{[\mathbf{x}] \in \mathbb{P}} P(\mathbf{X} \in [\mathbf{x}])} = \frac{\sum_{[\mathbf{x}] \in \mathbb{P}} P(\mathbf{X} \in [\mathbf{x}]) f_{\mathbf{X} | \mathbf{X} \in [\mathbf{x}]}(\mathbf{y})}{\sum_{[\mathbf{x}] \in \mathbb{P}} P(\mathbf{X} \in [\mathbf{x}])}$$

$$= \sum_{[\mathbf{x}] \in \mathbb{P}} \frac{P(\mathbf{X} \in [\mathbf{x}])}{\sum_{[\mathbf{x}] \in \mathbb{P}} P(\mathbf{X} \in [\mathbf{x}])} f_{\mathbf{X} | \mathbf{X} \in [\mathbf{x}]}(\mathbf{y}), \quad (12)$$

where $\delta_{\text{true}} = 1$ and $\delta_{\text{false}} = 0$ else. Since $f_{\mathbf{X}}$ is bounded, it is easily derived from (11):

$$f_{\mathbf{X} | \mathbf{X} \in \sqcup_{\mathbb{P}}}(\mathbf{y}) \xrightarrow{\alpha_{\mathbb{P}} \rightarrow 0} f_{\mathbf{X} | \mathbf{X} \in g^{-1}([\epsilon])}(\mathbf{y}) \quad (13)$$

Now, equation (12) shows clearly that $f_{\mathbf{X} | \mathbf{X} \in \sqcup_{\mathbb{P}}}$ may be sampled by applying two steps: first sample a box $[\mathbf{x}] \in \mathbb{P}$ according to the discrete probability $\frac{P(\mathbf{X} \in [\mathbf{x}])}{\sum_{[\mathbf{x}] \in \mathbb{P}} P(\mathbf{X} \in [\mathbf{x}])}$, then sample \mathbf{y} by the conditional law $f_{\mathbf{X} | \mathbf{X} \in [\mathbf{x}]}$. At last, we have here an efficient method for sampling $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$ (algorithm 1).

```

1 Function Sampling  $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$  (subpaving)
   input :  $\alpha, g, N$    output:  $y_{1:N}$ 
2   Build subpaving  $\mathbb{P}$  such that  $\alpha_{\mathbb{P}} < \alpha$ 
3   for  $k \leftarrow 1$  to  $N$  do
4     Select  $[\mathbf{x}] \in \mathbb{P}$  with proba.
        $\frac{P(\mathbf{X} \in [\mathbf{x}])}{\sum_{[\mathbf{x}] \in \mathbb{P}} P(\mathbf{X} \in [\mathbf{x}])}$ 
5     Build  $y_k$  by sampling  $[\mathbf{X} | \mathbf{X} \in [\mathbf{x}]]$ 
6   end
7 end

```

Algorithm 1: Based on a subpaving.

The approach is efficient on rare events of the form $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$. This immediate application of the interval-based inversion is only applicable to rather small dimensions. Taking inspiration of this preliminary approach, we address now the sampling problem in higher dimensions.

Naive Dichotomous Sampling. A key point of algorithm 1 is to be able to sample a box $[x]$ of a subpaving of $g^{-1}([\epsilon])$. It is noticeable that we do not need to build the full subpaving; if we were able to construct a box $[x]$ of the subpaving *on demand*, together with its *relative weight* within the subpaving, then we would be able to build sample y .

To begin with, we define what is a cut:

- A cut of box $[x] \in [\mathbb{R}^n]$ is a pair $([l], [r]) \in [\mathbb{R}^n]^2$ such that $[l] \cap [r] = \emptyset$ and $[l] \sqcup [r] = [x]$.
- A bisection is a cut $([l], [r])$ such that $[l]$ and $[r]$ are same-sized.

In general, bisections are often used in dichotomous processes, as there is a garanty of exponential volume decrease of the search area. Here, we speak in terms of cuts, which are more general, *being implied that an appropriate management of the box length is made in order to ensure the convergence*.

We assume that a weighting function is available:

$$\omega_{[x]} \simeq P(\mathbf{X} \in [x] \& g(\mathbf{X}) \in [\epsilon]). \quad (14)$$

Algorithm 2 implicitly builds a partial subpaving, and actually produces a weighted particle cloud as a result of the sampling of $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$.

The algorithm iterates (**for** loop) the same sampling process, that is the following successive steps, until $[x_j]$ is sufficiently small (*i.e.* $\rho([x_j]) \leq r$) or is inside an implied suppaving (*i.e.* $[g]([x_j]) \subset [\epsilon]$):

- Cut $[x_j]$ by means of $\text{Cut}([x_j])$. This function is designed so as to ensure that $\rho([x_j])$ vanishes,
- Select randomly one box of cut $([l_j], [r_j])$ in proportion to their weight, by mean of Bernoulli process $\text{Bern}([l_j], \omega_{[l_j]}), ([r_j], \omega_{[r_j]})$,
- Update π_k which computes the processed probability of $[x_j]$ in regards to the Bernoulli sequence.

Assume that J is the last value reached by parameter j after the **while** loop. Then, the corrected weight $w_k = \frac{1}{\pi_k} P(\mathbf{X} \in [x_j]) \delta_{[g]([x_j]) \subset [\epsilon]}$ is computed for $[x_j]$ and for y_k , and y_k is sampled from $[x_j]$.

Notice that $w_k = 0$ when $[g]([x_j]) \not\subset [\epsilon]$, which implies that only the boxes $[x_j]$ of a subpaving of $g^{-1}([\epsilon])$ are actually considered. Combined with loop constraint $\rho([x_j]) > r$ and $[g]([x_j]) \not\subset [\epsilon]$, it follows that a subpaving of $g^{-1}([\epsilon])$ is implicitly and partially built during the sampling process.

When $[g]([x_j]) \subset [\epsilon]$, we have $w_k = \frac{P(\mathbf{X} \in [x_j])}{\pi_k}$ where π_k evaluates the processed probability for $[x_j]$. As a result, the weighted particles (y_k, w_k) provide an unbiased estimation of $f_{\mathbf{X}|g(\mathbf{X}) \in [\epsilon]}$ in a subpaving of

²Recall that $[g]([x_j])$ is computable while $g([x_j])$ is not.

```

1 Function Sampling  $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$ 
   ( $\omega$ -based)
2   input :  $r, g, \omega, N$    output:  $(y_k, w_k)_{1:N}$ 
3   for  $k \leftarrow 1$  to  $N$  do
4      $([x_0], \pi_k, j) \leftarrow ([b], 1, 0)$ 
5     while  $\rho([x_j]) > r$  and  $[g]([x_j]) \not\subset [\epsilon]$ 
6       do
7          $([l_{j+1}], [r_{j+1}]) \leftarrow \text{Cut}([x_j])$ 
8          $j \leftarrow j + 1$ 
9          $[x_j] \leftarrow$ 
10           $\text{Bern}([l_j], \omega_{[l_j]}), ([r_j], \omega_{[r_j]})$ 
11          $\pi_k \leftarrow \frac{\omega_{[x_j]}}{\omega_{[l_j]} + \omega_{[r_j]}} \pi_k$ 
12       end
13      $w_k \leftarrow P(\mathbf{X} \in [x_j]) \delta_{[g]([x_j]) \subset [\epsilon]} / \pi_k$ 
14     Build  $y_k$  by sampling  $[\mathbf{X} | \mathbf{X} \in [x_j]]$ 
15   end

```

Algorithm 2: Based on a weighting function.

$g^{-1}([\epsilon])$. It is not the same at the border of $g^{-1}([\epsilon])$, but this case is neglected. However, the sampler is not at all efficient when considering its variance.

Let us consider a case which works perfectly. Assume $\omega_{[x]} = P(\mathbf{X} \in [x] \& g(\mathbf{X}) \in [\epsilon])$. In this ideal case, the weight along a **while** loop is computed by:

$$\frac{\omega_{[x_j]}}{\omega_{[l_j]} + \omega_{[r_j]}} = \frac{P(\mathbf{X} \in [x_j] \& g(\mathbf{X}) \in [\epsilon])}{P(\mathbf{X} \in [l_j] \cup [r_j] \& g(\mathbf{X}) \in [\epsilon])} = \frac{P(\mathbf{X} \in [x_j] \& g(\mathbf{X}) \in [\epsilon])}{P(\mathbf{X} \in [x_{j-1}] \& g(\mathbf{X}) \in [\epsilon])}, \quad (15)$$

and then:

$$\pi_k = \prod_{j=1}^J \frac{\omega_{[x_j]}}{\omega_{[l_j]} + \omega_{[r_j]}} = \frac{P(\mathbf{X} \in [x_j] \& g(\mathbf{X}) \in [\epsilon])}{P(\mathbf{X} \in [b] \& g(\mathbf{X}) \in [\epsilon])}. \quad (16)$$

Three cases potentially arise:

- $[g]([x_j]) \subset [\epsilon]$, *i.e.* $[x_j]$ is in implied subpaving. Now $P(\mathbf{X} \in [x_j] \& g(\mathbf{X}) \in [\epsilon]) = P(\mathbf{X} \in [x_j])$, because $g([x_j]) \subset [g]([x_j]) \subset [\epsilon]$. Then:

$$w_k = \frac{P(\mathbf{X} \in [x_j]) \delta_{[g]([x_j]) \subset [\epsilon]}}{\frac{P(\mathbf{X} \in [x_j] \& g(\mathbf{X}) \in [\epsilon])}{P(\mathbf{X} \in [b] \& g(\mathbf{X}) \in [\epsilon])}} = \frac{P(\mathbf{X} \in [x_j])}{\frac{P(\mathbf{X} \in [x_j])}{P(\mathbf{X} \in [b] \& g(\mathbf{X}) \in [\epsilon])}} = P(\mathbf{X} \in [b] \& g(\mathbf{X}) \in [\epsilon]) \triangleq P(g(\mathbf{X}) \in [\epsilon]) \quad (17)$$

- $[g]([x_j]) \cap [\epsilon] \neq \emptyset$ but $[g]([x_j]) \not\subset [\epsilon]$, *i.e.* $[x_j]$ is within the border of the implied subpaving. Then $w_k = 0$. Since we are at the border, these *lost* cases are negligible for small precision r .
- $[g]([x_j]) \cap [\epsilon] = \emptyset$, *i.e.* $[x_j]$ is outside the implied subpaving and its border. Then $g([x_j]) \cap [\epsilon] = \emptyset$

and $\omega_{[x_j]} = P(\mathbf{X} \in [x_j] \& g(\mathbf{X}) \in [\epsilon]) = 0$. Case is simply impossible from the Bernoulli process.

Equation (17) shows that the sampling process results in a cloud of same-weight particles over the implied subpaving. Other cases (rejections) are negligible. Here we have a sampler of $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$ with the best variance performance in regards to the number of particles. But this is achieved only when $\omega_{[x]} = P(\mathbf{X} \in [x] \& g(\mathbf{X}) \in [\epsilon])$. Of course such exact weighting function is almost never available.

Why it Does Not Work in General Cases? In the cases where $\omega_{[x]} \neq P(\mathbf{X} \in [x] \& g(\mathbf{X}) \in [\epsilon])$, the accumulated error will explode with the dimension, which will result in dramatically uneven weights on the particles. The resulting weighted particles cloud is then useless for practical applications.

3 A DICHOTOMOUS APPROACH FOR SAMPLING

Algorithms 1 and 2 illustrate the two main dimensional issues, that we have to deal with. These approaches are complementary:

- By building a complete subpaving of $g^{-1}([\epsilon])$, algorithm 1 makes possible a direct sampling of $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$, and incidentally an accurate computation of $P(\mathbf{X} \in [x] \& g(\mathbf{X}) \in [\epsilon])$. However, this construction of a complete subpaving is only possible for small dimensions.
- Algorithm 2 avoids the construction of a complete subpaving. Instead, it builds the boxes of an implied subpaving on demand throughout the sampling iteration. However, the algorithm is inefficient unless the weighting function $\omega_{[x]}$ is a good approximation of $P(\mathbf{X} \in [x] \& g(\mathbf{X}) \in [\epsilon])$. This condition is not accessible in general.

We propose now an intermediate approach which:

- keeps history of the subpaving construction throughout the sampling process,
- use this history to build an improved estimate of $P(\mathbf{X} \in [x] \& g(\mathbf{X}) \in [\epsilon])$.

By these tricks, it is expected that the sampling precision will increase with the number of samples. In order to avoid useless exploration, we also truncate the dichotomous process on the basis of some predictive assessment of final weight w_k . Thus, the algorithm tends to favor breadth search instead of depth search at the early stages of the sampling process.

3.1 Implementing Some Containment of the Curse of Dimension

From now on, it is assumed that:

$$0 \leq \omega_{[x]} \leq P(\mathbf{X} \in [x]), \quad (18)$$

and that:

$$\omega_{[x]} = \begin{cases} 0 & \text{if } [g]([x]) \cap [\epsilon] = \emptyset, \\ P(\mathbf{X} \in [x]) & \text{if } [g]([x]) \subset [\epsilon]. \end{cases} \quad (19)$$

Algorithm 3 is an evolution of algorithm 2. In addition, it builds an history of cuts, stored in map cuts, and computes dynamically from this history an improved weighting function, stored in map omg.

```

1 Function Sampling  $[\mathbf{X} | g(\mathbf{X}) \in [\epsilon]]$  (cut
   history)
   input :  $\sigma, r, g, \omega, N$    output:  $(y_k, w_k)_{1:N}$ 
2   (cuts, omg, k)  $\leftarrow$  (0, 0, 0)
3   omg([b])  $\leftarrow$   $\omega_{[b]}$ 
4   while  $k < N$  do
5      $([x_0], \pi_k, j) \leftarrow ([b], 1, 0)$ 
6     while  $\rho([x_j]) > r$  and  $[g]([x_j]) \not\subset [\epsilon]$ 
       do
7       if  $\left| \log_2 \left( \frac{\text{omg}([b])}{\text{omg}([x_j])} \pi_k \right) \right| > \sigma$ 
           goto 20
8       ifundef cuts  $([x_j]) \leftarrow$  Cut  $([x_j])$ 
9        $([l_{j+1}], [r_{j+1}]) \leftarrow$  cuts  $([x_j])$ 
10       $j \leftarrow j + 1$ 
11      ifundef omg  $([r_j]) \leftarrow$   $\omega_{[r_j]}$ 
12      ifundef omg  $([l_j]) \leftarrow$   $\omega_{[l_j]}$ 
13       $(v_{[l_j]}, v_{[r_j]}) \leftarrow$  (omg  $([l_j])$ , omg  $([r_j])$ )
14       $[x_j] \leftarrow$  Bern  $(([l_j], v_{[l_j]}), ([r_j], v_{[r_j]}))$ 
15       $\pi_k \leftarrow \frac{v_{[x_j]}}{v_{[l_j]} + v_{[r_j]}} \pi_k$ 
16      end
17       $w_k \leftarrow P(\mathbf{X} \in [x_j]) \delta_{[g]([x_j]) \subset [\epsilon]} / \pi_k$ 
18      Build  $y_k$  by sampling  $[\mathbf{X} | \mathbf{X} \in [x_j]]$ 
19       $k \leftarrow k + 1$ 
20      for  $i \leftarrow j$  to 1 do
21        | omg  $([x_{i-1}]) \leftarrow$  omg  $([l_i]) +$  omg  $([r_i])$ 
22      end
23    end
24 end

```

Algorithm 3: Based on cuts history.

The lines of this algorithm are colored in black, blue or dark blue. Black lines are inherited from algorithm 2. Blue lines are new additions to the previous algorithm. Dark blue lines (4, 19, 23 and 2 partially) correspond to the **for** loop rewritten as a **while** loop:

$k \leftarrow 0$ **while** $k < N$ **do** ... $k \leftarrow k + 1$... **end**

Variable `cuts` is a dictionary and is used to register the history of computed cuts. At start, `cuts` is defined empty (line 2). For a given box $[x_j]$, the cut on $[x_j]$ is computed only once, if it is computed, by line 8 :

ifundef `cuts` ($[x_j]$) \leftarrow `Cut` ($[x_j]$)

Keyword **ifundef** tests if `cuts` ($[x_j]$) is defined ; if still undefined, then `cuts` ($[x_j]$) is set to `Cut` ($[x_j]$) .

Variable `omg` is a dictionary which records the weighting function and its possible updates, when needed. At start, `omg` is only defined for $[b]$ and is set to $\omega_{[b]}$ (lines 1 and 3). Variable `omg` ($[r_j]$) is set to $\omega_{[r_j]}$, if it has not been initialized yet (line 11). The same is done for variable `omg` ($[l_j]$) at line 12. When the `cuts` sequence is done (second **while**), then the weighting function is updated by the **for** loop (lines 20,21,22). This ensures that `omg` ($[x]$) is computed as the sum of the weights `omg` ($[z]$) of the leaves $[z]$ of the cuts tree rooted on $[x]$. Then, property (19) ensures that `omg` ($[x]$) gets closer to $P(\mathbf{X} \in [x] \& g(\mathbf{X}) \in [\epsilon])$ when the cuts tree rooted on $[x]$ gets more refined.

Algorithm 3 is similar to algorithm 2, except that:

- the cut $([l_{j+1}], [r_{j+1}])$ is recovered from the history, when it is possible (line 9),
- the cut choice is done by means of updated weights $(v_{[l_j]}, v_{[r_j]}) \triangleq (\text{omg}([l_j]), \text{omg}([r_j]))$.

There is an interesting property here. Assume that J is the last value reached by j and that $J' < J$ is such that `omg` ($[l_j]$) and `omg` ($[r_j]$) are already defined for all $1 \leq j \leq J'$. Weighting functions are updated in these cases. Then, it comes for all $1 \leq j \leq J'$ that:

$$\text{omg}([x_{j-1}]) = \text{omg}([l_j]) + \text{omg}([r_j]).$$

The computation of π_k is then simplified:

$$\begin{aligned} \pi_k &= \prod_{j=1}^{J'} \frac{v_{[x_j]}}{v_{[l_j]} + v_{[r_j]}} \prod_{j=J'+1}^J \frac{v_{[x_j]}}{v_{[l_j]} + v_{[r_j]}} \\ &= \prod_{j=1}^{J'} \frac{v_{[x_j]}}{v_{[x_{j-1}]}} \prod_{j=J'+1}^J \frac{v_{[x_j]}}{v_{[l_j]} + v_{[r_j]}} \\ &= \frac{v_{[x_{J'}]}}{v_{[b]}} \prod_{j=J'+1}^J \frac{v_{[x_j]}}{v_{[l_j]} + v_{[r_j]}}. \end{aligned} \quad (20)$$

Thus, the error on π_k grows exponentially only within the newly explored cuts, that is here from $J' + 1$ to J . This is a reason for setting a certain restriction on the depth-oriented aspect of this sampling process. Another good reason is to prevent degenerate particle weights, w_k . Algorithm 3 thus implements some code (line 7) for testing the degeneracy of π_k and eventually restarting the sampling loop (second **while**):

if $\left| \log_2 \left(\frac{\text{omg}([b])}{\text{omg}([x_j])} \pi_k \right) \right| > \sigma$ **goto** 20

This code tests the logarithmic distance between the weight of $[x_j]$, `omg` ($[x_j]$), and the weight resulting from the sampling process, `omg` ($[b]$) π_k . If it is higher than σ , then the loop is stopped by going to line 20. By doing that, the incrementation of k is skipped, so that the sampling loop is restarted for the same indice k . However, the update of variable `omg` is done, and of course, the history of `cuts` stays incremented. So, although the sampling loop has been interrupted in this case, the sampling structure has been upgraded. This results in an adaptive process which will balance depth and breadth explorations when running the sampling. Breadth exploration is favored on the first sampling iterations, but the tendency becomes inverted after several samples.

3.2 Practical Implementation

Algorithm 3 draws the main principles of our sampling method. Some details were not described, although necessary for practical implementation:

- We implement the following definition of ω :

$$\omega_{[x]} = \frac{\mu([g]([x]) \cap [\epsilon])}{\mu([\epsilon])} P(\mathbf{X} \in [x]), \quad (21)$$

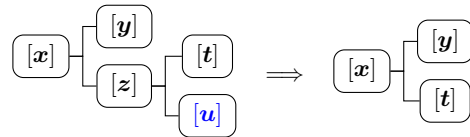
where μ is Borel measure on \mathbb{R}^m . This definition checks properties (18) and (19). It also tries a rough approximation for $P(\mathbf{X} \in [x] \& g(\mathbf{X}) \in [\epsilon])$.

- The definition of `Cut` is an important choice. There are n possible bisections of $[x] \in [\mathbb{R}^n]$. Our algorithm selects a bisection $([y], [z])$ randomly in regards to the following criterions:

- Favor cuts such that $\omega_{[y]} \ll \omega_{[z]}$ or $\omega_{[y]} \gg \omega_{[z]}$,
- Avoid overly elongated $[x]$, i.e. $\frac{\max_i(x_i^+ - x_i^-)}{\min_i(x_i^+ - x_i^-)} \gg 1$,

In addition, this cut process interacts with some cuts history simplifications (next point).

- Our implementation tries to optimize the structure of the cuts history. For example, assume that $([y], [z])$ is a cut of $[x]$, $([t], [u])$ is a cut of $[z]$ and $[g]([u]) \cap [\epsilon] = \emptyset$, i.e. $[u]$ is outside the subpaving and its border. Then, boxes $[z]$ and $[u]$ are useless and should be removed from the structure:



- M first samples are discarded before sampling, so as to initialize the structure of the sampler. After that, N samples are sampled and returned.

Whatever, the algorithm is rather simple to set up. Except for the choice of ω , which is structural, r , M and σ are the only parameters to be defined.

Parallelization. Data structures cuts and omg deal rather well with parallel processing. Our implementation is multi-threaded.

3.3 Some Ideas for Improvements

As will be shown in the tests, our method has been applied up to a space of dimension $n = 11$ (for a subvariety of dimension 10). This is much more than what is possible through a complete subpaving construction. But somehow, it is a reprieve in regards to the curse of the dimension.

Using Contractors. Contractors are used along with bisection process in order to speedup the subpaving construction (Chabert and Jaulin, 2009). Contractors are especially available when function g is expressed through some constraints. Such tool may be useful in our algorithm, in terms of improving the speed and reducing the complexity of the cuts history.

Better Weighting Function. Definition (21) is rather cheap. Defining ω as a better approximation of $P(X \in [x] \& g(X) \in [\epsilon])$ may be possible by means of local linear approximations (or high order) of g .

Relaxing the Constraint. Many rare event simulation methods work by starting from a relaxed constraint, *e.g.* $h(x) \geq \gamma_0$ where $\gamma_0 \ll \gamma_{\max}$, then by gradually tensing this constraint, *e.g.* up to $h(x) \geq \gamma_k$ where $\gamma_k \simeq \gamma_{\max}$. Our algorithm totally discarded such approaches. However, it may be profitable to mix both points of view in order to improve the behavior of our approach for higher dimensions. We have especially in mind a way for an incrementally better definition of the weighting function ω .

4 EXAMPLES AND TESTS

This section presents simulation tests followed by a small application in black-box function optimization.

4.1 Simulation Tests

The tests presented here are performed for sampling algorithm 3. The algorithm has been implemented in

Rust language (www.rust-lang.org) and was processed on 7 threads. In order to illustrate its performances, our choice was to consider a mathematically simple simulation problem, in order to make the statistics of the results clear enough to analyze.

Test Cases. Thorough the section, it is assumed that \mathbf{X} follows the uniform law on $\mathbf{b} = [-2, 2]^n$ with $n \in \{2, 3, \dots, 11\}$. Three cases are investigated:

Case (a): Are defined $g_a(\mathbf{x}) = \|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^n x_j^2}$ and $[\epsilon_a] = [0.95, 1.05]$. Then $g_a^{-1}([\epsilon_a])$ is a hyperannulus, which approximates the unit hypersphere of dimension $n - 1$.

Case (b): For $n = 11$ and $0 \leq k \leq 9$, it is considered:

$$g_b(\mathbf{x}) = \begin{bmatrix} \|\mathbf{x}\|_2 \\ x_3 \\ \vdots \\ x_{2+k} \end{bmatrix} \text{ and } [\epsilon_b] = [\epsilon_a] \times [\mathbf{z}]^k, \quad (22)$$

with $[\mathbf{z}] = [-0.05, 0.05]$. This case is similar to (a), but with additional constraints, so that $g_b^{-1}([\epsilon_b])$ approximates an hypersphere of dimension $n - 1 - k$. When $k = 0$, we are back to case (a) with $n = 11$. When $k = 9$, then $g_b^{-1}([\epsilon_b])$ approximates the unit circle \mathcal{C} within the first two dimensions.

Case (c): For $n = 11$, it is considered:

$$g_c(\mathbf{x}) = \begin{bmatrix} \|\mathbf{x}\|_2 \\ x_3 \\ \vdots \\ x_{11} \\ \min(|x_1|, |x_2|) \end{bmatrix}, \quad (23)$$

and $[\epsilon_c] = [\epsilon_a] \times [\mathbf{z}]^9 \times [\alpha]$ with $[\alpha] = [0, 0.5]$. It is noticed that this case is obtained by adding constraint $0 \leq \min(|x_1|, |x_2|) \leq 0.5$ to last subcase of (b). In other words, we are approximately sampling on the (disjoint) union of the 4 subsegments $\mathcal{A}_1, \dots, \mathcal{A}_4$ of the unit circle \mathcal{C} , defined by:

$$\mathcal{A}_j = \left\{ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathcal{C} \middle/ \arg \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) \in j\frac{\pi}{2} + \left[-\frac{\pi}{6}, \frac{\pi}{6} \right] \bmod 2\pi \right\}. \quad (24)$$

These 4 subsegments have the same size so that their probabilities are the same in regards to X .

Purpose of the Test Cases. Subsequently, case (a) is used in order to evaluate the performance of the sampling process both in accuracy and in efficiency for different dimensions. Case (b) is used in order to

evaluate the efficiency of the sampling when the number of constraints increases. Case (c) is used in order to evaluate the accuracy of the sampling in case of complex constraints which introduce disjoint modalities.

Results and Analysis. All the subsequent tests have been achieved with the following parameters:

- $r = 0.001$ is the radius bound for second **while** stop condition,
- $M = 5000$ is the number of first samples, which are discarded in order to initialize the sampler,
- $N = 50000$ is the number of sampled particles.
- $\sigma = 10$ on all tests.

Case (a): This case is mathematically easy to predict. In (Dezert and Musso, 2001), Dezert and Musso proposed a method, which may be used for uniformly sampling on an annulus of ellipsoid. Whatever, one must keep in mind that our approach is generic and can be applied to an infinite number of configurations.

Histograms. For each subcase $n \in \{3, 7, 11\}$, we have computed the radius of all samples x and built the associated histograms (figures 2, 3 and 4).

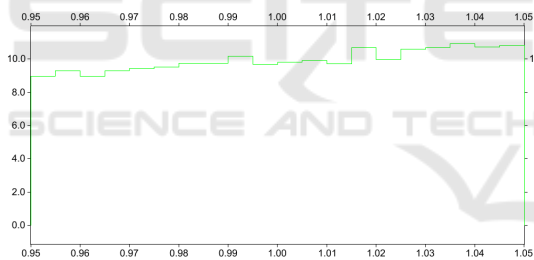


Figure 2: (a): Radius histogram (20 divisions); $n = 3$.

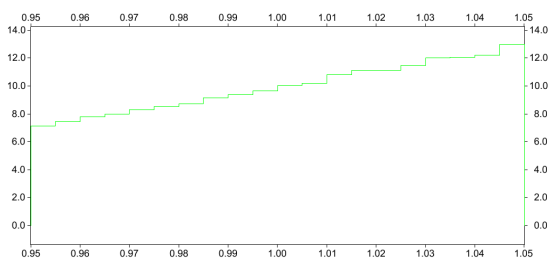


Figure 3: (a): Radius histogram (20 divisions); $n = 7$.

For each $n \in \{3, 7, 11\}$ and for all $1 \leq i < j \leq n$, we have computed the angle of all samples (x_i, x_j) and built the associated histograms. From these $\frac{n(n-1)}{2}$ histograms of each subcase, we have computed the minimal, mean and maximal histogram. The results are shown in blue, green and red, respectively, and

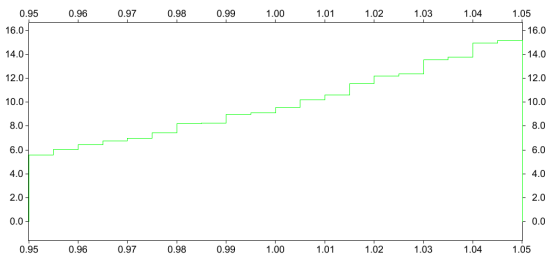


Figure 4: (a): Radius histogram (20 divisions); $n = 11$.

provide an hint on the error of the estimation (figures 5, 6 and 7).

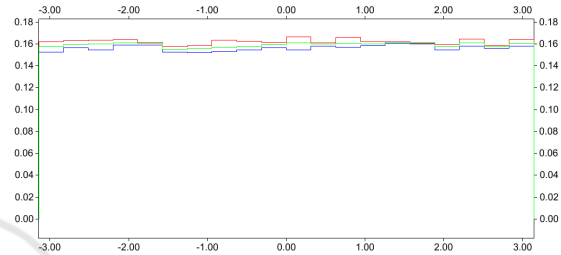


Figure 5: (a): Angle histogram (20 divisions); $n = 3$.

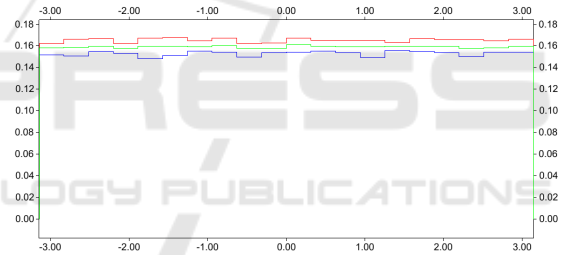


Figure 6: (a): Angle histogram (20 divisions); $n = 7$.

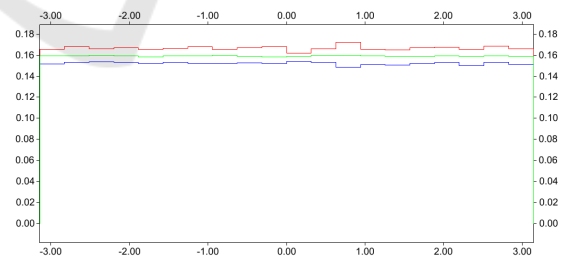


Figure 7: (a): Angle histogram (20 divisions); $n = 11$.

By symmetry, the *theoretical histograms* are uniform. The errors should be of the order of $\sqrt{\frac{20}{50000}} \simeq 0,02$. In comparison, the errors figured in the angular histograms are quite acceptable, even for the highest dimension. The most interesting point is that there is no rupture in the histogram, which shows that the sampler does manage the subvariety structure. We do not have an error estimation for the radius histograms.

Actually, the local probability should theoretically increase with the radius, this property being accentuated with the dimension. This is what is obtained on the histograms. It is noteworthy however that the sides of these histograms are subject to additional errors implied by the border of the subvariety.

Process Statistics: We present some elements of measurement on the behavior of the algorithm according to the number of generated samples (0 to 55000). Each figure presents the result for cases $n = 3, 7, 11$ with respective colors, red, blue and green.

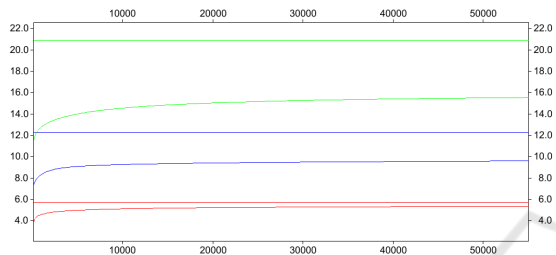


Figure 8: (a): $-\log_2(\text{omg}([b]))$ versus $P(g_a(\mathbf{X}) \in [\epsilon])$; $n = 3, 7, 11$.

Figure 8 shows how value $-\log_2(\text{omg}([b]))$ evolves and approximates $-\log_2(P(g_a(\mathbf{X}) \in [\epsilon]))$. Each curve increases to theoretical value (same color line), but performance decreases with dimension.

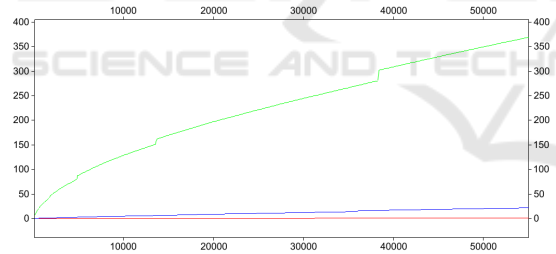


Figure 9: (a): Cumulative cpu time (s); $n = 3, 7, 11$.

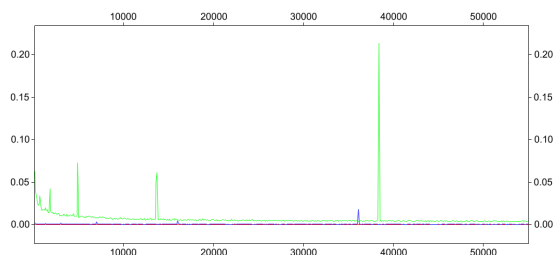


Figure 10: (a): Evolution of cpu time (s); $n = 3, 7, 11$.

Figures 9 and 10 present the cumulative cpu-time per thread and the evolution of the cpu-time per sample and per thread consumed by the process (expressed in second). The discontinuities in the curves are caused by intermittent memory allocations. But

we notice clearly that the sampling efficiency increases with the number of generated samples. However, the cumulative cpu time still increases dramatically with the dimension (the memory use evolves similarly). Although the curse of dimension has been delayed by our approach, it is still there.

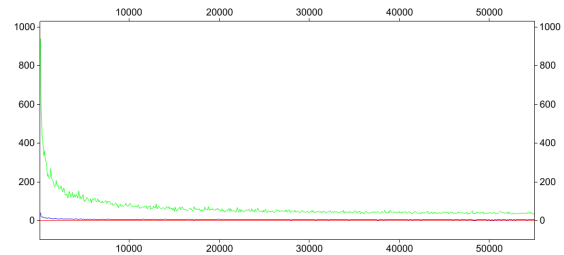


Figure 11: (a): Evolution of loop retry; $n = 3, 7, 11$.

The number of loop retries during the sampling is an interesting indication of the achievement of the sampling structure (figure 11). It decreases with the number of samples, and moreover becomes rather small, even for the highest dimension (around 40 for $n = 11$). This result should be compared to the probability of the subvariety (around 10^{-6} for $n = 11$).

Case b: We present synthetic curves, when the number of constraints increases. All curves are function of k , the number of additional constraints. The values are computed on the 100 last samples.

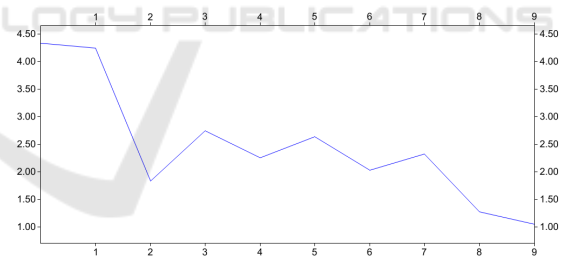
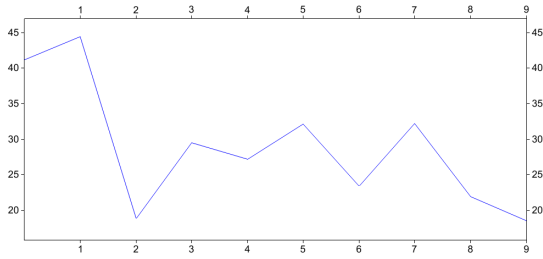


Figure 12: (b): Evolution of cpu time (ms); $k = 0 : 9$.

Figure 12 shows the evolution of cpu time per sample and per thread with k (expressed in millisecond). The curves tend to decrease and approach to zero. This tendency is not strict however.

Figure 13 presents the evolution of number of loop retries with k . Again, the tendency is similar. As a conclusion here, the performance of the sampler is likely to increase with the number of constraints, and this is a useful quality.

Case c: On this last test, we computed the radial histogram and angular histogram (there is only one) for the samples. These histograms are presented respectively in figures 14 and 15. The quality of the


 Figure 13: (b): Evolution of loop retry; $k = 0 : 9$.

histograms is comparable although slightly weaker than what was observed previously. Due to the constraint configuration, the *theoretical radius histogram* is uniform, and the *theoretical angular histogram* is uniform around each subsegment, $\mathcal{A}_1, \dots, \mathcal{A}_4$ with a gradual decrease on the borders. The generated histograms actually comply with these properties.

As a preliminary conclusion, we consider that our sampling method is globally performant in sampling conditionally to subvarieties. A future issue will be to increase the dimension of the sampling space.

4.2 Optimizing a Black-box Function

Such applications to black-box optimization were actually a great motivation for this work.

Assume that one needs to optimize a function which is not well known, and which may be computed by a highly costly process (a heavy simulation, tests made by human teams on the grounds, etc.). Of course the optimization should be made by sparing at best the number of calls to the costly evaluation. Is it possible to solve that? At first sight, it is tempting to

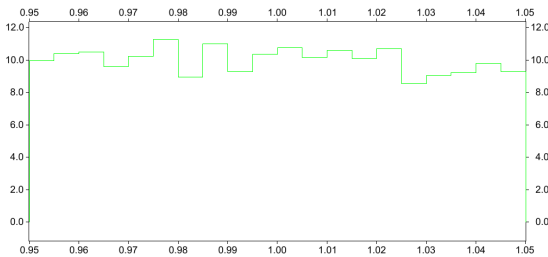


Figure 14: (c): Radius histogram (20 divisions).

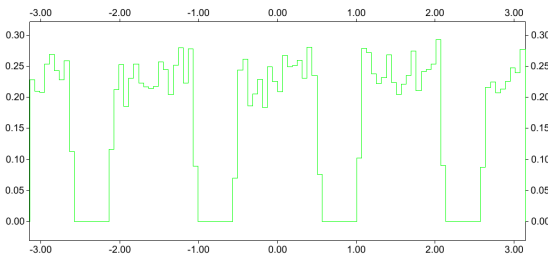


Figure 15: (c): Angle histogram (100 divisions).

say no!

In (Jones et al., 1998), Jones, Schonlau and Welch proposed the efficient global optimization method (EGO) for addressing such kind of problem. The idea is to use a surrogate function under the form of a functional random variable. This functional random variable is described by means of a Gaussian process with correlation depending on spatial distance (kriging). Based on such modelling, the construction of an *optimal* parameter sequence to be evaluated is obtained by iterating:

- Compute the posterior law of the functional random variable, according to the past evaluations,
- Compute the expected improvement function, in regards to the posterior law and the already best computed value,
- Choose the parameter optimizing this expected improvement function and evaluate it.

It is beyond the scope of this paper to detail this seminal work. Whatever, the approach relies on deep simplifications obtained from the Gaussian modelling, and it tends to be less efficient when the dimension of the optimization space increases. Works have been made in order to deal with this dimensional issue; *e.g.* (Bartoli et al., 2019). We describe subsequently a nonlinear interpretation of the EGO method.

A Nonlinear Formalisation. In (Dambreville, 2015), we proposed a nonlinear implementation of EGO, by considering a (nonlinear) function $g(\gamma, \mathbf{X})$ depending on noise model \mathbf{X} instead of a kriging-based surrogate function. The optimal measure sequence (γ_k) is obtained by iterating:

- 1 **Function Process next measure**
input : γ_j and $g_j \triangleq g(\gamma_j, \mathbf{x}^o)$ for $1 \leq j \leq k$
output: γ_{k+1} and $g_{k+1} \triangleq g(\gamma_{k+1}, \mathbf{x}^o)$
- 2 Make samples of $[\mathbf{X} | \forall j, g(\gamma_j, \mathbf{X}) = g_j]$
- 3 Compute $m_\gamma = \min_{1 \leq j \leq k} g_j$ and:

$$EI(\gamma) \simeq E_{\mathbf{X} | \forall j, g(\gamma_j, \mathbf{X}) = g_j} \min\{g(\gamma, \mathbf{X}), m_\gamma\}$$
// EI(γ) is approximated from the samples
- 4 Compute $\gamma_{k+1} \in \arg \min_\gamma EI(\gamma)$
- 5 Compute $g_{k+1} \triangleq g(\gamma_{k+1}, \mathbf{x}^o)$
- 6 **end**

By this way, we intend to solve this simple geometric problem: how to find the isobarycenter $\gamma = (a, b)$ of 4 *unknown points* $M_i = (x_i^o, y_i^o) \in [-5, 5]^2$ with $i \in \{1, 4\}$? The only approach that is possible for us is to test some solutions by requesting for a costly

measurement; this measurement evaluates function:

$$g(\gamma, \mathbf{x}^o) = \|\gamma - h(\mathbf{x}^o)\|_2, \quad (25)$$

where $h(\mathbf{x}^o) = \frac{1}{4} \sum_{i=1}^4 M_i$ and $\mathbf{x}^o = (x_1^o, y_1^o, \dots, x_4^o, y_4^o)$. Our purpose is then to optimize (a, b) by doing a minimum request to the costly evaluation $g(\gamma, \mathbf{x}^o)$.

We did not have an efficient method for sampling $[\mathbf{X} \mid \forall j, g(\gamma_j, \mathbf{X}) = g_j]$ at the time of (Dambreville, 2015). Now, we propose to apply algorithm 3 combined with a discretized method (we actually enumerated on 10^6 discretized points of $[-5, 5]^2$) for minimizing the expected improvement $EI(\gamma)$ in order to process minimizing sequence (γ_k) . This sequence converge to isobarycenter of $M_{1:4}$.

Tests and Results. Points M_1, \dots, M_4 are $(2, -1)$, $(3, 2)$, $(-\frac{3}{2}, 4)$, $(\frac{1}{2}, 3)$. Their barycenter is $(1, 2)$. We used a sampler with $M = 5000$, $N = 10000$ and $[\epsilon] = [-\frac{1}{100}, \frac{1}{100}]^k$. Variable \mathbf{X} is considered uniform on $[-5, 5]^8$. Process is stopped at step k_o , when optimum is found; best evaluation and solution are then respectively 0 and $(1, 2)$. The following table summarizes the results of 100 runs. cpu gives the average computation time:

k_o	3	4	5
%	46	53	1
cpu(s)	1670	2758	2939

Except for outlier, optimum is found almost equiprobably after 3 or 4 tries. This is similar to the geometric method³. It is noteworthy that our algorithm does not have geometric knowledge of the problem and deals with eight dimensions model noise.

5 CONCLUSIONS

We proposed an original dichotomous method for sampling a random vector conditionally to a subvariety. This generic approach, inspired from interval analysis, is accurate and efficient up to a space of dimension 11. We have shown how it could be applied efficiently to the optimization of expensive black-box function. The work is promising from an applicative point of view and offers several improvement perspectives. We will particularly investigate some relaxations techniques applied to the subvariety in order to enhance the efficiency of the approach in regards to higher dimensions.

³Each measure restricts the solution to a circle. After 2 measures, we usually have to choose between two points, and the solution is found equiprobably at step 3 or 4.

REFERENCES

- Abdallah, F., Gning, A., and Bonnifait, P. (2008). Box particle filtering for nonlinear state estimation using interval analysis. *Automatica*, 44(3):807–815.
- Alefeld, G. and Mayer, G. (2000). Interval analysis: theory and applications. *Journal of Computational and Applied Mathematics*, 121(1):421–464.
- Bartoli, N., Lefebvre, T., Dubreuil, S., Olivanti, R., Priem, R., Bons, N., Martins, J., and Morlier, J. (2019). Adaptive modeling strategy for constrained global optimization with application to aerodynamic wing design. *Aerospace Science and Technology*, 90:85–102.
- Bui Quang, P., Musso, C., and Le Gland, F. (2016). Particle filtering and the Laplace method for target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 52(1):350–366.
- Cérou, F., Del Moral, P., Furon, T., and Guyader, A. (2012). Sequential Monte Carlo for rare event estimation. *Statistics and Computing*, 22(3):795–908.
- Chabert, G. and Jaulin, L. (2009). Contractor Programming. *Artificial Intelligence*, 173:1079–1100.
- Dambreville, F. (2015). Optimizing a sensor deployment with network constraints computable by costly requests. In Thi, H. A. L., Dinh, T. P., and Nguyen, N. T., editors, *Modelling, Computation and Optimization in Information Systems and Management Sciences*, volume 360 of *Advances in Intelligent Systems and Computing*, pages 247–259. Springer.
- Dezert, J. and Musso, C. (2001). An efficient method for generating points uniformly distributed in hyperellipsoids. In *Workshop on Estimation, Tracking and Fusion: A Tribute to Bar-Shalom*, Monterey, California.
- Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. (2001). *Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer London Ltd.
- Jones, D., Schonlau, M., and Welch, W. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492.
- Morio, J., Balesdent, M., Jacquemart, D., and Vergé, C. (2014). A survey of rare event simulation methods for static input–output models. *Simulation Modelling Practice and Theory*, 49:287–304.
- Musso, C., Champagnat, F., and Rabaste, O. (2016). Improvement of the laplace-based particle filter for track-before-detect. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1095–1102.
- Rubinstein, R. Y. and Kroese, D. P. (2004). *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-Carlo Simulation (Information Science and Statistics)*. Springer-Verlag.