Stochastic methods for solving high-dimensional partial differential equations

Marie Billaud-Friess[*]   Arthur Macherey[*‡]   Anthony Nouy[*]   Clémentine Prieur[†]

May 15, 2019

Abstract

We propose algorithms for solving high-dimensional Partial Differential Equations (PDEs) that combine a probabilistic interpretation of PDEs, through Feynman-Kac representation, with sparse interpolation. Monte-Carlo methods and time-integration schemes are used to estimate pointwise evaluations of the solution of a PDE. We use a sequential control variates algorithm, where control variates are constructed based on successive approximations of the solution of the PDE. Two different algorithms are proposed, combining in different ways the sequential control variates algorithm and adaptive sparse interpolation. Numerical examples will illustrate the behavior of these algorithms.

1   Introduction

We consider the solution of an elliptic partial differential equation

$$
\begin{aligned}
\mathcal{A}(u) = g &\quad \text{in} \quad \mathcal{D}, \\
u = f &\quad \text{on} \quad \partial\mathcal{D},
\end{aligned}
\tag{1}
$$

where $u : \overline{\mathcal{D}} \to \mathbb{R}$ is a real-valued function, and $\mathcal{D}$ is an open bounded domain in $\mathbb{R}^d$. $\mathcal{A}$ is an elliptic linear differential operator and $f : \partial\mathcal{D} \to \mathbb{R}$, $g : \overline{\mathcal{D}} \to \mathbb{R}$ are respectively the boundary condition and the source term of the PDE.

We are interested in approximating the solution of (1) up to a given precision. For high dimensional PDEs ($d \gg 1$), this requires suitable approximation formats such as sparse tensors [5, 26] or low-rank tensors [24, 18, 19, 1, 23]. Also, this requires algorithms that provide approximations in a given approximation format. Approximations are typically provided by Galerkin projections using variational formulations of PDEs. Another path consists in using a probabilistic representation of the solution u through Feynman-Kac formula, and Monte-Carlo methods to provide estimations

---

[*]Centrale Nantes, LMJL, UMR CNRS 6629, 1 rue de la Noë, 44321 Nantes

[†]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP*, LJK, 38000 Grenoble, France

[‡]Corresponding author (arthur.macherey@ec-nantes.fr).

of pointwise evaluations of u (see e.g., [17]). This allows to compute approximations in a given approximation format through classical interpolation or regression [3, 27, 4]. In [14, 15], the authors consider interpolations on fixed polynomial spaces and propose a sequential control variates method for improving the performance of Monte-Carlo estimation. In this paper, we propose algorithms that combine this variance reduction method with adaptive sparse interpolation [6, 7].

The outline is as follows. In section 2, we recall the theoretical and numerical aspects associated to probabilistic tools for estimating the solution of (1). We also present the sequential control variates algorithm introduced in [14, 15]. In section 3 we introduce sparse polynomial interpolation methods and present a classical adaptive algorithm. In section 4, we present two algorithms combining the sequential control variates algorithm from section 2 and adaptive sparse polynomial interpolation. Finally, numerical results are presented in section 4.

2   Probabilistic tools for solving PDEs

We consider the problem (1) with a linear partial differential operator defined by $\mathcal{A}(u) = -\mathcal{L}(u) + ku$, where k is a real valued function defined on $\overline{\mathcal{D}}$, and where

$$\mathcal{L}(u)(x) = \frac{1}{2} \sum_{i,j=1}^{d} (\sigma(x)\sigma(x)^T)_{ij} \partial^2_{x_i x_j} u(x) + \sum_{i=1}^{d} b_i(x) \partial_{x_i} u(x) \tag{2}$$

is the *infinitesimal generator* associated to the d-dimensional diffusion process X^x solution of the stochastic differential equation

$$dX^x_t = b(X^x_t)dt + \sigma(X^x_t)dW_t, \quad X^x_0 = x \in \overline{\mathcal{D}}, \tag{3}$$

where W is a d-dimensional Brownian motion and $b := (b_1, \dots, b_d)^T : \mathbb{R}^d \to \mathbb{R}^d$ and $\sigma : \mathbb{R}^d \to \mathbb{R}^{d \times d}$ stand for the drift and the diffusion respectively.

2.1   Pointwise evaluations of the solution

The following theorem recalls the Feynman-Kac formula (see [11, Theorem 2.4] or [12, Theorem 2.4] and the references therein) that provides a probabilistic representation of $u(x)$, the solution of (1) evaluated at $x \in \overline{\mathcal{D}}$.

Theorem 2.1 (Feynman-Kac formula)**.** *Assume that*

(H1) *$\mathcal{D}$ is an open connected bounded domain of $\mathbb{R}^d$, regular in the sense that, if $\tau^x = \inf \{s > 0 \; : \; X^x_s \notin \mathcal{D}\}$ is the first exit time of $\mathcal{D}$ for the process X^x, we have*

$$\mathbb{P}(\tau^x = 0) = 1, \quad x \in \partial \mathcal{D},$$

(H2) *b, σ are Lipschitz functions,*

(H3) f is continuous on ∂D, g and $k \geq 0$ are Hölder-continuous functions on $\overline{\mathcal{D}}$,

(H4) (uniform ellipticity assumption) there exists $c > 0$ such that

$$\sum_{i,j=1}^{d} \left(\sigma(x)\sigma(x)^T\right)_{ij} \xi_i \xi_j \geq c \sum_{i=1}^{d} \xi_i^2, \quad \xi \in \mathbb{R}^d, \ x \in \overline{\mathcal{D}}.$$

Then, there exists a unique solution of (1) in $\mathcal{C}\left(\overline{\mathcal{D}}\right) \cap \mathcal{C}^2\left(\mathcal{D}\right)$, which satisfies for all $x \in \overline{\mathcal{D}}$

$$u(x) = \mathbb{E}\left[F(u, X^x)\right] \tag{4}$$

where

$$F(u, X^x) = u(X_{\tau^x}^x) \exp\left(-\int_0^{\tau^x} k(X_t^x) dt\right) + \int_0^{\tau^x} \mathcal{A}(u)(X_t^x) \exp\left(-\int_0^t k(X_s^x) ds\right) dt,$$

with $u(X_{\tau^x}^x) = f(X_{\tau^x}^x)$ and $\mathcal{A}(u)(X_t^x) = g(X_t^x)$.

Note that $F(u, X^x)$ in (4) only depends on the values of u on ∂D and $\mathcal{A}(u)$ on D, which are the given data f and g respectively. A Monte-Carlo method can then be used to estimate $u(x)$ using (4), which relies on the simulation of independent samples of an approximation of the stochastic process X^x. This process is here approximated by an Euler-Maruyama scheme. More precisely, letting $t_n = n\Delta t$, $n \in \mathbb{N}$, X^x is approximated by a piecewise constant process $X^{x,\Delta t}$, where $X_t^{x,\Delta t} = X_n^{x,\Delta t}$ for $t \in [t_n, t_{n+1}[$ and

$$\begin{aligned} X_{n+1}^{x,\Delta t} &= X_n^{x,\Delta t} + \Delta t \, b(X_n^{x,\Delta t}) + \sigma(X_n^{x,\Delta t}) \, \Delta W_n, \\ X_0^{x,\Delta t} &= x. \end{aligned} \tag{5}$$

Here $\Delta W_n = W_{n+1} - W_n$ is an increment of the standard Brownian motion. For details on time-integration schemes, the reader can refer to [20]. Letting $\{X^{x,\Delta t}(\omega_m)\}_{m=1}^M$ be independent samples of $X^{x,\Delta t}$, we obtain an estimation $u_{\Delta t, M}(x)$ of $u(x)$ defined as

$$\begin{aligned} u_{\Delta t, M}(x) &:= \frac{1}{M} \sum_{m=1}^{M} F\left(u, X^{x,\Delta t}(\omega_m)\right) \\ &= \frac{1}{M} \sum_{m=1}^{M} \left[f(X_{\tau^{x,\Delta t}}^{x,\Delta t}(\omega_m)) \exp\left(-\int_0^{\tau^{x,\Delta t}} k(X_t^{x,\Delta t}(\omega_m)) dt\right) \right. \\ &\left. + \int_0^{\tau^{x,\Delta t}} g(X_t^{x,\Delta t}(\omega_m)) \exp\left(-\int_0^t k(X_s^{x,\Delta t}(\omega_m)) ds\right) dt \right] \end{aligned} \tag{6}$$

where $\tau^{x,\Delta t}$ is the first exit time of D for the process $X^{x,\Delta t}(\omega_m)$, given by

$$\tau^{x,\Delta t} = \inf\left\{t > 0 \ : \ X_t^{x,\Delta t} \notin \mathcal{D}\right\} = \min\left\{t_n > 0 \ : \ X_{t_n}^{x,\Delta t} \notin \mathcal{D}\right\}.$$

Remark 2.2. In practice, f has to be defined over $\mathbb{R}^d$ and not only on the boundary ∂D. Indeed, although $X_{\tau^x}^x \in \partial \mathcal{D}$ with probability one, $X_{\tau^{x,\Delta t}}^{x,\Delta t} \in \mathbb{R}^d \setminus \overline{\mathcal{D}}$ with probability one.

3

The error can be decomposed in two terms

$$u(x) - u_{\Delta t, M}(x) = \overbrace{u(x) - \mathbb{E}\left[F\left(u, X^{x,\Delta t}\right)\right]}^{\varepsilon_{\Delta t}}$$
$$+ \underbrace{\mathbb{E}\left[F\left(u, X^{x,\Delta t}\right)\right] - \frac{1}{M}\sum_{m=1}^{M} F\left(u, X^{x,\Delta t}(\omega_m)\right)}_{\varepsilon_{MC}}, \tag{7}$$

where $\varepsilon_{\Delta t}$ is the time integration error and ε_{MC} is the Monte-Carlo estimation error. Before discussing the contribution of each of both terms to the error, let us introduce the following additional assumption, which ensures that $\mathcal{D}$ does not have singular points*.

(H5) Each point of $\partial\mathcal{D}$ satisfies the *exterior cone condition* which means that, for all $x \in \partial\mathcal{D}$, there exists a finite right circular cone K, with vertex x, such that $\overline{K} \cap \overline{\mathcal{D}} = \{x\}$.

Under assumptions (H1)-(H5), it can be proven [15, §4.1] that the time integration error $\varepsilon_{\Delta t}$ converges to zero. It can be improved to $O(\Delta t^{1/2})$ by adding differentiability assumptions on the boundary [16]. The estimation error ε_{MC} is a random variable with zero mean and standard deviation converging as $O(M^{-1/2})$. The computational complexity for computing a pointwise evaluation

---

*Note that together with (H4), assumption (H5) implies (H1) (see [15, §4.1] for details), so that the set of hypotheses (H1)-(H5) could be reduced to (H2)-(H5).

of $u_{\Delta t,M}(x)$ is in $O\left(M\Delta t^{-1}\right)$ in expectation for Δt sufficiently small[†], so that the computational complexity for achieving a precision ϵ (root mean squared error) behaves as $O(\epsilon^{-4})$. This does not allow to obtain a very high accuracy in a reasonable computational time. The convergence with Δt can be improved to $O(\Delta t)$ by suitable boundary corrections [16], therefore yielding a convergence in $O(\epsilon^{-3})$. To further improve the convergence, high-order integration schemes could be considered (see [20] for a survey). Also, variance reduction methods can be used to further improve the convergence, such as antithetic variables, importance sampling, control variates (see [17]). Multilevel Monte-Carlo [13] can be considered as a variance reduction method using several control variates (associated with processes $X^{x,\Delta t_k}$ using different time discretizations). Here, we rely on the sequential control variates algorithm proposed in [14] and analyzed in [15]. This algorithm constructs a sequence of approximations of u. At each iteration of the algorithm, the current approximation is used as a control variate for the estimation of u through Feynman-Kac formula.

2.2 A sequential control variates algorithm

Here we recall the sequential control variates algorithm introduced in [14] in a general interpolation framework. We let $V_\Lambda \subset \mathcal{C}^2(\overline{\mathcal{D}})$ be an approximation space of finite dimension $\#\Lambda$ and let $\mathcal{I}_\Lambda : \mathbb{R}^\mathcal{D} \to V_\Lambda$ be the interpolation operator associated with a unisolvent grid $\Gamma_\Lambda = \{x_\nu : \nu \in \Lambda\}$. We let $(l_\nu)_{\nu \in \Lambda}$ denote the (unique) basis of V_Λ that satisfies the interpolation property $l_\nu(x_\mu) = \delta_{\nu\mu}$ for all $\nu, \mu \in \Lambda$. The interpolation $\mathcal{I}_\Lambda(w) = \sum_{\nu \in \Lambda} w(x_\nu) l_\nu(x)$ of function w is then the unique function in V_Λ such that

$$\mathcal{I}_\Lambda(w)(x_\nu) = w(x_\nu), \quad \nu \in \Lambda.$$

The following algorithm provides a sequence of approximations $(\tilde{u}^k)_{k \geq 1}$ of u in V_Λ, which are defined by $\tilde{u}^k = \tilde{u}^{k-1} + \tilde{e}^k$, where $\tilde{e}^k$ is an approximation of e^k, solution of

$$\begin{aligned}
\mathcal{A}(e^k)(x) &= g(x) - \mathcal{A}(\tilde{u}^{k-1})(x), \quad x \in \mathcal{D}, \\
e^k(x) &= f(x) - \tilde{u}^{k-1}(x), \qquad x \in \partial\mathcal{D}.
\end{aligned}$$

Note that e^k admits a Feyman-Kac representation $e^k(x) = \mathbb{E}(F(e^k, X^x))$, where $F(e^k, X^x)$ depends on the residuals $g - \mathcal{A}(\tilde{u}^{k-1})$ on $\mathcal{D}$ and $f - \tilde{u}^{k-1}$ on $\partial\mathcal{D}$. The approximation $\tilde{e}^k$ is then defined as the interpolation $\mathcal{I}_\Lambda(e^k_{\Delta t,M})$ of the Monte-Carlo estimate $e^k_{\Delta t,M}(x)$ of $e^k_{\Delta t}(x) = \mathbb{E}(F(e^k, X^{x,\Delta t}))$ (using M samples of $X^{x,\Delta t}$).

Algorithm 2.3. (Sequential control variates algorithm)

1: Set $\tilde{u}^0 = 0$, $k = 1$ and $S = 0$.
2: **while** $k \leq K$ and $S < n_s$ **do**
3:     Compute $e^k_{\Delta t,M}(x_\nu)$ for $x_\nu \in \Gamma_\Lambda$.
4:     Compute $\tilde{e}^k = \mathcal{I}_\Lambda(e^k_{\Delta t,M}) = \sum_{\nu \in \Lambda} e^k_{\Delta t,M}(x_\nu) l_\nu(x)$.

[†]A realization of $X^{x,\Delta t}$ over the time interval $[0, \tau^{x,\Delta t}]$ can be computed in $O\left(\tau^{x,\Delta t}\Delta t^{-1}\right)$. Then, the complexity to evaluate $u_{\Delta t,M}(x)$ is in $O(\mathbb{E}(\tau^{x,\Delta t})M\Delta t^{-1})$ in expectation. Under $(H1)$-$(H5)$, it is stated in the proof of [15, Theorem 4.2] that $\sup_x \mathbb{E}[\tau^{x,\Delta t}] \leq C$ with C independent of Δt for Δt sufficiently small.

5:    Update $\tilde{u}^k = \tilde{u}^{k-1} + \tilde{e}^k$.

6:    If $\|\tilde{u}^k - \tilde{u}^{k-1}\|_2 \leq \epsilon_{tol}\|\tilde{u}^{k-1}\|_2$ then $S = S + 1$ else $S = 0$.

7:    Set $k = k + 1$.

8: **end while**

For practical reasons, Algorithm 2.3 is stopped using an heuristic error criterion based on stagnation. This criterion is satisfied when the desired tolerance ϵ_{tol} is reached for n_s successive iterations (in practice we chose $n_s = 5$).

Now let us provide some convergence results for Algorithm 2.3. To that goal, we introduce the time integration error at point x for a function h

$$e^{\Delta t}(h, x) = \mathbb{E}[F(h, X^{\Delta t, x})] - \mathbb{E}[F(h, X^x)]. \tag{8}$$

Then the following theorem [15, Theorem 3.1] gives a control of the error in expectation.

Theorem 2.4. *Assuming (H2)-(H5), it holds*

$$\sup_{\nu \in \Lambda} \left| \mathbb{E}\left[\tilde{u}^{n+1}(x_\nu) - u(x_\nu) \right] \right| \leqslant C(\Delta t, \Lambda) \sup_{\nu \in \Lambda} \left| \mathbb{E}\left[\tilde{u}^n(x_\nu) - u(x_\nu) \right] \right| + C_1(\Delta t, \Lambda)$$

with $C(\Delta t, \Lambda) = \sup_{\nu \in \Lambda} \sum_{\mu \in \Lambda} |e^{\Delta t}(l_\mu, x_\nu)|$ and $C_1(\Delta t, \Lambda) = \sup_{\nu \in \Lambda} \left| e^{\Delta t}(u - \mathcal{I}_\Lambda(u), x_\nu) \right|$.

Moreover if $C(\Delta t, \Lambda) < 1$, it holds

$$\limsup_{n \to \infty} \sup_{\nu \in \Lambda} |\mathbb{E}\left[\tilde{u}^n(x_\nu) - u(x_\nu) \right]| \leqslant \frac{C_1(\Delta t, \Lambda)}{1 - C(\Delta t, \Lambda)}. \tag{9}$$

The condition $C(\Delta t, \Lambda) < 1$ implies that in practice Δt should be chosen sufficiently small [15, Theorem 4.2]. Under this condition, the error at interpolation points uniformly converges geometrically up to a threshold term depending on time integration errors for interpolation functions l_ν and the interpolation error $u - \mathcal{I}_\Lambda(u)$.

Theorem 2.4 provides a convergence result at interpolation points. Below, we provide a corollary to this theorem that provides a convergence result in $L^\infty(\mathcal{D})$. This result involves the Lebesgue constants in L^∞-norm associated to $\mathcal{I}_\Lambda$, defined by

$$\mathcal{L}_\Lambda = \sup_{v \in \mathcal{C}^0(\overline{\mathcal{D}})} \frac{\|\mathcal{I}_\Lambda(v)\|_\infty}{\|v\|_\infty}, \tag{10}$$

and such that for any $v \in \mathcal{C}^0(\overline{\mathcal{D}})$,

$$\|v - \mathcal{I}_\Lambda(v)\|_\infty \leq (1 + \mathcal{L}_\Lambda) \inf_{w \in V_\Lambda} \|v - w\|_\infty. \tag{11}$$

Throughout this article, we adopt the convention that supremum exclude elements with norm 0. We recall also that the L^∞ Lebesgue constant can be expressed as $\mathcal{L}_\Lambda = \sup_{x \in \overline{\mathcal{D}}} \sum_{\nu \in \Lambda} |l_\nu(x)|$.

Corollary 2.5 (Convergence in L^∞).

Assuming $(H2)$-$(H5)$, it holds

$$\limsup_{n\to\infty} \|\mathbb{E}\left[\tilde{u}^n - u\right]\|_\infty \leqslant \frac{C_1(\Delta t, \Lambda)}{1 - C(\Delta t, \Lambda)}\mathcal{L}_\Lambda + \|u - \mathcal{I}_\Lambda(u)\|_\infty. \tag{12}$$

Proof. By triangular inequality, we have

$$\|\mathbb{E}\left[\tilde{u}^n - u\right]\|_\infty \leqslant \|\mathbb{E}\left[\tilde{u}^n - \mathcal{I}_\Lambda(u)\right]\|_\infty + \|\mathcal{I}_\Lambda(u) - u\|_\infty.$$

We can build a continuous function w such that $w(x_\nu) = \mathbb{E}\left[\tilde{u}^n(x_\nu) - u(x_\nu)\right]$ for all $\nu \in \Lambda$, and such that

$$\|w\|_\infty = \sup_{\nu\in\Lambda} |w(x_\nu)| = \sup_{\nu\in\Lambda} |\mathbb{E}\left[\tilde{u}^n(x_\nu) - u(x_\nu)\right]|.$$

We have then

$$\|\mathbb{E}\left[\tilde{u}^n - \mathcal{I}_\Lambda(u)\right]\|_\infty = \|\mathcal{I}_\Lambda(w)\|_\infty \leq \mathcal{L}_\Lambda\|w\|_\infty.$$

The result follows from the definition of the function w and Theorem 2.4. ∎

Remark 2.6. *Since for bounded domains $\mathcal{D}$, we have*

$$\|v\|_2 \leq |\mathcal{D}|^{1/2}\|v\|_\infty,$$

for all v in $\mathcal{C}^0(\overline{\mathcal{D}})$, where $|\mathcal{D}|$ denotes the Lebesgue measure of $\mathcal{D}$, we can deduce the convergence results in L^2 norm from those in L^∞ norm.

3 Adaptive sparse interpolation

We here present sparse interpolation methods following [6, 7].

3.1 Sparse interpolation

For $1 \leq i \leq d$, we let $\{\varphi_k^{(i)}\}_{k\in\mathbb{N}_0}$ be a univariate polynomial basis, where $\varphi_k^{(i)}(x_i)$ is a polynomial of degree k. For a multi-index $\nu = (\nu_1, \ldots, \nu_d) \in \mathbb{N}_0^d$, we introduce the multivariate polynomial

$$\varphi_\nu(x) = \prod_{i=1}^d \varphi_{\nu_i}^{(i)}(x_i).$$

For a subset $\Lambda \subset \mathbb{N}^d$, we let $\mathcal{P}_\Lambda = \text{span}\{\varphi_\nu : \nu \in \Lambda\}$. A subset Λ is said to be *downward closed* if

$$\forall \nu \in \Lambda, \ \mu \leq \nu \Rightarrow \mu \in \Lambda.$$

If Λ is downward closed, then the polynomial space $\mathcal{P}_\Lambda$ does not depend on the choice of univariate polynomial bases and is such that $\mathcal{P}_\Lambda = \text{span}\{x^\nu : \nu \in \Lambda\}$, with $x^\nu = x_1^{\nu_1} \ldots x_d^{\nu_d}$.

In the case where $\mathcal{D} = \mathcal{D}_1 \times \ldots \times \mathcal{D}_d$, we can choose for $\{\varphi_k^{(i)}\}_{k \in \mathbb{N}_0}$ an orthonormal basis in $L^2(\mathcal{D}_i)$ (i.e. a rescaled and shifted Legendre basis). Then $\{\varphi_\nu\}_{\nu \in \mathbb{N}_0^d}$ is an orthonormal basis of $L^2(\mathcal{D})$. To define a set of points Γ_Λ unisolvent for $\mathcal{P}_\Lambda$, we can proceed as follows. For each dimension $1 \leq i \leq d$, we introduce a sequence of points $\{z_k^{(i)}\}_{k \in \mathbb{N}_0}$ in $\overline{\mathcal{D}}_i$ such that for any $p \geq 0$, $\Gamma_p^{(i)} = \{z_k^{(i)}\}_{k=0}^p$ is unisolvent for $\mathcal{P}_p = \text{span}\{\varphi_k^{(i)} : 0 \leq k \leq p\}$, therefore defining an interpolation operator $\mathcal{I}_p^{(i)}$. Then we let

$$\Gamma_\Lambda = \{z_\nu = (z_{\nu_1}^{(1)}, \ldots, z_{\nu_d}^{(d)}) : \nu \in \Lambda\} \subset \overline{\mathcal{D}}.$$

This construction is interesting for adaptive sparse algorithms since for an increasing sequence of subsets Λ_n, we obtain an increasing sequence of sets Γ_{Λ_n}, and the computation of the interpolation on $\mathcal{P}_{\Lambda_n}$ only requires the evaluation of the function on the new set of points $\Gamma_{\Lambda_n} \setminus \Gamma_{\Lambda_{n-1}}$. Also, with such a construction, we have the following property of the Lebesgue constant of $\mathcal{I}_\Lambda$ in L^∞-norm. This result is directly taken from [7, Section 3].

Proposition 3.1. *If for each dimension $1 \leq i \leq d$, the sequence of points $\{z_k^{(i)}\}_{k \in \mathbb{N}_0}$ is such that the interpolation operator $\mathcal{I}_p^{(i)}$ has a Lebesgue constant $\mathcal{L}_p \leq (p+1)^s$ for some $s > 0$, then for any downward closed set Λ, the Lebesgue constant $\mathcal{L}_\Lambda$ satisfies*

$$\mathcal{L}_\Lambda \leq (\#\Lambda)^{s+1}. \tag{13}$$

Leja points or magic points [21] are examples of sequences of points such that the interpolation operators $\mathcal{I}_p^{(i)}$ have Lebesgue constants not growing too fast with p. For a given Λ with $\rho_i := \max_{\nu \in \Lambda} \nu_i$, it is possible to construct univariate interpolation grids $\Gamma_{\rho_i}^{(i)}$ with better properties (e.g., Chebychev points), therefore resulting in better properties for the associated interpolation operator $\mathcal{I}_\Lambda$. However for Chebychev points, e.g., $\rho_i \leq \rho_i'$ does not ensure $\Gamma_{\rho_i}^{(i)} \subset \Gamma_{\rho_i'}^{(i)}$. Thus with such univariate grids, an increasing sequence of sets Λ_n will not be associated with an increasing sequence of sets Γ_{Λ_n}, and the evaluations of the function will not be completely recycled in adaptive algorithms. However, for some of the algorithms described in Section 4, this is not an issue as evaluations can not be recycled anyway.

Note that for general domains $\mathcal{D}$ which are not the product of intervals, the above constructions of grids Γ_Λ are not viable since it may yield to grids not contained in the domain $\mathcal{D}$. For such general domains, magic points obtained through greedy algorithms could be considered.

3.2 Adaptive algorithm for sparse interpolation

An adaptive sparse interpolation algorithm consists in constructing a sequence of approximations $(u_n)_{n \geq 1}$ associated with an increasing sequence of downward closed subsets $(\Lambda_n)_{n \geq 1}$. According to (11), we have to construct a sequence such that the best approximation error and the Lebesgue constant are such that

$$\mathcal{L}_{\Lambda_n} \inf_{w \in \mathcal{P}_{\Lambda_n}} \|u - w\|_\infty \longrightarrow 0 \text{ as } n \to \infty$$

8

for obtaining a convergent algorithm. For example, if

$$\inf_{w \in \mathcal{P}_{\Lambda_n}} \|u - w\|_\infty = O((\#\Lambda_n)^{-r}) \tag{14}$$

holds[‡] for some $r > 1$ and if $\mathcal{L}_{\Lambda_n} = O((\#\Lambda_n)^k)$ for $k < r$, then the error $\|u - u_n\|_\infty = O(n^{-r'})$ tends to zero with an algebraic rate of convergence $r' = r - k > 0$. Of course, the challenge is to propose a practical algorithm that constructs a good sequence of sets Λ_m.

We now present the adaptive sparse interpolation algorithm with bulk chasing procedure introduced in [6]. Let θ be a fixed bulk chasing parameter in $(0, 1)$ and let $\mathcal{E}_\Lambda(v) = \|P_\Lambda(v)\|_2^2$, where P_Λ is the orthogonal projector over $\mathcal{P}_\Lambda$ for any subset $\Lambda \subset \mathbb{N}_0^d$.

Algorithm 3.2. (Adaptive interpolation algorithm)

1: Set $\Lambda_1 = \{\mathbf{0}_d\}$ and $n = 1$.
2: **while** $n \leq N$ and $\varepsilon^{n-1} > \varepsilon$ **do**
3:    Compute $\mathcal{M}_{\Lambda_n}$.
4:    Set $\Lambda_n^\star = \Lambda_n \cup \mathcal{M}_{\Lambda_n}$ and compute $\mathcal{I}_{\Lambda_n^\star}(u)$.
5:    Select $N_n \subset \mathcal{M}_{\Lambda_n}$ the smallest such that $\mathcal{E}_{N_n}(\mathcal{I}_{\Lambda_n^\star}(u)) \geq \theta\mathcal{E}_{\mathcal{M}_{\Lambda_n}}(\mathcal{I}_{\Lambda_n^\star}(u))$
6:    Update $\Lambda_{n+1} = \Lambda_n \cup N_n$.
7:    Compute $u_{n+1} = \mathcal{I}_{\Lambda_{n+1}}(u)$ (this step is not necessary in practice).
8:    Compute ε^n.
9:    Update $n = n + 1$.
10: **end while**

At iteration n, Algorithm 3.2 selects a subset of multi-indices N_n in the *reduced margin* of Λ_n defined by

$$\mathcal{M}_{\Lambda_n} = \{\nu \in \mathbb{N}^d \setminus \Lambda_n : \forall j \text{ s.t. } \nu_j > 0, \ \nu - e_j \in \Lambda_n\},$$

where $(e_j)_k = \delta_{kj}$. The reduced margin is such that for any subset $S \subset \mathcal{M}_{\Lambda_n}$, $\Lambda_n \cup S$ is downward closed. This ensures that the sequence $(\Lambda_n)_{n \geq 1}$ generated by the algorithm is an increasing sequence of downward closed sets. Finally, Algorithm 3.2 is stopped using a criterion based on

$$\varepsilon^n = \frac{\mathcal{E}_{\mathcal{M}_n}(\mathcal{I}_{\Lambda_n^\star}(u))}{\mathcal{E}_{\Lambda_n^\star}(\mathcal{I}_{\Lambda_n^\star}(u))}.$$

4   Combining sparse adaptive interpolation with sequential control variates algorithm

We present in this section two ways of combining Algorithm 2.3 and Algorithm 3.2. First we introduce a perturbed version of Algorithm 3.2 and then an adaptive version of Algorithm 2.3. At the end of the section, numerical results will illustrate the behavior of the proposed algorithms.

[‡]see e.g. [9] for conditions on u ensuring such a behavior of the approximation error.

4.1 Perturbed version of Algorithm 3.2

As we do not have access to exact evaluations of the solution u of (1), Algorithm 3.2 can not be used for interpolating u. So we introduce a perturbed version of this algorithm, where the computation of the exact interpolant $\mathcal{I}_\Lambda(u)$ is replaced by an approximation denoted $\tilde{u}_\Lambda$, which can be computed for example with Algorithm 2.3 stopped for a given tolerance ϵ_{tol} or at step k. This brings the following algorithm.

Algorithm 4.1. (Perturbed adaptive sparse interpolation algorithm)

 1: Set $\Lambda_1 = \{\mathbf{0}_d\}$ and $n = 1$.
 2: **while** $n \leq N$ and $\tilde{\varepsilon}^{n-1} > \varepsilon$ **do**
 3:     Compute $\mathcal{M}_{\Lambda_n}$.
 4:     Set $\Lambda_n^\star = \Lambda_n \cup \mathcal{M}_{\Lambda_n}$ and compute $\tilde{u}_{\Lambda_n^\star}$.
 5:     Select N_n as the smallest subset of $\mathcal{M}_{\Lambda_n}$ such that $\mathcal{E}_{N_n}(\tilde{u}_{\Lambda_n^\star}) \geq \theta \mathcal{E}_{\mathcal{M}_{\Lambda_n}}(\tilde{u}_{\Lambda_n^\star})$
 6:     Update $\Lambda_{n+1} = \Lambda_n \cup N_n$.
 7:     Compute $\tilde{u}_{\Lambda_{n+1}}$.
 8:     Compute $\tilde{\varepsilon}^n$.
 9:     Update $n = n + 1$.
10: **end while**

4.2 Adaptive version of Algorithm 2.3

As a second algorithm, we consider the sequential control variates algorithm (Algorithm 2.3) where at step 4, an approximation $\tilde{e}^k$ of e^k is obtained by applying the adaptive interpolation algorithm (Algorithm 4.1) to the function $e^k_{\Delta t, M}$, which uses Monte-Carlo estimations $e^k_{\Delta t, M}(x_\nu)$ of $e^k(x_\nu)$ at interpolation points. At each iteration, $\tilde{e}^k$ therefore belongs to a different approximation space $\mathcal{P}_{\Lambda_k}$. In the numerical section, we will call this algorithm **adaptive Algorithm 2.3**.

4.3 Numerical results

In this section, we illustrate the behavior of algorithms previously introduced on different test cases. We consider the simple diffusion equation

$$\begin{aligned} -\triangle u(x) &= g(x), & x \in \mathcal{D}, \\ u(x) &= f(x), & x \in \partial\mathcal{D}, \end{aligned} \tag{15}$$

were $\mathcal{D} =]-1, 1[^d$. The source terms and boundary conditions will be specified later for each test case.

The stochastic differential equation associated to (15) is the following

$$dX_t^x = \sqrt{2}dW_t, \quad X_0^x = x, \tag{16}$$

where $(W_t)_{t \geq 0}$ is a d-dimensional Brownian motion.

We use tensorized grids of magic points for the selection of interpolation points evolved in adaptive

10

algorithms.

Small dimensional test case. We consider a first test case (TC1) in dimension $d = 5$. Here the source term and the boundary conditions in problem (15) are chosen such that the solution is given by

$$u(x) = x_1^2 + sin(x_2) + exp(x_3) + sin(x_4)(x_5 + 1), \qquad x \in \overline{\mathcal{D}}. \tag{TC1}$$
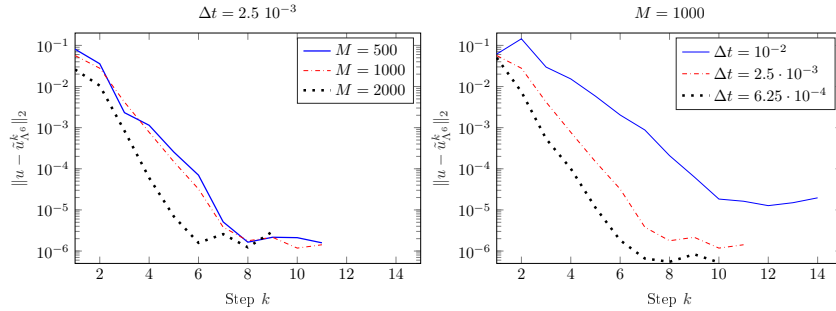

Figure 1: (TC1) Algorithm 2.3 for fixed Λ : evolution of $\|u - \tilde{u}_{\Lambda^6}^k\|$ with respect to k for various M (left figure), and various Δt (right figure).

We first test the influence of Δt and M on the convergence of Algorithm 2.3 when Λ is fixed. In that case, Λ is selected a priori with Algorithm 3.2 using samples of the exact solution u for (TC1), stopped for $\varepsilon \in \{10^{-6}, 10^{-8}, 10^{-10}\}$. In what follows, the notation Λ^i stands for the set obtained for $\varepsilon = 10^{-i}, i \in \{6, 8, 10\}$. We represent on Figure 1 the evolution of the absolute error in L^2-norm (similar results hold for the L^∞-norm) between the approximation and the true solution with respect to step k for $\Lambda = \Lambda^6$. As claimed in Corollary 1, we recover the geometric convergence up to a threshold value that depends on Δt. We also notice faster convergence as M increases and when Δt decreases. We fix $M = 1000$ in the next simulations.
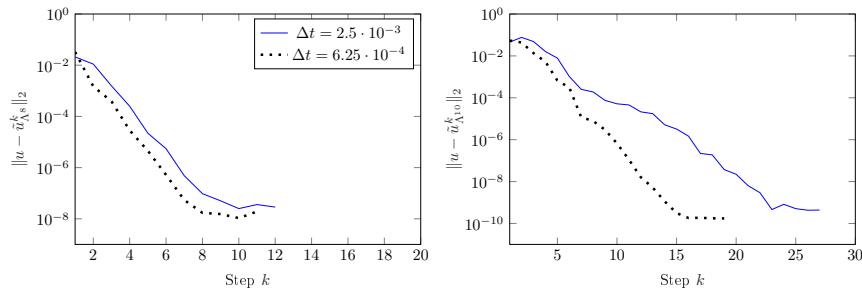

Figure 2: (TC1) Algorithm 2.3 for fixed Λ^i: evolution of $\|u - \tilde{u}_{\Lambda^i}^k\|_2$ with respect to k for $i = 8$ (left figure), and $i = 10$ (right figure).

We study the impact of the choice of Λ^i on the convergence of Algorithm 2.3. Again we observe on Figure 2 that the convergence rate gets better as Δt decreases. Moreover as $\#\Lambda$ increases the

threshold value decreases. This is justified by the fact that interpolation error decreases as $\#\Lambda^i$ increases (see Table 1). Nevertheless, we observe that it may also deteriorate the convergence rate if it is chosen too large together with Δt not sufficiently small. Indeed for the same number of iterations $k = 10$ and the same time-step $\Delta t = 2.5 \cdot 10^{-3}$, we have an approximate absolute error equal to 10^{-7} for Λ^8 against 10^{-4} for Λ^{10}.

Λ_n	$\#\Lambda_n$	ε^n	$||u - u_n||_2$	$||u - u_n||_\infty$
	1	6.183372e-01	1.261601e+00	4.213566e+00
	10	2.792486e-02	1.204421e-01	3.602629e-01
	20	2.178450e-05	9.394419e-04	3.393999e-03
Λ^6	26	9.632815e-07	4.270457e-06	1.585129e-05
	30	9.699704e-08	2.447475e-06	8.316435e-06
Λ^8	33	4.114730e-09	2.189518e-08	9.880306e-08
	40	1.936050e-10	6.135776e-10	1.739848e-09
Λ^{10}	41	1.008412e-11	9.535433e-11	4.781375e-10
	50	1.900248e-14	1.004230e-13	4.223288e-13
	55	7.453467e-15	2.905404e-14	1.254552e-13

Table 1: Algorithm 3.2 computed on the exact solution of (TC1): evolution of $\#\Lambda_n$, error criterion ε^n and interpolation errors in norms L^2 and L^∞ at each step n.

We present now the behavior of Algorithm 4.1. Simulations are performed with a bulk-chasing parameter $\theta = 0.5$. At each step n of Algorithm 4.1, we use Algorithm 2.3 with $(\Delta t, M) = (10^{-4}, 1000)$, stopped when a stagnation is detected. As shown on the left plot of Figure 3, for $\#\Lambda_n = 55$ we reach approximately a precision of 10^{-14} as for Algorithm 3.2 performed on the exact solution (see Table 1). According to the right plot of Figure 3, we also observe that the enrichment procedure behaves similarly for both algorithms ($\tilde{\varepsilon}^n$ and ε^n are almost the same). Here using the approximation provided by Algorithm 2.3 has a low impact on the behavior of Algorithm 3.2.
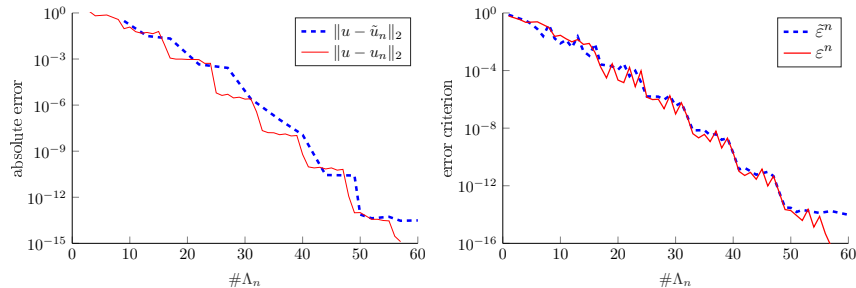

Figure 3: (TC1) Comparison of Algorithm 3.2 applied to exact solution and Algorithm 4.1 : (left) absolute error in L^2-norm (right) evolution of ε^n and $\tilde{\varepsilon}^n$ with respect to $\#\Lambda_n$.

We present then results provided with the adaptive Algorithm 2.3. The parameters chosen for the adaptive interpolation are $\varepsilon = 5 \cdot 10^{-2}$, $\theta = 0.5$. $K = 30$ ensures the stopping of Algorithm 2.3. As illustrated by Figure 4, we recover globally the same behavior as for Algorithm 2.3 without adaptive interpolation. Indeed as k increases, both absolute errors in L^2-norm and L^∞-norm decrease and then stagnate. Again, we notice the influence of Δt on the stagnation level. Nevertheless, the convergence rates are deteriorated and the algorithm provides less accurate approximations than Algorithm 4.1. This might be due to the sparse adaptive interpolation procedure, which uses here pointwise evaluations based on Monte-Carlo estimates, unlike Algorithm 4.1 which relies on pointwise evaluations resulting from Algorithm 2.3 stopping for a given tolerance.
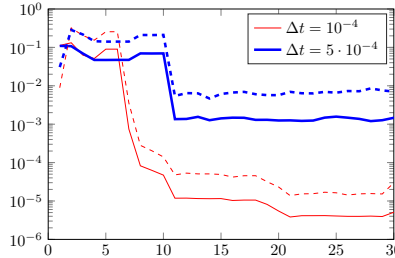

Figure 4: (TC1) Adaptive Algorithm 2.3: evolution of $\|u - u_{\Lambda_k}^k\|_2$ (continuous line) and $\|u - u_{\Lambda_k}^k\|_\infty$ (dashed line) with respect to step k and Δt.

Finally in Table 2, we compare the algorithmic complexity of these algorithms to reach a precision of $3 \cdot 10^{-5}$ for $(\Delta t, M) = (10^{-4}, 1000)$. For adaptive Algorithm 2.3, Λ_k refers to the set of multi-indices considered at step k of Algorithm 2.3. For Algorithm 4.1, N_n stands for the number of iteration required by Algorithm 2.3 to reach tolerance ε_{tol} at step n. Finally, Algorithm 2.3 is run with full-grid $\Lambda = \Lambda_{max}$ where $\Lambda_{max} = \{\nu \in \mathbb{N}^d \ : \ \nu_i \leq 10\}$ is the set of multi-indices allowing to reach the machine precision. In this case, N stands for the number of steps for this algorithm to converge.

	Adaptive Algorithm 2.3	Algorithm 4.1	Full-grid Algorithm 2.3
Th. Complexity	$M(\Delta t)^{-1}(\sum_k \#\Lambda_k)$	$M(\Delta t)^{-1}(\sum_n \#\Lambda_n N_n)$	$M(\Delta t)^{-1}\#\Lambda_{max}N$
Est. Complexity	$4 \cdot 10^9$ operations	$16 \cdot 10^9$ operations	$10^{12}N$ operations

Table 2: (TC1) Comparison of the algorithmic complexity to reach the precision $3 \cdot 10^{-5}$, with $(\Delta t, M) = (10^{-4}, 1000)$.

We observe that both the adaptive version of Algorithm 2.3 and Algorithm 4.1 have a similar complexity, which is better than for the full-grid version of Algorithm 2.3. Moreover, we observed that while adaptive version of Algorithm 2.3 stagnates at a precision of $3 \cdot 10^{-5}$, Algorithm 4.1, with the same parameters Δt and M, converges almost up to the machine precision. This is why the high-dimensional test cases will be run only with Algorithm 4.1.

13

Higher-dimensional test cases. Now, we consider two other test cases noted respectively (TC2) and (TC3) in higher dimension.

(TC2) As second test case in dimension $d = 10$, we define (15) such that its solution is the Henon-Heiles potential

$$u(x) = \frac{1}{2}\sum_{i=1}^{d} x_i^2 + 0.2\sum_{i=1}^{d-1}\left(x_i x_{i+1}^2 - x_i^3\right) + 2.5\ 10^{-3}\sum_{i=1}^{d-1}\left(x_i^2 + x_{i+1}^2\right)^2, \qquad x \in \overline{\mathcal{D}}.$$

We set $(\Delta t, M) = (10^{-4}, 1000)$ and $K = 30$ for Algorithm 2.3.

(TC3) We also consider the problem (15) whose exact solution is a sum of non-polynomial functions, like (TC1) but now in dimension $d = 20$, given by

$$u(x) = x_1^2 + sin(x_{12}) + exp(x_5) + sin(x_{15})(x_8 + 1).$$

Here, the Monte-Carlo simulations are performed for $(\Delta t, M) = (10^{-4}, 1000)$ and $K = 30$.

Since for both test cases the exact solution is known, we propose to compare the behavior of Algorithm 4.1 and Algorithm 3.2. Again, the approximations $\tilde{u}_n$, at each step n of Algorithm 4.1, are provided by Algorithm 2.3 stopped when a stagnation is detected. In both cases, the parameters for Algorithm 4.1 are set to $\theta = 0.5$ and $\varepsilon = 10^{-15}$.

In Table 3 and Table 4, we summarize the results associated to the exact and perturbed sparse adaptive algorithms for (TC2) and (TC3) respectively. We observe that Algorithm 4.1 performs well in comparison to Algorithm 3.2, for (TC2). Indeed, we get an approximation with a precision below the prescribed value ε for both algorithms.

$\#\Lambda_n$	ε^n	$\|u-u_n\|_\infty$	$\|u-u_n\|_2$	$\#\Lambda_n$	$\tilde{\varepsilon}^n$	$\|u-\tilde{u}_{\Lambda_n}\|_\infty$	$\|u-\tilde{u}_{\Lambda_n}\|_2$
1	4.0523e-01	3.0151e+00	1.2094e+00	1	3.9118e-01	8.3958e-01	6.9168e-01
17	1.6243e-01	1.8876e+00	5.9579e-01	17	1.6259e-01	5.2498e-01	3.4420e-01
36	5.4494e-02	7.0219e-01	2.0016e-01	36	5.4699e-02	1.9209e-01	1.2594e-01
46	1.2767e-02	1.6715e-01	4.9736e-02	46	1.2806e-02	4.6904e-02	2.8524e-02
53	9.6987e-04	2.9343e-02	4.8820e-03	53	1.0350e-03	7.8754e-03	2.8960e-03
60	7.6753e-04	1.5475e-02	4.1979e-03	61	7.0354e-04	3.0365e-03	1.7610e-03
71	3.2532e-04	8.4575e-03	2.1450e-03	71	3.1998e-04	2.3486e-03	1.2395e-03
77	1.7434e-16	3.9968e-15	1.5784e-15	77	7.3621e-16	6.2172e-15	1.2874e-15

Table 3: (TC2) Comparison of Algorithm 3.2 (first four columns) and Algorithm 4.1 (last four columns).

Similar observation holds for (TC3) in Table 4 and this despite the fact that the test case involves higher dimensional problem.

$\#\Lambda_n$	ε^n	$\|u - u_n\|_\infty$	$\|u - u_n\|_2$	$\#\Lambda_n$	$\tilde{\varepsilon}^n$	$\|u - \tilde{u}_{\Lambda_n}\|_\infty$	$\|u - \tilde{u}_{\Lambda_n}\|_2$
1	7.0155e-01	3.9361e+00	1.2194e+00	1	5.5582e-01	7.2832e-01	7.0771e-01
6	1.4749e-01	2.2705e+00	5.4886e-01	6	7.4253e-02	2.7579e-01	5.1539e-01
11	2.1902e-02	2.8669e-01	1.0829e-01	11	1.4929e-02	4.4614e-02	4.1973e-02
15	7.6086e-03	1.6425e-01	4.7394e-02	15	1.2916e-02	1.5567e-02	2.5650e-02
20	2.2275e-04	2.7715e-03	7.2230e-04	20	3.4446e-04	5.6927e-04	5.3597e-04
24	1.4581e-05	1.5564e-04	7.5314e-05	24	1.6036e-05	2.5952e-05	3.0835e-05
30	1.8263e-06	8.0838e-06	2.1924e-06	30	9.0141e-07	2.8808e-06	1.9451e-06
35	3.9219e-09	8.9815e-08	2.4651e-08	35	8.1962e-09	2.1927e-08	1.5127e-08
40	1.7933e-10	2.0152e-09	6.9097e-10	40	1.6755e-10	2.8455e-10	2.6952e-10
45	5.0775e-12	2.4783e-10	4.1600e-11	45	1.4627e-11	3.3188e-11	1.7911e-11
49	1.7722e-14	4.6274e-13	8.5980e-14	49	1.7938e-14	8.6362e-14	5.0992e-14
54	3.9609e-15	2.2681e-13	3.1952e-14	54	3.2195e-15	4.8142e-14	2.6617e-14
56	4.5746e-16	8.4376e-15	3.0438e-15	56	8.2539e-16	8.4376e-15	6.3039e-15

Table 4: (TC3) Comparison of Algorithm 3.2 (first four columns) and Algorithm 4.1 (last four columns).

5   Conclusion

In this paper we have introduced a probabilistic approach to approximate the solution of high-dimensional elliptic PDEs. This approach relies on adaptive sparse polynomial interpolation using pointwise evaluations of the solution estimated using a Monte-Carlo method with control variates. Especially, we have proposed and compared different algorithms. First we proposed Algorithm 2.3 which combines the sequential algorithm proposed in [14] and sparse interpolation. For the non-adaptive version of this algorithm we recover the convergence up to a threshold as the original sequential algorithm [15]. Nevertheless it remains limited to small-dimensional test cases, since its algorithmic complexity remains high. Hence, for practical use, the adaptive Algorithm 2.3 should be preferred. Adaptive Algorithm 2.3 converges but it does not allow to reach low precision with reasonable number of Monte-Carlo samples or time-step in the Euler-Maruyama scheme. Secondly, we proposed Algorithm 4.1. It is a perturbed sparse adaptive interpolation algorithm relying on inexact pointwise evaluations of the function to approximate. Numerical experiments have shown that the perturbed algorithm (Algorithm 4.1) performs well in comparison to the ideal one (Algorithm 3.2) and better than the adapted Algorithm 2.3 with a similar algorithmic complexity. Here, since only heuristic tools have been provided to justify the convergence of this algorithm, the proof of its convergence, under assumptions on the class of functions to be approximated, should be addressed in a future work.

References

[1] M. Bachmayr, R. Schneider & A. Uschmajew. Tensor Networks and Hierarchical Tensors for the Solution of High-Dimensional Partial Differential Equations. *Found Comput Math.* 2016, vol. 16, no 6, p. 1423-1472.

[2] M. Barrault, Y. Maday, N.C. Nguyen & A.T. Patera. An "empirical interpolation" method : application to efficient reduced-basis discretization of partial differential equations, *Comptes Rendus Mathématique*, 2004, vol. 339, no 9, p. 667-672.

[3] C. Beck, E. Weinan & A. Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *arXiv preprint arXiv:1709.05963*, 2017.

[4] C. Beck, S. Becker, P. Grohs, N. Jaafari & A. Jentzen. Solving stochastic differential equations and Kolmogorov equations by means of deep learning. *arXiv preprint arXiv:1806.00421*, 2018.

[5] H.-J. Bungartz & M. Griebel. Sparse grids. *Acta numerica.* 2004, vol. 13, p. 147-269.

[6] A. Chkifa, A. Cohen & R. DeVore. Sparse adaptive Taylor approximation algorithms for parametric and stochastic elliptic PDEs. *ESAIM: Mathematical Modelling and Numerical Analysis*, 2013, vol. 47, no 1, p. 253-280.

[7] A. Chkifa, A. Cohen & C. Schwab. High-dimensional adaptive sparse polynomial interpolation and applications to parametric PDEs. *Found. Comput. Math.*, 2014, vol. 14 pp. 601–633.

[8] A. Cohen, W. Dahmen & R. DeVore. Adaptive wavelet methods for elliptic operator equations: convergence rates. *Mathematics of Computation*, 2001, vol. 70, no 233, p. 27-75.

[9] A. Cohen & R. DeVore. Approximation of high-dimensional parametric PDEs. *Acta Numerica*, 2015, vol. 24, p. 1-159.

[10] A. Cohen & G. Migliorati. Multivariate approximation in downward closed polynomial spaces. *Contemporary Computational Mathematics-A Celebration of the 80th Birthday of Ian Sloan.* Springer, Cham, 2018. p. 233-282.

[11] F. Comets & T. Meyre. *Calcul stochastique et modèles de diffusions-2ème éd. Dunod*, 2015.

[12] A. Friedman. *Stochastic differential equations and applications.* Academic Press, New York, 1975.

[13] M. B. Giles. Multilevel monte carlo methods. *Monte Carlo and Quasi-Monte Carlo Methods 2012.* Springer, Berlin, Heidelberg, 2013. p. 83-103.

[14] E. Gobet & S. Maire. A spectral Monte Carlo method for the Poisson equation. *Monte Carlo Methods and Applications mcma*, 2004, vol. 10, no 3-4, p. 275-285.

[15] E. Gobet & S. Maire. Sequential control variates for functionals of Markov processes. *SIAM Journal on Numerical Analysis*, 2005, vol. 43, no 3, p. 1256-1275.

[16] E. Gobet & S. Menozzi. Stopped diffusion processes: boundary corrections and overshoot. *Stochastic Processes and Their Applications*, 2010, vol. 120, no 2, p. 130-162.

[17] E. Gobet. *Monte-Carlo methods and stochastic processes: from linear to non-linear.* Chapman and Hall/CRC, 2016.

[18] L. Grasedyck, D. Kressner & C. Tobler. A literature survey of low- rank tensor approximation techniques. *GAMM-Mitteilungen.*, 2013, vol. 36, no 1, p. 53-78.

[19] W. Hackbusch. Numerical tensor calculus, *Acta numerica*, 2014, vol. 23, p. 651-742.

[20] P. Kloeden & E. Platen. Numerical solution of stochastic differential equations. *Springer Science & Business Media* , 2013.

[21] Y. Maday, N.C. Nguyen, A.T. Patera, et al. A general, multipurpose interpolation procedure: the magic points. *Communications on Pure and Applied Analysis*, 2009, vol.8 no1 p. 383-404.

[22] F. Nobile, L. Tamellini, F. Tesei & R. Tempone. An Adaptive Sparse Grid Algorithm for Elliptic PDEs with Lognormal Diffusion Coefficient; Sparse Grids and Applications. *Sparse Grids and Applications-Stuttgart 2014. Springer, Cham* 2016, p. 191-220.

[23] A. Nouy. Low-Rank Methods for High-Dimensional Approximation and Model Order Reduction. *Model Reduction and Approximation, Chapter 4.* 2017.

[24] I. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.* 2011, vol. 33, no 5, p. 2295-2317.

[25] C. Schwab & C. J. Gittelson. Sparse tensor discretizations of high-dimensional parametric and stochastic PDEs. *Acta Numerica.*, 2011, vol. 20, p. 291-467.

[26] J. Shen & H. Yu. Efficient Spectral Sparse Grid Methods and Applications to High Dimensional Elliptic Problems. *SIAM J. Sci. Comput.* 2010, vol. 32, no 6, p. 3228-3250.

[27] E. Weinan, H. Jiequn & A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 2017, vol. 5, no 4, p. 349-380.