Monte-Carlo valuation of American options: facts and new algorithms to improve existing methods

Bouchard B., Warin X.

Abstract The aim of this paper is to discuss efficient algorithms for the pricing of American options by two recently proposed Monte-Carlo type methods, namely the Malliavian calculus and the regression based approaches. We explain how both techniques can be exploited with improved complexity and efficiency. We also discuss several techniques for the estimation of the corresponding hedging strategies. Numerical tests and comparisons, including the quantization approach, are performed.

1 Introduction

In the last decades, several Monte-Carlo type techniques have been proposed for the numerical computation of American option prices, or more generally the evaluation of value functions associated to semi-linear parabolic equations, with possible free boundary, see e.g. the survey paper [11]. The idea of combining Monte-Carlo methods with approximations of the expectation operator in backward induction schemes comes back to Carrière [14] and was popularized by Longstaff and Schwartz [31]. This was the starting point of fruitful researches involving, in particular, a series of papers by Pagès and its co-authors, see e.g. [3] or [4], on the quantization approach, Lions and Renier [30], and Bouchard, Ekeland and Touzi [10, 12] on the Malliavin calculus based formulation.

The aim of all the above mentioned papers is to compute prices for American or Bermudan options in (relatively) high dimensions, when purely deterministic techniques (finite differences or finite elements for partial differential equations,

Bruno Bouchard
Bruno Bouchard, CEREMADE-University Paris-Dauphine and CREST-ENSAE e-mail: bruno.bouchard@ceremade.dauphine.fr

Xavier Warin
Xavier Warin, EDF R&D & FiME, Laboratoire de Finance des Marchés de l'Energie (www.fime-lab.org) e-mail: xavier.warin@edf.fr

approximating trees) are made inefficient by the so-called curse of dimensionality. The rationality behind the purely Monte-Carlo based approaches of [31] and [12] is that the convergence speed of the proposed schemes does not a-priori depend on the dimension of the problem. This is the usual justification for the use of such techniques in numerical integration, although, like for any Monte-Carlo method, the dimension plays an import role at finite distance, usually through the variance of the estimation error or the complexity of the algorithm.

In the regression based approach of Longstaff and Schwartz [31], it appears in the choice of the basis of polynomials used for the numerical estimation of conditional expectations. Such a choice is made very difficult in practice when the dimension increases.

Many papers are devoted to such an issue and it works well on some particular (possibly complex) payoffs, see e.g. [37] among others. However, the question of choosing a good basis in a practical non-standard situation is in general difficult, and this approach does not allow to built efficient payoff independent algorithms, particularly in high dimensions. The reason is very simple: the error should essentially be controlled by the projection error on the basis. Hence the basis should be close to the pricing function, which is unknown (see e.g. Theorem 2 in [25] for explicit bounds obtained for non-reflected BSDEs). On the other hand, local basis based on hyper-cubes partitions of the whole space allow for easier error estimates and seem to be much more robust, see Section 6.1 in [25].

In the Malliavin based approach, the dimension of the problem appears through an exploding variance of the estimators of the conditional expectation operators. This is due to the Skorohod integrals (usually called *Malliavin weights*) which enter into the representation of conditional expectations as the ratio of two unconditional expectations obtained by integrating by parts the Dirac mass which shows up when applying the Bayes' rule. The variance of these terms explodes with the dimension of the underlying factor and with the number of time steps. Another important issue is the complexity of the algorithm, which, *a-priori*, seems to be of order of the number of simulated paths N to the square: $O(N^2)$. Since the variance explodes with the number of underlying factors and the number of times steps, a large number of simulated paths has to be used in order to achieve a good precision in high dimension. The above mentioned complexity thus makes this approach a-priori much too slow in practice.

The aim of this paper is to explain how both methods can be improved in order to circumvent the above mentioned criticisms. As for the non-parametric regression based method, we suggest to modify the purely non-parametric method of [25] by adapting the support of the function basis to the density of the underlying factors. The main advantage of this approach is that the regression basis is not chosen a-priori but automatically adapted to the distribution of the underlying process. Concerning the Malliavin based approach, we explain how an efficient algorithm with a reduced complexity can be constructed. We shall see in particular that the complexity of this algorithm is far from being of the order of the number of simulated paths to the square, as claimed in many papers. It is of order $O(N\ln(N)^{(d-1)\vee 1})$ where d is the dimension of the underlying factor.

For both methods, we will explain how, with essentially the same computation costs, two consistent estimators can be build at the same time. The first one corresponds to the approach of Longstaff and Schwartz [31], which consists in estimating the optimal exercise time. The second is based on the computation of the prices at each time through a pure backward induction procedure. Because, the estimator of the optimal exercise rule is by nature sub-optimal, the first price estimator is essentially biased from below. On the other hand, because of the convexity of the max operator, the second one is essentially biased from above. We suggest to consider the corresponding interval to test the accuracy of the estimations. This can be seen as a subsidy for the usual confidence interval in linear Monter-Carlo methods. We refer to [1], [2], [6] or [26] (see also the references therein) for other approaches leading to the construction of upper- and lower-bounds, and to [24] and [28] for numerical studies on penalization and regularization technics.

We shall also investigate different methods for the computation of the hedging strategy. In particular, we shall emphasize that the standard tangent process approach, widely used in the context of European type options, can be used for American options too. We will also consider Malliavin based techniques, following the ideas of the seminal paper [21].

The rest of the paper is organized as follows. In Section 2, we recall fundamental results on the pricing of American and Bermudan options. We discuss the error induced by the approximation of American option prices by their Bermudan counterparts. We also provide different representation for the hedging policy. In Section 3, we explain how these results can be exploited in order to build estimators of the price and the hedging strategy, assuming that we are given a way of approximating conditional expectations. Section 4 is dedicated to the presentation of improved versions of the regression based and the Malliavin based Monte-Carlo algorithms. Numerical experiments and comparisons, including the quantization approach of [4], are presented in Section 5. All over this paper, elements of $\mathbb{R}^d$ are viewed as column vectors and transposition is denoted by $'$.

2 Fundamental results for the construction of numerical algorithms

In this section, we review some fundamental results on the formulation of prices and the representation of hedging strategies that will be used in the algorithms described below.

All over this paper, we shall consider a d-dimensional Brownian motion W on a probability space $(\Omega, \mathscr{F}, \mathbb{P})$ endowed with the natural (completed and right-continuous) filtration $\mathbb{F} = (\mathscr{F}_t)_{t \leq T}$ generated by W up to some fixed time horizon $T > 0$. We assume, for sake of simplicity, that the interest rate is zero and that there exists only one risk neutral measure, which is given by the original probability measure $\mathbb{P}$ (or at least $\mathbb{P}$ will be considered to be the pricing measure). The

stock dynamics is modeled as the strong solution $X = (X^1, \dots, X^d)$ of the stochastic differential equation:

$$X_t = X_0 \int_0^t \sigma(s, X_s) dW_s \quad t \leq T, \tag{1}$$

where σ is a Lipschitz continuous function defined on $[0,T] \times \mathbb{R}^d$ and taking values in the set of d-dimensional square matrices. For sake of simplicity, we shall assume from now on that the stock price process X can be perfectly simulated on any finite time grid of $[0,T]$, which is the case in most standard market models.

These choices are made in order to simplify the presentation, but the above algorithms/results could clearly be extended to more general situations, see e.g. [11] for convergence issues.

2.1 Definitions and facts

We recall in this section some well-know facts on the pricing of American and Bermudan options.

From now on, the payoff of the American option is defined as a deterministic measurable function $g : [0,T] \times \mathbb{R}^d \mapsto \mathbb{R}$, i.e. the seller pay $g(t,x)$ at time t if the option is exercised at time t and the value of the underlying assets at time t is x. We shall assume all over this paper that g has linear growth and is Lipschitz continuous.

Under the above assumption, it follows from standard arguments, see e.g. [19], that the price at time t of the American option is given by a continuous supermartingale P satisfying

$$P_t = \operatorname{esssup}_{\tau \in \mathscr{T}_{[t,T]}} \mathbb{E}[g(\tau, X_\tau) \mid \mathscr{F}_t] \quad \text{for } t \leq T \ \ \mathbb{P}-\text{a.s.}, \tag{2}$$

where $\mathscr{T}_{[t,T]}$ denotes the set of stopping times with values in $[t,T]$.

Similarly, the price of a Bermudan option with the same payoff function, but which can be exercised only at times in

$$\pi := \{0 = t_0 < t_1 < t_2 < \cdots < t_\kappa = T\}, \text{ for some } \kappa \in \mathbb{N},$$

is given by a làdcàg supermartingale P^π satisfying

$$P_t^\pi = \operatorname{esssup}_{\tau \in \mathscr{T}^\pi_{[t,T]}} \mathbb{E}[g(\tau, X_\tau) \mid \mathscr{F}_t] \quad \text{for } t \leq T \ \ \mathbb{P}-\text{a.s.}, \tag{3}$$

where $\mathscr{T}^\pi_{[t,T]}$ denotes the set of stopping times with values in $[t,T] \cap \pi$.

It then follows from the Doob-Meyer decomposition and the martingale representation theorem, that we can find predictable processes ϕ and ϕ^π as well as non-decreasing processes A and A^π such that

$$\mathbb{E}\left[\int_0^T |\phi_s|^2 + |\phi_s^\pi|^2 ds\right] < \infty \, , A_0 = A_0^\pi = 0$$

and

$$P_t = P_0 + \int_0^t \phi_s' dW_s - A_t \, , \, P_t^\pi = P_0^\pi + \int_0^t \phi_s^{\pi\prime} dW_s - A_t^\pi \quad \text{for } t \le T \;\; \mathbb{P}-\text{a.s.} \quad (4)$$

The processes ϕ and ϕ^π are related to the hedging strategy of, respectively, the American and the Bermudan option. More precisely, the number of units of stocks to hold in the hedging portfolio are given by $\psi' := \phi'\sigma^{-1}(\cdot, X)$ and $\psi^{\pi\prime} := \phi^{\pi\prime}\sigma^{-1}(\cdot, X)$ whenever these quantities are well-defined.

Moreover,

$$P_t = \mathbb{E}\left[g(\hat{\tau}_t, X_{\hat{\tau}_t}) \mid \mathscr{F}_t\right] \;\text{ and }\; P_t^\pi = \mathbb{E}\left[g(\hat{\tau}_t^\pi, X_{\hat{\tau}_t^\pi}) \mid \mathscr{F}_t\right] \quad \text{for } t \le T \;\; \mathbb{P}-\text{a.s.}\,, \quad (5)$$

where

$$\hat{\tau}_t := \inf\{s \in [t,T] \;:\; P_s = g(s, X_s)\} \;\text{ and} \quad (6)$$
$$\hat{\tau}_t^\pi := \inf\{s \in [t,T] \cap \pi \;:\; P_s^\pi = g(s, X_s)\} \quad (7)$$

are the (first) optimal exercise times, after t. In particular, P and P^π are martingales on $[t, \hat{\tau}_t]$ and $[t, \hat{\tau}_t^\pi]$ respectively, for all $t \le T$.

2.2 From Bermudan to American options

Most numerical methods for the pricing of American options are based on the approximation by Bermudan options with time grid π with mesh $|\pi| := \max_{i<\kappa}(t_{i+1} - t_i)$ going to 0. The approximation is justified theoretically by the following result, see [3] and [9].

Theorem 1. *The following holds:*

$$\max_{i \le \kappa} \mathbb{E}\left[|P_{t_i} - P_{t_i}^\pi|^2\right]^{\frac{1}{2}} \le O(|\pi|^{\frac{1}{4}}) \, .$$

If moreover there exists $\rho_1 : \; [0,T] \times \mathbb{R}^d \mapsto \mathbb{R}^{d+1}$ *and* $\rho_2 : \; [0,T] \times \mathbb{R}^d \mapsto \mathbb{R}_+$ *such that*

$$|\rho_1(t,x)| + |\rho_2(t,x)| \;\le\; C_L(1 + |x|^{C_L})$$

$$g(t,x) - g(s,y) \;\le\; \rho_1(t,x)'(s-t, y-x) + \rho_2(t,x)(|t-s|^2 + |x-y|^2) \;\; , \forall\, x,y \in \mathbb{R}^d, t,s \in [0,T] \, . \quad (8)$$

for some constant $C_L > 0$*, then*

$$\max_{i\leq\kappa}\mathbb{E}\left[|P_{t_i}-P^{\pi}_{t_i}|^2\right]^{\frac{1}{2}}+\mathbb{E}\left[\int_0^T|\phi_s-\phi^{\pi}_s|^2ds\right]^{\frac{1}{2}}\leq O(|\pi|^{\frac{1}{2}})\ .$$

Note that the assumption (8) is satisfied by most payoffs in practice.

In view of this convergence result, it is enough to focus on the pricing of Bermudan options. We will therefore concentrate on this in the following.

Remark 1. We refer to [3] and [9] for the additional error due to the approximation of X by its Euler scheme. For a time step of size $h>0$, It is of order $O(h^{1/4})$ in general, and of order $O(h^{1/2})$ under (8).

2.3 Delta representations

For practical purposes, the computation of the hedging strategy is as important as the estimation of the price process. As for European type options, at least three different methods can be used in practice. In this section, we restrict to the case of Bermudan options, recall the convergence results of Section 2.2.

2.3.1 Finite difference approach

The finite difference approach consist is estimating the price process for different initial conditions. More precisely, let $P^{\pi,\delta}$ be defined as in (1)-(3) with X_0 replaced by $X_0+\delta$, $\delta\in\mathbb{R}^d$. Then, following the standard approach for European options, one could approximate the i-th component of $\phi_0^{\pi\prime}\sigma(0,X_0)^{-1}$ by $(P_0^{\pi,\delta_i}-P_0^{\pi})/h$ or $(P_0^{\pi,\delta_i}-P_0^{\pi,-\delta_i})/2h$ where δ_i is the vector of $\mathbb{R}^d$ defined by $\delta_i^j=h\mathbf{1}_{i=j}$ and $h>0$ is small. A large literature is available on this approach for European type options, see e.g. [17] and the references therein. To our knowledge, no rigorous convergence result is available for American type options. However, in the case of Bermudan options, the results obtained for European options can still be applied at time 0 by considering the deterministic price function $p^{\pi}(t_1,\cdot)$, where p^{π} is implicitly defined by $p^{\pi}(\cdot,X_\cdot)=P^{\pi}$ on $[0,t_1]$, as a given terminal payoff at time t_1.

Note that this requires the computation of two different values of the American option price, for two different initial conditions, which is, a-priori, much too time consuming in comparison to the techniques proposed below. On the other hand the algorithms presented below, Algorithms A1, A2 and A2b, can be easily adapted to this context. Indeed, they produce (or can produce for Algorithm 1), simulated values of option prices on a grid of time corresponding to simulated values of the stock prices. If one starts the simulations of the stock prices at time $-\delta$, $\delta>0$ small, they will thus produce values of the option price at time 0 for simulated, but close if δ is small, values of the stock prices. These can be used to compute the finite differences. Obviously there is no hope that this method will be convergent and the choice of the value of δ is not clear. We will therefore not test this approach here.

2.3.2 Tangent process approach

Assume that g, σ is C_b^1. Then, under a standard uniform ellipticity condition on σ and mild additional regularity assumptions, ensuring the usual smooth pasting property for the American option price on the associated free boundary, it is shown in [23] that there exists a version of ϕ satisfying

$$\phi_0' = \mathbb{E}\left[\nabla g(\hat{\tau}_0, X_{\hat{\tau}_0})' \nabla X_{\hat{\tau}_0}\right] \sigma(0, X_0)$$

where ∇g denote the gradient of g with respect to its space variable and ∇X is the first variation (or tangent) process of X defined as the solution of

$$\nabla X_t = I_d + \int_0^t \sum_{j=1}^{d} \nabla \sigma^j(X_r) \nabla X_r dW_r^j$$

where I_d is the identity matrix of $\mathbb{M}^d$, σ^j is the j-th column of σ, and $\nabla \sigma^j$ the Jacobian matrix of σ^j. This is a natural extension of the well-known result for European options, see [13].

This result was then extended in [9], see also [37], to Bermudan options in terms of the Malliavin derivative process of X, without ellipticity condition. Here, we state it in terms of the first variation process ∇X, compare with Corollary 5.1 and see (5.3) in [9].

Theorem 2. *Assume that $g, \sigma \in C_b^1$ then there exists a version of ϕ^π satisfying*

$$\phi_t^{\pi\prime} = \mathbb{E}\left[\nabla g(\hat{\tau}_t^\pi, X_{\hat{\tau}_t^\pi})' \nabla X_{\hat{\tau}_t} \mid \mathscr{F}_t\right] (\nabla X_t)^{-1} \sigma(t, X_t) \ , \ t \leq T \ . \tag{9}$$

Remark 2. Note that the payoff function g is assumed to be C_b^1 in the above assertion. However, it should be clear that it can be extended to many situations where g is only differentiable a.e. with bounded derivatives. In particular, for one dimensional put options with strike K, it is clear that $X_{\hat{\tau}_t^\pi} < K$ $\mathbb{P}-$a.s. since $g(t,K) = 0$, at least under suitable ellipticity conditions on σ ensuring that $P^\pi > 0$ on $[0,T)$. Since K is the only point where the payoff function is not differentiable, the above representation can be easily extended.

2.3.3 Malliavin calculus approach

An extension of the formulation of the delta similar to the one introduced for European type options in the seminal paper [21] was first proposed in [32]. However, it involves the non-decreasing process A (or A^π) which is difficult to estimate in practice. In the case where we restrict to Bermudan options, then things simplify and the result of Proposition 5.1 in [9], together with a standard integration by parts argument in the Malliavin calculus sense, leads to the following representation.

Theorem 3. *Assume that $g, \sigma \in C_b^1$ and that σ is invertible with bounded inverse, Then there exists a version of ϕ^π satisfying for $t \in [t_i, t_{i+1}),\ i \leq \kappa$*

$$\phi_t^{\pi\prime} = \frac{1}{t_{i+1}-t}\mathbb{E}\left[P_{t_{i+1}}^\pi \int_t^{t_{i+1}} \sigma(s,X_s)^{-1}\nabla X_s dW_s \mid \mathscr{F}_t\right]' (\nabla X_t)^{-1}\sigma(t,X_t). \quad (10)$$

Since P^π is a martingale on each interval $[t, \hat{\tau}_t]$, it can alternatively be written in the following form.

Theorem 4. *Assume that $g, \sigma \in C_b^1$ and that σ is invertible with bounded inverse, Then there exists a version of ϕ^π satisfying for $t \in [t_i, t_{i+1}),\ i \leq \kappa$*

$$\phi_t^{\pi\prime} = \frac{1}{t_{i+1}-t}\mathbb{E}\left[g(\hat{\tau}_t^\pi, X_{\hat{\tau}_t^\pi}) \int_t^{t_{i+1}} \sigma(s,X_s)^{-1}\nabla X_s dW_s \mid \mathscr{F}_t\right]' (\nabla X_t)^{-1}\sigma(t,X_t). \quad (11)$$

Remark 3. In Black-Scholes type models, i.e. $\sigma(t,x) = \mathrm{diag}[x]\,\tilde{\sigma}(t)$ where $\mathrm{diag}[x]$ is the diagonal matrix with i-th diagonal component equal to x^i and $\tilde{\sigma}$ is deterministic with bounded inverse, then the above results still holds true. Also note that the payoff function g is assumed to be C_b^1 in the above assertion. However, it should be clear that it can be extended to more general situations where g can be uniformly approximated by a sequence of C_b^1 functions. This follows from standard stability results for reflected backward stochastic differential equations.

3 Abstract algorithms

3.1 Backward induction for the pricing of Bermudan options

It follows from the formulation (3) of P^π in terms of an optimal stopping problem on a finite time grid, that the price process of the Bermudan option satisfies the *so-called* backward American dynamic programming equation for $i = \kappa - 1, \ldots, 0$

$$P_T^\pi = g(T,X_T) \ \text{ and } \ P_{t_i}^\pi = \max\left\{g(t_i,X_{t_i})\ ,\ \mathbb{E}\left[P_{t_{i+1}}^\pi \mid \mathscr{F}_{t_i}\right]\right\}. \quad (12)$$

or equivalently, thanks to the martingale property of P^π on each interval $[t, \hat{\tau}_t^\pi]$,

$$P_T^\pi = g(T,X_T) \ \text{ and } \ P_{t_i}^\pi = \max\left\{g(t_i,X_{t_i})\ ,\ \mathbb{E}\left[g(\hat{\tau}_{t_{i+1}}^\pi, X_{\hat{\tau}_{t_{i+1}}^\pi}) \mid \mathscr{F}_{t_i}\right]\right\}. \quad (13)$$

Assuming that the involved conditional expectation can be perfectly estimated, this leads to two kind of possible algorithms for the computation of the price of the Bermudan option at time 0. In practice, these operators have to be replaced by a numerical estimation. In what follows, we denote by $\hat{\mathbb{E}}[\cdot \mid \mathscr{F}_{t_i}]$ an approximation of the true condition expectation operator $\mathbb{E}[\cdot \mid \mathscr{F}_{t_i}]$. For $\hat{\mathbb{E}}$ given, the corresponding

approximation schemes are:

Algorithm A1 [optimal exercise time estimation]:
1. Initialization : Set $\hat{\tau}_\kappa^{1,\pi} := T$.
2. Backward induction : For $i = \kappa - 1$ to 0, set $\hat{\tau}_i^{1,\pi} := t_i \mathbf{1}_{A_i^1} + \hat{\tau}_{i+1}^{1,\pi} \mathbf{1}_{(A_i^1)^c}$
where $A_i^1 := \{g(t_i, X_{t_i}) \geq \hat{\mathbb{E}}[g(\hat{\tau}_{i+1}^{1,\pi}, X_{\hat{\tau}_{i+1}^{1,\pi}}) \mid \mathscr{F}_{t_i}]\}$.
3. Price estimator at 0: $\hat{P}_0^{1,\pi} := \hat{\mathbb{E}}[g(\hat{\tau}_0^{1,\pi}, X_{\hat{\tau}_0^{1,\pi}})]$.

Algorithm A2 [price process computation]:
1. Initialization: Set $\hat{P}_T^{2,\pi} := g(T, X_T)$
2. Backward induction: For $i = \kappa - 1$ to 0, set $\hat{P}_{t_i}^{2,\pi} := \max\{g(t_i, X_{t_i}) \, , \, \hat{\mathbb{E}}[\hat{P}_{t_{i+1}}^{2,\pi} \mid \mathscr{F}_{t_i}]\}$.
3. Price estimator at 0: $\hat{P}_0^{2,\pi}$.

Note that the optimal exercise strategy can also be approximated in the Algorithm A2 as follows:

Algorithm A2b [with optimal exercise time estimation]:
1. Initialization: Set $\hat{\tau}_\kappa^{2,\pi} = T$
2. Backward induction: For $i = \kappa - 1$ to 0, $\hat{\tau}_i^{2,\pi} := t_i \mathbf{1}_{A_i^2} + \hat{\tau}_{i+1}^{2,\pi} \mathbf{1}_{(A_i^2)^c}$ where $A_i^2 := \{g(t_i, X_{t_i}) = \hat{P}_{t_i}^{2,\pi}\}$.
3. Price estimator at 0: $\hat{P}_0^{2b,\pi} := \hat{\mathbb{E}}[g(\hat{\tau}_0^{2,\pi}, X_{\hat{\tau}_0^{2,\pi}})]$.

The algorithm A1 corresponds to the approach of [31] in which the conditional expectation operators are estimated by non-parametric regression techniques based on a suitable choice of regression polynomials.

The algorithm A2 corresponds to the approach of [30] and [10] in which the conditional expectation operators are estimated by pure Monte-Carlo methods based on the representation of conditional expectations in terms of a suitable ratio of unconditional expectations obtained by using some Malliavin calculus techniques, see below.

Assume for a moment that $\xi \in L^1 \mapsto \hat{\mathbb{E}}[\xi \mid \mathscr{F}_{t_i}] \in \mathscr{F}_{t_i}$ and that this approximation is conditionally unbiased, i.e. $\mathbb{E}[\hat{\mathbb{E}}[\cdot \mid \mathscr{F}_{t_i}] \mid \mathscr{F}_{t_i}] = \mathbb{E}[\cdot \mid \mathscr{F}_{t_i}]$, then a backward induction argument combined with Jensen's inequality implies that $\mathbb{E}\left[\hat{P}_0^{2,\pi}\right] \geq P_0^\pi$. On the other hand, the fact that the estimated optimal exercise policy $\hat{\tau}_0^{i,\pi}$ is suboptimal by definition, for $i = 1, 2$, implies that $\mathbb{E}\left[\hat{P}_0^{1,\pi}\right] \leq P_0^\pi$ and $\mathbb{E}\left[\hat{P}_0^{2b,\pi}\right] \leq P_0^\pi$. It follows that:

$$\mathbb{E}\left[\hat{P}_0^{1,\pi}\right], \mathbb{E}\left[\hat{P}_0^{2b,\pi}\right] \leq P_0^\pi \leq \mathbb{E}\left[\hat{P}_0^{2,\pi}\right] . \tag{14}$$

The above formal relation can then be used for the construction of *confidence intervals* for the true price of the Bermudan option: $[\hat{P}_0^{1,\pi}, \hat{P}_0^{2,\pi}]$ or $[\hat{P}_0^{2b,\pi}, \hat{P}_0^{2,\pi}]$. If the computation of the conditional expectations is accurate, then the effect of the convexity bias should be small and therefore $\hat{P}_0^{2,\pi}$ should be close to P_0^π. Similarly,

the error in estimating the exact counterpart of the exercise regions A_i^1 and A_i^2 should be small and therefore the estimation of the optimal stopping times should be good, leading to $\hat{P}_0^{1,\pi}$ and $\hat{P}_0^{2b,\pi}$ close to P_0^π. Thus, a tiny *confidence interval* should reveal a good approximation of the exact price, while a large *confidence interval* should be a sign that the estimation was poor. In practice, it seems better to use the interval $[\hat{P}_0^{2b,\pi}, \hat{P}_0^{2,\pi}]$ as both quantities can be computed at the same time with almost no additional cost.

In practice, the approximation operators $\hat{\mathbb{E}}[\cdot \mid \mathscr{F}_{t_i}]$ will be based on future values of simulated paths of X, see Section 4, so that the above reasoning can not be applied rigorously, and the terminology *confidence interval* should be taken with care. Still, numerical tests, see Section 5.2 below, show that such intervals provide a good idea of the quality of the approximation.

3.2 Hedging strategy approximation

As above, we restrict to the case of a Bermudan option. Recalling that the number of units ψ^π of stocks to hold in the hedging portfolio is given by $\phi^{\pi\prime}\sigma^{-1}(\cdot, X)$, whenever this quantity is well-defined, one can estimate the hedging policy by using one of the representation of ϕ^π presented in Section 2.3.

The finite difference approach mentioned in Section 2.3.1 can be combined with Algorithms A1 and A2 in an obvious manner.

As for the tangent process approach and the Malliavin calculus based one, we can also use Algorithms A1 and A2. Algorithms A1 and A2b provide an estimation of the optimal exercise strategy. Plugged into (9) or (11) this leads to two possible approximations of the hedging strategy at time 0:

$$\phi_0^{\pi\prime} \sim \hat{\mathbb{E}}\left[\nabla g(\hat{\tau}_0^\pi, X_{\hat{\tau}_0^\pi})' \nabla X_{\hat{\tau}_0^\pi}\right] \sigma(0, X_0) \tag{15}$$

or

$$\phi_0^{\pi\prime} \sim \frac{1}{t_1}\hat{\mathbb{E}}\left[g(\hat{\tau}_0^\pi, X_{\hat{\tau}_0^\pi}) \int_0^{t_1} \sigma(s, X_s)^{-1} \nabla X_s dW_s\right]' \sigma(0, X_0)\,, \tag{16}$$

with $\hat{\tau}_0^\pi = \hat{\tau}_0^{1,\pi}$ or $\hat{\tau}_0^{2,\pi}$.

Algorithm A2 provides an estimation of the price process at time t_1. Plugged into (10) this leads to

$$\phi_0^{\pi\prime} \sim \frac{1}{t_1}\hat{\mathbb{E}}\left[\hat{P}_{t_1}^{2,\pi} \int_0^{t_1} \sigma(s, X_s)^{-1} \nabla X_s dW_s\right]' \sigma(0, X_0)\,. \tag{17}$$

4 Improved algorithms for the estimation of conditional expectations

As above, we focus on the pricing of Bermudan options. We shall also assume here that the process X can be perfectly simulated on the time grid π. If this is not the case, then it has to be replaced by its Euler scheme. The convergence results of Section 2.2 justify these approximations.

4.1 The regression based approach

We first address the basis function regression method and show how to numerically improve the methodology proposed by [31]. We compute the complexity of the method depending on the number of particles and the number of basis functions.

4.1.1 Generalities

The common fundamental idea in the regression based and the Malliavin based approach consists in using simulated paths of the stock prices $(X^{(j)})_{j\leq N}$ and to apply one of the backward induction Algorithms A1, A2 (possibly A2b) described in Section 3.1 by using the simulations in order to estimate the involved conditional expectations.

In the context of Algorithm A1, the numerical procedure reads as follows:

Algorithm A1 with regression [optimal exercise time estimation]:
1. Initialization : Set $\hat{\tau}_\kappa^{1,\pi,(j)} := T,\ j \leq N$
2. Backward induction : For $i = \kappa - 1$ to 1, set $\hat{\tau}_i^{1,\pi,(j)} := t_i \mathbf{1}_{A_i^{1,(j)}} + \hat{\tau}_{i+1}^{1,\pi,(j)} \mathbf{1}_{(A_i^{1,(j)})^c}$
where $A_i^{1,(j)} := \{g(t_i, X_{t_i}^{(j)}) \geq \hat{\mathbb{E}}^N[g(\hat{\tau}_{i+1}^{1,\pi}, X_{\hat{\tau}_{i+1}^{1,\pi}}) \mid X_{t_i} = X_{t_i}^{(j)}]\},\ j \leq N,$
3. Price estimator at 0: $\hat{P}_0^{1,\pi,} := \frac{1}{N}\sum_{j=1}^N g(\hat{\tau}_0^{1,\pi,(j)}, X_{\hat{\tau}_0^{1,\pi,(j)}}^{(j)}),$

where the estimation

$$\hat{F}^N(t_i, X_{t_i}^{(j)}) := \hat{\mathbb{E}}^N[g(\hat{\tau}_{i+1}^{1,\pi}, X_{\hat{\tau}_{i+1}^{1,\pi}}) \mid X_{t_i} = X_{t_i}^{(j)}]$$

of the true conditional expectation

$$F(t_i, X_{t_i}^{(j)}) := \mathbb{E}[g(\hat{\tau}_{i+1}^{1,\pi}, X_{\hat{\tau}_{i+1}^{1,\pi}}) \mid X_{t_i} = X_{t_i}^{(j)}]$$

is computed by regressing $(g(\hat{\tau}_{i+1}^{1,\pi,(\ell)}, X^{(\ell)}_{\hat{\tau}_{i+1}^{1,\pi,(\ell)}}))_{\ell\leq N}$ on $(\psi_1(X_{t_i}^{(\ell)}),\dots,\psi_M(X_{t_i}^{(\ell)}))_{\ell\leq N}$, where $\psi_1,\dots,\psi_M$ are given functions, i.e.

$$\hat{F}^N(t_i,x) := \sum_{k=1}^{M} \hat{\alpha}_k^{t_i,N}\psi_k(x)$$

where $(\hat{\alpha}_k^{t_i,N})_{k\leq M}$ minimizes

$$\sum_{\ell=1}^{N}\left|g(\hat{\tau}_{i+1}^{1,\pi,(\ell)}, X^{(\ell)}_{\hat{\tau}_{i+1}^{1,\pi,(\ell)}}) - \sum_{k=1}^{M}\alpha_k\psi_k(X_{t_i}^{(\ell)})\right|^2$$

over $(\alpha_k)_{k\leq M}\in\mathbb{R}^M$.

Clearly, the same ideas can be combined with Algorithm A2 (and its variation A2b).

We shall not discuss here the theoretical convergence of the method, we refer to [16] for rigorous statements, but rather describe how it can be improved from a numerical point of view. See also [18] and [40] for convergence rates and [22] for a discussion on the fact that the number of simulated paths has to be increased rapidly with the number of polynomials. See also [28] for further numerical tests.

4.1.2 General comments on the regression procedure

Note that at each step of the above algorithm, we have to solve a quadratic optimization problem of the form

$$\min_{\alpha\in\mathbb{R}^M} \|A\alpha - B\|^2 \tag{18}$$

Different solutions are available. First, we can deduce the induced normal equations

$$A'A\alpha = A'B\,, \tag{19}$$

and use a Choleski decomposition LL' of $A'A$ to solve it. This method is the most efficient in term of computational time but is rather sensible to roundoff error. Besides, it is not memory consuming because the A matrix does not need to be constructed.

A more stable approach consists in using a QR decomposition of A, i.e. write A as QR where Q is a N-dimensional orthonormal matrix and R is a N-dimensional upper triangular matrix, and solve $R\alpha = Q'B$. This is much more time consuming. Moreover, the $N\times M$ matrix A has to be stored. When using standard basis functions for the ψ_k's, it is typically full, which may become highly demanding in terms of memory when the number of basis functions is high.

In order to completely avoid roundoff errors, the Singular Value Decomposition can be used: write $A = UWV'$ with U in $\mathbb{R}^N\times\mathbb{R}^N$ an orthogonal square matrix , V

an orthogonal square matrix in $\mathbb{R}^M \times \mathbb{R}^M$, and W a diagonal matrix in $\mathbb{R}^N \times \mathbb{R}^M$ with only positive or zero entries w_i on the diagonal. Replacing $1/w_i$ by zero if $w_i = 0$, the solution to (18) is given by

$$\alpha = V[\mathrm{diag}\,[1/w_i]]U'B \tag{20}$$

Still this method suffers the same problem as the QR algorithm in term of memory needed to create the matrix A and is the most time consuming.

4.1.3 Drawbacks of polynomial regressions

The idea of taking $\psi_1, \ldots, \psi_M$ has polynomials, or more generally as functions with global support, was first introduced by [39], in the context of general optimal stopping problems, and then used to price American options by [31]. In [31], the authors use monomial basis functions and Laguerre polynomials. Most of the following papers have consisted in extending this approach to Hermite, Hyperbolic and Chebyshev polynomials (see [15] for example).

Some basis may be chosen orthogonal in order to invert easily the associated normal equation (19). Note however that it should then be orthogonal in the law induced by X and not with respect to the Lebesgue measure as usually chosen.

Although very easy to implement in practice, this kind of function basis has a major flaw. For a given number of particles it is not easy to find an optimal degree of the functional basis. Besides, an increase in the number of function basis often leads to a deterioration in the accuracy of the result. This is due to rare events that the polynomials try to fit, leading to some oscillating representation of the function.

Figure 1 below shows how the regression behaves at the first backward step for a put option in dimension one, for different choices of monomial basis functions.

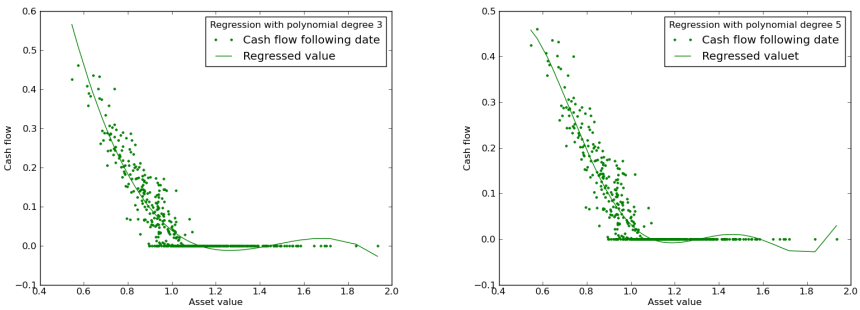


Fig. 1 Regression with global function

Clearly, the regression procedure could be improved by adding the payoff function at the first steps of the algorithm, when the price function should still be close

to the payoff. However, the number of steps for which the payoff should be included in the basis is somehow arbitrary and difficult to determine in practice.

Note that, in the case where an explicit formula is available for the corresponding European option, one can replace the estimator $\hat{\mathbb{E}}[\hat{P}^{2,\pi}_{t_{i+1}} \mid \mathscr{F}_{t_i}]$ in Algorithms A2 and A2b by $\hat{\mathbb{E}}[\hat{P}^{2,\pi}_{t_{i+1}} - P^{euro}(t_{i+1}, X_{t_{i+1}}) \mid \mathscr{F}_{t_i}] + P^{euro}(t_i, X_{t_i})$ where $P^{euro}(t,x)$ denotes the price of the corresponding European option at time t if $X_t = x$. The rationality behind this comes from the fact that the European price process $P^{euro}(\cdot, X)$ is a martingale, and that it typically explains a large part of the Bermudan price. Alternatively, $P^{euro}(t_i, \cdot)$ could also be included in the regression basis.

4.1.4 The adaptive local basis approach

We propose to use a different technique. It essentially consists in applying a non-conform finite element approach rather than a spectral like method as presented above.

The idea is to use, at each time step t_i, a set of functions $\psi_q, q \in \{0, 1, \ldots, M\}$ having local hypercube support $D_{i_1,i_2,..,i_d}$ where $i_j = 1$ to I_j, $M = \prod_{1\leq k\leq N} I_k$, and $\{D_{i_1,..i_d}\}_{(i_1,..,i_d)\in\{1,...,I_1\}\times\cdots\times\{1,...,I_d\}}$ is a partition of the hypercube $[\min_{1\leq k\leq N} X^{1,(k)}_{t_i}, \max_{1\leq k\leq N} X^{1,(k)}_{t_i}] \times \cdots \times [\min_{1\leq k\leq N} X^{d,(k)}_{t_i}, \max_{1\leq k\leq N} X^{d,(k)}_{t_i}]$. On each D_l, for $l = (i_1, .., i_d)$, ψ_l is a linear function with $1+d$ degrees of freedom. This approximation is "non-conform" in the sense that we do not assure the continuity of the approximation. However, it has the advantage to be able to fit any, even discontinuous, function.

The number of degrees of freedom is equal to $M * (1+d)$. In order to avoid oscillations, the support are chosen so that they contain roughly the same number of particles.

On Figure 2, we plot the solution of the previous regression problem (Fig. 1) obtained for a number of basis functions chosen so as to have the same number of degrees of freedom as when using polynomials.

Clearly, the method behaves much better than when the basis is made of monomials.

Moreover, the normal equation is sparse when using such local functions, and far better conditioned, leading to the possibility to use the Choleski method, which, as claimed above, is the most efficient for solving the regression problem. At the opposite, basis with global support, like polynomials, typically require the use of QR or SVD factorization, because the global resolution typically fails when a lot of particles are used.

Note that, in order to ensure that each hypercube approximately contains the same number of particles, it is necessary to "localize" them in order to appropriately define the support of the local functions. In dimension one, this can be achieved by

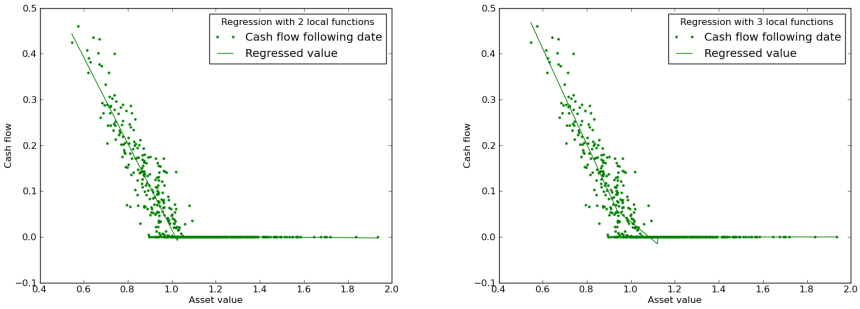


Fig. 2 Regression with local function

using a partial sort procedure whose complexity is of order of $O(I_1 N)$. In dimension d, two methods are available :

(i) Realize partial sorts in each direction and derive the support of the basis functions so that each contains approximatively the same number of particles. This option is particularly efficient when the particles have rather independent coordinates. The operation is realized in $O(\sum_{k=1}^{d}(I_k+1)N)$. For example in dimension 2, it leads to a first partial sort to get the particle with the lowest x-coordinates x_0, and successive partial sorts are achieved giving points x_i such that $[x_{i-1}, x_i]$, $i = 1$ to I_1, contains N/I_1 particles. Doing independently the same in the second dimension, we get I_2 intervals of the form $[y_{i-1}, y_i]$, $i = 1, \ldots, I_2$. The $I_1 I_2$ hypercubes are defined by the sets $(x_{i-1}, x_i] \times (y_{j-1}, y_j]$, $i = 1, \ldots, I_1,\ j = 1, \ldots, I_2$.

(ii) Use a Kd tree, see e.g. [38] and [8], with depth d so that each node of depth i has I_{i+1} sons. Use a partial sort at each node of depth i to sort the particles following the coordinate $i+1$, and use this sort to define the I_{i+1} sons. For example in dimension 2, (I_1+1) partial sorts are achieved in the first dimension leading to I_1 sets of N/I_1 particles. Then, for each set of particles, partial sorts are achieved in the second dimension giving subsets of $N/(I_1 I_2)$ particles. The hypercubes are given by the minimum and maximum coordinates of the points in the subsets. The difference with (i) is that the partial sorts are no more performed independently in each direction. The complexity of the procedure remains in $O(\sum_{k=1}^{d}(I_k+1)N)$.

Although the complexity of the two techniques is the same, the first one uses only $\sum_{k=1}^{d}(I_k+1)$ partial sorts, while the second one uses $1+\sum_{l=1}^{d}\prod_{j<l} I_j(I_l+1)$ calls to the partial sort procedure and is far more expensive.

On Figure 3, we have plotted an example of supports in the case of $16 = 4 \times 4$ local basis functions, in dimension 2.

Importantly, this approach allows to increase the number of basis functions without any instability in the resolution of the regression problem. The construction of the normal matrix $A'A$ has a complexity of order $O(N)$ when storing only non zero elements, the resolution time is negligible (linear with M because of the sparsity

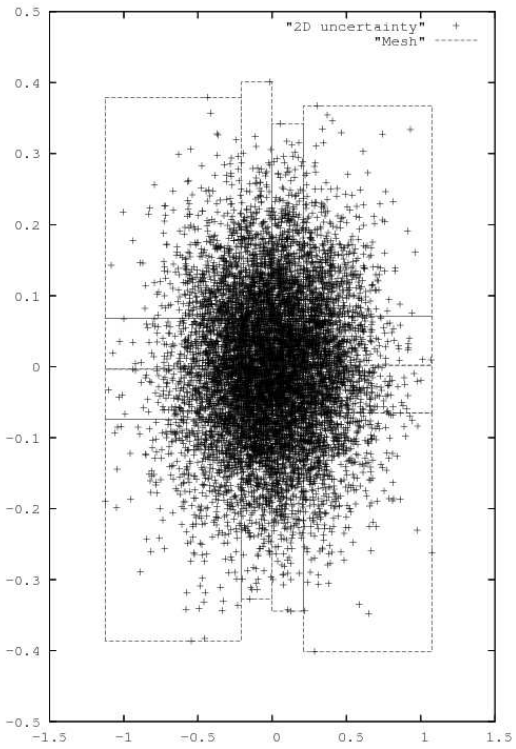


Fig. 3 Support of 2D function basis

of the matrix), and the reconstruction procedure is in $O(N)$. The complexity of the global resolution is therefore of order of $O(N(1+\sum_{k=1}^{d}(I_k+1)))$.

Note that the use of local basis functions was already discussed in [25] but for (non-reflected) BSDEs. The difference is that the support of the hyper-cubes is fixed once for all. In this paper, one can see that the approximation error is very robust, in the sense that it essentially only depends on the first moments of the underlying process X, see their Section 6.2.

4.2 The Malliavin based approach

4.2.1 The alternative representation for conditional expectations

The idea of using Malliavin calculation to provide efficient estimators of conditional expectations first appeared in [20], and was then further used in [30] and [12] to price American options, see also [10].

Given a measurable map f, the main idea consists in writing

$$r(t_i,x) := \mathbb{E}\left[f(X_{t_{i+1}}) \mid X_{t_i} = x\right]$$

as

$$r(t_i,x) = \frac{\mathbb{E}\left[\delta_x(X_{t_i}) f(X_{t_{i+1}})\right]}{\mathbb{E}\left[\delta_x(X_{t_i})\right]} \tag{21}$$

where δ_x denotes the Dirac mass at x. Then, under suitable regularity and uniform ellipticity conditions on σ, a formal integration by parts in the Malliavin calculus sense allows to rewrite $r(t_i,x)$ as

$$r(t_i,x) = \frac{\mathbb{E}\left[f(X_{t_{i+1}}) H_x(X_{t_i}) \mathscr{S}_{t_{i+1}}\right]}{\mathbb{E}\left[H_x(X_{t_i}) \mathscr{S}_{t_{i+1}}\right]} \tag{22}$$

where H_x denotes the Heaviside function, $H_x(y) = 1$ if $y^i > x^i$ for all $i \leq d$ and $H_x(y) = 0$ otherwise, and $\mathscr{S}_{t_{i+1}}$ is a Skorohod integral which depends only on the path of X on $[0,t_{i+1}]$, see [10] for details and (25) below for an example of representation in a simple Gaussian framework.

In the context of Algorithm A2 of Section 3.1, the numerical procedure reads as follows:

Algorithm A2 with Malliavin [price process computation]:
1. Initialization: Set $\hat{P}_T^{2,\pi,(j)} := g(T, X_T^{(j)})$, $j \leq N$.
2. Backward induction: For $i = \kappa - 1$ to 0, set

$$\hat{P}_{t_i}^{2,\pi,(j)} := \max\{g(t_i, X_{t_i}^{(j)}) \, , \, \hat{\mathbb{E}}^N[\hat{P}_{t_{i+1}}^{2,\pi} \mid X_{t_i} = X_{t_i}^{(j)}]\}$$

,

3. Price estimator at 0: $\hat{P}_0^{2,\pi,(1)}$,

where the estimation

$$\hat{F}^N(t_i, X_{t_i}^{(j)}) := \hat{\mathbb{E}}^N[\hat{P}_{t_{i+1}}^{2,\pi} \mid X_{t_i} = X_{t_i}^{(j)}]$$

of the true conditional expectation

$$F(t_i, X_{t_i}^{(j)}) := \mathbb{E}[\hat{P}_{t_{i+1}}^{2,\pi} \mid X_{t_i} = X_{t_i}^{(j)}]$$

is computed by considering the Monte-Carlo counterparts of the numerator and denominator in (22), i.e.

$$\hat{F}^N(t_i,x) := \frac{\sum_{j=1}^N H_x(X_{t_i}^{(j)}) \hat{P}_{t_{i+1}}^{2,\pi,(j)} \mathscr{S}_{t_{i+1}}^{(j)}}{\sum_{j=1}^N H_x(X_{t_i}^{(j)}) \mathscr{S}_{t_{i+1}}^{(j)}} \tag{23}$$

where $\mathscr{S}^{(j)}_{t_{i+1}}$ is the Skorohod integral associated to the path $X^{(j)}$, and where we use the convention $0/0 = 0$.

One can similarly combine this approach with Algorithm A1, by using the approximation

$$\mathbb{E}[g(\hat{\tau}_{t_{i+1}}, X_{\hat{\tau}_{t_{i+1}}}) \mid X_{t_i} = X^{(j)}_{t_i}] \sim \frac{\sum_{j=1}^N H_x(X^{(j)}_{t_i}) g(\hat{\tau}^{(j)}_{t_{i+1}}, X^{(j)}_{\hat{\tau}^{(j)}_{t_{i+1}}}) \mathscr{S}^{(j)}_{t_{i+1}}}{\sum_{j=1}^N H_x(X^{(j)}_{t_i}) \mathscr{S}^{(j)}_{t_{i+1}}} \tag{24}$$

Here again, we shall not discuss the theoretical convergence of the method, we refer to [12] for rigorous statements, but rather describe how this method should be implemented in practice so as to be efficient. However, it should be noted that the convergence obtained in [12] is based on the fact that they use independent simulations for each time step, although this does not seem necessary in practice. The convergence analysis in the above setting is certainly much more involved, and left open.

4.2.2 General comments

First, it can be shown that the variance of the Skohorod integral $\mathscr{S}_{t_{i+1}}$ is of order of $(\min\{t_i, t_{i+1} - t_i\})^{-d}$, see (25) below as an example. As will be explained below in the Gaussian case, it can be partially compensated by incorporating a localization function. Still, this method will in general perform rather badly when the length of the time interval between two dates of possible exercise in small. In particular, if we are interested by the approximation of American option prices by their Bermudan counterparts, then a large number of simulations will be required in order to compensate for the explosion of the variance of the estimator as the time step goes to 0.

Second, we should note that the term $\mathbb{E}\left[H_x(X_{t_i})\mathscr{S}_{t_{i+1}}\right]$ in (22) is just the density of X_{t_i} at x. If it is known, it can be used directly instead of being estimated. In practice, it turns out that that the fact that both the numerator and the denominator in (23) are influenced by the $\mathscr{S}^{(j)}_{t_{i+1}}$'s leads to numerical compensations which seem to stabilize the algorithm.

In most applications, natural upper and lower bounds are known for the true price process P^π is terms of deterministic functions of X. They can be used to truncate the numerical results obtained at each step of the algorithm, in order to stabilize it.

The computation of the Skorohod integrals involved in the above estimators is rather tricky in general, even when X is replaced by its Euler scheme. This is due to the (possible) dependence of the different components of X with respect to the different components of the Brownian motion. However, if up to a suitable transformation and possibly a deterministic time change, we can reduce to the case where $X = W$, then things simplify significantly, as will be shown in the next section.

Finally, this approach requires a non-degeneracy assumption on σ, while the regression based method may be applied to large class of Markov processes, see [29].

4.2.3 Simplifications in the Gaussian case

In the case where we can reduce to $W = X$, for instance in the Black-Scholes model, the Skorohod integrals entering in the representation (22) can be taken in the form

$$\mathscr{S}_{t_{i+1}} = \prod_{k\le d}\left(\frac{W^k_{t_i}}{t_i} - \frac{W^k_{t_{i+1}} - W^k_{t_i}}{t_{i+1}-t_i}\right) \tag{25}$$

In order to reduce the variance of the estimator, we can also incorporate a localization function as explained in [10]. Given a smooth bounded function φ on $\mathbb{R}_+$ such that $\varphi(0) = 1$, $r(t_i,x)$ can also be written with $\mathscr{S}_{t_{i+1}}(x)$ in place of $\mathscr{S}_{t_{i+1}}$ in (22), where

$$\mathscr{S}_{t_{i+1}}(x) := \prod_{k\le d}\left[\varphi(W^k_{t_i} - x^k)\left(\frac{W^k_{t_i}}{t_i} - \frac{W^k_{t_{i+1}} - W^k_{t_i}}{t_{i+1}-t_i}\right) - \varphi'(W^k_{t_i} - x^k)\right]$$

In such a class of functions, it was shown in [10] that the one that minimizes the integrated variance of the numerator and the denominator is of exponential type, $\varphi(y) = \exp(-\eta y)$ with $\eta > 0$. The parameter η should be theoretically of order of $1/\sqrt{\min\{t_i, t_{i+1}-t_i\}}$.

Note that the numerator in (22) could be simplified by using $\tilde{\mathscr{S}}_{t_{i+1}}(x)$ instead of $\mathscr{S}_{t_{i+1}}(x)$, where

$$\tilde{\mathscr{S}}_{t_{i+1}}(x) := \prod_{k\le d}\left[\varphi(W^k_{t_i} - x^k)\frac{W^k_{t_i}}{t_i} - \varphi'(W^k_{t_i} - x^k)\right]$$

because the increments of the Brownian motion are independent. However, we have noticed from our tests that it is better to use the same integration weights at the numerator and denominator so as to play with possible numerical compensations.

4.2.4 Improved numerical methods

A crude application of the algorithm described in Section 4.2.1 leads, at each time step, to the computation of N sums composed of N terms each. This leads to a complexity of order N^2, which makes this procedure completely inefficient in practice.

However, it should be noted that the above calculation problem for the numerator and the denominator of our estimator can be reduced to the following one: Given N point $y_1,\dots,y_N$ in $\mathbb{R}^d$, and N real numbers $f_1,\dots,f_N$,

$$\text{Compute} \quad q_i := \sum_{j \leq N} H_{y_i}(y_j) f_j \ , \quad \text{for each } i \leq N. \tag{26}$$

It is clear that both terms in (23) are of this form with $y_j = X_{t_i}^{(j)}$ and $f_j = \hat{P}_{t_{i+1}}^{2,\pi,(j)} \mathscr{S}_{t_{i+1}}^{(j)}$ or $f_j = \mathscr{S}_{t_{i+1}}^{(j)}$. Even, if $\mathscr{S}_{t_{i+1}}$ is replaced by $\mathscr{S}_{t_{i+1}}(\cdot)$ this remains the case, whenever $\mathscr{S}_{t_{i+1}}(\cdot)$ is defined with respect to an exponential localizing function:

$$\begin{aligned}
\mathscr{S}_{t_{i+1}}(x) &= \prod_{k \leq d} \left[e^{-\eta(W_{t_i}^k - x^k)} \left(\frac{W_{t_i}^k}{t_i} - \frac{W_{t_{i+1}}^k - W_{t_i}^k}{t_{i+1} - t_i} \right) + \eta e^{-\eta(W_{t_i}^k - x^k)} \right] \\
&= \left(\prod_{k \leq d} e^{\eta x^k} \right) \prod_{k \leq d} e^{-\eta W_{t_i}^k} \left[\left(\frac{W_{t_i}^k}{t_i} - \frac{W_{t_{i+1}}^k - W_{t_i}^k}{t_{i+1} - t_i} \right) + \eta \right],
\end{aligned}$$

where the only terms which depends on x, $\left(\prod_{k \leq d} e^{\eta x^k}\right)$, can be added at the end of the procedure. Note that, when the same weights are used for the numerator and the denominator, these terms actually compensate.

We now discuss how Problem (26) can be solved efficiently.

The one dimensional case. In the one dimensional case, the main idea is to reduce to the situation where $y_1 > y_2 > \ldots > y_N$, assuming that none of them are equal for simplicity. Indeed, in this case, the q_i's can be computed in N steps by induction: $q_1 = 0$, $q_{i+1} = q_i + f_i$ for $i = 1, \ldots, N-1$. In order to reduce to the case where the y_i's are sorted, it suffices to use a quick sort algorithm whose complexity is of order of $N \ln(N)$. Hence, the complexity of Problem (26) is $O(N \ln N)$ and not $O(N^2)$.

The two dimensional case. In dimension two, it is no more possible to sort the data. However, Problem (26) is related to the well-documented "dominance reporting problem", which was solved efficiently in dimension two by [7] with the classical divide and conquer algorithm. The algorithm is based on the construction of two dimensional K-d tree that stores the points, see [8]. Its construction is achieved in $O(N \ln N)$, and a query for reporting dominance over one point can be achieved in $O(\sqrt{N})$, see [38] and [8]. The global dominance reporting problem for a set of N points can thus be solved in $O(N\sqrt{N})$. We modify this algorithm in the sequel such that our problem can be solved $O(N \ln N)$.

To show how the algorithm works, imagine for example that $N = 8$ as on figure 4. After a sort according to the first coordinate, we split the points into two groups with the same cardinality : points 6,2,5 and 4 define the first set, 3,1,8,7 the second set. All points of the first set can be dominated[1] by all points of set two but no point of the second set is dominated by a point of set one.
We then compare the points from the second set with the points of the first set according to the second coordinate, while keeping the partial summation, say *psum*.

[1] Hereafter, we say that a point x_j dominates a point x_k if $x_j^i > x_k^i$ for all $i \leq d$.

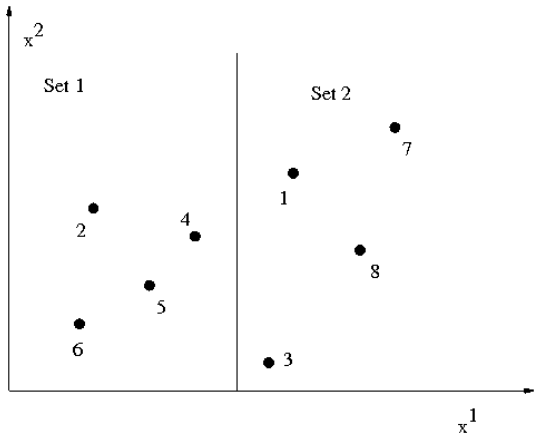


Fig. 4 First step to calculate g

The algorithm is initialized with $psum = 0$. Then, point 7 has the highest second coordinate of set 2 and dominates all points of set 1: add f_7 to *psum*. The second one, point 1 dominates all points of set one: add f_1 to *psum*. The third one, point 8, does not dominate points 2 and 4 of set one : add *psum* to q_2 and q_4, then add f_8 to *psum*. The last point, point 3, does not dominate any point of set one: add *psum* to q_5 and q_6. We have achieved the last point of set 1, we thus stop the algorithm.
Graphically, the algorithm can be understood as follows. Draw a horizontal line crossing the vertical axis at the level of the highest second coordinate of the two sets, then lower down this line. Each time the line crosses a point x_j of set 2, add the corresponding f_j value to *psum*, each time the line crosses a point x_k of set 1, add *psum* to the corresponding q_k.

In a second step, we split the first set into two sets (set 3 and 4) and the second set into two sets (set 5 and 6), see figure 5. We apply the same procedure as before on these new pair of sets. For example for set one, we first set $psum = 0$. Then, the point of set 4 with the highest second coordinate is number 4 and it does not dominate point 2 of the set 3 : add f_4 to *psum*. The second one, point 5, does the same : add f_5 to *psum*. Then add *psum* to q_6 which has the lowest second coordinate.
We iterate the procedure until each subset contains only one point.

Below, we provide the algorithm for the dimension 2. It is composed of two functions :

- a one dimensional merge function given by algorithm 1,
- a recursive Divide and Conquer function given by algorithm 2.

Merge algorithm *Merge1D*: We are given two sets. The first set has cardinality $nbp1$, and the second has cardinality $nbp2$. We are also given sorting tables of indexes $isort1$ and $isort2$ so that $(x_{isort1(j)})_{j \leq nbp1}$ (resp. $(x_{isort2(j)})_{j \leq nbp2}$) corresponds

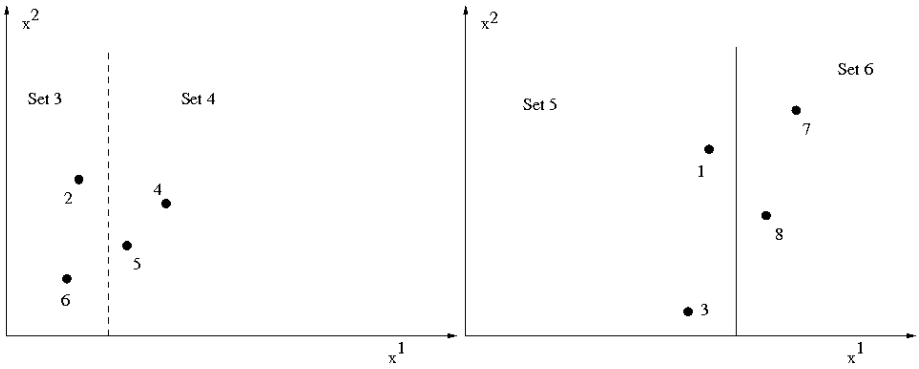


Fig. 5 Second step to calculate g

to the sequence of points of set 1 (resp. set 2) sorted increasingly with respect to the second coordinate. The array Y in the algorithm below corresponds to the second coordinate of the points $(x_j)_{j\leq N}$, i.e. $Y(j) := x_j^2$. The other input is the array f of the values $(f_j)_{j\leq N}$, $f(j) := f_j$. The output are the updated values of q, $q(j) = q_j$, for the values of the index j corresponding to set 1.

Algorithm 1 Merge algorithm $Merge1D(Y, f, isort1, isort2, nbp1, nbp2)$

$sp = 0$, $ip = nbp2$
for $i = nbp1$ to 1 **do**
  $ipoint2 = isort2(ip)$
  $ipoint1 = isort1(i)$
  while $Y(ipoint2) >= Y(ipoint1)$ **do**
    $sp = sp + f(ipoint2)$
    $ip = ip - 1$
    if $ip = 0$ **then**
      Break
    end if
    $ipoint2 = isort2(ip)$
  end while
  $q(ipoint1) = sp$
  if $ip = 0$ **then**
    for $j = 1$ to $i - 1$ **do**
      ipoint1 = $isort1(j)$
      $q(ipoint1) = sp$
    end for
    Break
  end if
end for
return q

Divide and conquer algorithm *Divide2D*: We are given one set of points $(x_j)_{j\leq nbp}$. X and Y are the arrays corresponding to the first and second coordinates, $X(j) :=$

x_j^1 and $Y(j) := x_j^2$. The arrays *isortX* and *isortY* are tables of indexes so that $(X(isortX(j))_{j\leq nbp}$ and $(Y(isortY(j))_{j\leq nbp}$ are sorted increasingly. The input of this function is the range of indexes corresponding to the set of points to be sorted. The result is a table of indexes. The output of the global algorithm is the array q, $q(j) = q_j$.

Algorithm 2 Divide and conquer algorithm $Divide2D(X,Y,f,isortX,isortY,nbp)$

$imed = nbp/2$
$imedp = nbp - imed$
$xmed = (X(isortX(imed)) + X(isortX(imed+1)))/2$ // compute the median point which delimitates set 1 (first coordinate lower than xmed) and set 2 (first coordinate bigger than xmed)
$isortX_1 = isortX(1 : imed)$ // sort data according to the first coordinate in set 1
$isortX_2 = isortX(imed+1 : nbp)$ // sort data according to the first coordinate in set 2
$iy1 = 0$
$iy2 = 0$
for $i = 1$ to nbp **do**
  $ipoint = isortY(i)$
  if $X(ipoint) <= xmed)$ and $(iy1 < imed)$ **then**
    $isortY_1(iy1) = ipoint$
    $iy1 = iy1 + 1$
  else
    $isortY_2(iy2) = ipoint$
    $iy2 = iy2 + 1$
  end if
end for// sort data according to the second coordinate in each subset
$Divide2D(X,Y,f,isortX_1,isortY_1,imed)$
$Divide2D(X,Y,f,isortX_2,isortY_2,imedp)$ // recursive call to *Divide2D* on each subset
$qloc = Merge1D(Y,f,isortY_1,isortY_2,imed,imedp)$ // call of *Merge1D* on the two subsets
for $i = 1$ to $imedp$ **do**
  $ipoint1 = isortX_1(i)$
  $q(ipoint1) = q(ipoint1) + qloc(ipoint1)$
end for// update of q

The divide and conquer leads implicitly to the construction of a binary tree of depth $O(\ln(N)/\ln(2))$. At each father node at depth p of this tree corresponds a subtree which contains $N/2^p$ points. The cost of the Divide and Conquer function is linear, and we merge the points corresponding to the son nodes with a linear cost (as seen in the Merge algorithm). At depth p, we have 2^p father nodes, so the cost of merging all subtrees and spent in the Divide and Conquer function at depth p is $O(N) = 2^p O(N/2^p)$. Since the length of the tree is $O(\ln(N)/\ln(2))$, the global cost of the algorithm $O(N\ln(N))$. This is the cost of the calculation of the conditional expectation in dimension 2.

Higher dimensions. In [33] some specific algorithm based on binary trees has been developed for the 3D problem. The query time is said to be equal to $O(\ln N + k)$ where k is the number of point to report. Recently in [27], an algorithm generalizing

the previous approach and using a fusion tree of a certain degree, instead of a binary tree, was proved to solve one query search in $O(\ln N/\ln\ln N+k)$. All the geometric algorithm suffers the same flaw: for our problem the number of points dominating another is on average $\frac{N}{2^d}$ so the global answer remains in $O(N^2)$.
The key point in the calculation of q is to try to keep information about the partial summation in order to report geometrically which point dominates another.This implies that it is possible to reduce drastically the number of operations by using a similar structure as k-D trees.

It turns out that the generalization of the previous algorithm is indeed rather straightforward. We use the same divide and conquer algorithm in the first dimension. This reduces the problem to merging the points in dimension $d-1$. Using once again a binary tree in a new merge function, we are then able to compare the two sets of points generated by the Divide and Conquer algorithm. To do this, we use recursively the merge algorithm with a divide and conquer approach in order to decrease the dimension of the final merge to dimension one. The idea of dominance merge is described page 367 of [7]. For example in dimension three, the main divide and conquer, see Algorithm 3, is identical to the two dimensional algorithm. The only difference is that it asks for a merge in dimension 2.

Divide and conquer algorithm *Divide*3*D*: We are given one set of points $(x_j)_{j\leq nbp}$. X, Y and Z are the arrays corresponding to the first, the second and the third coordinates, $X(j):=x_j^1$, $Y(j):=x_j^2$ and $Z(j):=x_j^3$. The arrays *isortX*, *isortY* and *isortZ* are tables of indexes so that $(X(isortX(j))_{j\leq nbp}$, $(Y(isortY(j))_{j\leq nbp}$ and $(Z(isortZ(j))_{j\leq nbp}$ are sorted increasingly. The input of this function is the range of indexes corresponding to the set of points to be sorted. The result is a table of indexes. The output of the global algorithm is the array q, $q(j)=q_j$.

The merge in dimension 2 is given by Algorithm 4. It is a recursive algorithm that calls the Merge1D, Algorithm 1. Two sets A and B in dimension 2 with associated second and third coordinates are used as input. Due to the divide and conquer part, we know that potentially each point in B dominates the points in A, because they have bigger first coordinates. A split is achieved on $A\cup B$ according to the second coordinate leading to four subsets A_1, A_2, and B_1, B_2, see figure 6. Then a call to

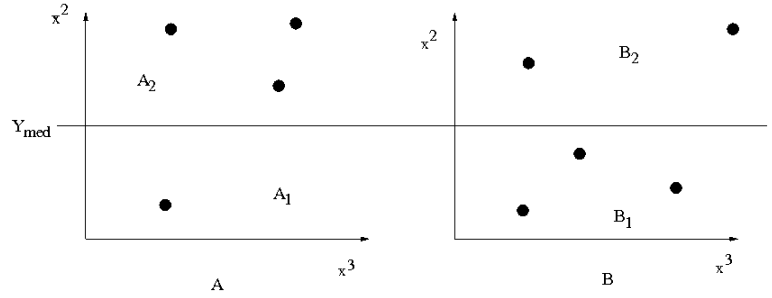


Fig. 6 2D merge for two subsets A and B

Algorithm 3 Divide and conquer algorithm

$$Divide3D(X,Y,Z,f,isortX,isortY,isortZ,nbp)$$

$imed = nbp/2$
$imedp = nbp - imed$
$xmed = (X(isortX(imed)) + X(isortX(imed+1)))/2$
$isortx_1 = isortx(1:imed)$
$isortx_2 = isortx(imed+1:nbp)$
$iy1 = 0$
$iy2 = 0$
for $i = 1$ to nbp **do**
  $ipoint = isortY(i)$
  if $X(ipoint) <= xmed)$ and $(iy1 < imed)$ **then**
    $isortY_1(iy1) = ipoint$
    $iy1 = iy1 + 1$
  else
    $isortY_2(iy2) = ipoint$
    $iy2 = iy2 + 1$
  end if
end for
$iz1 = 0$
$iz2 = 0$
for $i = 1$ to nbp **do**
  $ipoint = isortZ(i)$
  if $X(ipoint) <= xmed)$ and $(iz1 < imed)$ **then**
    $isortZ_1(iz1) = ipoint$
    $iz1 = iz1 + 1$
  else
    $isortZ_2(iz2) = ipoint$
    $iz2 = iz2 + 1$
  end if
end for
$Divide3D(X,Y,Z,f,isortX_1,isortY_1,isortZ_1,imed)$
$Divide3D(X,Y,Z,f,isortX_2,isortY_2,isortZ_2,imedp)$
$qloc = Merge2D(Y,f,isortY_1,isortY_2,isortZ_1,isortZ_2)$
for $i = 1$ to $imedp$ **do**
  $ipoint1 = isortX_1(i)$
  $q(ipoint1) = q(ipoint1) + qloc(ipoint1)$
end for

the Merge2D function is achieved on A_1 and B_1 and on A_2 and B_2. All point of B_2 dominate the points of A_1 according to the second coordinate. It only remains to check in the third direction, which is performed by a Merge1D on B_2 and A_1 according to the third coordinate.

The cost $c(N)$ of the merge with N points can be decomposed in:

- some operation with linear cost in N,
- two merges in dimension 2 with $N/2$ points,
- one merge in dimension one with $N/2$ points realized in $O(N/2)$.

Algorithm 4 Merge algorithm $Merge2D(Y,Z,f,isortY_1,isortY_2,isortZ_1,isortZ_2)$

Create subset A_1, A_2, B_1, B_2 (figure 6)
Create $isortY_{11}$,$isortZ_{11}$ associated to A_1, $isortY_{12}$ $isortZ_{12}$ associated to A_2,
Create $isortY_{21}$,$isortZ_{21}$ associated to B_1, $isortY_{22}$ $isortZ_{22}$ associated to B_2,
$qloc = Merge2D(Y,Z,f,isortY_{11},isortY_{21},isortZ_{11},isortZ_{21})$
$qloc+ = Merge2D(Y,Z,f,isortY_{12},isortY_{22},isortZ_{12},isortZ_{22})$
$qloc+ = Merge1D(Y,Z,f,isortZ_{11},isortZ_{22})$
return $qloc$

Hence, c satisfies $c(N) = 2c(N/2) + O(N)$, which leads to a global cost $c(N) = O(N \ln N)$.

The divide and conquer algorithm with cost $D(N)$ achieves :

- two divide an conquer with $N/2$ points,
- one merge with cost in $O(N \ln N)$,
- some extra work with linear cost $O(N)$.

Hence, we have $D(N) = D(N/2) + O(N \ln N)$ leading to a global cost of $D(N) = O(N(\ln N)^2)$.
For N points, the algorithm has a complexity of order $N(\ln N)^2$.

The same procedure can be used in dimension $d \geq 3$. The call of the merge function in dimension d at a father node with $N/2^p$ particles leads to

- some work with linear cost $O(N/2^p)$,
- two calls of the merge at the son node with $N/2^{p+1}$ particles, in dimension d,
- a call of the merge function with $N/2^{p+1}$ particles in dimension $d-1$.

So the complexity of the merge function c^d can be calculated recursively. A merge for N particles in 3D satisfies $c^3(N) = 2c^2(N/2) + O(N/2\ln(N/2))$ which leads to a complexity $c^3(N) = O(N(\ln N)^2)$. Similarly a merge in dimension d, $d > 2$, will lead to $c^d(N) = O(N(\ln N)^{d-1})$.
So the divide and conquer algorithm with cost $D^d(N)$, $d > 2$, achieves :

- two divide and conquer with $N/2$ points with cost $2D^d(N/2)$,
- one merge in dimension $d-1$ with N points and a cost in $O(N(\ln N)^{d-2})$,
- some extra work with linear cost $O(N)$.

Hence, we have $D(N) = 2D(N/2) + O(N(\ln N)^{d-2})$ leading to a global cost of $D(N) = O(N(\ln N)^{d-1})$.

In table 1, we apply the algorithm and compute the time spent for different dimensions and different numbers of particles. In dimension 1 and 2, we effectively observe that the complexity is the same and that the time spent divided by $N\ln(N)$ is constant. For dimension 4, it appears numerically that the time spent is between $O(N\ln(N)^2)$ and $O(N\ln(N)^3)$. For dimension 9, we observe that the time spent is in $O(N(\ln N)^6)$. Our numerical results thus show a complexity slightly better than the theoretical one.

Table 1 Time spend in seconds to calculate g for a given particles number

Part nb	1D	2D	3D	4D	5D	6D	7D	8D	9D
10.000	0.	0.01	0.07	0.22	0.48	0.78	1.08	1.32	1.52
100.000	0.01	0.13	1.05	3.94	9.85	18.95	29.96	41.04	50.3
1.000.000	0.17	1.92	15.2	62.24	178.45	396	717	1110	1518

5 Numerical experiments

In this part, we produce some numerical tests for the pricing of American options associated to different payoffs.

5.1 Model and payoffs

We now set the interest rate to $r = 5\%$ annually. This means that we have to add a drift term in (1) and take discounting into account in all our algorithms.

All the assets are non correlated and follow a Black and Scholes type dynamics with annual volatility $\sigma = 20\%$, and initial value equal to 1:

$$X_t^i = 1 + \int_0^t r X_s^i ds + \int_0^t \sigma X_s^i dW_s^i \ , \ i \leq d \ .$$

We consider three different Bermudan options with maturity one year and 11 equally distributed possible exercise dates:

Option 1: a geometrical put option with strike $K = 1$ and payoff $(K - \prod_{i=1}^d X_t^i)^+$,
Option 2: a geometrical digital put option with strike $K = 0.9$ and payoff $\mathbf{1}_{K > \prod_{i=1}^d X_t^i}$,
Option 3: a basket put option with strike $K = 1$ and payoff $(K - \frac{1}{d}\sum_{i=1}^d X_t^i)^+$.

Note that the two first payoffs involve the process $\prod_{i=1}^d X^i$ which can be identified to a one-dimensional non standard exponential Brownian motion. This implies that the pricing of both Bermudan options reduces to a one dimensional optimal stopping problem which can be efficiently solved by PDE techniques. In table 2, 3 we give reference prices and delta values computed for geometrical put and digital options.

Table 2 Reference option prices and delta for geometrical put

	1D	2D	3D	4D	5D	6D
Option value	0.06033	0.07815	0.08975	0.09837	0.10512	0.11074
Delta value	-0.4090	-0.3858	-0.3734	-0.3607	-0.3577	-0.3498

Table 3 Reference option prices and delta for digital options

	1D	2D	3D	4D	5D	6D
Option value	0.4223	0.5035	0.5375	0.5556	0.5662	0.5727
Delta value	-3.067	-2.466	-2.116	-1.886	-1.721	-1.593

This will serve as a benchmark. Obviously, we do not use this trick when applying our algorithms. In table 4, we give option values computed for basket options with 10^d meshes and $8d^2$ millions of particules. Notice that for basket options these values should be considered with care, since there is no guarantee that the deterministic schemes have converged. In the figures below, estimated prices and deltas for

Table 4 Most accurate computed option prices for basket options

	1D	2D	3D	4D	5D	6D
Option value	0.06031	0.03882	0.02947	0.02404	0.02046	0.01831

Options 1 and 2 are normalized by their *true* value computed by PDE techniques. Since no easily accessible benchmark are available for Option 3, results will be presented in absolute values.

5.2 Numerical results on prices

In the different tests, we compare:

1. Algorithm A1 and Algorithm A2 for the regression based approach of Sections 4.1.1 and 4.1.4 with a number of meshes equal to 8^d,
2. Algorithm A1 and Algorithm A2 for the Malliavin based approach, recall (23)-(24). We use an exponential parameter $\eta = 1/\sqrt{\Delta t}$ in the localization function, with $\Delta t = 1/10$.
3. We also compare our results with the quantization method, see [5], [3], [35], [4], [36]. The quantization method is a recombining tree method where the nodes are optimally calculated, see [35]. Once a time discretization has been fixed, a number of quantization points at each time step is chosen according to [4]. The quantization points are calculated off line and are available on the website http://www.quantize.maths-fi.com. Once the quantization points have been chosen a Monte Carlo approach is used to calculated the transition probabilities linking nodes in the tree. This technique being time consuming, we use the Principal Axis Tree method, see [34], to accelerate the computations. The number of Monte Carlo simulations used to calculate the probabilities is fixed to 4 millions.

Results does not change with more than 10 millions.

For each option, dimension and number of simulated paths, we apply the different algorithms with the same set of particles.

For all the methods, no special knowledge on the payoff has been used: no control variate (which could be used for each method and is very efficient in practice), no special guess of the regression function.

For Option 1, results are given on figure 7 for $d = 1$ to 6 for the Malliavin and Regression based approaches depending on the ln of the number N of particles used. We do not provide the results obtained with the Malliavin approach for large values of N because it is too time consuming. For instance, in dimension 6, the last option price calculated with 2 millions particles takes more than two hours to be calculated. It is clearly a limitation to this approach. Recall however that no (even natural) control variate technique has been used here.

We observe that Algorithm A2 generally provides results above the exact value of the option while the results obtained with Algorithm A1 are slightly below the analytic value as expected, see Section 3.1. The Malliavin approach gives very good results for dimension 1 to 3. The regression based method seems to exhibit a very small bias which is due to the fact that the number of basis function is limited. From dimension 4, the convergence is becoming slow and the time needed becomes prohibitive, especially for the Malliavin based approach.

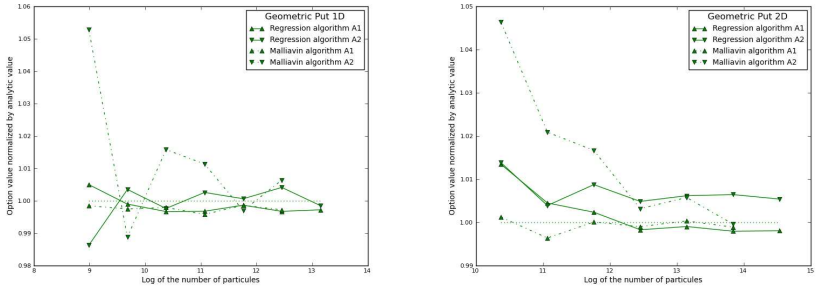
In table 5, we give the time spent for the different calculation with different dimensions.[2]

Table 5 Time spent for calculation of the Malliavin and Regression based approaches for different numbers of particles (particule number in thousands)

Dimension	1D	1D	2D	2D	3D	3D	4D	4D	5D	5D	6D	6D
part nb	8	256	256	1024	256	2000	250	2000	500	2000	1000	2000
ln of part nb	8.98	12.45	12.45	13.84	12.45	14.50	12.42	14.50	13.12	14.50	13.81	14.50
Regression	0.025	0.80	1.3	5.3	2.	16.6	2.8	23.	8.8	34	32.	59.
Malliavin	0.020	0.95	1.03	23.5	31.	360.	256.	2782	694	4010	3650.	9080.

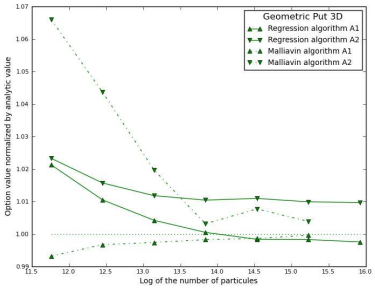
We observe that the cost of the regression approach is linear with respect to the number of particles (instead of the expected $N\ln(N)$ due to the sort algorithm).

[2] For all the computations, we use a core i7 2,9 GHz processor.

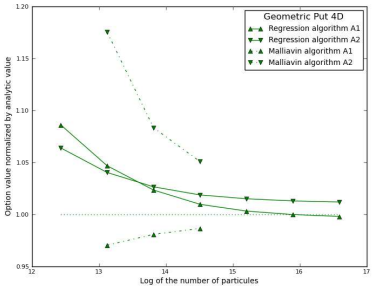


(a) 1D

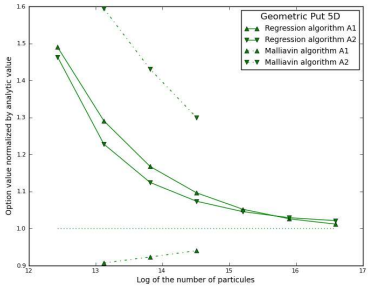
(b) 2D



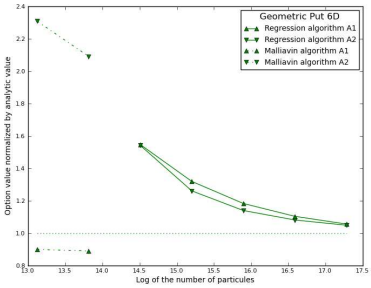
(c) 3D



(d) 4D



(e) 5D



(f) 6D

Fig. 7 Comparison between the regression and the Malliavin based methods for the Bermudean geometric put option

If we compare the two methods for Option 1 and Algorithm A1 (the most accurate), we can conclude that for a given level of accuracy:

- the Malliavin approach is more attractive in dimension 1 (similar cost but more accurate).
- the Malliavin approach seems to be more attractive in dimension 2 too. For example, with 32.000 particles and a cost of 0.45 seconds, the Malliavin approach

provides the same accuracy as the regression approach with 258.000 particles and a cost of 1.8 seconds (the relative error is of order of 0.2%).
- the regression approach seems to become more attractive for dimensions greater or equal to 3. For instance, in dimension 3, with 2 millions particles and a cost of 41 seconds, it provides the same accuracy as the Malliavin approach with 500.000 particles and a cost of 70 seconds (the relative error is of order of 0.2%).

On figure 8, we provide the results obtained with the quantization approach for Option 1 depending on the number of global quantization points. We use two different approaches:

- The backward approach: it consists in applying Algorithm A2 to the quantized process.
- The forward approach: we first apply the backward Algorithm A1 to the quantized process so as to compute an estimation $\hat{C}^\pi$ of the continuation region $C^\pi := \{(t,x) \in \pi \times (0,\infty)^d \ : \ p^\pi(t,x) > g(t,x)\}$, where $p^\pi(t,x)$ is the price of the Bermudean option at time t if the stock price is x. We then simulate forward N paths, $(X^{(j)})_{j\leq N}$, of the stock price process X and approximate the Bermudean option price by $N^{-1}\sum_{j\leq N} g(\hat{\tau}_0^{(j)}, X^{(j)}_{\hat{\tau}_0^{(j)}})$ where $\hat{\tau}_0^{(j)} := \min\{t \in \pi \ : \ (t, X_t^{(j)}) \notin \hat{C}^\pi\}$. We use 4 millions particles.

In dimension 1, the quantization method requires 1.600 points[3] for an accuracy of 0.2%. Once probabilities have been calculated, the backward and the forward resolutions are achieved in 0.02 seconds. An equivalent accuracy can be obtained with the regression approach in 0.350 seconds. It takes 0.050 seconds with the Malliavin approach. Obviously this does not take into account the time spend to compute the transition probabilities, nor the construction of the quantization tree.
In dimension 2, we could only obtain an error of 0.8% with a total of 6.400 quantization points and a quantization of the last time step of 815 points. With 25.600 points, the maximum accuracy was 2% in dimension 3, 8% in dimension 4, 15% in dimension 5, and 22% in dimension 6, when only using Algorithm A2.
Algorithm A1 combined with a forward Monte-Carlo simulation provides better results.
Overall, the method is converging and is certainly the less time consuming once a grid and the associated transition probabilities have been computed. However, the grids proposed on the website http://www.quantize.maths-fi.com are not thin enough to provide accurate results.

On figures 9 and 10, we give our results for the digital put. All the methods have difficulties to converge. Algorithm A1 always gives better results than Algorithm A2, i.e. the approximation by the continuation value seems to converge slower than the one based on stopping times. For dimensions equal or greater to 4, only the

[3] Here and below, the number of points corresponds to the sum of the numbers of points used at each time step. There are distributed according to [4]

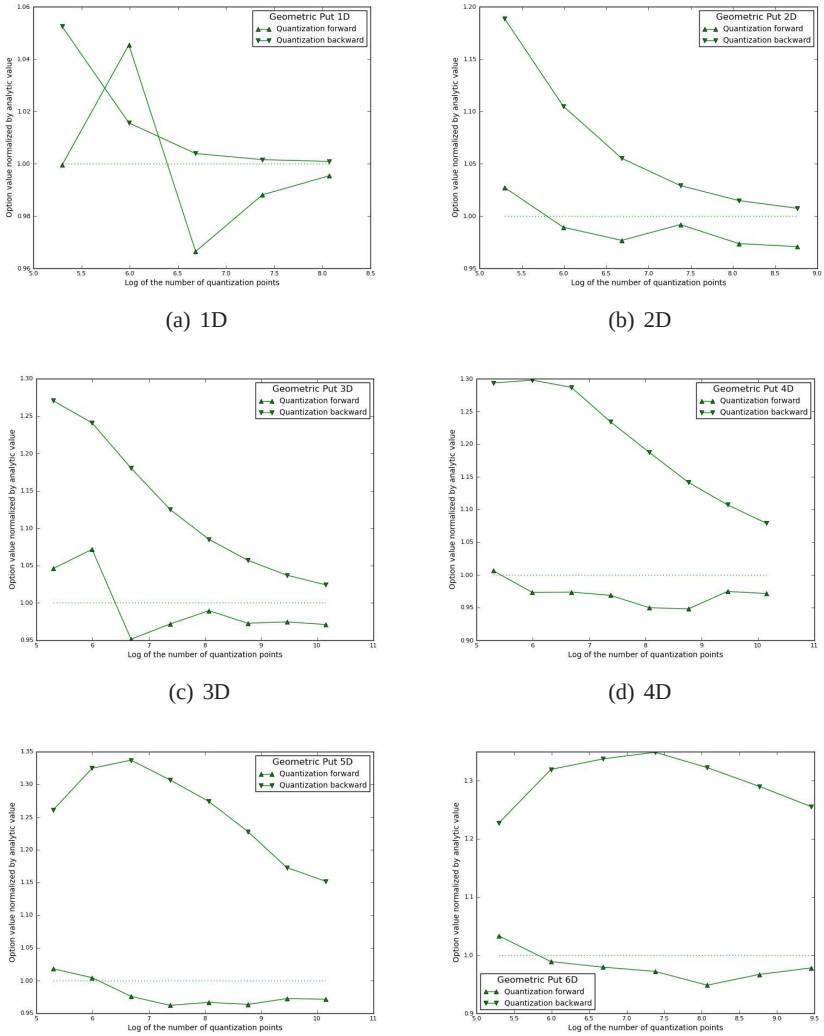
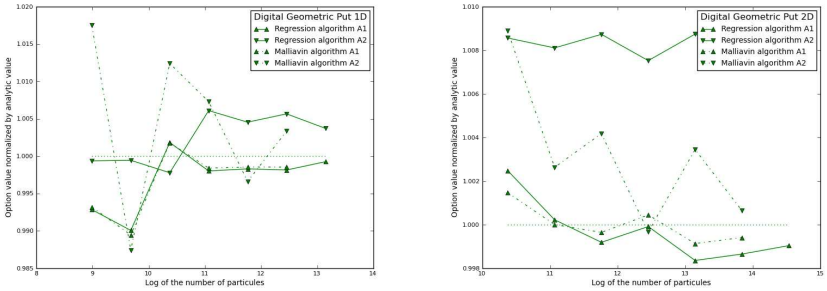


Fig. 8 Convergence of the quantization method for the geometric Bermudean put option

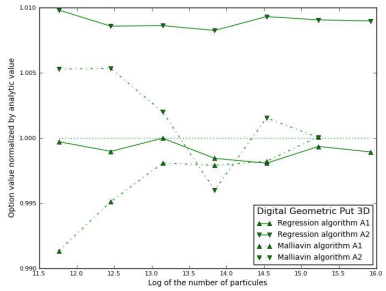
regression method provides good results. In dimension 3, for a given number of particles, the results obtained by the Malliavin and the regression approach are similar for Algorithm A1. Because of the difference in computation time, the regression approach is more appropriate. In dimension 2, the Malliavin approach combined with Algorithm A1 seems to be more attractive but it is not clear in dimension 1.

In dimension 1, the quantization approach only achieves an accuracy of 1.2% for the finest meshes while the regression and Malliavin approaches achieves a 0.3% error. In dimensions 2 and 3, it provides good results but the accuracy of the two other approaches is much better. Results in dimension greater than 4 shows that far more quantization points are needed.

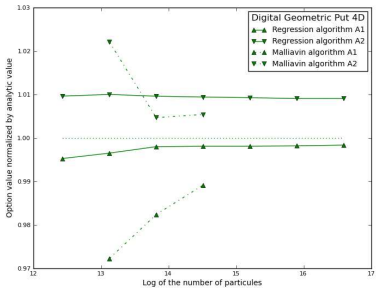


(a) 1D

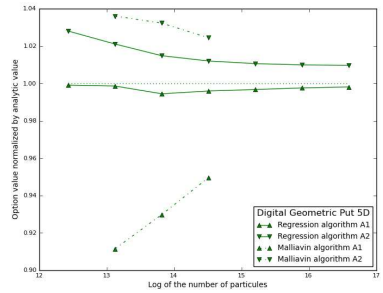
(b) 2D



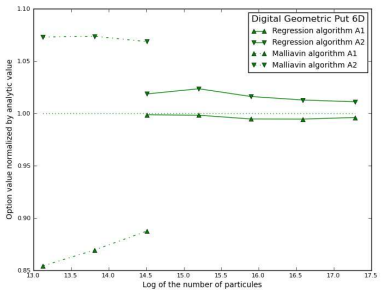
(c) 3D



(d) 4D

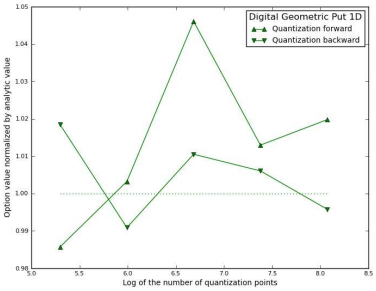


(e) 5D

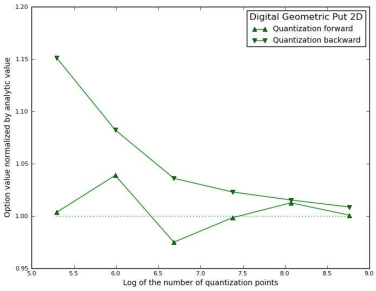


(f) 6D

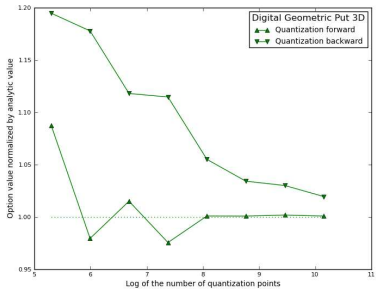
Fig. 9 Comparison between the regression and the Malliavin approaches for the Bermudean geometric digital option



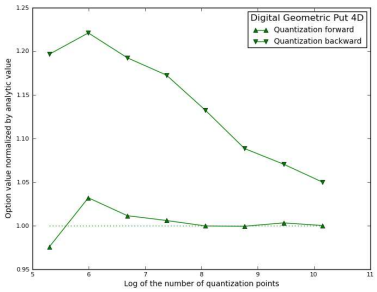
(a) 1D



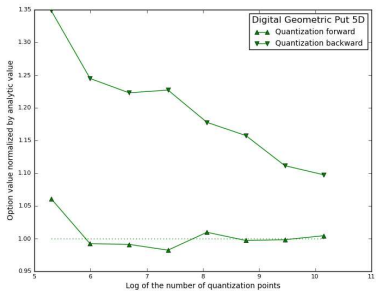
(b) 2D



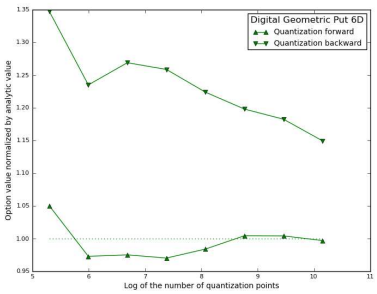
(c) 3D



(d) 4D



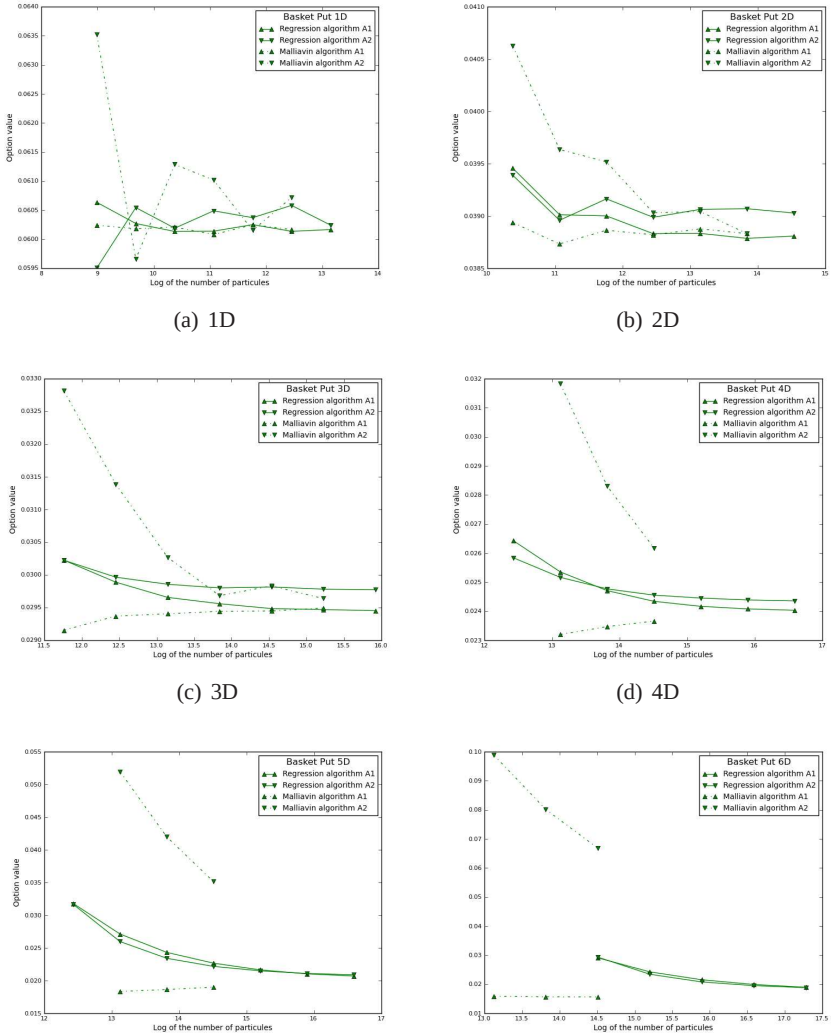
(e) 5D



(f) 6D

Fig. 10 Convergence of the quantization method for the geometric Bermudean digital option

On figures 11 and 12, we provide the results obtained for the Bermudean basket put option. It confirms our previous observations.



(a) 1D (b) 2D

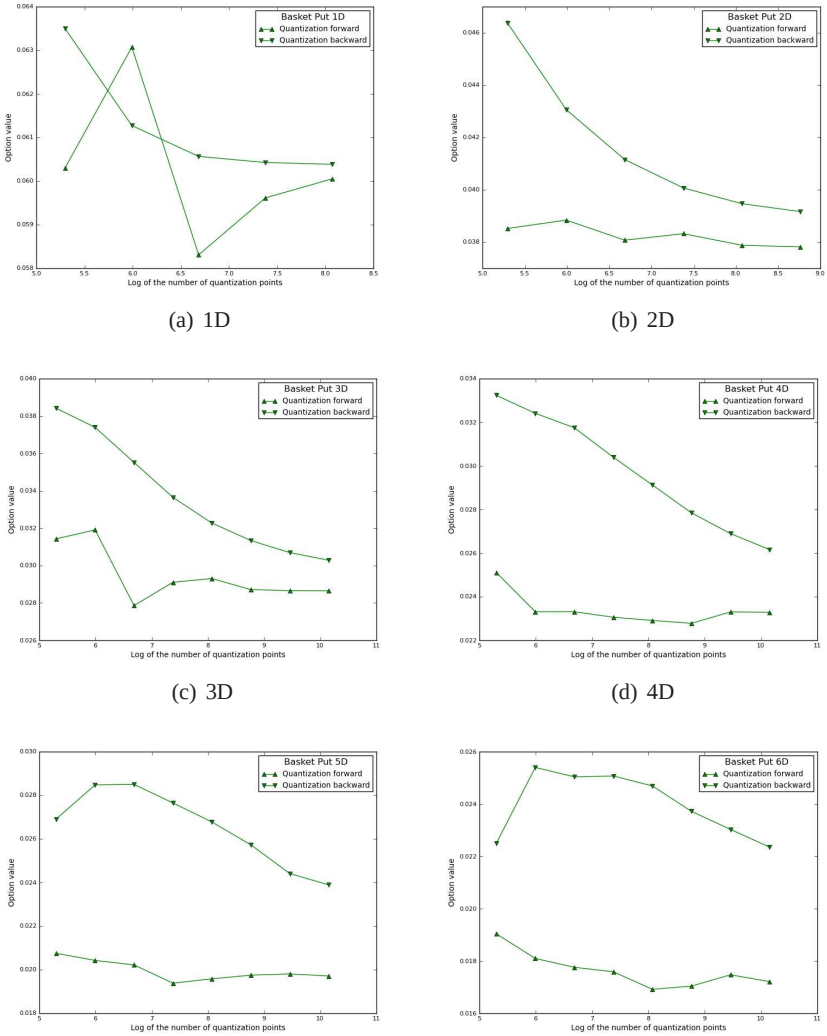
(c) 3D (d) 4D

(e) 5D (f) 6D

Fig. 11 Comparison between the regression and the Malliavin based methods for the Bermudean basket put option

5.3 Numerical results on hedging policies

In figures 13 and 14, we provide the results obtained by combining the regression and the Malliavin approach with the representations (9) and (11) for the Bermudean geometric put option. We provide the results obtained by using the representation



(a) 1D (b) 2D

(c) 3D (d) 4D

(e) 5D (f) 6D

Fig. 12 Convergence of the quantization method for the Bermudean basket put option

(11) for the digital option. We only provide the results for prices computed with Algorithm A1, Algorithm A2 being less accurate.

In the figures, we use the following terminology:

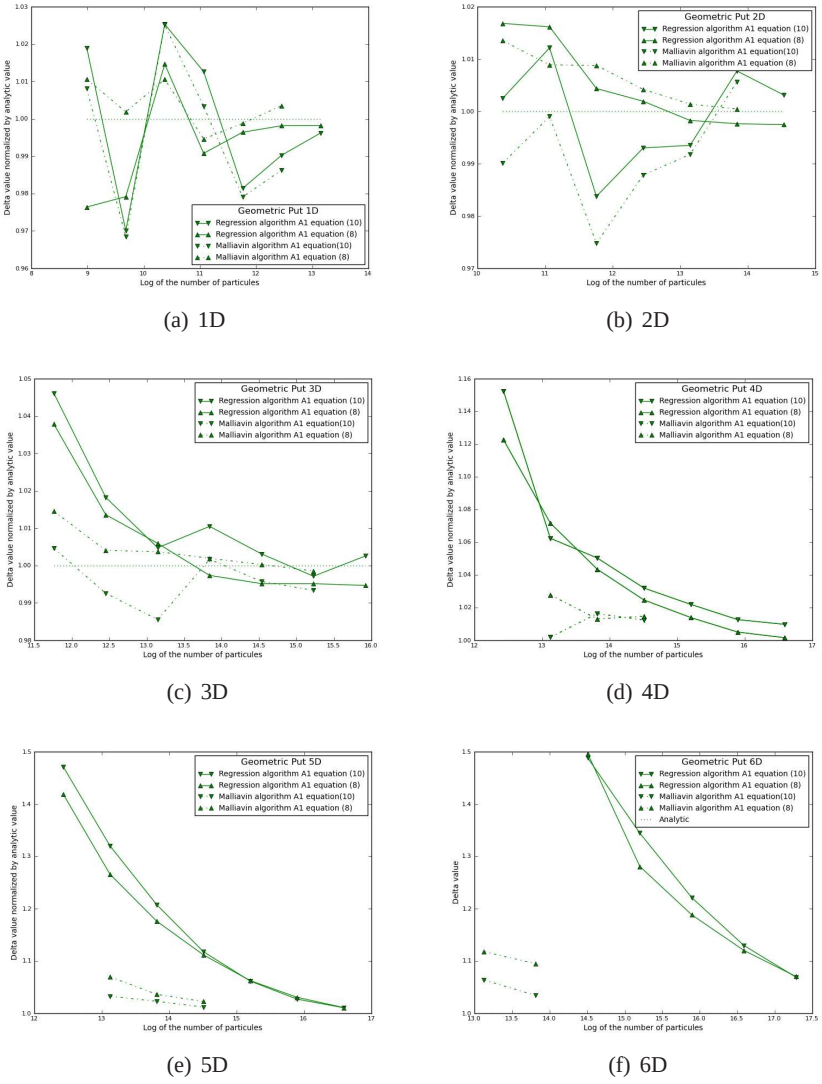
- Regression algorithm A1 means that prices are computed by using algorithm A1 and the regression based technique.

- Malliavin algorithm A1 means that prices are computed by using algorithm A1 and the Malliavin based representation of conditional expectations.
- equation (8), resp. (10), means that we then use the representation of the delta given in (8), resp. (10).

Note that the problem is symmetric in the different components, so that only one figure is provided. For more clarity, we normalize our result by dividing the estimation by the *true* value computed by analytical methods.

Both representations seem to provide equally good results.

Acknowledgements We are grateful to Christos Makris, Paul Masurel and to the two anonymous referees for helpful suggestions.

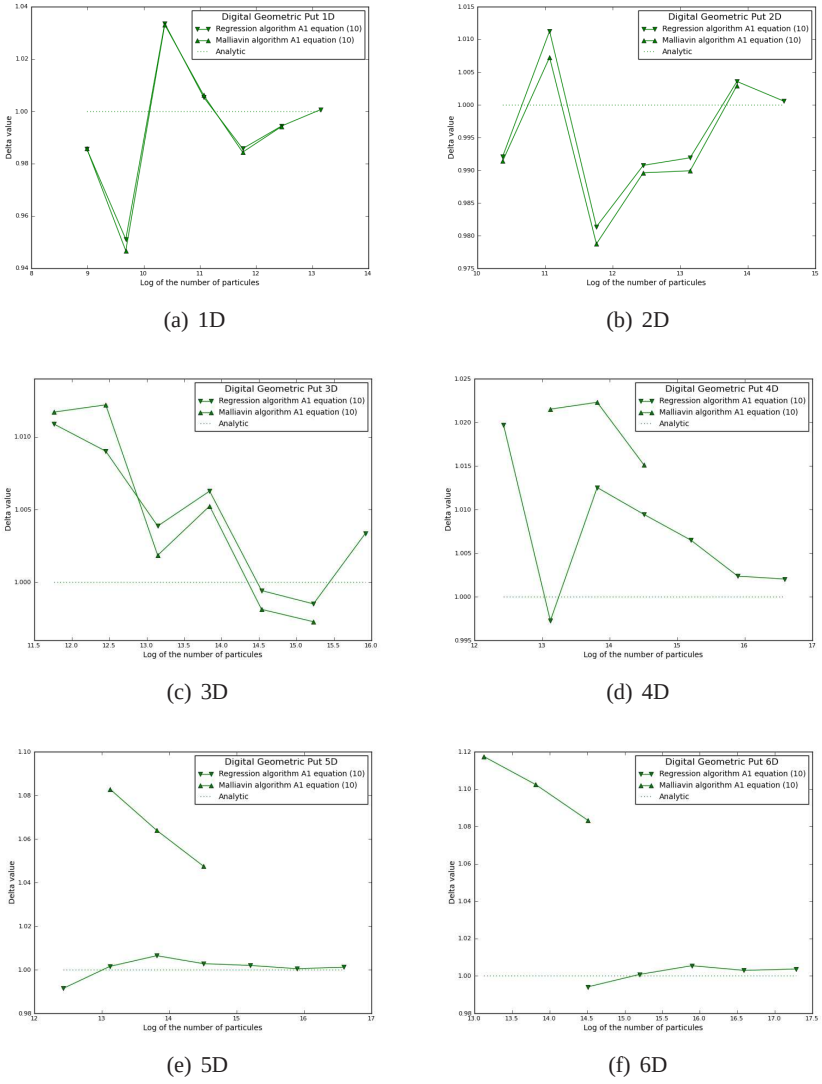


(a) 1D (b) 2D

(c) 3D (d) 4D

(e) 5D (f) 6D

Fig. 13 Convergence of the delta for the geometric Bermudean put option



(a) 1D (b) 2D

(c) 3D (d) 4D

(e) 5D (f) 6D

Fig. 14 Convergence of the delta for the geometric Bermudean digital option

References

1. L. Andersen and M. Broadie: A Primal-Dual Simulation Algorithm for Pricing Multi-Dimensional American Options. Management Science, 50 (9), 1222-1234 (2004).
2. V. Bally, M.E. Caballero, B. Fernandez and N. El Karoui: Reflected BSDEs, PDEs and variational inequalities. INRIA Report 4455, (2002).
3. V. Bally, G. Pagès: Error analysis of the quantization algorithm for obstacle problems. Stochastic Processes and their Applications **106**, 1-40, 2003.
4. V. Bally, G. Pagès, J. Printems : A quantization method for pricing and hedging multi-dimensional American style options. Mathematical Finance **15**, 1 (2005)
5. O. Bardou, S. Bouthemy, G. Pagès: Optimal quantization for the pricing of swing options. Applied Mathematical Finance **16(2)**, 183-217 (2009)
6. D. Belomestny, Ch. Bender, J. Schoenmakers: True upper bounds for Bermudan products via non-nested Monte Carlo. Mathematical Finance, 19(1), 53-71 (2009).
7. J.-L. Bentley and M.-I. Shamos: Divide-and-Conquer in multidimensional space. Proc. Eighth ACM Annual Symp. on Theory of Comput 220-230 (1976)
8. M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf: Computational geometry, Springer (2000)
9. B. Bouchard, J.-F. Chassagneux : Discrete time approximation for continuously and discretely reflected BSDE's. Stochastic Processes and their Applications, **118**, 2269-2293 (2008)
10. B. Bouchard, I. Ekeland, N. Touzi: On the Malliavin approach to Monte Carlo approximation of conditional expectations. Finance and Stochastics, **8(1)**, 45-71 (2004)
11. B. Bouchard, E. Elie and N. Touzi: Discrete-Time Approximation of BSDEs and Probabilistic Schemes for Fully Nonlinear PDEs. Radon Series Comp. Appl. Math. **8**, 133, de Gruyter ed. (2009)
12. B. Bouchard and N. Touzi : Discrete-time approximation and Monte Carlo simulation of backward stochastic differential equations. Stochastic Processes and their Applications, **111**, 175-206 (2004)
13. M. Broadie, P. Glasserman: Estimating security price derivatives using simulation. Manag. Sci., **42**, 269285 (1996)
14. J.-F. Carrière: Valuation of the Early-Exercise Price for Options using Simulations and Nonparametric Regression. Insurance : mathematics and Economics, **19**, 19-30 (1996)
15. A. R. Choudhury, A. King, S. Kumar, Y. Sabharwal: Optimizations in financial engineering: The Least-Squares Monte Carlo method of Longstaff and Schwartz. In Proc. of 2008 IEEE International Symposium on Parallel and Distributed Processing. (IPDPS 2008):pp 1-11 April 2008
16. E. Clément, D. Lamberton, P. Protter: An analysis of a least squares regression method for American option pricing. Finance and Stochastics, **6**, 449-472, 2002.
17. J. Detemple, R. Garcia, M. Rindisbacher: Asymptotic Properties of Monte Carlo Estimators of Derivatives. Management science, **51(11)**, 1657-1675 (2005)
18. D. Egloff: Monte Carlo algorithms for optimal stopping and statistical learning. Ann. Appl. Probab. , 15 (2), 1396-1432 (2005).
19. N. El Karoui: Les aspects probabilistes du contrôle stochastique, Ecole d'Eté de Probabilités de Saint Flour, IX, Lecture Notes in Mathematics 876, Springer Verlag (1979)
20. E. Fournier, J.-M. Lasry, J. Lebuchoux, P.-L. Lions: Applications of Malliavin calculus to Monte Carlo methods in finance II. Finance and Stochastics , **5**, 201-236 (2001)
21. E. Fournier, J.-M. Lasry, J. Lebuchoux, P.-L. Lions, N. Touzi: Applications of Malliavin calculus to Monte Carlo methods in finance. Finance and Stochastics, **3**, 391-412 (1999)
22. P. Glasserman and B. Yu: Number of paths versus number of basis functions in American option pricing. Ann. Appl. Probab. 14(4), 2090-2119 (2004).
23. E. Gobet: Revisiting the Greeks for European and American options. Proceedings of the "International Symposium on Stochastic Processes and Mathematical Finance" at Ritsumeikan University, Kusatsu, Japan (2003)

24. E. Gobet and J. P. Lemor: Numerical simulation of BSDEs using empirical regression methods : theory and practice. In Proceedings of the Fifth Colloquium on BSDEs (29th May - 1st June 2005, Shan- gai), Available on http ://hal.archives-ouvertes.fr/hal-00291199/fr/, (2006).
25. E. Gobet, J.P. Lemor, X. Warin: A regression-based Monte-Carlo method to solve backward stochastic differential equations. Annals of Applied Probability, **15(3)**, 2172-2002 (2005)
26. M. B. Haugh and L. Kogan: Pricing American Options: A Duality Approach. Operation research, 52(2), 258-270 (2004)
27. J. JaJa, C. Mortensen, Q. Shi: Space Efficient and Fast Algorithms for Multidimensional Dominance Reporting and Counting. Proceedings of the 2004 Annual Symposium on Algorithms and Computation, Hong Kong (2004)
28. J.P. Lemor: Approximation par projections et simulations de Monte-Carlo des équations différentielles stochastiques rétrogrades. PhD thesis, Ecole Polytechnique, http ://www.imprimerie.polytechnique.fr/Theses/Files/lemor.pdf, (2005).
29. J.P. Lemor, E. Gobet, and X. Warin: Rate of convergence of an empirical regression method for solving generalized backward stochastic differential equations. Bernoulli, 12(5), 889916 (2006).
30. P.-L. Lions, H. Regnier: Calcul du prix et des sensibilités d'une option américaine par une méthode de Monte Carlo, preprint (2001)
31. F. Longstaff and E. Schwartz: Valuing American options by simulation: A simple least-squares. Review of Financial Studies, **1(14)**, 113-147, 2001.
32. J. Ma and J. Zhang: Representations and regularities for solutions to BSDEs with reflections. Stochastic processes and their applications, **115**, 539-569 (2005)
33. C. Makris, A.-K. Tsakalidis: Algorithms for three dimensional dominance searching in linear space. Information Processing Letters, **66**, 6 (1998)
34. J. McNames: A Fast Nearest-Neighbor Algorithm Based on a Principal Axis Search Tree. IEEE Transactions on Pattern Analysis and Machine Intelligence, **23(9)**, 964-976 (2001)
35. G. Pagès, J. Printems: Optimal quadratic quantization for numerics: the Gaussian case. Monte Carlo Methods & Applications, **2**, 9, 135-166 (2003)
36. G. Pagès, J. Printems: Functional quantization for numerics with an application to option pricing. Monte Carlo Methods & Applications , **4**, 11, 407-446 (2005)
37. V.-V. Piterbarg: Risk sensitivities of Bermuda options. Technical report, Bank of America, http://ssrn.com/abstract=367920 (2002)
38. F.-P. Preparata, M.-I.Shamos:Computational geometry (an introduction), Springer (1985)
39. J.-N. Tsitsiklis, B. Van Roy: Optimal Stopping of Markov Processes: Hilbert Spaces theory, Approximations Algorithms and an application to pricing high-dimensional financial derivatives. IEEE Transactions on Automatic Control, **10(44)**, 1840-1851 (1999)
40. D. Zanger: Convergence of a least-squares Monte Carlo algorithm for bounded approximating sets. Applied Mathematical Finance, 16(2), 123150 (2009).