

A spectral Monte Carlo method for the Poisson equation

Emmanuel Gobet* and Sylvain Maire†

December 4th, 2003

ABSTRACT. Using a sequential Monte Carlo algorithm, we compute a spectral approximation of the solution of the Poisson equation in dimension 1 and 2. The Feynman-Kac computation of the pointwise solution is achieved using either an integral representation or a modified walk on spheres method. The variances decrease geometrically with the number of steps. A global solution is obtained, accurate up to the interpolation error. Surprisingly, the accuracy depends very little on the absorption layer thickness of the walk on spheres.

KEYWORDS : Spectral method, Sequential Monte Carlo, Poisson equation, Variance reduction, Modified walk on spheres, Feynman-Kac formula.

1 Introduction

The Feynman-Kac formula is a very powerful tool to achieve stochastic representations of the pointwise solution of numerous partial differential equations like, for instance, diffusion or transport equations [Fri76][Fre85][CDL⁺89][LPS98]. Consider the Dirichlet boundary value problem in a domain $D \subset \mathbb{R}^d$:

$$\begin{cases} -\frac{1}{2}\Delta u = f & \text{in } D, \\ u = g & \text{on } \partial D. \end{cases} \quad (1)$$

Under some assumptions, we have $\forall x \in D$

$$u(x) = \mathbf{E}_x(g(B_{\tau_D}) + \int_0^{\tau_D} f(B_s)ds), \quad (2)$$

where $B = (B_t)_{t \geq 0}$ is a standard d -dimensional Brownian motion and where τ_D is the exit time of this process from the domain D . The Monte Carlo computation of this pointwise solution leads to two kinds of numerical errors. The first one is due to the simulation error of the Brownian motion B . Different approximation schemes can be used [Bal95][Gob01], even for more general stochastic processes, but for the Brownian case, the most efficient ones are the walk on spheres (WOS) methods. We refer to [Mul56][Sab91] and to the recent work by Hwang et al.[HMG03], which allows to keep the advantages of the original method even with a source term f . The second kind of error is the standard Monte Carlo error using N simulations, that is $\frac{\sigma}{\sqrt{N}}$, where σ is the standard deviation of the approximation of $g(B_{\tau_D}) + \int_0^{\tau_D} f(B_s)ds$.

*CMAP, Ecole Polytechnique, 91128 Palaiseau cedex, France. Email: emmanuel.gobet@polytechnique.fr

†ISITV, Université de Toulon et du Var, avenue G. Pompidou, BP56, 83262 La Valette du Var cedex, France. Email:mair@univ-tln.fr

Our goal here is to try to reduce the Monte Carlo error, using an adaptive Monte Carlo algorithm based on the ideas of sequential Monte Carlo methods [Hal62]. These methods have been used for Markov chains [BCP00], transport equations [Boo85] and recently for numerical integration [Mai03]. They rely on an intelligent use of the random drawings from a step to another which allows to reduce geometrically the variance up to an approximation error. We intend to use the same idea here to compute a spectral approximation [CHQZ88][BM92] of the solution of the Poisson equation.

We will perform this method on square domains in dimension one and two, using Tchebychef polynomials in the approximation algorithm. On an interval, we can make the simulation using an integral representation. This allows to get totally rid of the discretization error. In dimension 2, we will use the modified WOS method developed in [HMG03]. Its accuracy depends on the absorption layer thickness ε which can be taken very small without increasing too much the CPU times.

We will see that variances decrease geometrically with the number of iterations, and surprisingly, the bias due to the discretization error behaves analogously. Finally, this will lead to a global and accurate Monte Carlo approximation of the solution of the Poisson equation.

2 Description of the approximation algorithm

Consider the Poisson equation (1). We can compute approximate values $u_i^{(1)}$ of the solution of this equation at some points x_i using a Monte Carlo method based on the representation (2). Using this information, we can then build a global and regular approximation $u^{(1)}(x)$ of the Poisson equation. This can be achieved, for example, by a simple polynomial interpolation or by fitting the data to a polynomial model. In sections 3.2 and 4.2 below, we will mainly use global Tchebychef polynomial interpolations built from pointwise computations of the solution at the Tchebychef abscissae over $[-1, 1]^d$, in dimension one or two. We can now use the control variate method with $u^{(1)}$ as an approximation of u . We put

$$y = u - u^{(1)}$$

which solves the new Poisson equation

$$\begin{cases} -\frac{1}{2}\Delta y = -\frac{1}{2}\Delta(u - u^{(1)}) = f + \frac{1}{2}\Delta u^{(1)} & \text{in } D, \\ y = g - u^{(1)} & \text{on } \partial D. \end{cases}$$

A global solution $y^{(1)}$ can be computed using the same method than the one used to compute $u^{(1)}$. If we now approximate the solution of the initial equation by

$$u^{(2)}(x) = u^{(1)}(x) + y^{(1)}(x),$$

we can expect this solution to be more accurate than the previous one. We can then iterate this method to achieve an approximation $u^{(n)}(x)$ at the n th step of the relevant algorithm.

A basic example

We aim at illustrating how fast the convergence of the sequential algorithm above can be. For this, we consider the most simple Poisson equation, that is

$$\begin{cases} u''(x) = 0, & x \in]0, 1[, \\ u(0) = 0, & u(1) = 1. \end{cases}$$

The solution being $u(x) = \mathbf{P}_x(B_{\tau_D} = 1) = x$, we chose to compute it at two points (say $x_1 = \frac{1}{4}$ and $x_2 = \frac{3}{4}$ for instance) and use a linear interpolation to get the values elsewhere.

At the n th step, we assume that unbiased estimators $p_1^{(n)}$ and $p_2^{(n)}$ of $\mathbf{P}_{x_1}(B_{\tau_D} = 1)$ and $\mathbf{P}_{x_2}(B_{\tau_D} = 1)$ are computed using N simulations of Bernoulli variables. Moreover, all simulations are independently drawn. In particular, we have

$$\mathbf{E}(p_1^{(n)}) = \frac{1}{4}, \quad \mathbf{E}(p_2^{(n)}) = \frac{3}{4}, \quad \text{Var}(p_1^{(n)}) \leq \frac{C}{N}, \quad \text{Var}(p_2^{(n)}) \leq \frac{C}{N}. \quad (3)$$

Step 1. We start with $u_1^{(1)} = p_1^{(1)}$ and $u_2^{(1)} = p_2^{(1)}$. Then, we compute a global approximation $u^{(1)}(x)$ as

$$u^{(1)}(x) = 2u_2^{(1)}\left(x - \frac{1}{4}\right) - 2u_1^{(1)}\left(x - \frac{3}{4}\right).$$

It is easy to check $u^{(1)}(x)$ is unbiased and has a variance of magnitude $\frac{1}{N}$.

Step 2. We now put

$$y(x) = u(x) - u^{(1)}(x)$$

and we have to solve

$$\begin{cases} -\frac{1}{2}\Delta y = -\frac{1}{2}\Delta(u - u^{(1)}) = 0, \\ y(0) = u(0) - u^{(1)}(0) = -u^{(1)}(0), \quad y(1) = u(1) - u^{(1)}(1) = 1 - u^{(1)}(1). \end{cases}$$

We still compute the solution at x_1 and x_2 using the previous method and we now have

$$\begin{aligned} \tilde{y}\left(\frac{1}{4}\right) &= [1 - u^{(1)}(1)]p_1^{(2)} - u^{(1)}(0)(1 - p_1^{(2)}), \\ \tilde{y}\left(\frac{3}{4}\right) &= [1 - u^{(1)}(1)]p_2^{(2)} - u^{(1)}(0)(1 - p_2^{(2)}), \\ y^{(1)}(x) &= -2\tilde{y}\left(\frac{1}{4}\right)\left(x - \frac{3}{4}\right) + 2\tilde{y}\left(\frac{3}{4}\right)\left(x - \frac{1}{4}\right). \end{aligned}$$

The approximation at step two is

$$u^{(2)}(x) = u^{(1)}(x) + y^{(1)}(x).$$

To study the performances of the algorithm, we only have to look at $u^{(2)}(0)$ and $u^{(2)}(1)$ because the approximation at the next step will only depend on these values. After some computations, we obtain

$$u^{(2)}(0) = \left[\frac{3}{2}p_1^{(2)} - \frac{1}{2}p_2^{(2)}\right][1 + u^{(1)}(0) - u^{(1)}(1)].$$

It comes $\mathbf{E}(u^{(2)}(0)) = 0$ and $\text{Var}(u^{(2)}(0)) \leq \frac{C}{N^2}$. Similarly, we have $\mathbf{E}(u^{(2)}(1)) = 1$ and $\text{Var}(u^{(2)}(1)) \leq \frac{C}{N^2}$. Then, because of the linear interpolation, it follows that

$$\mathbf{E}(u^{(2)}(x)) = x \quad \text{and} \quad \text{Var}(u^{(2)}(x)) \leq \frac{C'}{N^2}.$$

This shows that the variance goes to zero geometrically with the number of steps if we take sufficiently many random drawings in the computation of the solution at the points x_1 and x_2 . Indeed, the error is

multiplied by $\frac{C}{\sqrt{N}}$ at each step of the algorithm. We have observed exactly the same phenomenon in [Mai03] where the same ideas were used to compute iteratively the approximation

$$f(x) = \sum_{k=1}^p a_k e_k(x) + r(x)$$

of a function f on an orthonormal basis $(e_k)_k$. If the function f is in the approximation space, that is $r(x) \equiv 0$, then the variance of the Monte Carlo estimators of the coefficients a_k goes to zero geometrically with the number of steps. In the general case, the approximation of these coefficients is accurate up to the truncation error. We will see in the numerical experiments that the approximate solution of the Poisson equation will verify the same property. We will reach exactly the same accuracy than the interpolation polynomial of the exact solution at the Tchebychev abscissae.

3 The monodimensional case

We address the evaluation of the solution of the Poisson equation

$$-\frac{1}{2}u'' = f \text{ on }]-1, 1[\quad (4)$$

with boundary conditions $u(-1) = a$ and $u(1) = b$, using the method described before.

3.1 Monte-Carlo computation of the pointwise solution using an integral representation

The probabilistic solution is given by

$$u(x) = a\mathbf{P}_x(B_{\tau_D} = -1) + b(1 - \mathbf{P}_x(B_{\tau_D} = -1)) + \mathbf{E}_x\left(\int_0^{\tau_D} f(B_s)ds\right).$$

As $\mathbf{P}_x(B_{\tau_D} = -1) = \frac{1-x}{2}$, the contribution of the boundary conditions to the solution can be easily simulated. To make the Monte Carlo simulation of the source term possible, we note that

$$\mathbf{E}_x\left(\int_0^{\tau_D} f(B_s)ds\right) = (1-x^2)\mathbf{E}(f(Y_x)) \quad (5)$$

where Y_x is a random variable with density

$$\frac{1+r}{1+x}\mathbf{1}_{-1 \leq r \leq x} + \frac{1-r}{1-x}\mathbf{1}_{x \leq r \leq 1}.$$

Indeed, an easy computation shows that the r.h.s. of (5) solves the Poisson equation with $a = b = 0$. To simulate Y_x , first consider the Bernoulli variable Z_x such that

$$\mathbf{P}(Z_x = 0) = \int_{-1}^x \frac{1+r}{1+x} dr = \frac{1}{2}(1+x).$$

Then, define the random variables A_x and B_x which have for density respectively

$$\frac{2(1+r)}{(1+x)^2}\mathbf{1}_{-1 \leq r \leq x} \quad \text{and} \quad \frac{2(1-r)}{(1-x)^2}\mathbf{1}_{x \leq r \leq 1}.$$

Using the inverse of their cumulative distribution function, they can be simulated respectively by $-1 + (x+1)\sqrt{U}$ and $1 + (x-1)\sqrt{U}$ where U is uniform on $[0, 1]$. We finally set

$$Y_x = (1 - Z_x)A_x + Z_x B_x.$$

3.2 Tchebychef interpolation

Tchebychef polynomials are the orthogonal polynomials on $[-1, 1]$ with respect to the inner product $\langle P, Q \rangle = \int_{-1}^1 \frac{P(x)Q(x)}{\sqrt{1-x^2}} dx$. Their expressions are given by $T_n(x) = \cos(n \arccos(x))$. We define $\mathcal{P}_N([-1, 1])$ as the linear space of the polynomials of degree $\leq N$ on $[-1, 1]$. The orthogonal projection $\pi_N(u)$ of a function $u \in L^2([-1, 1])$ on this space is

$$\pi_N(u) = \sum_{n=0}^N \mu_n T_n \quad \text{with} \quad \mu_n = \frac{1}{\|T_n\|^2} \int_{-1}^1 \frac{u(x)T_n(x)}{\sqrt{1-x^2}} dx.$$

If we assume that $u \in C^{2m}([-1, 1])$, the coefficients μ_n decrease very quickly as $\frac{C}{n^{2m}}$. If we replace $\pi_N(u)$ by the interpolation polynomial $P_N(u)$ of the function u at the Tchebychef abscissae

$$x_k = \cos\left(\frac{2k+1}{N+1} \frac{\pi}{2}\right), \quad k = 0, 1, \dots, N,$$

the approximation is still as good [BM92]. Moreover, this interpolation has optimal properties with respect to the sup norm. We have [Bjö96]

$$P_N(u) = \sum_{n=0}^N \alpha_n T_n \quad \text{with} \quad \alpha_n = \frac{\pi}{\|T_n\|^2 (N+1)} \sum_{k=0}^N u(x_k) T_n(x_k).$$

We will use this approximation for the solution of the Poisson equation. We also need to compute the second derivative of $P_N(u)$ to build the new Poisson equation. We have [CHQZ88]

$$(P_N(u))'' = \sum_{n=0}^{N-2} \beta_n T_n \quad \text{with} \quad \beta_k = \sum_{p=k+2, p+k \text{ even}}^N p(p^2 - k^2) \alpha_p.$$

3.3 Numerical results

The initial global approximation of the solution of the Poisson equation is given by

$$u^{(1)} = \sum_{n=0}^N \bar{\alpha}_n T_n$$

where the coefficients $\bar{\alpha}_n$ are built using the Monte Carlo approximations $u_k^{(1)}$ of $u(x_k)$. We also have

$$(u^{(1)})'' = \sum_{n=0}^{N-2} \bar{\beta}_n T_n \quad \text{with} \quad \bar{\beta}_k = \sum_{p=k+2, p+k \text{ even}}^N p(p^2 - k^2) \bar{\alpha}_p.$$

The algorithm is efficient if the Monte Carlo computation of the correction

$$y(x) = u(x) - u^{(1)}(x)$$

is more accurate than the direct computation of $u(x)$. This computation is achieved by solving a new Poisson equation where the source term becomes

$$f + \frac{1}{2} \Delta(u^{(1)}) = f + \frac{1}{2} \sum_{n=0}^{N-2} \bar{\beta}_n T_n$$

and where the boundary conditions are

$$y(-1) = a - \sum_{n=0}^N \overline{\alpha}_n T_n(-1), \quad y(1) = b - \sum_{n=0}^N \overline{\alpha}_n T_n(1).$$

This shows that the boundary conditions will be close to zero even if the computations of the $\overline{\alpha}_n$ are not very accurate. Because of the derivation of the interpolation polynomial, a rather accurate approximation of the $\overline{\beta}_k$ could not be achieved unless the approximations of the $\overline{\alpha}_n$ are accurate enough. This is especially true when N increases. We will need a lot more sample values to make the algorithm converge in this case. We now give a numerical example to show the efficiency of the algorithm.

Example. We study the equation (4), with $a = \frac{1}{e}$ and $b = e$, so that the solution of this equation is $u(x) = \exp(x)$. We give in the following table the error

$$e(j) = \sup_{0 \leq k \leq N} \left\| u(x_k) - u^{(j)}(x_k) \right\|$$

as a function of the number j of steps and of the number M of sample values to compute the pointwise approximation at the x_k . The accuracy of the crude Monte Carlo method with M sample values is given by $e(0)$. For a given value of M , L is the number of steps until $[e(j)]_{j \geq 0}$ stabilize. We give the CPU time until convergence in seconds.

N	M	L	$e(0)$	$e(5)$	$e(L)$	CPU
5	100	15	8×10^{-2}	2×10^{-3}	1×10^{-4}	0.01
7	300	24	9×10^{-2}	4×10^{-3}	4×10^{-6}	0.1
10	1000	72	5×10^{-2}	2×10^{-3}	6×10^{-10}	0.9
13	5000	63	4×10^{-2}	2×10^{-3}	8×10^{-15}	6.8

M has to be chosen large enough with respect to N to make the algorithm converge. For example, if we take only $M = 100$ in the case $N = 7$, the algorithm diverges. We have checked that the error at convergence $e(L)$ corresponds exactly to the interpolation error of the interpolation polynomial $P_N(u)$ of the exact solution at the Tchebychef abscissae. This accuracy is obviously out of reach of the standard Monte Carlo method. The interpolation polynomial at convergence

$$P_N^{(L)}(u) = \sum_{n=0}^N \alpha_n^{(L)} T_n$$

already gives an accurate pointwise approximation of the solution u at any point of the domain. If one wants to achieve an even more accurate approximation at a given point, one can use once more the control variate method after convergence.

3.4 Quasi-Monte Carlo acceleration

In order to speed up the convergence of the algorithm, we will now replace the random drawings in the numerical computation of the correction on the source term by low-discrepancy sequences. Indeed, these quasi-random sequences achieve a rate of convergence of $\frac{(\log(N))^{(d-1)}}{N}$ for numerical integration in dimension d instead of $\frac{1}{\sqrt{N}}$ for crude Monte Carlo integration [Nie92][KU98]. This mean that we can

expect that the accuracy on this correction will be a lot better using these sequences. We now give the numerical results on the previous example based on Halton sequences.

N	M	L	$e(0)$	$e(5)$	$e(L)$
5	100	10	9×10^{-2}	2×10^{-3}	3×10^{-5}
7	300	20	5×10^{-2}	6×10^{-4}	1×10^{-6}
10	1000	38	5×10^{-2}	6×10^{-3}	7×10^{-11}
13	5000	40	3×10^{-2}	1×10^{-3}	5×10^{-15}

We can see, on this example, that the number of steps until convergence has been divided by around two. This can be heuristically explained because the accuracy on the correction at each step is multiplied by a $O(\frac{1}{N})$ instead of a $O(\frac{1}{\sqrt{N}})$ for the Monte Carlo method. This also explains why we obtain a better accuracy on $e(L)$.

4 The bidimensional case on $D =]-1, 1[^2$

4.1 Description of the modified WOS

Even if we only focus on the Poisson equation, we should make our approach as global as possible. We can not use the Green function for any domain D because it is usually unknown. In order to be as fast and as accurate as possible in the simulation of

$$g(B_{\tau_D}) + \int_0^{\tau_D} f(B_s) ds,$$

we will replace this Green function by a modified WOS method [HMG03] which can take into account the source term f . In the original WOS method [Boo82][Sab91], we walk from x to the boundary ∂D from a sphere to another until the motion reaches the ε -absorption layer (which occurs in n random steps). The spheres are built so that the jumps are as large as possible. The radius of the next sphere from a starting point x_n is $d(x_n, \partial D)$. The next point is chosen uniformly on this sphere because of the isotropy of the Brownian motion. To compute the contribution of the source term in this walk for a problem in dimension two, it is proposed in [HMG03] to compute this contribution in each of the balls from the passage to their centers to the boundary, conditioned by the exit point. This is achieved using the explicit expression of the Green function conditioned by the exit point z .

For the centered unit ball $\mathcal{S} = \mathcal{B}_1$, we can write

$$\mathbf{E}\left(\int_0^{\tau_S} f(B_s) ds \mid B_0 = 0, B_{\tau_S} = z\right) = \mathbf{E}(f(Y))$$

where $Y = (R \cos(\theta), R \sin(\theta))$ with (R, θ) having some appropriate distributions. Namely, the cumulative radial distribution function is

$$f_R(r) = r^2(1 - 2 \ln(r)) 1_{0 \leq r \leq 1}$$

and the cumulative conditional angular distribution function is

$$f_{\theta|R=r}(\theta) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{1+r \tan(\theta - \theta_0)}{1-r} \frac{1}{2}\right) 1_{-\pi \leq \theta \leq \pi}$$

where $\exp(i\theta_0) = z$. More generally, for a ball $S = \mathcal{B}_j$ of radius r_j centered at (x_j, y_j) , we have

$$\mathbf{E}\left(\int_0^{\tau_S} f(B_s) ds \mid B_0 = (x_j, y_j), B_{\tau_S} = z\right) = \frac{r_j^2}{2} \mathbf{E}(f(Y_j)) \quad (6)$$

with $Y_j = (r_j R \cos(\theta) + x_j, r_j R \sin(\theta) + y_j)$. The Monte Carlo computation of $\mathbf{E}_x(g(B_{\tau_D}) + \int_0^{\tau_D} f(B_s) ds)$ is then achieved as $\frac{1}{N} \sum_{i=1}^N Z_i$ where

$$Z_i = g(B_{\tau_D}^{(i)}) + \sum_{j=1}^{n_i} \frac{[r_j^i]^2}{2} f(Y_j^{(i)}),$$

using one simulation to evaluate the expectation (6).

4.2 Description of the approximation

The Tchebychef approximation of a function $u \in L^2([-1, 1]^2)$ is built using the same process than in dimension one. We give directly the interpolation polynomial $P_N(u)$ at the points of the Tchebychef grid of this function :

$$P_N(u) = \sum_{n=0}^N \sum_{m=0}^N \alpha_{n,m} T_n T_m$$

where the $\alpha_{n,m}$ are defined by

$$\alpha_{n,m} = \frac{\pi^2}{\|T_n\|^2 \|T_m\|^2 (N+1)^2} \sum_{k=0}^N \sum_{j=0}^N u(x_k, x_j) T_n(x_k) T_m(x_j).$$

The quality of this approximation is studied very precisely in [BM92]. We have

$$\Delta P_N(u) = \sum_{n=0}^N \sum_{m=0}^N \alpha_{n,m}^{(2)} T_n T_m$$

with

$$\alpha_{n,m}^{(2)} = \sum_{p=n+2, p+n \text{ even}}^N p(p^2 - n^2) \alpha_{p,m} + \sum_{p=m+2, p+m \text{ even}}^N p(p^2 - m^2) \alpha_{n,p}.$$

4.3 Numerical results

We still study a Poisson equation where the solution is very regular. The complexity of our method increases compared to the one-dimensional case. The pointwise solution will have to be computed at $(N+1)^2$ points instead of $N+1$ and the source term will be constituted of a $O(N^2)$ terms instead of a $O(N)$. The WOS method is obviously more consuming than the method based on the integral representation. Nevertheless, we will see that the number of sample values M to make the algorithm converge, remains approximately the same.

Example. Consider the equation

$$-\frac{1}{2} \Delta u(x, y) = -\exp(x + y)$$

with Dirichlet boundary conditions chosen so that the solution of this equation is $u(x, y) = \exp(x + y)$. We take $\varepsilon = 10^{-6}$ to make sure that only the Monte Carlo error remain.

N	M	L	$e(0)$	$e(L)$	CPU
5	100	8	0.15	6×10^{-5}	6
7	300	25	0.17	4×10^{-7}	112
10	1000	40	0.15	7×10^{-11}	1720

We still achieve an accuracy on the solution of this equation corresponding exactly to the interpolation error. Moreover, we observe that we obtain an accuracy of 7×10^{-11} when $N = 10$ with $\varepsilon = 10^{-6}$: this means that the final error is significantly smaller than the discretization error. It seems to show that we both reduce the variance and the bias. The value $\varepsilon = 10^{-2}$ seems to be the upper limit, for which we observe such a nice convergence. Beyond this limit, the algorithm diverges. Taking $\varepsilon = 10^{-2}$, we have the following results.

N	M	L	$e(0)$	$e(L)$	CPU
5	100	8	0.21	7×10^{-5}	1.8
7	300	25	0.23	3×10^{-7}	34
10	1000	40	0.15	8×10^{-11}	540

The numerical accuracy on the solution is exactly the same than with $\varepsilon = 10^{-6}$. The CPU times have been divided by about three, which is really significant. We could certainly also use a quasi-Monte Carlo version of this algorithm.

5 Conclusion

We have developed and studied a Monte Carlo algorithm to compute a spectral approximation of the solution of the Poisson equation in dimension one and two over simple domains. The numerical examples have shown that we have simultaneously drastically reduced the Monte Carlo error and the error due to the simulation of the Brownian trajectories using the modified WOS method. Moreover, a global solution is obtained up to the interpolation error. We have nevertheless only proved the geometric convergence of the method in a very simple case. This work was the first step to check the efficiency of this method. Our future work [GMon] will consist in proving the convergence of the algorithm on the studied example of the Poisson equation and in extending this approach to general elliptic and even parabolic operators.

Références

- [Bal95] P. Baldi. Exact asymptotics for the probability of exit from a domain and applications to simulation. *The Annals of Probability*, 23(4) :1644–1670, 1995.
- [BCP00] K. Baggerly, D. Cox, and R. Picard. Exponential convergence of adaptive importance sampling for Markov chains. *J. Appl. Prob.*, 37 :342–358, 2000.
- [Bjö96] A. Björck. *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996.
- [BM92] C. Bernardi and Y. Maday. *Approximations spectrales de problèmes aux limites elliptiques*. Mathématiques & Applications [Mathematics & Applications], 10. Springer-Verlag, Paris, 1992.

- [Boo82] T.E. Booth. Exact Monte Carlo solution of elliptic partial differential equation. *J. Comput. Phys.*, 47 :396–404, 1982.
- [Boo85] T.E. Booth. Exponential convergence for Monte Carlo particle transport ? *Trans. Amer. Nucl. Soc.*, 50 :267–268, 1985.
- [CDL⁺89] M. Cessenat, R. Dautray, G. Ledanois, P.L. Lions, E. Pardoux, and R. Sentis. *Méthodes probabilistes pour les équations de la physique*. Collection CEA, Eyrolles, 1989.
- [CHQZ88] C. Canuto, M.Y. Hussaini, A. Quarteroni, and T.A. Zang. *Spectral methods in fluid dynamics*. Springer Series in Computational Physics. Springer-Verlag, New York, 1988.
- [Fre85] M. Freidlin. *Functional integration and partial differential equations*. Annals of Mathematics Studies - Princeton University Press, 1985.
- [Fri76] A. Friedman. *Stochastic differential equations and applications*. Vol. 2. New York - San Francisco - London : Academic Press, a subsidiary of Harcourt Brace Jovanovich, Publishers. XIII, 1976.
- [GMon] E. Gobet and S. Maire. In preparation.
- [Gob01] E. Gobet. Euler schemes and half-space approximation for the simulation of diffusions in a domain. *ESAIM : Probability and Statistics*, 5 :261–297, 2001.
- [Hal62] J.H. Halton. Sequential Monte Carlo. *Proc. Camb. Phil. Soc.*, 58 :57–78, 1962.
- [HMG03] C.O. Hwang, M. Mascagni, and J.A. Given. A Feynman-Kac path-integral implementation for Poisson’s equation using an h -conditioned Green’s function. *Math. Comput. Simulation*, 62(3-6) :347–355, 2003. 3rd IMACS Seminar on Monte Carlo Methods—MCM 2001 (Salzburg).
- [KU98] A.R. Krommer and C.W. Ueberhuber. *Computational integration*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.
- [LPS98] B. Lapeyre, E. Pardoux, and R. Sentis. *Methodes de Monte Carlo pour les processus de transport et de diffusion*. Collection Mathématiques et Applications 29 - Springer Verlag, 1998.
- [Mai03] S. Maire. Reducing variance using iterated control variates. *The Journal of Statistical Computation and Simulation*, 73(1) :1–29, 2003.
- [Mul56] M.E. Muller. Some continuous Monte Carlo methods for the Dirichlet problem. *Ann. Math. Statist.*, 27(569–589), 1956.
- [Nie92] H. Niederreiter. *Random number generation and quasi-Monte Carlo methods*, volume 63 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992.
- [Sab91] K.K. Sabelfeld. *Monte Carlo methods in boundary value problems*. Springer Series in Computational Physics. Springer-Verlag, Berlin, 1991. Translated from the Russian.