

Recursion

Let us assume that we have an integral representation of the quantity $\mathcal{A}$ that we seek to estimate, but that this integral involves the quantity $\mathcal{A}$ itself : $\mathcal{A}$ is known as the solution of an integral equation (a Fredholm equation). Let us further assume that this equation is linear :

$$\mathcal{A} = \int_{\mathcal{D}} \alpha(x)\mathcal{A} + \beta(x) dx \quad (1)$$

where a and b are known functions on $\mathcal{D}$.

We then proceed in three steps :

1. We probabilize the integral by introducing a sampling variable for $\mathcal{D}$, as we did for the first integral in Lab 2.
2. We probabilize the sum $\alpha(x)\mathcal{A} + \beta(x)$ so as to write it as an expectation. We then have a Bernoulli algorithm for estimating this expectation, exactly as we did in the Lab 1 example where we estimated $4 + 1$ by Monte Carlo.
3. We use double randomization to combine the two, as we did to combine the two integrals in Lab 2.

Probabilization of the integral We introduce a random variable X on $\mathcal{U}_X = \mathcal{D}$ with probability density p_X . This probability density must be nonzero at every point of the domain. We multiply and divide the integrand by p_X to write

$$\mathcal{A} = \int_{\mathcal{D}} p_X(x) dx \left(\frac{\alpha(x)}{p_X(x)} \mathcal{A} + \frac{\beta(x)}{p_X(x)} \right) \quad (2)$$

This expression can be read as a first algorithm : we generate one realization x of the random variable X and the corresponding Monte Carlo weight realization is $w = \frac{\alpha(x)}{p_X(x)} \mathcal{A} + \frac{\beta(x)}{p_X(x)}$. The average of a large number of such weights will produce an estimate of $\mathcal{A}$.

Probabilization of the sum We introduce a probability P and write

$$\frac{\alpha(x)}{p_X(x)} \mathcal{A} + \frac{\beta(x)}{p_X(x)} = P \left[\frac{\alpha(x)}{p_X(x)P} \mathcal{A} \right] + (1 - P) \left[\frac{\beta(x)}{p_X(x)(1 - P)} \right] \quad (3)$$

This expression can be read as a second algorithm : we generate one realization b of a Bernoulli random variable B with probability P ; if the realization is $b = 1$, we retain the Monte Carlo weight $w = \frac{\alpha(x)}{p_X(x)P} \mathcal{A}$; if the realization is $b = 0$, we retain the Monte Carlo weight $w = \frac{\beta(x)}{p_X(x)(1 - P)}$. The average of a large number of such weights will produce an estimate of $\frac{\alpha(x)}{p_X(x)} \mathcal{A} + \frac{\beta(x)}{p_X(x)}$.

Double randomization We combine the two previous expressions to obtain

$$\mathcal{A} = \int_{\mathcal{D}} p_X(x) dx \left(P \left[\frac{\alpha(x)}{p_X(x)P} \mathcal{A} \right] + (1 - P) \left[\frac{\beta(x)}{p_X(x)(1 - P)} \right] \right) \quad (4)$$

which can be read as the following algorithm : we generate one realization x of the random variable X ; we generate one realization b of a Bernoulli random variable B with probability P ; if the realization is $b = 1$, we retain the Monte Carlo weight $w = \frac{\alpha(x)}{p_X(x)P} \mathcal{A}$; if the realization is $b = 0$, we retain the Monte Carlo weight $w = \frac{\beta(x)}{p_X(x)(1 - P)}$. The average of a large number of such weights will produce an estimate of $\mathcal{A}$.

Recursion The preceding algorithm estimates $\mathcal{A}$ but is practicable only if $\mathcal{A}$ is already known. Indeed, when the realization of the Bernoulli variable is $b = 1$, the Monte Carlo weight $w = \frac{\alpha(x)}{p_X(x)P} \mathcal{A}$ contains $\mathcal{A}$ itself. However, this weight is a linear function of $\mathcal{A}$. In the Monte Carlo literature, when certain branches of the algorithm lead to a weight expression that is a linear function of the quantity to be estimated, the algorithmic translation of the integral formulation should be conceived recursively : the appearance of $\mathcal{A}$ means that the algorithm continues, with a new realization of X and a new realization of B , leading to a new expression for an intermediate weight that takes the place of $\mathcal{A}$ in $w = \frac{\alpha(x)}{p_X(x)P} \mathcal{A}$. Of course, this new expression for the total weight may itself involve $\mathcal{A}$, which translates into yet another realization of X and a new realization of B , etc. This recursion requires management of the prefactor $\frac{\alpha(x)}{p_X(x)P}$, which is illustrated in algorithm 1.

Algorithme 1 : Exemple d'algorithme récursif

```

1  somme = 0;
2  sommeDesCarres = 0;
3  pour chaque  $i \in \{1 : n\}$  faire
4      recursivite = vrai;
5      prefacteur = 1;
6      tant que recursivite faire
7          Réalisation  $x$  de  $X$ ;
8          Réalisation  $b$  d'une loi de Bernoulli  $B$  de probabilité  $P$ ;
9          si  $b=1$  alors
10             prefacteur = prefacteur *  $\frac{\alpha(x)}{p_X(x)P}$ ;
11          sinon
12              $w = \text{prefacteur} * \frac{\beta(x)}{p_X(x)(1-P)}$ ;
13             recursivite = faux;
14          fin
15      fin
16      somme = somme +  $w$ ;
17      sommeDesCarres = sommeDesCarres +  $w^2$ ;
18 fin
19  $m = \frac{\text{somme}}{n}$ ;
20  $s = \frac{1}{\sqrt{n-1}} \sqrt{\frac{\text{sommeDesCarres}}{n} - m^2}$ ;
```

Corresponding derivation We formally justify this algorithm by writing the expressions as expectations. Equation 4 means

$$\mathcal{A} = \mathbb{E} \left(\mathcal{H}(B) \left[\frac{\alpha(X)}{p_X(X)P} \mathcal{A} \right] + \mathcal{H}(1-B) \left[\frac{\beta(X)}{p_X(X)(1-P)} \right] \right) \quad (5)$$

where the Heaviside function $\mathcal{H}$ is defined so that $\mathcal{H}(1) = 1$ and $\mathcal{H}(0) = 0$. We can then define two identically distributed and independent random variables X_1 and X_2 that follow the same law as X , and likewise define two identically distributed and independent random variables B_1 and B_2 that follow the same law as B , in order

to write $\mathcal{A}$ in two different ways :

$$\begin{aligned}\mathcal{A} &= \mathbb{E} \left(\mathcal{H}(B_1) \left[\frac{\alpha(X_1)}{p_X(X_1)P} \mathcal{A} \right] + \mathcal{H}(1 - B_1) \left[\frac{\beta(X_1)}{p_X(X_1)(1 - P)} \right] \right) \\ \mathcal{A} &= \mathbb{E} \left(\mathcal{H}(B_2) \left[\frac{\alpha(X_2)}{p_X(X_2)P} \mathcal{A} \right] + \mathcal{H}(1 - B_2) \left[\frac{\beta(X_2)}{p_X(X_2)(1 - P)} \right] \right)\end{aligned}\tag{6}$$

Substituting one into the other gives :

$$\mathcal{A} = \mathbb{E} \left(\mathcal{H}(B_1) \left[\frac{\alpha(X_1)}{p_X(X_1)P} \mathbb{E} \left(\mathcal{H}(B_2) \left[\frac{\alpha(X_2)}{p_X(X_2)P} \mathcal{A} \right] + \mathcal{H}(1 - B_2) \left[\frac{\beta(X_2)}{p_X(X_2)(1 - P)} \right] \right) \right] + \mathcal{H}(1 - B_1) \left[\frac{\beta(X_1)}{p_X(X_1)(1 - P)} \right] \right)\tag{7}$$

Using again the same essential idea of double randomization (the expectation of a linear function of an expectation is an expectation), we obtain

$$\mathcal{A} = \mathbb{E} \left(\mathcal{H}(B_1) \left[\frac{\alpha(X_1)}{p_X(X_1)P} \mathcal{H}(B_2) \left[\frac{\alpha(X_2)}{p_X(X_2)P} \mathcal{A} \right] + \mathcal{H}(1 - B_2) \left[\frac{\beta(X_2)}{p_X(X_2)(1 - P)} \right] \right] + \mathcal{H}(1 - B_1) \left[\frac{\beta(X_1)}{p_X(X_1)(1 - P)} \right] \right)\tag{8}$$

that is, by expanding,

$$\begin{aligned}\mathcal{A} &= \mathbb{E} \left(\mathcal{H}(B_1) \frac{\alpha(X_1)}{p_X(X_1)P} \mathcal{H}(B_2) \frac{\alpha(X_2)}{p_X(X_2)P} \mathcal{A} \right. \\ &\quad \left. + \mathcal{H}(B_1) \frac{\alpha(X_1)}{p_X(X_1)P} \mathcal{H}(1 - B_2) \frac{\beta(X_2)}{p_X(X_2)(1 - P)} \right. \\ &\quad \left. + \mathcal{H}(1 - B_1) \frac{\beta(X_1)}{p_X(X_1)(1 - P)} \right)\end{aligned}\tag{9}$$

This expansion can be extended to infinity (introducing X_3 , B_3 , X_4 , B_4 ... to replace $\mathcal{A}$ at each step) to obtain

$$\mathcal{A} = \mathbb{E} \left(\sum_{i=1}^{+\infty} \left[\prod_{j=1}^{i-1} \mathcal{H}(B_j) \frac{\alpha(X_j)}{p_X(X_j)P} \right] \mathcal{H}(1 - B_i) \frac{\beta(X_i)}{p_X(X_i)(1 - P)} \right)\tag{10}$$

which is indeed represented by algorithm 1 .

Remark on the limitations of the Iterated Kernel Method : The method described above is the iterated-kernel method, which is often referred to in the literature as a technique for obtaining a formalization of the solution as a Neumann series. It is essential to bear in mind that this method relies on specific convergence conditions that are not always satisfied. Indeed, this approach consists in expressing the solution of the integral equation as a series obtained by successive iteration of the kernel. For this series to converge, the kernel of the equation must have a "small norm" (often considered in an appropriate function space such as L^2) to guarantee absolute convergence of the series. Finally, it is also crucial to consider the nature of the integral equation itself. First-kind integral equations are particularly sensitive to convergence problems with the iterated-kernel method, whereas second-kind equations are generally more amenable to applying this method. If the approach is purely mathematical, a prior analysis of the kernel norm and its overall behavior is essential before considering application of the iterated-kernel method.

Symbolic

Let us return to the example we used for the sensitivity calculation in the introductory handout. We assume that we have a Monte Carlo algorithm estimating $\mathcal{A}(\alpha)$ as follows (see the section on integral formulation) :

$$\mathcal{A}(\alpha) = \int_{\mathcal{U}_X} p_X(x; \alpha) dx \tilde{w}(x; \alpha) \quad (11)$$

In the sensitivity calculation, the objective was to estimate the derivative $\frac{\partial \mathcal{A}}{\partial \alpha}$, that is, the derivative of the quantity under study with respect to a problem parameter. The parameter affects both the sampling distribution p_X and the weight function $\tilde{w}$. It does not affect the sample space X because sensitivity questions involving the sample space are much more difficult : they are the subject of very active research, and we can guide you toward the corresponding literature, some advances already having reached a high level of operational maturity. But as an introduction to sensitivity questions, we leave aside situations in which the parameter changes the integration domain.

Without saying so explicitly, the sensitivities we considered through the derivative $\frac{\partial \mathcal{A}}{\partial \alpha}$ were linear sensitivities. The derivative gives us the response of $\mathcal{A}$ to a small perturbation of the parameter, small (infinitesimal) in the sense that the response of $\mathcal{A}$ is linear, with the derivative $\frac{\partial \mathcal{A}}{\partial \alpha}$ then carrying all the information about this response.

The so-called symbolic Monte Carlo methods, by contrast, are concerned with perturbations such that the response of $\mathcal{A}$ to a variation in α is nonlinear. We therefore seek to characterize the function $\mathcal{A}(\alpha)$ itself, still with the idea that α takes values in the neighborhood of a reference value α_{ref} but allowing non-infinitesimal deviations $\alpha - \alpha_{ref}$. The adjective symbolic is used to indicate that a symbolic form will be proposed for $\mathcal{A}(\alpha)$ and that the Monte Carlo algorithm will produce not an estimate of $\mathcal{A}$ for a given value of α , but an estimate of the parameters of this symbolic form. Several approaches are possible, and here we describe only the one that gave rise to this approach, based on importance sampling.

When we introduced importance sampling, the idea was that one could choose sampling distributions in order to reduce the estimator variance. Starting from equation 11, one could change $p_X(x; \alpha)$ by taking another probability density $p_{X,opt}(x)$ and correcting the weight by multiplying it by the prefactor $\frac{p_X(x; \alpha)}{p_{X,opt}(x)}$. Here we use this flexibility of the method not to reduce variance, but to always use α_{ref} for sampling and therefore move the dependence on α into the weight only. We therefore choose $p_{X,"opt"}(x) = p_X(x; \alpha_{ref})$:

$$\mathcal{A}(\alpha) = \int_{\mathcal{U}_X} p_X(x; \alpha_{ref}) dx \frac{p_X(x; \alpha)}{p_X(x; \alpha_{ref})} \tilde{w}(x; \alpha) \quad (12)$$

In the corresponding Monte Carlo algorithm, we will generate n realizations x_i . The estimate of $\mathcal{A}(\alpha)$, denoted $m(\alpha)$, and the estimate of the variance of the estimator of $\mathcal{A}(\alpha)$, denoted $s(\alpha)^2$, will then be

$$\begin{aligned} m(\alpha) &= \frac{1}{n} \sum_{i=1}^n \frac{p_X(x_i; \alpha)}{p_X(x_i; \alpha_{ref})} \tilde{w}(x_i; \alpha) \\ s(\alpha)^2 &= \frac{1}{n-1} \left(\left(\frac{1}{n} \sum_{i=1}^n \left[\frac{p_X(x_i; \alpha)}{p_X(x_i; \alpha_{ref})} \tilde{w}(x_i; \alpha) \right]^2 \right) - m(\alpha)^2 \right) \end{aligned} \quad (13)$$

These two estimates are indeed symbolic forms of functions of α . This is the first possible level of a symbolic approach : using the "importance" sampling technique, we write an algorithm in which the samples no longer depend on α and we store all realizations x_i so that we can replay the estimation algorithm for any new value of α without having to reconstruct these realizations. This procedure is particularly interesting when sampling requires the greatest computational power. This is often the case. In the engineering applications we have in mind, X is often replaced by a "path" visiting several coupled physical domains involving large-dimensional databases, most often in large CAD systems, and computational power is primarily required for sampling these paths. In this case, being able to use the same paths for different values of α (using the symbolic forms of equation 13) is already useful without any additional work.

But most often, this symbolic form can be simplified by grouping terms. Let us assume, for example, that $\tilde{w}$ does not depend on α , i.e. $\tilde{w} \equiv \tilde{w}(x)$, and that $p_X(x; \alpha)$ has the following form,

$$p_X(x; \alpha) = f(x) (\mathcal{H}(x)g_1(\alpha) + (1 - \mathcal{H}(x))g_2(\alpha)) \quad (14)$$

where the Heaviside function is such that $\mathcal{H}(x) = 1$ if $x > 0$ and $\mathcal{H}(x) = 0$ otherwise, the functions f , g_1 and g_2 being known. In this case, it is easy to group the terms in equation 13 to obtain

$$\begin{aligned} m(\alpha) &= m_1 \frac{g_1(\alpha)}{g_1(\alpha_{ref})} + m_2 \frac{g_2(\alpha)}{g_2(\alpha_{ref})} \\ s(\alpha)^2 &= \frac{1}{n-1} \left(s_1 \frac{g_1(\alpha)^2}{g_1(\alpha_{ref})^2} + s_2 \frac{g_2(\alpha)^2}{g_2(\alpha_{ref})^2} - m(\alpha)^2 \right) \end{aligned} \quad (15)$$

with

$$\begin{aligned} m_1 &= \frac{1}{n} \sum_{i=1}^n \mathcal{H}(x_i) f(x_i) \tilde{w}(x_i) \\ m_2 &= \frac{1}{n} \sum_{i=1}^n (1 - \mathcal{H}(x_i)) f(x_i) \tilde{w}(x_i) \\ s_1 &= \frac{1}{n} \sum_{i=1}^n \mathcal{H}(x_i) f(x_i)^2 \tilde{w}(x_i)^2 \\ s_2 &= \frac{1}{n} \sum_{i=1}^n (1 - \mathcal{H}(x_i)) f(x_i)^2 \tilde{w}(x_i)^2 \end{aligned} \quad (16)$$

In such a case, the output of the Monte Carlo algorithm is this set of four values, m_1 , m_2 , s_1 and s_2 which are the parameters of the symbolic form of the estimator of $\mathcal{A}(\alpha)$, namely $m(\alpha)$ in equation 15, and the parameters of the symbolic form of the variance of this estimator, namely $s(\alpha)^2$ in equation 15.

It is important to remember that we have not only an estimator of $\mathcal{A}(\alpha)$ that gives us the function of α itself (analyzing its form is often rich in terms of physical interpretation), but also the level of uncertainty, which is itself a function of α . Often the uncertainty turns out to be greater when α moves away from α_{ref} , and it is sometimes necessary to increase the sample size to capture the nonlinearity of the response over a large variation range, in which case several Monte Carlo simulation calculations may be necessary for different values of α_{ref} , for example evenly distributed over the range of interest of α .

Nonlinear

When we discussed double randomization, as a means of combining expectations of nested random functions (up to infinite dimension through recursion), the random functions always involved an expectation appearing linearly. Indeed, double randomization is an idea deeply associated with linearity, to the point that most theoretical writings on the Monte Carlo method state that extension to the nonlinear world is impossible, hence swarm Monte Carlo ideas, where one accepts sacrificing some of the method's essential strengths in pursuit of nonlinearity. However, other ideas have emerged that preserve the full theoretical framework. Here we describe the main idea; the rest is either restricted to very specific physical models, for example kinetic physics, or too embryonic, for example recent resonances with the Laplace transform.

The idea appears both in the applied mathematics literature and in that of process engineering. Starting from the same type of integral formulation corresponding to a Monte Carlo algorithm, we consider a weight function $\tilde{w}_A$ that depends on a parameter $\mathcal{C}$, namely

$$\mathcal{A} = \int_{\mathcal{U}_X} p_X(x) dx \tilde{w}_A(x; \mathcal{C}) \quad (17)$$

the parameter $\mathcal{C}$ being known only in the form of an integral expression, itself corresponding to a Monte Carlo algorithm, namely

$$\mathcal{C} = \int_{\mathcal{U}_Y} p_Y(y) dy \tilde{w}_{\mathcal{C}}(y) \quad (18)$$

In terms of expectations, these two expressions mean

$$\begin{aligned} \mathcal{A} &= \mathbb{E}(\tilde{w}_{\mathcal{A}}(X; \mathcal{C})) \\ \mathcal{C} &= \mathbb{E}(\tilde{w}_{\mathcal{C}}(Y)) \end{aligned} \quad (19)$$

and if the dependence of $\tilde{w}_{\mathcal{A}}$ on $\mathcal{C}$ is linear, we can combine the two expectations using the fact that the expectation of a linear function of an expectation is an expectation :

$$[\text{cas linéaire}] \quad \mathcal{A} = \mathbb{E}(\tilde{w}_{\mathcal{A}}(X; \tilde{w}_{\mathcal{C}}(Y))) \quad (20)$$

Thus it is no longer necessary to begin by estimating $\mathcal{C}$ using its Monte Carlo algorithm before using the algorithm for $\mathcal{A}$, replacing $\mathcal{C}$ by its estimate (which would in any case introduce a bias). We can directly use a new algorithm that combines the two previous ones, where each weight w is constructed using a realization x of X and a realization y of Y by combining the two weight functions : $w = \tilde{w}_{\mathcal{A}}(x; \tilde{w}_{\mathcal{C}}(y))$.

But this holds only under the assumption of linearity. If $\tilde{w}_{\mathcal{A}}$ depends nonlinearly on $\mathcal{C}$ this is not possible. However, we can consider the special case where the nonlinearity can be expanded in a power series. We now assume that $\tilde{w}_{\mathcal{A}}(x; \mathcal{C})$ can be written as

$$\tilde{w}_{\mathcal{A}}(x; \mathcal{C}) = \sum_{i=0}^{+\infty} a_i(x) (\mathcal{C} - \mathcal{C}_{ref})^i \quad (21)$$

where $\mathcal{C}_{ref}$ is a known quantity and the a_i are known functions of x . One possible strategy is then as follows :

- probabilize the successive sums of the series ;
- treat the power $(\mathcal{C} - \mathcal{C}_{ref})^i$ as a product with i terms and in each term j replace $\mathcal{C}$ by the expectation $\mathbb{E}(\tilde{w}_{\mathcal{C}}(Y_j))$ where the Y_j are IID random variables with the same distribution as Y .

Probabilizing the series We introduce a set of probabilities $P_0, P_1, P_2 \dots$ such that $\sum_{i=0}^{+\infty} P_i = 1$. This defines a random variable I such that $I = 0$ with probability P_0 , $I = 1$ with probability P_1 , etc. We can then write

$$\tilde{w}_{\mathcal{A}}(x; \mathcal{C}) = \sum_{i=0}^{+\infty} P_i \left[\frac{a_i(x)}{P_i} (\mathcal{C} - \mathcal{C}_{ref})^i \right] \quad (22)$$

which can be interpreted as a Monte Carlo algorithm in which we generate one realization i of I and the Monte Carlo weight is $w = \frac{a_i(x)}{P_i} (\mathcal{C} - \mathcal{C}_{ref})^i$. In terms of expectations,

$$\tilde{w}_{\mathcal{A}}(x; \mathcal{C}) = \mathbb{E} \left(\frac{a_I(x)}{P_I} (\mathcal{C} - \mathcal{C}_{ref})^I \right) \quad (23)$$

Unfolding the power The power-series expansion summarizes all the nonlinearity in a power $(\mathcal{C} - \mathcal{C}_{ref})^i$. Equation 24 allows us to write $\mathcal{C}$ in integral form, and since p_Y is a probability density on $\mathcal{U}_Y$, we also know that $\int_{\mathcal{U}_Y} p_Y(y) dy = 1$, hence

$$\mathcal{C}_{ref} = \int_{\mathcal{U}_Y} p_Y(y) dy \mathcal{C}_{ref} \quad (24)$$

We can then write

$$\mathcal{C} - \mathcal{C}_{ref} = \int_{\mathcal{U}_Y} p_Y(y) dy (\tilde{w}_{\mathcal{C}}(y) - \mathcal{C}_{ref}) \quad (25)$$

that is,

$$\mathcal{C} - \mathcal{C}_{ref} = \mathbb{E}(\tilde{w}_{\mathcal{C}}(Y) - \mathcal{C}_{ref}) \quad (26)$$

The important step at this point is to define a sequence $Y_1, Y_2, Y_3, \dots$ of IID random variables with the same distribution as Y . For every i we can then write

$$\begin{aligned} (\mathcal{C} - \mathcal{C}_{ref})^i &= \left(\int_{\mathcal{U}_Y} p_Y(y_1) dy_1 (\tilde{w}_{\mathcal{C}}(y_1) - \mathcal{C}_{ref}) \right) \left(\int_{\mathcal{U}_Y} p_Y(y_2) dy_2 (\tilde{w}_{\mathcal{C}}(y_2) - \mathcal{C}_{ref}) \right) \dots \left(\int_{\mathcal{U}_Y} p_Y(y_i) dy_i (\tilde{w}_{\mathcal{C}}(y_i) - \mathcal{C}_{ref}) \right) \\ &= \int_{\mathcal{U}_Y} p_Y(y_1) dy_1 \int_{\mathcal{U}_Y} p_Y(y_2) dy_2 \dots \int_{\mathcal{U}_Y} p_Y(y_i) dy_i (\tilde{w}_{\mathcal{C}}(y_1) - \mathcal{C}_{ref}) (\tilde{w}_{\mathcal{C}}(y_2) - \mathcal{C}_{ref}) \dots (\tilde{w}_{\mathcal{C}}(y_i) - \mathcal{C}_{ref}) \end{aligned} \quad (27)$$

that is,

$$(\mathcal{C} - \mathcal{C}_{ref})^i = \mathbb{E} \left(\prod_{j=1}^i (\tilde{w}_{\mathcal{C}}(Y_j) - \mathcal{C}_{ref}) \right) \quad (28)$$

The nonlinearity is thus reduced to a single expectation.

Substituting the unfolded power into the probabilized series We can then substitute this single expectation into equation 23 and substitute the whole expression into $\mathcal{A}$ in equation 19, using double randomization at each of these two steps, to obtain an expression equivalent to that of equation 20, but which now holds in the nonlinear case, namely

$$\begin{aligned} \tilde{w}_{\mathcal{A}}(x; \mathcal{C}) &= \mathbb{E} \left(\frac{a_I(x)}{P_I} \mathbb{E} \left(\prod_{j=1}^I (\tilde{w}_{\mathcal{C}}(Y_j) - \mathcal{C}_{ref}) \right) \right) \\ &= \mathbb{E} \left(\frac{a_I(x)}{P_I} \prod_{j=1}^I (\tilde{w}_{\mathcal{C}}(Y_j) - \mathcal{C}_{ref}) \right) \end{aligned} \quad (29)$$

and

$$\mathcal{A} = \mathbb{E} \left(\mathbb{E} \left(\frac{a_I(X)}{P_I} \prod_{j=1}^I (\tilde{w}_{\mathcal{C}}(Y_j) - \mathcal{C}_{ref}) \right) \right) \quad (30)$$

therefore

$$[\text{cas non-linéaire}] \quad \mathcal{A} = \mathbb{E} \left(\frac{a_I(X)}{P_I} \prod_{j=1}^I (\tilde{w}_C(Y_j) - \mathcal{C}_{ref}) \right) \quad (31)$$

The corresponding algorithm consists of generating one realization x of X , then generating one realization i of I (the order of the series), and then generating i realizations $y_1, y_2 \dots y_i$ of the parent random variables $Y_1, Y_2 \dots Y_i$, with the Monte Carlo weight being $w = \frac{a_i(x)}{P_i} \prod_{j=1}^i (\tilde{w}_C(Y_j) - \mathcal{C}_{ref})$.

Differential equations : introduction

In the preceding discussion, we started from an integral expression without specifying the origin of this expression. Monte Carlo always starts from an existing integral formulation, but in some cases it is useful to retain the memory of the formal developments that led to establishing this formulation. This is particularly the case when the integral quantity under study is the solution of a differential equation. We may then wish to preserve the meaning of the formal ingredients appearing in the integral expression, for example the fact that they represent an initial condition, a boundary condition, a source, a propagator, etc. We illustrate this point here with the example of a linear transport equation. ce point avec l'exemple d'une équation de transport linéaire.

We consider a geometric space represented by a coordinate x in $\mathbb{R}^+$, we introduce an observation time t also in $\mathbb{R}^+$ and consider a function f of x and t , namely $f \equiv f(x, t)$. This function is defined as the solution of the following partial differential equation,

$$\frac{\partial f}{\partial t} + c \frac{\partial f}{\partial x} = -\nu(f - f^*) \quad (32)$$

with the initial condition

$$f = f_I \quad (33)$$

and the boundary condition

$$f = f_L \quad (34)$$

where the collision frequency ν and the velocity c are independent of space and time and are known, the initial condition $f_I \equiv f_I(x)$ is a known function of space, the boundary condition $f_L \equiv f_L(t)$ is a known function of time and the source $f^* \equiv f^*(x, t)$ is a known function of space and time.

The solution of this differential equation can be formulated as follows (this is the solution of a typical linear Boltzmann equation in neutron transport and radiative transfer : we do not give the derivation) :

$$\begin{aligned} f(x, t) = \int_0^{+\infty} \nu \exp(-\nu t') dt' \{ \\ H(t' - t) \{ H(x - ct) f_I(x - ct) + H(ct - x) f_L(t - x/c) \} \\ + H(t - t') \{ H(x - ct') f^*(x - ct', t - t') + H(ct' - x) f_L(t - x/c) \} \\ \} \end{aligned} \quad (35)$$

where the function H is the Heaviside function.

In this integral formulation, we recognize the probability density $p_{T'}(t') = \nu \exp(-\nu t')$ of a random variable T' distributed according to an exponential law with parameter ν (the sampling of a collision time). It is constructed in a time-reversal logic, therefore $T' = t$ corresponds at the initial instant to $t = 0$. Thus, the test

$H(T' - t)$ is interpreted as "the collision does not occur because the initial condition was encountered before the collision time." Likewise, still within this time-reversal logic, the "particle" moves at the velocity $-c$ and the test $H(cT' - x)$ is interpreted as "the collision does not occur because the boundary condition (at $x = 0$) was encountered before the collision location)".

The two points to remember here are the following :

- We started from a differential equation, wrote its solution in integral form, and the form of this integral (inspired by work on the Boltzmann equation) is directly that of an expectation : celle d'une espérance :

$$f(x, t) = \mathbb{E}(\tilde{w}(T')) \quad (36)$$

with

$$\tilde{w}(t') = H(t' - t) \{ H(x - ct) f_I(x - ct) + H(ct - x) f_L(t - x/c) \} + H(t - t') \{ H(x - ct') f^*(x - ct', t - t') + H(ct' - x) f_L(t - x/c) \} \quad (37)$$

the Monte Carlo algorithm consists in sampling T' and then performing tests and, depending on these tests, retaining f_I , f_L or f^* .

- There is an intuitive interpretation associated with this algorithm in terms very directly inspired by Green's work on linear partial differential equations : we recognize the exponential propagator of the transport equations, and its statistical translation amounts to sampling the three types of "sources" (in Green's sense) : the initial condition, the boundary condition, or the volumetric source.

Differential equations : recursion

While retaining the possibility of inspiration from the physics of linear transport, it is easy to extend the class of partial differential equations that can be approached in the same way, by adding only the idea of recursion. This is notably the case for all particle scattering problems (scattering in English), where the source term f^* from the previous section itself is expressed as a function of another unknown quantity, typically of the function f in another direction. In that case, the formulation remains the same, but f^* involves a quantity for which we also have an integral formulation, and it is sufficient to nest the two, recursively, as we did in the Recursion section.

Let us take a very simple example, with only two directions and with two functions : f^+ for the direction of increasing x , f^- for the direction of decreasing x . We are interested in the solution of the following system of two coupled equations :

$$\begin{cases} \frac{\partial f^+}{\partial t} + c \frac{\partial f^+}{\partial x} = -\nu(f^+ - f^-) \\ \frac{\partial f^-}{\partial t} - c \frac{\partial f^-}{\partial x} = -\nu(f^- - f^+) \end{cases} \quad (38)$$

with the initial conditions

$$\begin{cases} f^+ = f_I^+ \\ f^- = f_I^- \end{cases} \quad (39)$$

and the boundary conditions

$$\begin{cases} f^+ = f_{LG}^+ \\ f^- = f_{LD}^- \end{cases} \quad (40)$$

where the collision frequency ν and the velocity c are independent of space and time and are known, the initial conditions $f_I^+ \equiv f_I^+(x)$ and $f_I^- \equiv f_I^-(x)$ are known functions of space for every $x \in [0, a]$ (where a is the length of the interval under study), the boundary conditions $f_{LG}^+ \equiv f_{LG}^+(t)$ at $x = 0$ and $f_{LD}^- \equiv f_{LD}^-(t)$ at $x = a$ are known functions of time.

Each of these two equations can be treated as in the previous section, with $f^* = f^-$ for the first and $f^* = f^+$ for the second. We then use recursion to write a single Monte Carlo algorithm that switches from the algorithm for f^+ to that for f^- , and conversely, each time the volumetric source f^* is sampled.

Differential equations : nonlinear

In the same spirit as in the previous section, we can add a nonlinearity (inspired by reactive chemistry) :

$$\begin{cases} \frac{\partial f^+}{\partial t} + c \frac{\partial f^+}{\partial x} = -\nu(f^+ - \beta f^+ f^-) \\ \frac{\partial f^-}{\partial t} - c \frac{\partial f^-}{\partial x} = -\nu\beta f^+ f^- \end{cases} \quad (41)$$

The preceding recursive approach must then be combined with the fact that $\nu\beta f^+ f^-$ induces a nonlinearity, hence branching in the sense in which we treated it in the Nonlinear section.

This is exactly how the Monte Carlo method can be used, without bias, to solve the nonlinear Boltzmann equation (see the work of Guillaume Terrée).

TP

TP7

Estimate the following integral :

$$\mathcal{A}(p, q, \gamma) = \int_0^1 dx \left(\gamma x \mathcal{A}(p, q, \gamma) + (1 - \gamma x) \int_0^x dy x^p y^q \right) \quad (42)$$

where $\gamma \in [0, 1]$, which makes it possible to interpret the product γx as a probability and proceed towards an algorithm involving a Bernoulli random variable B with probability $P = \gamma x$. The success or failure of the Bernoulli test leads either to recursion (we must estimate $\mathcal{A}$ again), or to termination of the recursion (with the sampling of a y on $[0, x]$ according to a distribution to be chosen).

Solution to TP7 A code corresponding to TP7 is provided in the directory `/home/lafrier/livre_informatique_graphique/Codes/somme_xp_yq_recuratif`.

TP8

Modify the TP7 algorithm so as to simultaneously estimate the sensitivity of $\mathcal{A}$ to γ .

Solution to TP8 A code corresponding to TP8 is provided in the directory `/home/lafrier/livre_informatique_graphique/Codes/sensib_somme_xp_yq_recuratif`.

TP9

Design a symbolic Monte Carlo algorithm for the nonlinear dependence of $\mathcal{A}$ on γ .

TP10

Revisit TP7 by modifying the definition of γ , which now becomes a function of x :

$$\gamma(x) = \int_0^x z^k dz \quad (43)$$

We will choose not to precompute this integral. Instead, we will begin by designing a Monte Carlo algorithm that estimates $\gamma(x)$ for a given value of x . The objective will then be to combine this algorithm with the TP7 algorithm, knowing that the combination of the two corresponding expectations is nonlinear.

Solution to TP10 A code corresponding to TP10 is provided in the directory `/home/lafrier/livre_informatique_graphique/Codes/somme_xp_yq_recuratif_non_lineaire`.

TP11

Code the algorithm described in the section *Differential equations : introduction*

TP12

Write the integral formulation, then design and code the algorithm corresponding to the linear system of the two differential equations described in the section *Differential equations : recursion*

TP13

Write the integral formulation, then design and code the algorithm corresponding to the nonlinear system of the two differential equations described in the section *Differential equations : nonlinear*

