APPLICATION OF KERNEL-BASED STOCHASTIC GRADIENT ALGORITHMS TO OPTION PRICING

KENGY BARTY, PIERRE GIRARDEAU, JEAN-SÉBASTIEN ROY, AND CYRILLE STRUGAREK

Abstract. We present an algorithm for American option pricing based on stochastic approximation techniques. Besides working on a finite subset of the exercise dates (e.g. considering the associated Bermudean option), option pricing algorithms generally involve another step of discretization, either on the state space or on the underlying functional space. Our work, which is an application of a more general perturbed gradient algorithm introduced recently by the authors, consists in approximating the value functions of the classical dynamic programming equation at each time step by a linear combination of kernels. The so-called kernel-based stochastic gradient algorithm avoids any a priori discretization, besides the discretization of time. Thus, it converges toward the optimum of the non-discretized Bermudan option pricing problem.
We present a comprehensive methodology to implement efficiently this algorithm, including discussions on the numerical tools used, like the Fast Gauss Transform, or Brownian bridge.
We also compare our results to some existing methods, and provide empirical statistical results.

1. Introduction

In many fields of application, evaluating the price of an option relying on multiple assets is a fundamental task. Since the early 70s, the field of mathematical finance and, more specifically, the option valuation theory, has been studied very thoroughly. When early exercise is possible, for American or Bermudean options, the problem can be formulated as an optimal stopping problem. Numerous techniques exist to price american options, and most of them rely on a parametrization. After having discretized the time period, they either discretize the state space (see e.g. the binomial or multinomial approaches in [Ros76], [CRR79], the stochastic mesh method in [BG04], or the quantization algorithm in [BPP05]), or they consider a truncated functional basis for the computation of the conditional expectation (see e.g. [LS01] and [TV99]) and try to optimize basis coefficients using a least squares approach, or use Malliavin calculus to compute these conditional exectations (see e.g. [FLL$^+$99] and [FLLL01]). A survey of these Monte-Carlo techniques can be found in [Gla03].

For classical diffusion models, there also exist PDE methods, like, for example, numerous finite differences methods.

We propose an alternative approach for these problems that is non-parametric and avoids any a priori discretization, besides the usual time discretization. Our method relies on the convolution by kernels, typically Gaussians, and has been introduced in [BRS05]. It is different from the approaches developped in [LS01] and [TV99] since they consider a fixed truncated basis of L^2 and tend to optimize the coefficients by regression. On the contrary, we do not restrict ourselves to a fixed subspace of L^2, and the coefficients of kernel functions are not optimized by regression, but obtained once for all by a single computation. Moreover this algorithm is quite easy to implement.

Our method can be introduced as an extension of the Robbins-Monro stochastic approximation algorithm [RM51] and, more recently, the temporal difference algorithm TD(0). Temporal difference learning introduced by Sutton[Sut88] provides a way to carry out the Bellman operator fixed point iterations while approximating the expectation through random sampling. Unfortunately, this approach still requires a discretization of the state space which, in the large scale case, might

Date: 17th May 2006.
2000 *Mathematics Subject Classification.* Primary 65C05; Secondary 91B28, 93E35.
Key words and phrases. American Option Pricing, Monte-Carlo Methods, Kernel Approximation, Stochastic Algorithms.
The authors would like to thank A. Zanette for his helpful remarks on a previous version of this paper.

not be practicable. To overcome the curse of dimensionality most approaches so far have proposed to approximate the value function as a linear combination of basis functions. This approach, called approximate dynamic programming, and first described in [BD59], has been thoroughly studied. See [SB98] and [BT96] for detailed introductions to temporal difference and approximate dynamic programming methods. Recent and promising approaches to this problem include a formulation of dynamic programming through a linear program which can ensure performance guarantees [dFVR04]. Nevertheless, all these approaches require the use of a predefined finite functional basis and therefore give up optimality, even asymptotically. Moreover, while the quality of the approximation might increase with the number of functions used in the basis, the complexity of each iteration (usually a least-square regression or linear program), renders the use of large basis impracticable.

The alternative approach we introduce is based on functional gradient descent and uses an infinite kernel basis, that preserves optimality under very light conditions while being implementable in practice. In contrast to finite functional basis methods, where the a priori basis is used to abitrarily generalize the local gradient information provided by each sample, we aim at generalizing using only regularity assumptions of the value fonction and therefore better exploiting the information provided.

Similar ideas date back to recursive nonparametric density estimation [WW69], and have been proposed in the context of econometry in [CW98]. Our approach aims at providing more sensible assumptions in the context of optimization and simpler proofs, based on a completely different theory.

In our method, the iterates (the value functions) are represented by a sum of kernels, that are usually Gaussian functions. In order to speed up the algorithm, we make use of the Improved Fast Gauss Transform to approximate a sum of Gaussian functions. This technique has already been used in mathematical finance, for instance in the multinomial and the stochastic mesh methods in [BY03] for the case of discrete-time American-style options.

Several other techniques are presented and used to improve the rate of convergence of the algorithm, such as averaging of the iterates [PJ92] and low discrepancy random number generators combined with a Brownian bridge. The performance of our implementation is assessed through some numerical experiments and comparisons against some well-known pricing algorithms.

This paper is organized as follows. In section 2, we draw the theoretical framework in which we consider option pricing. We then introduce our algorithm to solve the classical Q-learning formulation. We provide a convergence proof for the algorithm, based on a theorem on infinite dimensional stochastic approximation developped in [BRS05].

In section 3, we discuss several implementation issues for the algorithm that are not specific to option pricing, like the use of the Fast Gauss Tranform to compute efficiently a sum of Gaussian functions, or the choice of the stepsizes used in the algorithm. We also describe our use of averaging methods to accelerate the rate of convergence, based on the work by Polyak and Juditsky [PJ92].

In section 4, we introduce the use of quasi-Monte Carlo simulations enhanced with Brownian bridge in the case of classical diffusion processes. Then, we statistically compare the results of the kernel method to a few references in the literature. Finally, we give results for the pricing of discrete-time American-Asian options.

2. Theoretical framework

2.1. **Problem.** An option is typically the right to sell or buy a stock at prescribed dates, before a deadline called maturity. European options can be exercised only at maturity. On the contrary, American options can be exercised whenever before maturity. Hence American option pricing is a stopping time problem.

It is common to discretize American options as Bermudan options (see [CRR79], [BPP05] or [BM95] for example), for which the exercise dates belong to a finite subset. In the rest, we only consider Bermudan options.

Let us note x_{t_0} the initial stock price. We consider exercise dates $\{t_0, t_1, \dots, t_N = T\}$ and define $\delta t = \min_{0 \le j \le N-1} (t_{j+1} - t_j)$.
The price process X is assumed to be a Markov chain $\left\{X_{t_j} \in S, 0 \le j \le N\right\}$ with values in $S \subseteq \mathbb{R}^d$ and $\mathcal{F}_j$ is the σ-field generated by the random variable $(X_{t_0}, X_{t_1}, \dots, X_{t_j})$.
For all $j = 0, \dots, N$, we denote recursively by π_j the probability ditribution of X_{t_j} and note $\pi = \otimes_{j=0}^{N} \pi_j$.
For any $J, \tilde{J} : S^{N+1} \to \mathbb{R}^{N+1}$, let us define the following inner product and norm:

$$\left\langle J, \tilde{J} \right\rangle_{\pi} := \sum_{j=0}^{N} \int_S J_j(x)\, \tilde{J}_j(x)\, \pi_j(dx) = \sum_{j=0}^{N} \left\langle J_j, \tilde{J}_j \right\rangle_{\pi_j}, \quad \|J\|_{\pi}^2 := \langle J, J \rangle_{\pi}.$$

We then define: $\mathcal{L}^2\left(S^{N+1}, \mathbb{R}^{N+1}, \pi\right) = \left\{J : S^{N+1} \to \mathbb{R}^{N+1} \text{ measurable so that } \|J\|_{\pi}^2 < +\infty\right\}$.
Moreover, we denote by $L^2\left(S^{N+1}, \mathbb{R}^{N+1}, \pi\right)$ the Kolmogorov quotient of $\mathcal{L}^2\left(S^{N+1}, \mathbb{R}^{N+1}, \pi\right)$, i.e. where we divide out the kernel of the norm $\|\cdot\|_{\pi}$ (we identify two functions if they are equal almost everywhere).

Let $g : [0, T] \times S \to \mathbb{R}$ be the intrinsic value of the option. Then the price $J_0(x)$ of the Bermudan option with maturity T is given by

$$J_0(x) = \max_{\tau \in \mathcal{T}(t_0, t_N)} \mathbb{E}\left[g(\tau, X_\tau) \middle| X_{t_0} = x\right], \tag{1}$$

with $\mathcal{T}(t_0, t_N)$ the set of stopping times adapted to $(\mathcal{F}_j)_{0 \le j \le N}$.
From this definition we deduce the dynamic programming formulation equivalent to (1):

$$J_{N+1}(x) = 0, \quad \forall x \in S \tag{2a}$$

$$J_j(x) = \max\left(g(t_j, x), \mathbb{E}\left[J_{j+1}(X_{t_{j+1}}) \middle| X_{t_j} = x\right]\right), \ \forall x \in S, \ \forall j \in \{0, \dots, N\}. \tag{2b}$$

Let us now define Q_j as the expected payoff at time t_j if we do not exercise the option. From (2): $Q_j(x) = \mathbb{E}\left[J_{j+1}(X_{t_{j+1}}) \middle| X_{t_j} = x\right], \forall x \in S$. Hence it comes:

$$\forall x \in S, \forall j \in \{0, \dots, N\}, \quad J_j(x) = \max\left(g(t_j, x), Q_j(x)\right). \tag{3}$$

Equation (2) now reads:

$$Q_N(x) = 0, \ \forall x \in S, \tag{4a}$$

$$Q_j(x) = \mathbb{E}\left[\max\left(g\left(t_{j+1}, X_{t_{j+1}}\right), Q_{j+1}\left(X_{t_{j+1}}\right)\right) \middle| X_{t_j} = x\right], \ \forall x \in S, \ \forall j \in \{0, \dots, N-1\}. \tag{4b}$$

We propose an algorithm that builds sequences of functions $Q^k = \left(Q_j^k\right)_{0 \le j \le N}$ that converge to the solution $Q^* = \left(Q_j^*\right)_{0 \le j \le N}$ of the Q-learning equation (4). Then we come back to the solution J^* of the classical dynamic programming equation (2), using equation (3).

2.2. **Algorithm.** We use the following notational convention: $x_{t_j}^k$ is k-th drawing (realization) of the random process X_{t_j}.
Then our algorithm works as follows: our estimates $\left(Q_j^k\right)_{0 \le j \le N}$ of the optimal value functions $\left(Q_j^*\right)_{0 \le j \le N}$ are represented by sums of k kernels, typically Gaussian functions. At every step, we draw a trajectory of the price process. Then, starting with t_N and subsequently t_{N-1} down to t_0, we update the value functions Q_j^k in a neighbourhood of the drawings $x_{t_j}^k$, beginning with function Q_N^k. Updates are computed using relations (4), by adding to Q_j^k a kernel function centered on $x_{t_j}^k$ in order to obtain Q_j^{k+1}, which is therefore a sum of $(k+1)$ kernel functions.
We propose the following algorithm:

Algorithm 2.1. *Initialize* $Q_j^0(\cdot) = 0$ *for* $j \in \{0, \dots, N\}$,
Step $k \ge 0$*:*

- *Draw* $x^k = \left(x_{t_j}^k\right)_{0 \le j \le N}$ *independently from the past drawings with respect to the law of the Markov chain* $\left(X_{t_j}\right)_{0 \le j \le N}$*;*

- *Update* $Q_j, j \in \{0, \dots, N\}$*:*

$$\begin{cases} Q_N^{k+1}(\cdot) &= 0, \\ Q_{N-1}^{k+1}(\cdot) &= Q_{N-1}^k(\cdot) + \rho_{N-1}^k \, \Delta_{N-1}^k \, K_{N-1}^k(x_{t_{N-1}}^k, \cdot), \\ &\vdots \\ Q_j^{k+1}(\cdot) &= Q_j^k(\cdot) + \rho_j^k \, \Delta_j^k \, K_j^k(x_{t_j}^k, \cdot), \\ &\vdots \\ Q_0^{k+1}(\cdot) &= Q_0^k(\cdot) + \rho_0^k \, \Delta_0^k \, K_0^k(x_{t_0}^k, \cdot). \end{cases}$$

where

$$\Delta_j^k = \max\left(g\left(t_{j+1}, x_{t_{j+1}}^k\right), Q_{j+1}^k\left(x_{t_{j+1}}^k\right)\right) - Q_j^k(x_{t_j}^k), \ \forall j \in \{0, \dots, N-1\}.$$

Functions K_j^k are kernels, i.e. bounded mappings $S \times S \to \mathbb{R}$. A typical choice of these kernels is Gaussian function:

$$K^k(x, y) = e^{-\left\| \frac{x-y}{\eta^k} \right\|^2}, \text{ with } \eta^k \to 0 \text{ when } k \to +\infty.$$

For this particular kind of kernel function, we call η^k the bandwidth of the kernel. Barty et al. proved in a particular case that the sequence strongly converges to $Q^* = \left(Q_j^*\right)_{0 \le j \le N}$. Steps ρ_j^k and bandwidths η^k of the kernels must be decreasing scalar sequences, whose decreasing speed is ruled by relations discussed in subsection 3.2.

As one can see, we are working directly in the infinite dimensional state space to which the solution belongs. In spite of the infinite dimension, this method remains numerically tractable since, for a Gaussian kernel and for each j in $\{0, \dots, N\}$, Q_j^k may be represented by k triplets of real numbers: the centers, the bandwidths, and the heights of the kernels. Indeed, it holds from the description of Algorithm 2.1 that:

$$Q_j^k(\cdot) = \sum_{i=0}^{k} \rho_j^i \, \Delta_j^i \, K_j^i(x_{t_j}^i, \cdot), \quad \forall j \le N, \ \forall k \in \mathbb{N}.$$

In other words, $\left(\Delta_j^i, x_{t_j}^i, \eta^i\right)_{0 \le i \le k}$ describes completely Q_j^k.

2.3. **Comparison with the Robbins-Monro and the TD(0) algorithms.** Let us concentrate on pointing out the main differences of algorithm 2.1 compared to the Robbins-Monro [RM51] and the TD(0) [Sut88] algorithms.

In order to estimate the regression function $Q_j(x) = \mathbb{E}\left[J_{j+1}(X_{t_{j+1}}) \middle| X_{t_j} = x\right]$, Robbins and Monro [RM51] introduced an iterative algorithm that averages the drawings of $X_{t_{j+1}} | X_{t_j} = x$, for all x in the state space S. The Robbins-Monro stochastic approximation algorithm is the following:

$$Q_j^k(x) = Q_j^{k-1}(x) + \rho_j^k \, \Delta_j^k\left(x, y^k(x)\right),$$
$$\text{where } \Delta_j^k\left(x, y^k(x)\right) = \max\left(g\left(t_{j+1}, y^k(x)\right), Q_{t_{j+1}}^{k+1}\left(y^k(x)\right)\right) - Q_j^k(x),$$

and $y_j^k(x)$ is a realization of the process $X_{t_{j+1}} | X_{t_j} = x$.

Note that the update here concerns every state x and that it can be rewritten using the underlying random variable X_{t_j} and a Dirac mass δ:

$$Q_j^k(\cdot) = Q_j^{k-1}(\cdot) + \rho_j^k \, \mathbb{E}\left(\Delta_j^k\left(X_{t_j}, y\right) \delta_{X_{t_j}}(\cdot)\right).$$

Instead of updating Q_j for every state x, Sutton [Sut88] proposed to randomize this operation by drawing realizations of the random variable X_{t_j}. We hence obtain the TD(0) algorithm:

$$Q_j^k(x) = \begin{cases} Q_j^{k-1}\left(x_{t_j}^k\right) + \rho_j^k \, \Delta_j^k\left(x_{t_j}^k, y^k\left(x_{t_j}^k\right)\right) & \text{if } x_{t_j}^k = x, \\ Q_j^{k-1}(x) & \text{else.} \end{cases}$$

Unfortunately, this algorithm can not be implemented if the state space is continuous and is untractable if the state space is discrete but too large.

Our algorithm consists in approximating the Dirac mass $\delta_{x_{t_j}^k}(\cdot)$ by a convolution with kernels, whose bandwidths decrease along the iterations, using a well-known analysis result, for kernels having certain properties (see e.g. [Boc55], Theorem 1.3.2):

$$f(\cdot) = \lim_{k \to +\infty} \mathbb{E}\left(f(X) \frac{1}{\varepsilon^k} K^k(X, \cdot) \right),$$

where $\varepsilon^k = \left(\eta^k\right)^d$. Recall that η^k is the bandwidth of the kernel $K^k(\cdot,\cdot)$ and d is the dimension of the state space S.

2.4. **Comparison with the Longstaff-Schwartz and the Tsitsiklis-Van Roy algorithms.** On the other hand, algorithm 2.1 can seem very similar to the Longstaff-Schwartz [LS01] or the Tsitsiklis-Van Roy [TV99] algorithms. Let us now recall these two techniques.
Longstaff and Schwartz's algorithm works directly on the dynamic programming equation (2) and is the result of two approximation steps:

(i) A linear regression that consists in replacing the conditional expectation $\mathbb{E}\left[J_{j+1}(X_{t_{j+1}}) \middle| X_{t_j} = x \right]$ by a projection P_j^m onto the vector space $(e_i(x))_{0 \le i \le m}$ generated by m a priori chosen measurable real valued functions on S (taken from a suitable basis, like polynomial basis or wavelet basis for example).
(ii) Monte Carlo simulations and a least squares regression on the coefficients of the basis $(e_i(x))_{0 \le i \le m}$ to achieve the projection.

Thus, this method computes functions $J_j(\cdot)$ backwards:

- Knowing $J_{N+1}(\cdot) = 0$, simulate prices at time t_N and compute the regression to obtain $\tilde{J}_N$ with the dynamic programming equation (2).
- For all $j \ge 1$, knowing $J_j(\cdot)$, simulate prices at time t_{j-1} and compute the regression to obtain $\tilde{J}_{j-1}$ with the dynamic programming equation.
- Calculate $J_0(x) = \max\left(g(t_0, x), J_1(x)\right)$.

Hence Longstaff and Schwartz's algorithm consists in discretizing the functional space $L^2\left(S^N, \mathbb{R}^N, \pi\right)$ to replace conditional expectations by projections. It is a kind of Galerkin method, for conditional expectations. Then it approximates the projection by a classical Monte Carlo method.

Concerning Tsitsiklis and Van Roy's algorithm [TV99], they introduce the Q-functions and rewrite the dynamic programming equations like in (4). They also approximate the conditional expectation by a suitable projection and compute Monte Carlo simulations. The difference with Longstaff-Schwartz is that the use of Q-functions allows to exchange the maximum and the expectation in (2); it allows henceforth to proceed pathwise the least squares regression, i.e. on the contrary to Longstaff and Schwartz's algorithm, every new drawing of a price trajectory leads to a better approximation of all Q_j.

In our approach, we also consider the Q-functions but we have two main differences with the two preceding algorithms:

(i) We do not replace the conditional expectation by any projection on a vector subspace of L^2.
(ii) We do not optimize the coefficients behind the kernel functions, since it would become rapidly a heavy burden: the coefficients are computed once for all with a single temporal difference computation, i.e. a functional gradient step.

2.5. **Convergence proof.** Let us now present a convergence proof of 2.1, by means of perturbed gradient analysis [BRS05].
We first introduce $h_j(x, y) := \max\left(g(t_j, x), y\right)$, and:

$$H_j(Q)(\cdot) = \mathbb{E}\left[h_{j+1}\left(X_{t_{j+1}}, Q_{j+1}\left(X_{t_{j+1}}\right)\right) \middle| X_{t_j} = \cdot \right], \forall j \in \{0, \dots, N\}.$$

American option pricing consists in solving the fixed point equation described componentwise in equation (4), and summed up by:

$$HQ = Q. \tag{5}$$

In order to simplify notations, we consider that the steps ρ_j^k and ε_j^k are the same along the time steps t_j, and thus can be written ρ^k and ε^k. Moreover, we introduce:

$$r_j^k = H_j(Q^k) - Q_j^k, \quad \gamma^k = \rho^k \varepsilon^k, \quad \eta^k = \left(\varepsilon^k\right)^{\frac{1}{d}}.$$

Theorem 2.2. *The solution Q^* of the fixed point equation (5) exists and belongs to $L^2\left(S^N, \mathbb{R}^N, \pi\right)$. Moreover, if, for all $j \in \{0, \dots, N\}$, there exist real numbers b_1 and b_2 such that the following assumptions are fulfilled:*

$$\left\| r_j^k(\cdot) - \int r_j^k(x) \frac{1}{\varepsilon^k} K_j^k(x, \cdot) \pi_j(dx) \right\|_{\pi_j} \leq b_1 \eta^k \left(1 + \left\| r_j^k \right\|_{\pi_j}\right), \ \forall k \in \mathbb{N} \tag{6a}$$

$$\forall y \in \mathbb{R}^d, \int \left(K_j^k(x, y)\right)^2 \pi_j(dx) \leq b_2 \varepsilon^k, \ \forall k \in \mathbb{N} \tag{6b}$$

$$\varepsilon^k \xrightarrow[k \to +\infty]{} 0, \quad \sum_{k \in \mathbb{N}} \gamma^k = +\infty, \quad \sum_{k \in \mathbb{N}} \frac{(\gamma^k)^2}{\varepsilon^k} < +\infty, \quad \sum_{k \in \mathbb{N}} \gamma^k \eta^k < +\infty, \tag{6c}$$

the functions $Q_j^k(\cdot)$ defined by Algorithm 2.1 a.s. converge, when $k \to +\infty$, to the solution Q^* of the fixed point equation (5).

Proof:
We first introduce a new norm, that will be useful for the proof. For any $J, \tilde{J} : S^n \to \mathbb{R}^N$, let us define the following inner product and norm:

$$\left\langle J, \tilde{J} \right\rangle_{\pi'} := \sum_{j=0}^{N} e^{t_j} \int J_j(x) \tilde{J}_j(x) \pi_j(dx) = \sum_{j=0}^{N} e^{t_j} \left\langle J_j, \tilde{J}_j \right\rangle_{\pi_j}, \quad \|J\|_{\pi'}^2 := \langle J, J \rangle_{\pi'}. \tag{7}$$

One easily remarks that π and π' are equivalent, since $\sqrt{e^{t_0}} \|J\|_\pi \leq \|J\|_{\pi'} \leq \sqrt{e^{t_N}} \|J\|_\pi$.

1. Let us first prove that there exists a real number $0 \leq a < 1$ such that:

$$\forall Q, \tilde{Q} \in L^2\left(S^N, \mathbb{R}^N, \pi\right), \ \left\| H(Q) - H(\tilde{Q}) \right\|_{\pi'} \leq a \left\| Q - \tilde{Q} \right\|_{\pi'}. \tag{8}$$

Recall that we have by definition of H:

$$H_j(Q)(x) = \begin{cases} \mathbb{E}\left[\max\left(g(t_{j+1}, X_{t_{j+1}}), Q_{j+1}(X_{t_{j+1}})\right) \middle| X_{t_j} = x\right], & \text{if } j \in \{0, \dots, N-1\}, \\ 0 & \text{if } j = N. \end{cases}$$

By Jensen's inequality, for $Q, \tilde{Q} \in L^2\left(S^N, \mathbb{R}^N, \pi\right)$:

$$\begin{aligned} \left| H_j(Q)(X_{t_j}) - H_j(\tilde{Q})(X_{t_j}) \right|^2 &\leq \mathbb{E}\Big[\Big(\max\left(g(t_{j+1}, X_{t_{j+1}}), Q_{j+1}(X_{t_{j+1}})\right) \\ &\qquad - \max\left(g(t_{j+1}, X_{t_{j+1}}), \tilde{Q}_{j+1}(X_{t_{j+1}})\right)\Big)^2 \Big| X_{t_j} \Big], \\ &\leq \mathbb{E}\left[\left(Q_{j+1}(X_{t_{j+1}}) - \tilde{Q}_{j+1}(X_{t_{j+1}}) \right)^2 \middle| X_{t_j} \right]. \end{aligned}$$

By taking the expectation on both sides, we obtain:

$$\begin{aligned} e^{t_j} \mathbb{E}\left[\left(H_j(Q)(X_{t_j}) - H_j(\tilde{Q})(X_{t_j}) \right)^2 \right] &\leq e^{t_j} \mathbb{E}\left[\left(Q_{j+1}(X_{t_{j+1}}) - \tilde{Q}_{j+1}(X_{t_{j+1}}) \right)^2 \right], \\ &= \underbrace{e^{t_j - t_{j+1}}}_{\leq e^{-\delta t}} e^{t_{j+1}} \mathbb{E}\left[\left(Q_{j+1}(X_{t_{j+1}}) - \tilde{Q}_{j+1}(X_{t_{j+1}}) \right)^2 \right], \end{aligned}$$

since $\delta t = \min_{0\leq j\leq N-1}(t_{j+1}-t_j)$. Now, we can sum on index j, and by using the terminal condition $H_N(Q)(x) = 0 = Q_N(x)$, we obtain:

$$\begin{aligned}\sum_{j=0}^{N-1} e^{t_j}\mathbb{E}\left[\left(H_j(Q)(X_{t_j}) - H_j(\tilde{Q})(X_{t_j})\right)^2\right] &\leq e^{-\delta t}\sum_{j=0}^{N-2} e^{t_{j+1}}\mathbb{E}\left[\left(Q_{j+1}(X_{t_{j+1}}) - \tilde{Q}_{j+1}(X_{t_{j+1}})\right)^2\right],\\ &\leq e^{-\delta t}\sum_{j=1}^{N-1} e^{t_j}\mathbb{E}\left[\left(Q_j(X_{t_j}) - \tilde{Q}_j(X_{t_j})\right)^2\right]\\ &\quad + e^{-\delta t}e^{t_0}\mathbb{E}\left[\left(Q_0(X_{t_0}) - \tilde{Q}_0(X_{t_0})\right)^2\right],\end{aligned}$$

i.e.: $\left\|H(Q) - H(\tilde{Q})\right\|_{\pi'} \leq \sqrt{e^{-\delta t}}\left\|Q - \tilde{Q}\right\|_{\pi'}$.
Hence H is a contraction mapping and there exists a solution Q^* to the fixed point equation (5) that belongs to $L^2\left(S^N, \mathbb{R}^N, \pi'\right)$. Since π and π' are equivalent, the solution Q^* also belongs to $L^2\left(S^N, \mathbb{R}^N, \pi\right)$.

2. Now, we transform our fixed point problem into a minimization problem and claim that our algorithm consists in a perturbed gradient algorithm, in the sense defined in [BRS05], of which we verify the assumptions.
Fixed point equation (5) may be rewritten as a minimization problem:

$$\min_{Q\in L^2\left(S^N,\mathbb{R}^N,\pi'\right)} \frac{1}{2}\left\|Q - Q^*\right\|_{\pi'}^2$$

with Q^* the solution of the fixed point problem.
Our update equation can be written as:

$$Q_j^{k+1}(\cdot) = Q_j^k(\cdot) + \gamma^k\left(\underbrace{-Q_j^k + H_j(Q^k)}_{s_j^k} + \underbrace{-H_j(Q^k) + Q_j^k + \Delta_j^k\frac{1}{\varepsilon^k}K_j^k(x_j^k,\cdot)}_{w_j^k}\right),$$

where s^k represents the "true" descent direction we should choose, and w^k is the perturbation we introduce by replacing the functional gradient direction by a local approximation.
First, we have to prove that s^k is a descent direction in the sense of the assumption of [BRS05]. This is achieved by applying Cauchy-Schwarz inequality and the contraction property obtained above. More precisely:

$$\begin{aligned}\sum_{j=0}^{N} e^{t_j}\left\langle s_j^k, Q_j^k - Q_j^*\right\rangle_{\pi_j} &= \sum_{j=0}^{N} e^{t_j}\left[\left\langle -Q_j^k + Q_j^*, Q_j^k - Q_j^*\right\rangle_{\pi_j} + \left\langle H_j(Q^k) - H_j(Q^*), Q_j^k - Q_j^*\right\rangle_{\pi_j}\right],\\ &\leq -\left\|Q^k - Q^*\right\|_{\pi'}^2 + \left\|H(Q^k) - H(Q^*)\right\|_{\pi'}^2 \times \left\|Q^k - Q^*\right\|_{\pi'}^2,\\ &\leq \underbrace{\left(-1 + e^{-\delta t}\right)}_{<0}\left\|Q^k - Q^*\right\|_{\pi'}^2,\end{aligned}$$

and we can conclude that s_j^k is a descent direction for the minimization problem.

Now we must verify the two assumptions of [BRS05] concerning the perturbation w^k.

3. We denote $\mathcal{F}^k$ the σ-field generated by the random variable $\left(x^1, \ldots, x^k\right)$ and $\mathbb{E}^k[\cdot]$ the conditional expectation with respect to $\mathcal{F}^k$. One can directly observe that Q^k is $\mathcal{F}^k$-measurable.
The first result we need on w^k is that there exists a real number b_1 such that:

$$\forall k\in\mathbb{N},\ \left\|\mathbb{E}^k\left[w^k\right]\right\|_{\pi'} \leq b_1\eta^k\left(1 + \left\|H(Q^k) - Q^k\right\|_{\pi'}\right).$$

Using measurability and conditional expectation properties, we have:

$$\begin{aligned}
\mathbb{E}^k\left[w_j^k(\cdot)\right] &= Q_j^k(\cdot) - H_j(Q^k)(\cdot) \\
&\quad + \mathbb{E}^k\left[h_{j+1}\left(x_{t_{j+1}}^{k+1}, Q_{j+1}^k(x_{t_{j+1}}^{k+1})\right)\frac{1}{\varepsilon^k}K_j^k(x_{t_j}^{k+1},\cdot) - Q_j^k(x_{t_j}^{k+1})\frac{1}{\varepsilon^k}K_j^k(x_{t_j}^{k+1},\cdot)\right], \\
\mathbb{E}^k\left[w_j^k(\cdot)\right] &= Q_j^k(\cdot) - H_j(Q^k)(\cdot) \\
&\quad + \mathbb{E}\left[\underbrace{\mathbb{E}\left[h_{j+1}\left(x_{t_{j+1}}, Q_{j+1}^k(x_{t_{j+1}})\right)\middle| x_{t_j}\right]}_{H_j(Q^k)(x_{t_j})}\frac{1}{\varepsilon^k}K_j^k(x_{t_j},\cdot)\right] \\
&\quad - \int Q_j^k(x)\frac{1}{\varepsilon^k}K_j^k(x,\cdot)\pi_j(dx)
\end{aligned}$$

Recall that $r_j^k = H_j(Q^k) - Q_j^k$. Hence:

$$\mathbb{E}^k\left[w_j^k(\cdot)\right] = -r_j^k(\cdot) + \int r_j^k(x)\frac{1}{\varepsilon^k}K_j^k(x,\cdot)\pi_j(dx),$$

and:

$$\left\|\mathbb{E}^k\left[w_j^k\right]\right\|_{\pi_j} \leq b_1\eta^k\left(1 + \left\|r_j^k\right\|_{\pi_j}\right), \tag{9}$$

by assumption (6a). The result is obtained by summing relation (9) on index j.

4. Now we need to prove that:

$$\forall k \in \mathbb{N},\ \mathbb{E}^k\left[\left\|w^k\right\|_{\pi'}^2\right] \leq A\left(1 + \frac{1}{\beta_k}\left\|r^k\right\|_{\pi'}^2\right), \tag{10}$$

with β^k having the same behavior as ε^k (assumption (6c)).
Recall that:

$$w_j^k(\cdot) := -H_j(Q^k)(\cdot) + Q_j^k(\cdot) + \Delta_j^{k+1}\frac{1}{\varepsilon^k}K_j^k(x_{t_j}^k,\cdot).$$

We already have $\left\|H_j(Q^k) - Q_j^k\right\|_{\pi_j}^2 = \left\|r_j^k\right\|_{\pi_j}^2$. Let us give an upper bound for the third term in w_j^k. By Assumption (6b):

$$\begin{aligned}
\left\|\Delta_j^{k+1}\frac{1}{\varepsilon^k}K_j^k(x_{t_j}^k,\cdot)\right\|_{\pi_j}^2 &\leq \frac{1}{\varepsilon^k}\left|\Delta_j^{k+1}\right|^2, \\
&\leq \frac{1}{\varepsilon^k}\Big|\left(h_{j+1}\left(x_{t_{j+1}}^{k+1}, Q_{j+1}^k(x_{t_{j+1}}^{k+1})\right) - H_j(Q^*)(x_{t_j}^{k+1})\right) \\
&\quad - \left(Q_j^k(x_{t_j}^{k+1}) - Q_j^*(x_{t_j}^{k+1})\right)\Big|^2.
\end{aligned}$$

By taking the conditional expectation on both sides and summing on index j:

$$\begin{aligned}
\sum_{j=0}^N e^{t_j}\mathbb{E}^k\left[\left\|\Delta_j^{k+1}\frac{1}{\varepsilon^k}K_j^k(x_{tj}^k,\cdot)\right\|_{\pi_j}^2\right] &\leq \frac{1}{\varepsilon^k}\sum_{j=0}^N e^{t_j}\left(2\left\|H_j(Q^k) - H_j(Q^*)\right\|_{\pi_j}^2 + 2\left\|Q_j^k - Q_j^*\right\|_{\pi_j}^2\right), \\
&\leq 2\frac{1}{\varepsilon^k}\left(\left(e^{-\delta t}\right)^2 + 1\right)\left\|Q^k - Q^*\right\|_{\pi}^2.
\end{aligned}$$

Let us note $r^k = \left(r_j^k\right)$. By triangle inequality:

$$\left\|Q^k - Q^*\right\|_{\pi'} \leq \left\|r^k\right\|_{\pi'} + \left\|H(Q^k) - H(Q^*)\right\|_{\pi'},$$

so that, by equation (8):

$$\left\|Q^k - Q^*\right\|_{\pi'}^2 \leq \frac{1}{(1 - e^{-\delta t})^2}\left\|r^k\right\|_{\pi'}^2,$$

Hence:

$$\left\|w^k\right\|_{\pi'}^2 \leq 2\underbrace{\left\|Q^k - H(Q^k)\right\|_{\pi'}^2}_{\left\|r^k\right\|_{\pi'}^2} + 2\sum_{j=0}^N e^{t_j}\left\|\Delta_j^{k+1}\frac{1}{\varepsilon^k}K^k\left(x_{t_j}^k,\cdot\right)\right\|_{{\pi'}_j}^2,$$

and:

$$\mathbb{E}\left[\left\|w^k\right\|_{\pi'}^2\right] \leq 2\left\|r^k\right\|_{\pi'}^2 + 2\underbrace{\sum_{j=0}^{N} e^{t_j}\mathbb{E}\left[\left\|\Delta_j^{k+1}\frac{1}{\varepsilon^k}K^k\left(x_{t_j}^k,\cdot\right)\right\|_{\pi'_j}^2\right]}_{\leq 2\frac{1}{\varepsilon^k}\left((e^{-\delta t})^2+1\right)\left\|Q^k-Q^*\right\|_{\pi'}^2},$$
$$\leq 2\left\|r^k\right\|_{\pi'}^2 + 4\frac{1}{\varepsilon^k}\left((e^{-\delta t})^2+1\right)\underbrace{\left\|Q^k-Q^*\right\|_{\pi'}^2}_{\leq\frac{1}{\left(1-e^{-\delta t}\right)^2}\left\|r^k\right\|_{\pi'}^2},$$

Finally:

$$\mathbb{E}\left[\left\|w^k\right\|_{\pi'}^2\right] \leq 2\left\|r^k\right\|_{\pi'}^2 + 4\frac{1}{\varepsilon^k}\frac{1+\left(e^{-\delta t}\right)^2}{\left(1-e^{-\delta t}\right)^2}\left\|r^k\right\|_{\pi'}^2,$$
$$\leq \frac{1}{\beta^k}\left\|r^k\right\|_{\pi'}^2,$$

with $\frac{1}{\beta^k} := 2 + 4\frac{1}{\varepsilon^k}\frac{1+\left(e^{-\delta t}\right)^2}{\left(1-e^{-\delta t}\right)^2}$. Hence Assumption (10) is fulfilled.

Since all assumptions of [BRS05] are satisfied, we deduce that (Q_j^k) a.s. strongly converges to (Q_j^*). □

In practice, we use the implicit version of algorithm 2.1. Indeed, when we update $Q_j^k(\cdot)$ into $Q_j^{k+1}(\cdot)$, we already have $Q_{j+1}^{k+1}(\cdot)$ because we compute the update backward in time. Hence we can use as temporal difference the quantity:

$$\Delta_j^k = \max\left(g\left(t_{j+1}, x_{t_{j+1}}^k\right), \underbrace{Q_{j+1}^{k+1}\left(x_{t_{j+1}}^k\right)}_{\text{instead of } Q_{j+1}^k\left(x_{t_{j+1}}^k\right)}\right) - Q_j^k(x_{t_j}^k).$$

3. Practical Discussion

We here discuss several numerical tools that are useful to ensure a reasonable practical convergence speed for our algorithm. We explain how we make use of the Fast Gauss Transform to compute sums of Gaussian functions. Then we discuss the choice of the stepsizes for the algorithm in subsections 3.2 and 3.3.

3.1. Implementation with Fast Gauss Transform.

Even if algorithm 2.1 is quite easy to implement, its computation is a priori quadratic in the number of iterations. Indeed, the number of kernels grows linearly with the number of iterations. Recall that, at iteration k, for time step t_j, one has to compute:

$$Q_j^k(\cdot) = Q_j^{k-1}(\cdot) + \rho_j^k\ \Delta_j^k\ K_j^k(x_{t_j}^k,\cdot),$$

where: $\Delta_j^k = \max\left(g\left(t_{j+1}, x_{t_{j+1}}^k\right), Q_{j+1}^{k-1}\left(x_{t_{j+1}}^k\right)\right) - Q_j^{k-1}(x_{t_j}^k)$.

The main problem is thus that at every step k, one needs to compute the temporal difference, which is a difference between two sums of k Gaussian functions, for every time step $t_j, j \in \{0,\ldots,N\}$. Hence the global cost of the algorithm is $O\left(\frac{k(k+1)}{2}\right)$. Challenge is here to reduce the computation time at each step, which is achieved by using Fast Gauss Transform (FGT) techniques.

FGT has been introduced by Greengard and Strain [GS91] to efficiently compute sums of Gaussian functions. Their original problem is slightly different from ours since they consider a large number of Gaussian functions to be evaluated on a large number of points. Their idea is an adaption of the more general Fast Multipole Method, introduced by Greengard and Rokhlin [GR87], which consists in considering two different expansions of the Gaussian above its center: either a far-field Hermite series or a local Taylor series. This method has been applied to many examples in low-dimensional cases, because it reduces the computation to something constant with respect to k. Unfortunately, this constant grows exponentially with the dimension, for two reasons:

(i) The number of terms in the Hermite series grows exponentially with dimension,
(ii) The space subdivision scheme used by the FGT is a uniform box subdivision scheme which is tolerable in low dimensions but is extremely inefficient in higher dimensions.

The drawbacks of the FGT in dimension above 3 comes from the blind extension of the one-dimensional case to the multi-dimensional case, by treating the multivariate Gaussian as a product of univariate Gaussians. A new approach proposed by Yang et al. [YDGD03] overcomes this weakness. We now present this technique, when applied to our algorithm.

We restrict the explanation to the time step t_j and to a fixed kernel bandwidth η. Let us first describe the space subdivion scheme, which builds a set of clusters, each represented by its center, incrementally along the iterations:

- Initialize the list of centers with the first drawing $x_{t_j}^1$.
- At every step, say k, if the distance between the new drawing $x_{t_j}^k$ and the nearest center is greater than η, then add $x_{t_j}^k$ to the list of centers, else associate $x_{t_j}^k$ with the nearest center from it.

This procedure defines a set of N_k clusters that draw a partition of the drawings. Moreover, the operation of finding the nearest center from a drawing is done in constant time, the constant being the number of clusters N_k.

Now let us precise how we proceed with the evaluation of the sum of Gaussians. Suppose that x and y are two real vectors in $\mathbb{R}^d$, and that c is the the center of the cluster associated to x. Let us expand the Gaussian in the following way:

$$e^{-\|y-x\|^2/\eta^2} = e^{-\|\Delta y\|^2/\eta^2} e^{-\|\Delta x\|^2/\eta^2} e^{2\Delta y\cdot\Delta x/\eta^2}, \tag{11}$$

where $\Delta y = y - c$ and $\Delta x = x - c$. In expression (11), the first two exponential terms can be evaluated individually at either x or y. The only problem left is to evaluate the last term where x and y are entangled. One way of breaking the entanglement is to expand the exponential while paying attention to the fact that the inner product is nothing else than a real number. Let us note:

$$|\alpha| = \alpha_1 + \cdots + \alpha_d, \quad x^\alpha = x_1^{\alpha_1} \cdots x_d^{\alpha_d}, \quad \alpha! = \alpha_1! \cdots \alpha_d!.$$

For any x and y real vectors in $\mathbb{R}^d$, we have:

$$e^{2x\cdot y} = \sum_{n\geq 0} \frac{2^n (x\cdot y)^n}{n!}, \quad \text{and} \quad (x\cdot y)^n = \sum_{|\alpha|=n} \frac{n!}{\alpha_1! \ldots \alpha_d!} x^\alpha \times y^\alpha.$$

Thus we obtain the following expansion formula for (11):

$$e^{-\|y-x\|^2/\eta^2} = e^{-\|\Delta y\|^2/\eta^2} e^{-\|\Delta x\|^2/\eta^2} \sum_{\alpha\geq 0} \frac{2^{|\alpha|}}{\alpha!} \left(\frac{\Delta y}{\eta}\right)^\alpha \times \left(\frac{\Delta x}{\eta}\right)^\alpha. \tag{12}$$

In (12) lies the very improvement of this technique when compared to the FGT. Indeed, a truncation of this multivariate expansion at degree p means that we consider $\alpha_1 + \alpha_2 + \cdots + \alpha_d \leq p$, while with FGT, it means that $\alpha_1 \leq p, \ldots, \alpha_d \leq p$. The number of coefficients in the expansion is thus reduced from p^d in the FGT to $\frac{(p+d)!}{p!d!}$ in the IFGT. As an example, when $d = 12$ and $p = 10$, the original FGT needs 10^{12} terms, while the IFGT needs only 646646. For $d \to \infty$ and moderate p, the number of terms becomes $O(d^p)$.

Let us now apply this technique to our incremental algorithm. Recall that, at iteration k and time step t_j, we need to calculate:

$$Q_j^k(x_{t_j}^k) = \sum_{i=0}^{k} \rho_j^i \; \Delta_j^i \; K_j^i(x_{t_j}^i, x_{t_j}^k), \quad \forall j \leq N.$$

Recall that we have divided the space into N_k η-wide clusters B_s with centers c_s. Let us note $I_s, 1 \leq s \leq N_k$ the set of indexes such that:

$$i \in I_s \iff x_{t_j}^i \in B_s \iff c_s \text{ nearest center from } x_{t_j}^i,$$

we obtain the following evaluation rule for the temporal difference computation in Algorithm 2.1:

$$Q_j^k(x_{t_j}^k) = \sum_{s=1}^{N_k} \sum_{\alpha \geq 0} \frac{2^{|\alpha|}}{\alpha!} C_s(\alpha) e^{-\frac{\|x_{t_j}^k - c_s\|^2}{\eta^2}} \left(\frac{x_{t_j}^k - c_s}{\eta} \right)^\alpha, \tag{13a}$$

$$C_s(\alpha) = \sum_{i \in I_s} \rho_j^i \, \Delta_j^i \, e^{-\frac{\|x_{t_j}^i - c_s\|^2}{\eta^2}} \left(\frac{x_{t_j}^i - c_s}{\eta} \right)^\alpha. \tag{13b}$$

Thus, one can see in (13) that the number of operations for an evaluation is linear with respect to the number of centers, which is supposed to be much lower than the number of points. Practically, we use the following scheme:

- At iteration k, compute the temporal difference using (13) and the drawing $x_{t_j}^k$. It can be seen as collecting the influence of the boxes around $x_{t_j}^k$.
- Update $C_s(\alpha)$, with s the index of the box in which $x_{t_j}^k$ lies. This can be seen as adding the influence of the new kernel in its box:

$$C_s(\alpha) = C_s(\alpha) + \rho_j^k \, \Delta_j^k \, e^{-\frac{\|x_{t_j}^k - c_s\|^2}{\eta^2}}.$$

As one can see, in order to apply IFGT, one needs constant bandwidths η. However, our algorithm works with decreasing bandwidths η^k. Thus, for the application of the IFGT to our algorithm, we choose piecewise constant decreasing bandwidths, and associate a space partition and a set of coefficients $C_s^{(l)}(\alpha)$ to every step l in the piecewise constant sequence (η^k). In other words, each time a lower bandwidth size is attained, only the coefficients associated with the new partition are updated. However, coefficients associated to the "old" partitions are still used for evaluation.

3.2. **Choice of the steps.** Recall that ρ^k is the multiplying factor in the temporal difference and η^k is the bandwidth of kernel K^k. To ensure the convergence of the algorithm, one needs the following relations:

$$\sum_{k \in \mathbb{N}} \rho^k \varepsilon^k = +\infty, \quad \sum_{k \in \mathbb{N}} \left(\rho^k\right)^2 \varepsilon^k < +\infty, \quad \sum_{k \in \mathbb{N}} \rho^k \varepsilon^k \eta^k < +\infty,$$

where $\eta^k = (\varepsilon^k)^{\frac{1}{d}}$ and d is the dimension of the space.

Let us choose $\rho^k \simeq \frac{1}{k^\alpha}$, $\varepsilon^k \simeq \frac{1}{k^\beta}$, with $\alpha, \beta \in \mathbb{R}$. Then we have the following relations:

$$\begin{cases} 0 & \leq & \alpha + \beta \leq 1, \\ 1 & < & \alpha + 2\beta, \\ 1 & < & \alpha + \beta(1 + \frac{1}{d}). \end{cases}$$

We draw in Figure 1 the domain where the powers of the steps ε^k and ρ^k can be chosen. When the dimension of the space grows, the relations between the powers of the steps become:

$$\alpha + \beta = 1, \quad \alpha > 0, \quad \beta > 0, \tag{14}$$

i.e. the triangle vanishes on the segment $\alpha + \beta = 1$.

To choose the most efficient powersteps within this triangle, statistical tests on a large number of runs seems to be the most reasonable. It is a tradeoff between robustness and speed of convergence. In section 4, we detail how we performed such a statistical analysis.

3.3. **Acceleration of the rate of convergence by averaging.** We know that for a finite dimensional stochastic gradient type algorithm, the highest rate of convergence is given by methods requiring a large amount of a priori data. Another way of developping optimal algorithms (algorithms having the highest rate of convergence) has been studied by Polyak and Juditsky [PJ92]. It is based on the idea of averaging the iterates, while using larger stepsizes for the approximation. This improvement is one of the most important in this field since the 1960s. It is based on the paradoxical idea that a slow algorithm having less than optimal convergence rate can be averaged and attain an optimal rate of convergence.
We note $\hat{Q}_j^k$ the k-th averaged estimate of function Q_j^* and replace the update equation (2.1) by

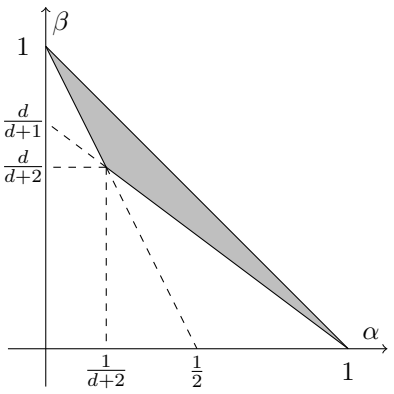

FIGURE 1. Domain where α and β can be chosen

the following two-step update for all $j \leq N$:

$$\begin{cases} Q_j^{k+1}(\cdot) &= Q_j^k(\cdot) + \rho_j^k \Delta_j^k K_j^k(x_j^k, \cdot), \\ \hat{Q}_j^{k+1}(\cdot) &= \frac{1}{k+1} \sum_{l=1}^{k+1} Q_j^l(\cdot). \end{cases}$$

We could also write the more practical update equation $\hat{Q}_j^{k+1}(\cdot) = \hat{Q}_j^k(\cdot) + \frac{1}{k+1}(Q_j^{k+1}(\cdot) - \hat{Q}_j^k(\cdot))$.

It has been shown in [PJ92], for finite dimensional algorithms, that the variance of the residue decreases like $\sqrt{k}$.

Practitioners acknowledge that the best way to use this method is to begin the averaging when the iterates have already done a great part of the approximation. Moreover, since the shape of our iterates in a kernel-based stochastic gradient algorithm is the following:

$$Q_j^{k+1}(\cdot) = \sum_{i=0}^{k} \rho_j^i \Delta_j^i K_j^i(x_j^i, \cdot),$$

we can rewrite the averaging process in the following way:

$$\hat{Q}_j^{k+1}(\cdot) = \begin{cases} \hat{Q}_j^k(\cdot) + \rho_j^k \Delta_j^k K_j^k(x_j^k, \cdot) & \text{if } k < k_0 \\ \hat{Q}_j^k(\cdot) + \left(\frac{k_{max} - k + 1}{k_{max} - k_0 + 1}\right) \rho_j^k \Delta_j^k K_j^k(x_j^k, \cdot) & \text{if } k \geq k_0 \end{cases}$$

where k_0 denotes the step when we begin the averaging and k_{max} is the total number of iterations desired.

We show in Figure 4 how averaging reduces the variance of the iterates and smoothes the values on a simple pricing example presented in the next section. Moreover, one can observe on Figure 2 how this technique accelerates the convergence of the estimates on a simple one-dimensional option pricing example.

4. Numerical Applications

We here focus on the simulation of price processes driven by Brownian motions (e.g. Black-Scholes model). Let us consider a general Brownian diffusion:

$$dX_t = \mu(t, X_t)\, dt + \sigma(t, X_t)\, dW_t, \tag{15}$$

where $\mu : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$ and $\sigma : [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d \times d}$ are Lipschitz-continuous vector fields and W_t is a d-dimensional Brownian motion with correlation matrix Σ.

It is common (see [BPP05]) to discretize (15) using a classical Euler scheme:

$$X_{t_{j+1}} = X_{t_j} + \mu\left(t_j, X_{t_j}\right)(t_{j+1} - t_j) + \sigma\left(t_j, X_{t_j}\right)\left(W_{t_{j+1}} - W_{t_j}\right), \quad j = 0, \ldots, N,$$

with $t_0 = 0 < t_1 < \cdots < t_N = T$.

It just remains to simulate the Brownian motion increments $W_{t_{j+1}} - W_{t_j}$. We choose to consider low-discrepancy random sequences, combined with Brownian bridge simulation, in order to get an equally distributed repartition of the trajectories and to obtain a good representation of the space.

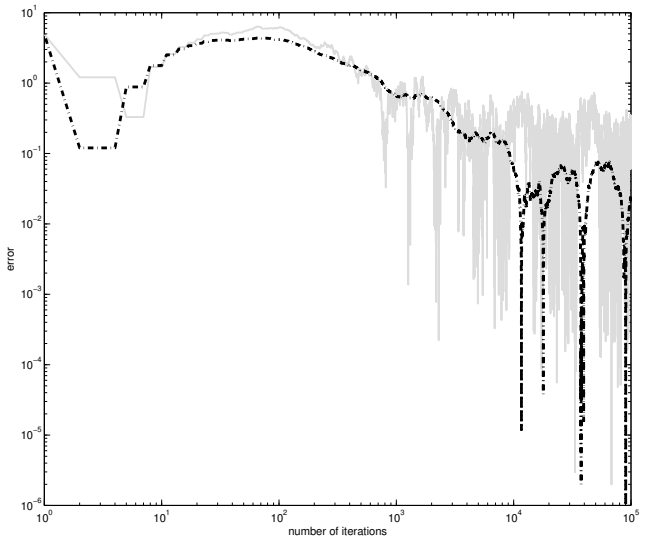

FIGURE 2. Convergence of the estimate of the price of an option with our algorithm, enhanced with averaging technique (dash-dotted black line), or without averaging (solid gray line).

4.1. **Brownian bridge and low-discrepancy random sequences.** An important factor of accuracy in many methods that aim at estimating conditional expectations is the repartition of the random values (here the price trajectory). Most of the techniques use Monte Carlo sequences, also named pseudo-random sequences, because they rely on deterministic procedures. Many implementations of such sequences are available.
Unfortunately, in a certain sense, none of them draws equitable distributions along the iterations. Let us precise in what sense and denote by D_N the discrepancy of an a priori uniformly distributed sequence $(x_i)_{i=1,\dots,N}$ on $[0,1]^d$:

$$D_N = \sup_{\text{Q rectangle in } [0,1]^d} \left| \frac{\text{number of } x_i \in Q}{N} - vol\,(Q) \right|.$$

Thus, for every rectangle Q, its discrepancy is the difference between the relative weight of Q in the sequence $(x_i)_{i=1,\dots,N}$ and its actual volume ($vol\left([0,1]^d\right) = 1$). The discrepancy of a sequence is just the supremum over all rectangles in $[0,1]^d$.

Quasi-random sequences are low-discrepancy random sequences. They aim at distributing equally the numbers along the iterations. Moreover, low-discrepancy random sequences provide the best convergence speed for the numerical computation of expectations, by Koksma-Hlawka inequality ([Nie92], Theorem 2.11). There exist numerous methods of Quasi-Monte Carlo sampling, see [Nie92] for a survey of these methods.

Because the most information we have is on the last time step (the limit condition in (4)), it seems reasonable to require a good representation of the state space at this time step. This implies to draw directly from x_{t_0} the prices x_{t_N}, with quasi-random transitions. But now the problem is: how to draw $x_{t_j}, 0 < j < N$, knowing x_{t_0} and x_{t_N} without introducing a bias ?

In the case of Brownian diffusion processes, the Brownian bridge brings the answer. Indeed, one has the equivalence between the two following drawing procedures for Brownian motions:

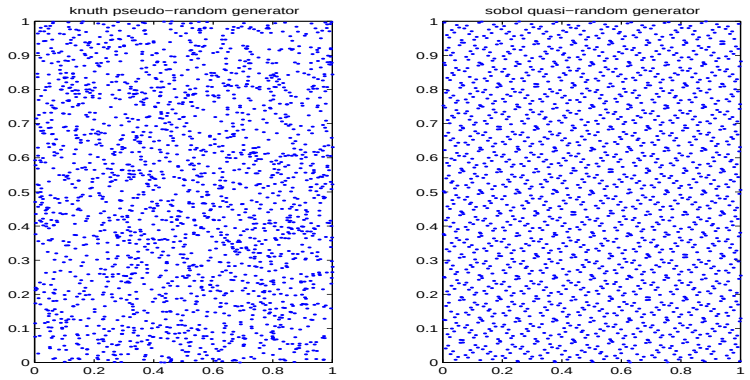

FIGURE 3. Repartition of pseudo and quasi random sequences, uniformly distributed on $[0,1]^2$

- The classical approach: for $1 \leq j \leq N$, knowing w_{t_j}, draw $\delta w_{t_j} = w_{t_{j+1}} - w_{t_j}$:
$$\delta w_{t_j} \sim \mathcal{N}\left(0, \left(t_{j+1} - t_j\right)\Sigma\right),$$
$$x_{t_{j+1}} = x_{t_j} + \mu\left(t_j, x_{t_j}\right)\left(t_{j+1} - t_j\right) + \sigma\left(t_j, x_{t_j}\right)\delta w_{t_j}.$$
- The Brownian bridge: knowing w_{t_0}, draw w_{t_N} with transition following $\mathcal{N}\left(0, \left(t_N - t_0\right)\Sigma\right)$. Then, knowing $w_{t_{j+1}}$ and w_{t_0}, compute w_{t_j} backwards by drawing
$$w_{t_j} \sim \mathcal{N}\left(w_{t_0} + \frac{j}{j+1}(w_{t_{j+1}} - w_{t_0}), \frac{j\left(t_{j+1} - t_j\right)}{j+1}\Sigma\right).$$
Finally, compute x_{t_j} with transitions $\delta w_{t_j} = w_{t_{j+1}} - w_{t_j}$.

The reader may refer to [Dud02], section 12.3, for detailed explanations on the Brownian bridge.

Remark 4.1. The properties listed above are specific to Brownian motions. With Black and Scholes processes, the price follows a log-normal diffusion and one has to write the Brownian bridge for $log\left(x\right)$.

4.2. **Two-dimensional option pricing.** We apply algorithm 2.1 to some one and two-dimensional option pricing problems. Since the algorithm does not depend on any sort of structure in the problem besides requiring the Markov property of the price process, we can consider any sort of payoff. Let us consider the following problem:

- X_t and Y_t are transfer rates and both follow Black and Scholes diffusions with annual interest rate r and volatility σ:
$$dX_t = X_t\left(rdt + \sigma dW_t^{(1)}\right),$$
$$dY_t = Y_t\left(rdt + \sigma dW_t^{(2)}\right),$$
discretized by a classical Euler scheme on $(t_0, \ldots, t_N)$. Moreover, we denote ρ the covariance between the Brownian motions $W_t^{(1)}$ and $W_t^{(2)}$.
- The classical intrinsic value of the option on transfer rates at time t is $g\left(t, X_t, Y_t\right) = e^{-rt}\max\left(0, X_t - Y_t\right)$,
- the maturity of the option is T_{max},
- we want to estimate the price of the option for all maturities less or equal to T_{max}, knowing the prices at the beginning are $X_0 = x_0$ and $Y_0 = y_0$.

Results with $T_{max} = 1$ and evenly spaced t_j with $\delta t = \frac{1}{25}$ are shown in Figure 4. We choose $\sigma = 0.2$, $r = 0.05$ annually, $x_0 = 40.0$ and $y_0 = 36.0$, and $\rho = 0$. Estimates by algorithm 2.1 (on the left) and by the averaged iterates of algorithm 2.1 (on the right) are compared with the solution obtained by dynamic programming with a finely discretized state space, which is supposed to be an accurate estimate of the real solution. As Polyak and Juditsky [PJ92] suggest, we choose the steps to decrease more slowly than what [BRS05] suggest, as well as the sequence of bandwidths. With these setups, even though non-avaraged estimates do not seem to converge very well, this

ensures a good convergence of the averaged estimates.

We draw in Figure 4 the behaviour of $\left(Q_j^k\left(x_0\right)\right)_{0\leq j\leq N}$ for averaged and non-averaged iterates, with $k = 500$, 1000 and 10000 iterations. One can observe the influence of averaging on the stability of the algorithm.

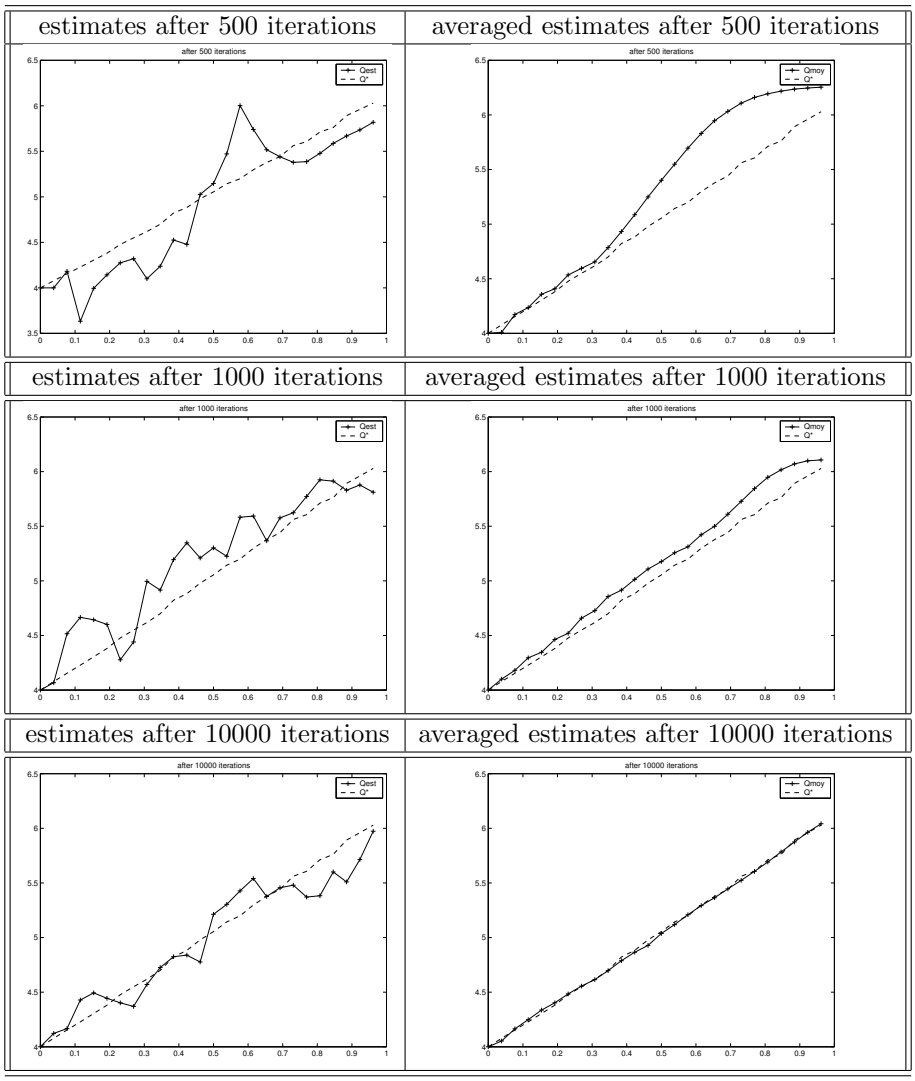

FIGURE 4. Results of convergence for two-dimensional option pricing. Our algorithm (solid line) compared with finely discretized dynamic programming (dotted line)

4.3. **Statistical comparisons.** Since our algorithm has been implemented in the option pricer *Premia*[1], we can compare statistically our results with numerous known algorithms. Statistical tests are made on an American put option relying on the minimum of stocks: $g(x_1, x_2) = e^{-rt}\max\left(0, K - \min\left(x_1, x_2\right)\right)$, with K the value of the strike.
Both stocks follow a Black-Scholes dynamic, discretized by an Euler scheme. We randomly draw a large number of experiments, choosing the values of the following inputs uniformly among the sets specified below:

- Original prices in $\{50, 55, \ldots, 145\}$,
- Volatilities in $\{0.1, 0.2, \ldots, 1\}$,
- Correlation in $\{0, 0.1, \ldots, 0.9\}$,

[1]More information on this software can be found on the webpage : `http://www.premia.fr`

- Annual interest rate in $\{1, 2, \dots, 10\}$,
- Maturity in $\{0.5, 0.6, \dots, 1.4\}$,
- Strike in $\{50, 55, \dots, 145\}$.

Reference prices and hedging values are computed using the dynamic programming equation (2) with a finely discretized state space and a large number of Monte Carlo simulations to compute the expectations. This is a time consuming method but it ensures to have the reference prices and hedging values.

We then compute the price using our algorithm and draw the boxplots that represent the distribution of the errors on prices and on deltas.

On Figure 5, one can observe the convergence of our algorithm for nearly every experiment, drawn with boxplots. In the central box lies half part of the runs, either above or under the box lies 25% of the runs. The median is represented by a red line.

On Figures 6, 7 and 8, one can observe the errors of our algorithm on the price, the first hedging value and the second hedging value respectively, when compared with classical pricing methods.

We compute the hedging values with finite differences, but the Markov property of the chain allows us not to perform the algorithm three times (in dimension 2), but to consider three different starting points for our price process $((x_1, x_2), (x_1 + \delta, x_2), (x_1, x_2 + \delta))$ and to rotate on them along with the iterations. With this method, we obtain very accurate approximations of the hedging values, as one can see on Figures 7 and 8.

We compare our results with the Longstaff-Schwartz (cf [LS01]), Barraquand-Martineau (cf [BM95]) and Lions-Regnier (cf [BCZ05]) algorithms that are respectively a linear regression method, a dynamic programming method with state aggregation, and a Malliavin calculus.

For reference, the three methods are implemented in the option pricer *Premia* and their parameters, which are the default parmeterers in *Premia*, are the following:

- The Longstaff-Schwartz algorithm here uses 50000 iterations and a 9-dimensional canonical regression basis.
- The Barraquand-Martineau here uses 20000 iterations, 100 cells and the size of the grid initializing sample is equal to 300.
- The Lions-Regnier algorithm uses 1000 iterations.

For all the methods, the relative value of the delta increment for the finite difference computation is equal to 10% and the number of exercise dates is equal to 20. More information is available in the *Premia* documentation.

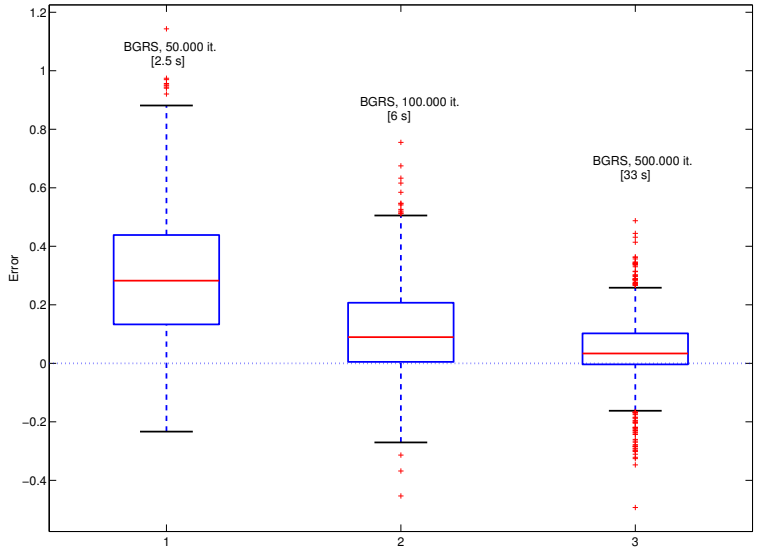

FIGURE 5. Error on the price. Performance of our algorithm on a large number of experiments, drawn with boxplots.

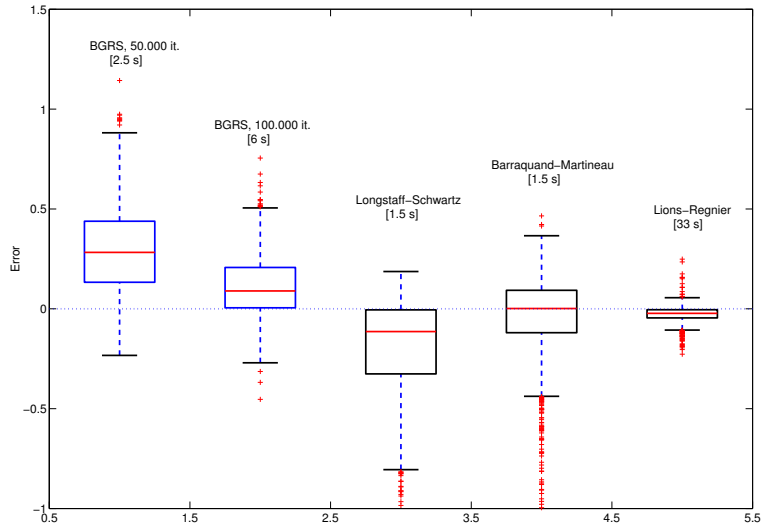

FIGURE 6. Error on the price. Performance of our algorithm and others on a large number of experiments, drawn with boxplots.

4.4. **Results on Asian options.** Since we prove the convergence of our algorithm with no requirements on the distribution of the underlying price process (except its Markov property), we can easily use our method on more exotic options.

As an example, we present results on Asian options, whose payoff depends on the average of the price over a time period. Thus, the value of this option can be written :

$$J_0(x) = \max_{\tau \in \mathcal{T}(t_0, t_N)} \mathbb{E}\left[g\left(\tau, X_\tau, A_{X,\tau}\right) \middle| X_{t_0} = x \right] \tag{16}$$

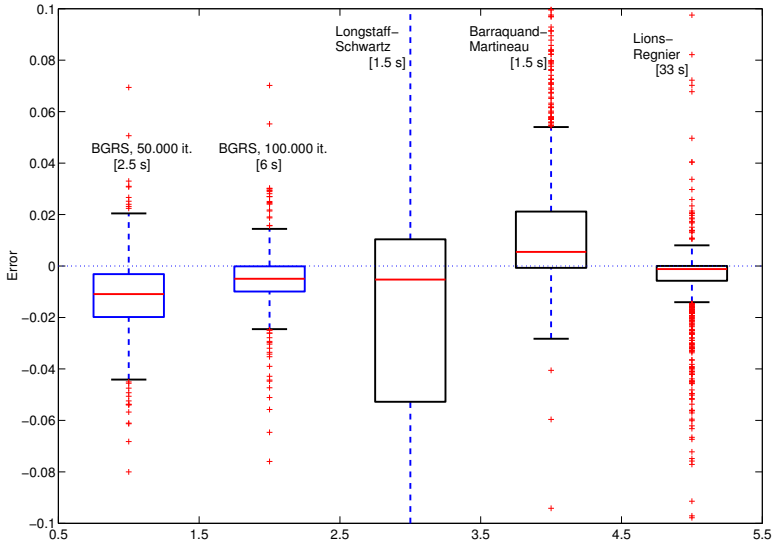

FIGURE 7. Error on the first hedging value. Performance of our algorithm and others on a large number of experiments, drawn with boxplots.

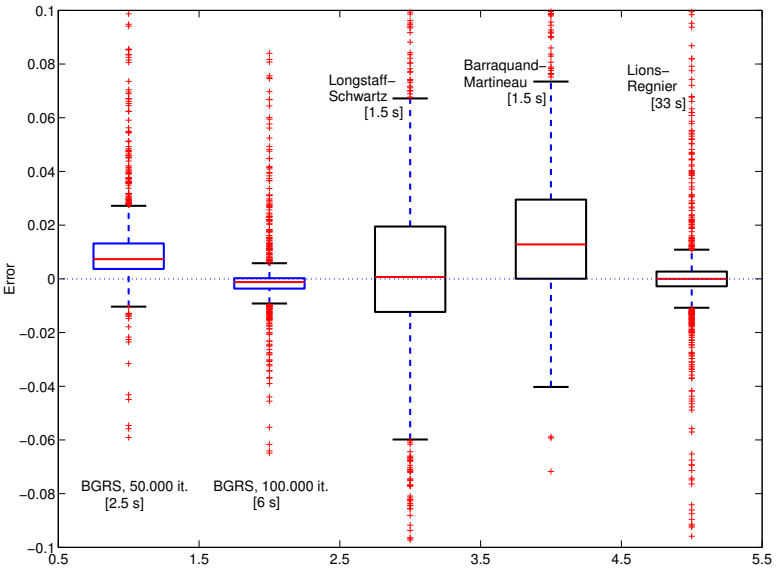

FIGURE 8. Error on the second hedging value. Performance of our algorithm and others on a large number of experiments, drawn with boxplots.

where :

$$A_{X,t} = \frac{1}{t - t_0} \int_{t_0}^{t} X_s ds$$

is the average of the price process X over the time period $[t_0, t]$. Formally, we consider a single stock $(X_\tau, A_{X,\tau})$ whose first component is the price and the second is its average. This process is Markovian, and there are many ways of simulating it. The simpler (and probably poorer) way, if

we denote by X_t the price process and $\tilde{A}_t$ its estimated average, is to perform Riemann sums:

$$\begin{cases} \tilde{A}_{X,t_0} &= X_{t_0}, \\ \tilde{A}_{X,t_{j+1}} &= \dfrac{j \cdot \tilde{A}_{X,t_j} + X_{t_{j+1}}}{j+1}. \end{cases}$$

A better way of simulating the average of the stochastic process X_t would be to simulate trajectories with a small discretization on every subset $[t_i, t_{i+1}]$, and then to simulate the average using these points. Another way is to perform a higher accuracy integration scheme, like a trapezoidal method (see [LT01] for details on the accuracy of these schemes).

With the same validation procedure as for American options, on the particular case of an Asian call fixed payoff, for which the payoff at time t, with stock X_t and average $A_{X,t}$ is $g(t, X_t, A_{X,t}) = e^{-rt} \max(0, K - A_{X,t})$, we draw the errors on the price of the option on Figure 9. Actual prices of the options are between 0 and 100.

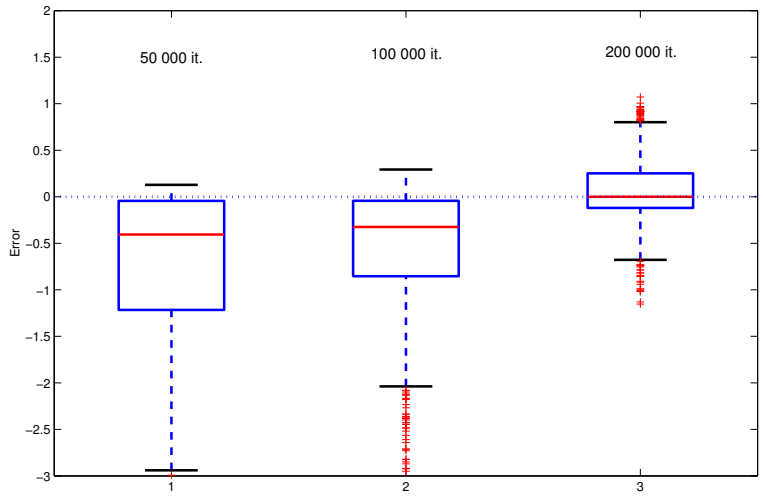

FIGURE 9. Error on the price of Asian American-type options (reference is a finely discretized dynamic programming method). Performance of our algorithm after 50 000 iterations, drawn with boxplots.

5. CONCLUSION

In this paper we present the application of a kernel-based stochastic gradient algorithm to American option pricing. Our approach avoids any a priori discretization, besides the usual time discretization. Thus, it converges toward the optimum of the Bermudean option pricing problem. After presenting the algorithm, we provide a convergence proof by means of stochastic approximation schemes. We also present the numerical tools used for accelerating the convergence, and we compare our method to some classical methods on a two-dimensional American option pricing problem. It appears that our results are relevant, when compared to other algorithms, especially for the evaluation of the hedging values. Moreover, since the algorithm only requires the price process to be a Markov chain, it is readily applicable to a large class of exotic option pricing problems.

Future research directions include avoiding the time discretization as well, by adding a new component in the kernels, and applying our algorithm straightforward. Further studies will be done in this direction in order to describe precisely the assumptions required for this extension.

References

[BCZ05] V. Bally, L. Caramellino, and A. Zanette, *Pricing and hedging American options by Monte Carlo methods using a Malliavin calculus approach*, Monte Carlo Methods and Applications **11** (2005), no. 2, 97–134.

[BD59] R. Bellman and S.E. Dreyfus, *Functional approximations and dynamic programming*, Math tables and other aides to computation **13** (1959), 247–251.

[BG04] M. Broadie and P. Glasserman, *A stochastic mesh method for pricing high-dimensional american options*, Journal of Computational Finance **7** (2004), 35–72.

[BM95] J. Barraquand and D. Martineau, *Numerical valuation of high dimensional multivariate american securities*, Journal of Financial and Quantitative Analysis **30** (1995), no. 3, 383–405.

[Boc55] S. Bochner, *Harmonic analysis and the theory of probability*, University of California Press, Berkeley, 1955.

[BPP05] V. Bally, G. Pagès, and J. Printems, *A quantization method for pricing and hedging multi-dimensional american style options*, Mathematical Finance **15** (2005), no. 1, 119–168.

[BRS05] K. Barty, J.-S. Roy, and C. Strugarek, *A perturbed gradient algorithm in Hilbert spaces*, Optimization Online (2005), http://www.optimization-online.org/DB_HTML/2005/03/1095.html.

[BT96] D.P. Bertsekas and J.N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996.

[BY03] M. Broadie and Y. Yamamoto, *Application of the fast Gauss transform to option pricing*, Management Science **49** (2003), no. 8, 1071–1088.

[CRR79] J. Cox, S. Ross, and M. Rubinstein, *Option pricing: A simplified approach*, Journal of Financial Economics **7** (1979), 229–263.

[CW98] X. Chen and H. White, *Nonparametric adaptive learning with feedback*, J. Econ. Theory **82** (1998), 190–222.

[dFVR04] D.P. de Farias and B. Van Roy, *A linear program for bellman error minimization with performance guarantees*, submitted to Math. Oper. Res., November 2004.

[Dud02] R.M. Dudley, *Real analysis and probability*, Cambridge Uuniversity Press, Cambridge, UK, 2002.

[FLL+99] É. Fournié, J.M. Lasry, J. Lebouchoux, P.L. Lions, and N. Touzi, *Applications of Malliavin calculus to Monte Carlo methods in Finance*, Finance & Stochastics **3** (1999), 391–412.

[FLLL01] É. Fournié, J.M. Lasry, J. Lebouchoux, and P.L. Lions, *Applications of Malliavin calculus to Monte Carlo methods in Finance II*, Finance & Stochastics **5** (2001), 201–236.

[Gla03] P. Glasserman, *Monte Carlo methods in financial engineering*, Springer, 2003.

[GR87] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulation*, Journal of Computational Physics (1987), 73, 2:325–348.

[GS91] L. Greengard and J. Strain, *The fast gauss transform*, SIAM Journal on Scientific and Statistical Computing **12** (1991), no. 1, 79–94.

[LS01] F. A. Longstaff and E. S. Schwartz, *Valuing american options by simulation: A simple least squares approach*, Rev. Financial Studies **14** (2001), no. 1, 113–147.

[LT01] B. Lapeyre and E. Temam, *Competitive Monte Carlo methods for the pricing of asian options*, 2001, pp. 39–59.

[Nie92] H. Niederreiter, *Random number generation and quasi-monte carlo methods*, SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia (1992).

[PJ92] B. T. Polyak and A. B. Juditsky, *Acceleration of stochastic approximation by averaging*, SIAM Journal on Control and Optimization (1992), 838 – 855.

[RM51] H. Robbins and S. Monro, *A stochastic approximation method*, Annals of Mathematical Statistics **22** (1951), 400–407.

[Ros76] S. Ross, *The arbitrage theory of capital asset pricing*, Journal of Economic Theory **13** (1976), 341–360.

[SB98] R.S. Sutton and A.G. Barto, *Reinforcement learning, an Introduction*, MIT press Cambridge, 1998.

[Sut88] R.S. Sutton, *Learning to predict by the method of temporal difference*, IEEE Trans. Autom. Control **37** (1988), 332–341.

[TV99] J. Tsitsiklis and B. Van Roy, *Optimal stopping for markov processes: Hilbert space theory, approximation algorithm and an application to pricing high-dimensional financial derivatives*, IEEE Trans. Autom. Control **44** (1999), 1840–1851.

[WW69] T.J. Wagner and C.T. Wolverton, *Recursive estimates of probability densities*, IEEE Trans. Syst. Man. Cybern. **5** (1969), 307.

[YDGD03] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis, *Improved fast gauss transform and efficient kernel density estimation*, IEEE International Conference on Computer Vision (2003), 464–471.

K. Barty, EDF R&D, 1, avenue du Général de Gaulle, F-92141 Clamart Cedex,
E-mail address: kengy.barty@edf.fr

P. Girardeau, École Nationale Supérieure de Techniques Avancées (ENSTA), also with EDF R&D
E-mail address: pierre.girardeau@ensta.fr

J.-S. Roy, EDF R&D, 1, avenue du Général de Gaulle, F-92141 Clamart Cedex,
E-mail address: jean-sebastien.roy@edf.fr

C. Strugarek, EDF R&D, also with the École Nationale Supérieure de Techniques Avancées (ENSTA) and the École Nationale des Ponts et Chaussées (ENPC)
E-mail address: cyrille.strugarek@edf.fr