A new sequential algorithm for L2-approximation and application to Monte-Carlo integration

Emmanuel Gobet, Khushboo Surana

A new sequential algorithm for L_2-approximation and application to Monte-Carlo integration[☆]

Emmanuel Gobet[1,α⋆], Khushboo Surana[α⋆α⋆]

Abstract

We design a new stochastic algorithm (called SALT) that sequentially approximates a given function in L_2 w.r.t. a probability measure, using a finite sample of the distribution. By increasing the sets of approximating functions and the simulation effort, we compute a L_2-approximation with higher and higher accuracy. The simulation effort is tuned in a robust way that ensures the convergence under rather general conditions. Then, we apply SALT to build efficient control variates for accurate numerical integration. Examples and numerical experiments support the mathematical analysis.

Keywords: Sequential approximation, Monte-Carlo simulations, Stochastic algorithms, Spectral decomposition
2010 MSC: 65C05, 62Lxx, 74S25

1. Introduction

1.1. The problem

Given a d-dimensional random variable Y and a measurable function $f : \mathbb{R}^d \mapsto \mathbb{R}$ such that $\mathbb{E}(f^2(Y)) < +\infty$, we aim at computing the L_2 decom-

[☆]The first author research is part of the Chair *Financial Risks* of the *Risk Foundation* and the *Finance for Energy Market Research Centre*. This work has been partly done during the visit of the second author at Ecole Polytechnique during the spring 2013, with the support of International Exchange Programs of Ecole Polytechnique.

[α⋆] Centre de Mathématiques Appliquées, Ecole Polytechnique and CNRS, Route de Saclay, 91128 Palaiseau Cedex, France
[α⋆α⋆] Indian Institute of Technology, Kanpur, Kalyanpur, U.P., India
Email addresses: emmanuel.gobet@polytechnique.edu (Emmanuel Gobet), khushboosurana21@gmail.com (Khushboo Surana)
[1] Corresponding author

position of f on the vector space spanned by the functions $(\phi_k)_{k\geq 0}$ according to the law induced by Y, i.e. finding coefficients $(\alpha_k^\star)_{k\geq 0}$ (from now on, we assume they exist) such that

$$f(Y) = \sum_{k\geq 0} \alpha_k^\star \phi_k(Y) \quad \text{in } L_2 \tag{1.1}$$

(assuming that $\mathbb{E}(\phi_k^2(Y)) < +\infty$ for any k). To achieve this decomposition, we use a finite number of independent simulations of Y, say $(Y^m)_{1\leq m\leq M}$, which serve to build a stochastic algorithm to compute $(\alpha_k^\star)_{k\geq 0}$. We aim at designing a scheme with robust convergence properties (robust w.r.t. f and the distribution of Y). Replacing f by an approximation on the basis functions is useful for instance in applications where the evaluation of f is particularly costly (e.g. the output of a complex computer program) and for some reasons, in a further step, we need to evaluate it many times which justifies the use of simpler and cheaper representations. The main application we develop below is the Monte-Carlo computation of $\mathbb{E}(f(Y))$ (numerical integration) using the approximative decomposition (1.1) as control variating. Anther potential field of application is Uncertainty Quantification [16] where the emulation of f serves to build statistics about the uncertainty $f(Y)$.

Throughout the paper, we assume that $(\phi_k)_{k\geq 0}$ are orthonormalized basis functions in the following sense:

(H1) $\mathbb{E}\left(\phi_k(Y)\phi_l(Y)\right) = \delta_{k,l}$, for any $k, l \geq 0$.

Therefore from (1.1), $\alpha_k^\star = \mathbb{E}(f(Y)\phi_k(Y))$ and a naive Monte-Carlo estimator is

$$\alpha_k^M = \frac{1}{M}\sum_{m=1}^{M} f(Y^m)\phi_k(Y^m). \tag{1.2}$$

Since $f(Y)\phi_k(Y)$ is integrable (because each factor is square integrable), the strong law of large numbers yields the *a.s.*-convergence of α_k^M towards $\alpha_k^\star$ for every k, as $M \to +\infty$. In other words

$$\sum_{k\geq 0}^{K} \alpha_k^M \phi_k(.) \xrightarrow[M\to+\infty]{a.s.} \sum_{k\geq 0}^{K} \alpha_k^\star \phi_k(.) \xrightarrow[K\to+\infty]{L_2(\mathbb{P}\circ Y^{-1})} f(.).$$

Nevertheless, the global L_2-convergence of the left hand side to the right one is a non-trivial issue since:

i) there is no guarantee that each coefficient α_k^M belongs to L_2 (except under stronger assumptions ensuring that $f(Y)\phi_k(Y)$ is square-integrable);

ii) as usually in statistics with the analysis of the bias-variance trade-off, we have to appropriately tune the joint convergence of K and M to infinity, in order to let both estimation and statistical errors converge to 0.

In this work we design a sequential algorithm which n-th step provides an estimation of the coefficients $(\alpha_k^\star)_{0\leq k\leq K_n}$ with non-decreasing order K_n. Knowing the values of the former step coefficients helps in improving the evaluation at the next step: the algorithm takes the form of a sequential learning algorithm that we call **SALT** (Sequential Approximation in L-Two). The convergence is studied in L_2: therefore, in view of the item i) above, we strengthen the hypothesis on the functions basis by assuming

(H2) for any $k \geq 0$, there is a constant c_k such[2] that $|\phi_k(Y)| \leq c_k < +\infty$ *a.s.* .

1.2. Literature background and applications

In the last fifteen years, several important algorithms based on Monte-Carlo L_2-approximations have been developed, they are popular owing to the flexibility of the Monte-Carlo simulations. Among them, we mention empirical regression methods that are now a standard approach to solve optimal stopping problems [18, 10, 7], Backward Stochastic Differential Equations [13, 17, 15]... They are several attempts to improve numerically L_2-projections, in order to reduce statistical inaccuracies: see for instance [5, 4]. Our current work is in this vein. However, a major difference is that our algorithm SALT works sequentially, in several steps, giving at each step a valuable approximation, which accuracy increases step after step: the computational effort increases coherently according to the number of steps, so that the algorithm can be stopped at any time to produce a solution, expected to be reasonably accurate given the computational effort. For the computation of $\mathbb{E}(f(Y))$ with smooth functions f, we show in Section 3 that our algorithm achieves optimal convergence rates (about optimal numerical integration, see [1] or [2] and references therein).

[2]observe that $c_k \geq 1$ since $\mathbb{E}(\phi_k^2(Y)) = 1$; in some cases, c_k does not depend on k, see Section 3.

Some ideas related to the iteration of our algorithm are exposed in [19, 20] where iterative control variates are built to speed-up Monte-Carlo integration methods (extensions to stochastic processes are designed in [14, 12]). But in [19, 20], essentially only the numerical integration error (i.e. about computing the expectation) is investigated and the number of control variates is fixed. Here, we consider the full L_2-approximation and we aim at achieving an infinite number of basis functions, with a robust strategy ensuring the convergence in a rather general setting.

1.3. Organization of the paper

In Section 2, we present the algorithm and state the main convergence results. An application to numerical integration (computation of $\mathbb{E}(f(Y))$) is then developed. In Section 3, examples of basis functions are discussed, with explicit convergence rates in the case of smooth functions f and random variables Y taking values in a cube. Despite these specific examples, we argue that our algorithm applies to more general settings. These examples show that we achieve rate-optimality in the computation of $\mathbb{E}(f(Y))$. Section 4 is devoted to numerical tests showing the performance of the algorithm SALT, in particular for the Monte-Carlo evaluation of $\mathbb{E}(f(Y))$. Intermediate results are proved in Appendix.

Extra notations.

- The residual at order $K \geq 0$ of the approximation of f on the basis functions is defined by

$$r_K(y) := f(y) - \sum_{k=0}^{K} \alpha_k^\star \phi_k(y). \tag{1.3}$$

  The identity (1.1) means

$$\mathbb{E}(r_K^2(Y)) = \sum_{k=K+1}^{+\infty} [\alpha_k^\star]^2 \to 0$$

  as $K \to +\infty$. The more appropriate the functions basis, the faster the convergence rate of the truncation error $\mathbb{E}(r_K^2(Y))$. Examples are given later.

- The notation $A \leq_c B$ means $A \leq CB$ for a generic constant $C > 0$ possibly changing from line to line, independent of the algorithm step number n. Similarly, $A \geqslant_c B$ means $A \leq_c B$ and $B \leq_c A$.

2. Algorithm SALT

2.1. Heuristics

The principle of our algorithm is to take advantage of the coefficients computed at steps $i = 0, \dots, n-1$ to speed-up those at step n, like a sequential learning algorithm.

We first expose the intuition of the algorithm. Consider that (non exact) coefficients $(\alpha_0, \dots, \alpha_K)$ have been already computed, then write (owing to **(H1)**)

$$\alpha_k^\star := \alpha_k + \beta_k \quad \text{with} \quad \beta_k := \mathbb{E}\left(\left[f(Y) - \sum_{l=0}^{K} \alpha_l \phi_l(Y)\right]\phi_k(Y)\right), \quad 0 \le k \le K,$$

and replace the expectation by an empirical mean over M independent simulations of Y: this provides an approximated correction to the pre-computed value α_k. The variance of this estimator is proportional to $1/M$ and to the variance of $Z_{k,K} := (f(Y) - \sum_{l=0}^{K} \alpha_l \phi_l(Y))\phi_k(Y)$. In particular, if $\sum_{l=0}^{K} \alpha_l \phi_l(.)$ approximates well $f(.)$, the random variable $Z_{k,K}$ and its variance are expected to be relatively small. Consequently the new estimation of $\alpha_k^\star$ will be particularly accurate, and likely much more accurate compared to the previous step. Iterating the procedure and incorporating more and more coefficients to compute, we obtain the algorithm SALT described below.

2.2. Detailed algorithm

Let $(K_n)_{n\ge1}$ be a non-decreasing sequence of integers, related to the number of basis functions used at each algorithm step. At step n, we compute the coefficients for the indices $k = 0, \dots, K_n$ using M_n simulations: the k-th coefficient is to be denoted by α_k^{n,M_n}.

The algorithm is initialized at $n = 0$ with $\alpha_k^{n,M_n} = 0$ for any k. The step $n \ge 1$ works as follows.

(Step n)-i) Generate a sample $(Y^{n,m})_{1\le m\le M_n}$ of M_n independent r.v. with same distribution as Y, and independent of other simulations.

(Step n)-ii) Define the corrections $(\beta_k^{n,M_n})_k$ and the coefficients $(\alpha_k^{n,M_n})_k$ as follows:

$$\begin{cases} \beta_k^{n,M_n} := \dfrac{1}{M_n}\displaystyle\sum_{m=1}^{M_n}\Big(f(Y^{n,m}) - \sum_{l=0}^{K_{n-1}} \alpha_l^{n-1,M_{n-1}}\phi_l(Y^{n,m})\Big)\phi_k(Y^{n,m}) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } 0 \le k \le K_n, \\ \alpha_k^{n,M_n} := \alpha_k^{n-1,M_{n-1}} + \beta_k^{n,M_n} \qquad\qquad \text{for } 0 \le k \le K_n, \\ \beta_k^{n,M_n} := \alpha_k^{n,M_n} := 0 \qquad\qquad\qquad\qquad \text{for } k > K_n. \end{cases}$$

2.3. Convergence results

For $n \ge 1$, denote by $\mathbb{E}^{n-1}(\cdot)$ and $\mathbb{V}\mathrm{ar}^{n-1}(\cdot)$ the expectation and variance conditionally to the sigma-field generated by the random variables $(Y^{i,m} : 1 \le m \le M_i, i \le n-1)$. To justify the algorithm convergence, we preliminarily analyse the conditional mean and variance of the coefficients. The proof of the above lemma is postponed to Appendix.

Lemma 2.1. *Let $n \ge 1$ and $0 \le k \le K_n$. We have*

$$\begin{aligned} \mathbb{E}^{n-1}(\alpha_k^{n,M_n}) &= \alpha_k^\star, \\ \mathbb{V}\mathrm{ar}^{n-1}(\alpha_k^{n,M_n}) &\le \frac{c_k^2}{M_n}\Big(\mathbb{E}[r_{K_{n-1}}^2(Y)] + \sum_{l=0}^{K_{n-1}}(\alpha_l^{n-1,M_{n-1}} - \alpha_l^\star)^2\Big). \end{aligned}$$

At step n of the algorithm, the function $f(.)$ is approximated by

$$f_n(.) := \sum_{k=0}^{K_n} \alpha_k^{n,M_n}\phi_k(.) \tag{2.4}$$

and the resulting quadratic error is given by

$$\mathcal{E}_n := \mathbb{E}\left(\big[f_n(Y) - f(Y)\big]^2\right) = \mathbb{E}(r_{K_n}^2(Y)) + \sum_{k=0}^{K_n}\mathbb{E}\left((\alpha_k^{n,M_n} - \alpha_k^\star)^2\right). \tag{2.5}$$

Note that $\mathcal{E}_0 = \mathbb{E}(f^2(Y))$. From Lemma 2.1, α_k^{n,M_n} estimates $\alpha_k^\star$ without bias and the conditional expectation of α_k^{n,M_n} is constant: this implies

$$\mathbb{E}\left((\alpha_k^{n,M_n} - \alpha_k^\star)^2\right) = \mathbb{V}\mathrm{ar}(\alpha_k^{n,M_n}) = \mathbb{E}\left(\mathbb{V}\mathrm{ar}^{n-1}(\alpha_k^{n,M_n})\right) \le \frac{c_k^2}{M_n}\mathcal{E}_{n-1}. \tag{2.6}$$

Combining (2.5) and (2.6), we establish

Theorem 2.1 (Error propagation along algorithm steps). *For any $n \geq 1$, we have*

$$\mathcal{E}_n \leq \mathbb{E}(r_{K_n}^2(Y)) + \frac{1}{M_n}\left[\sum_{k=0}^{K_n} c_k^2\right]\mathcal{E}_{n-1}.$$

In case there exists an order $K^\star < +\infty$ such that $r_{K^\star}(.) \equiv 0$ (i.e. f can be linearly represented by a finite number of $(\phi_k)_k$), it is enough to take $K_n = K^\star$ to directly obtain

Corollary 2.1 (Finite-dimensional approximation). *In the case of finite $K^\star$, we have*

$$\mathcal{E}_n \leq \frac{1}{M_n}\left[\sum_{k=0}^{K^\star} c_k^2\right]\mathcal{E}_{n-1}.$$

In particular, choosing a simulation effort rate which is constant and large enough (in the sense $M_n = M$ with $\rho := \frac{1}{M}\sum_{k=0}^{K^\star} c_k^2 < 1$), the convergence to 0 is geometric along steps, at rate ρ:

$$\mathcal{E}_n \leq \rho^n \mathbb{E}(f^2(Y)), \quad n \geq 0.$$

The global simulation effort after n steps being $\mathcal{C} = Mn$, the error is decreasing at most like $\exp(-\alpha\mathcal{C})$ for some $\alpha > 0$: therefore, when the L_2-approximation problem is only finite-dimensional, the convergence is exponentially fast w.r.t. the computational effort. This is fully different from the naive Monte-Carlo procedure. Similar features have been reported in some Sequential Monte-Carlo algorithms, see [6, 3]. We do not elaborate further in this direction since the focus of this work is rather on the general case $K^* = +\infty$.

The next result easily follows from Theorem 2.1, we leave the proof to the reader.

Corollary 2.2 (Infinite-dimensional approximation, general convergence). *The choice $M_n := \lceil 2\sum_{k=0}^{K_n} c_k^2 \rceil$ yields*

$$\mathcal{E}_n \leq \mathbb{E}(r_{K_n}^2(Y)) + \frac{1}{2}\mathcal{E}_{n-1}, \tag{2.7}$$

$$\mathcal{E}_n \leq \sum_{k=1}^{n} 2^{k-n}\mathbb{E}(r_{K_k}^2(Y)) + 2^{-n}\mathbb{E}(f^2(Y)). \tag{2.8}$$

Thus, for any choice of non-decreasing unbounded sequence $(K_n)_n$ (in particular $K_n \to +\infty$ as $n \to +\infty$), we have $\mathbb{E}(r_{K_n}^2(Y)) \to 0$ and thus $\mathcal{E}_n \to 0$.

This specification of M_n is very simple and depends only on the a priori knowledge of $(c_k)_k$ and the choice of the sequence $(K_n)_n$: this yields robustness concerning the tuning of the local simulation effort.

2.4. Application to integration

We now turn to the important application to the computation of $\mathbb{E}(f(Y))$. For this, assume that $\phi_0 \equiv 1$: in view of **(H1)**, it implies that $\phi_k(Y)$ is a centered r.v. for $k \geq 1$, and $\alpha_0^\star = \mathbb{E}(f(Y))$.

Algorithm 1. A first possibility is to estimate $\mathbb{E}(f(Y))$ simply by $I_n^{(1)} := \alpha_0^{n,M_n}$: it is unbiased (see Lemma 2.1) and leads to a quadratic error controlled by

$$\mathbb{E}\left(|I_n^{(1)} - \mathbb{E}(f(Y))|^2\right) = \mathbb{E}\left(|\alpha_0^{n,M_n} - \alpha_0^\star|^2\right) \leq \mathcal{E}_n. \tag{2.9}$$

Algorithm 2. A second possibility is to generate M other independent samples of Y serving to build a Monte-Carlo estimator of $\mathbb{E}(f(Y))$ with control variates given by $f^n(Y)$ (removing the constant term): it writes

$$I_n^{(2)} := \frac{1}{M}\sum_{m=1}^{M}\left(f(Y^m) - \sum_{k=1}^{K_n}\alpha_k^{n,M_n}\phi_k(Y^m)\right). \tag{2.10}$$

As for the first algorithm, it is an unbiased estimator of $\mathbb{E}(f(Y))$ and its quadratic error is given by

$$\begin{aligned}\mathbb{E}\left(|I_n^{(2)} - \mathbb{E}(f(Y))|^2\right) &= \frac{1}{M}\mathbb{E}\left(\mathbb{V}\mathrm{ar}^n\Big(f(Y) - \sum_{k=1}^{K_n}\alpha_k^{n,M_n}\phi_k(Y)\Big)\right)\\ &= \frac{1}{M}\left(\sum_{k=1}^{K_n}\mathbb{E}\left((\alpha_k^{n,M_n} - \alpha_k^\star)^2\right) + \sum_{k=K_n+1}^{+\infty}[\alpha_k^\star]^2\right)\\ &\leq \frac{\mathcal{E}_n}{M}.\end{aligned} \tag{2.11}$$

Hence, once computed a "good" L_2-approximation of f, it gives the possibility of more accurately evaluating its integral via Monte-Carlo simulations.

2.5. Comparison with a naive Monte-Carlo algorithm

A naive Monte-Carlo algorithm would consist in truncating the decomposition (1.1) at order K and in using M independent copies $(Y^m)_{1\leq m\leq M}$ with the same distribution as Y to approximate each coefficient $\alpha_k^\star$ by α_k^M defined in (1.2). Then, the approximation of f writes

$$f_{K,M}^{\text{Naive MC}}(.) := \sum_{k=0}^{K} \alpha_k^M \phi_k(.). \tag{2.12}$$

The related quadratic error is equal to

$$\begin{aligned}
\mathbb{E}\left(\left[f_{K,M}^{\text{Naive MC}}(Y) - f(Y)\right]^2\right) &= \mathbb{E}(r_K^2(Y)) + \sum_{k=0}^{K} \mathbb{E}\left((\alpha_k^M - \alpha_k^\star)^2\right) \\
&= \mathbb{E}(r_K^2(Y)) + \sum_{k=0}^{K} \frac{\mathbb{V}\text{ar}\left(\phi_k(Y) f(Y)\right)}{M} \\
&\leq \mathbb{E}(r_K^2(Y)) + \frac{\mathbb{E}\left(f^2(Y)\right)}{M} \sum_{k=0}^{K} c_k^2
\end{aligned}$$

using **(H2)** and bounding the variance by the second moment. By appropriately tuning M in order to balance both contributions, we obtain

Proposition 2.1. *Let $K \geq 0$ and assume that $\mathbb{E}(r_K^2(Y)) \leq \xi(K)$ for an explicit positive function $\xi(.)$ that measures the accuracy of the L_2-approximation w.r.t. the order K. For $M := \lceil \frac{\sum_{k=0}^{K} c_k^2}{\xi(K)} \rceil$, the quadratic error of the naive Monte-Carlo estimator* (2.12) *is at most equal (up to a constant C independent of K) to the truncation error:*

$$\mathbb{E}\left(\left[f_{K,M}^{\text{Naive MC}}(Y) - f(Y)\right]^2\right) \leq C\ \xi(K).$$

This result serves as a benchmark for the theoretical performance of sequential algorithms, see the discussion in Section 3.1.

3. Examples of Y and vector spaces

3.1. Chebyshev polynomials

Let us consider the case of d-dimensional random variable Y which distribution has the density

$$p(x) = \prod_{i=1}^{d} \left(\frac{1}{\pi\sqrt{1-x_i^2}}\right) \mathbf{1}_{(-1,1)^d}(x) \tag{3.13}$$

with respect to the Lebesgue measure on $\mathbb{R}^d$. Its coordinates $(Y_1, \ldots, Y_d)$ are independent and each one can be simulated owing to the inversion method: $Y_i \overset{\mathrm{d}}{=} \sin(\pi U_i - \pi/2)$ where U_i is uniformly distributed on $(0,1)$.

Chebyshev polynomials provide natural basis functions orthonormalized w.r.t. the law of Y. The one-dimensional polynomial of degree $k \in \mathbb{N}$ is defined by

$$T_0(x) = 1 \quad \text{and} \quad T_k(x) = \sqrt{2}\cos(k \arccos(x)) \text{ for } k \geq 1, \tag{3.14}$$

so that $\mathbb{E}(T_k(Y)T_l(Y)) = \delta_{k,l}$ for any $k, l \geq 0$. For usual properties related to Chebyshev polynomials, we refer the reader to [11, Sections 1.5 and 2.2]. These polynomials have the advantage of being bounded (useful to check **(H2)**). The approximation of f on polynomial basis is often referred to as spectral method [11, 16].

The multidimensional case is achieved by tensorization: for a multi-integers $\mathbf{k} = (k_1, \ldots, k_d) \in \mathbb{N}^d$, set

$$T_{\mathbf{k}}(x) = \prod_{i=1}^{d} T_{k_i}(x_i). \tag{3.15}$$

We readily check that the basis functions $(T_{\mathbf{k}})_{\mathbf{k}\in\mathbb{N}^d}$ satisfy **(H1)**and **(H2)** (with $c_{\mathbf{k}} = 2^{d/2}$).

First strategy for K_n. Define $|\mathbf{k}| = \max_{i=1,\ldots,d} k_i$: at the n-th step of the algorithm, we aim at approximating f by

$$\sum_{\mathbf{k}:|\mathbf{k}|\leq n} \alpha^{\star}_{\mathbf{k}} T_{\mathbf{k}},$$

i.e by taking all the polynomials with degree at most n in each variable: therefore,

$$K_n = (n+1)^d. \tag{3.16}$$

This numerical strategy with *Polynomially Growing approximation space* is referred to in the sequel as *PG (or SALT-PG)*. The convergence rate of $\mathbb{E}(r^2_{K_n}(Y))$ can be analyzed by standard approximation results in the field of spectral methods, see for instance [9, Inequality (5.8.29), Section 5.8.4]. Namely, if f is L-times differentiable ($L \geq 1$) with derivatives in $L_2(\mathbb{P}\circ Y^{-1})$, we have

$$\mathbb{E}\left(r^2_{K_n}(Y)\right) \leq C n^{-2L}, \quad n \geq 1, \tag{3.17}$$

for a constant C independent on n. Then, the choice $M_n = \lceil 2 \sum_{\mathbf{k}:|\mathbf{k}|\leq n} c_k^2 \rceil = 2^{d+1}(n+1)^d$ leads to the error estimate

$$\mathcal{E}_n \leq_c \sum_{k=1}^{n} 2^{k-n} k^{-2L} + 2^{-n} \leq_c n^{-2L}$$

(the second inequality follows by easy computations exploiting the fast increase of 2^k compared to k^{2L} as $k \to +\infty$). The number of simulations used up to step n (representing the simulation effort) is equal $\mathcal{M}_n := \sum_{k=1}^{n} M_k \geqslant_c n^{d+1}$, therefore

$$\mathcal{E}_n \leq C \mathcal{M}_n^{-\frac{2L}{d+1}}. \tag{3.18}$$

We recover the curse of dimensionality effect, with a competition between the smoothness L and the dimension d. We mention that the effective algorithm complexity may be different from $\mathcal{M}_n$: it depends on the way the user encodes the evaluation of Chebyshev polynomials and the related finite approximation.

Second strategy for K_n. Actually the factor $d+1$ in (3.18) can be improved into a factor d by choosing another numerical strategy with *Exponentially Growing approximation space* (denoted by *EG or SALT-EG).* For $\gamma > 0$, set

$$K_{n+1} = \max(K_n + 1, \lfloor K_n e^{\gamma} \rfloor)$$

for a given K_1: the ordering of indices $\mathbf{k}$ of the polynomials that we incorporate at each step is not really important in our analysis, for instance in dimension 2 we can take $1, x_1, x_2, x_1x_2, x_1^2, x_1^2x_2, x_1^2x_2^2, x_1x_2^2, x_2^2, x_1^3, \ldots$ and similarly in higher dimension. On the one hand, it is clear that $K_n \leq_c e^{\gamma n}$. On the other hand, since $K_{n+1} \geq K_n e^{\gamma} - 1$ and $K_{n+1} \geq n + K_1 \to +\infty$, we easily deduce $e^{\gamma n} \leq_c K_n$. Hence, to summarize,

$$K_n \geqslant_c e^{\gamma n}. \tag{3.19}$$

Denote by $\tilde{n}$ the integer such that $(\tilde{n}+1)^d \leq K_n < (\tilde{n}+2)^d$: then, from (3.17) the residual at step n is bounded as

$$\mathbb{E}\left(r_{K_n}^2(Y)\right) \leq C \tilde{n}^{-2L} \geqslant_c e^{-2L\frac{\gamma}{d}n}. \tag{3.20}$$

Moreover, the number of simulations used up to step n can be estimated as $\mathcal{M}_n := \sum_{k=1}^n M_k \geqslant_c \sum_{k=1}^n 2^{d+1} K_k \geqslant_c e^{\gamma n}$. At last, after simple computations, we can show that the quadratic error (2.8) is estimated as follows:

$$\mathcal{E}_n \leq_c \sum_{k=1}^{n} 2^{k-n} e^{-2L\frac{\gamma}{d}k} + 2^{-n} \leq_c \begin{cases} e^{-2L\frac{\gamma}{d}n} & \text{if } \gamma < \gamma^\star := \frac{d\log(2)}{2L}, \\ n2^{-n} & \text{if } \gamma = \gamma^\star, \\ 2^{-n} & \text{if } \gamma > \gamma^\star, \end{cases} \tag{3.21}$$

$$\leq C \begin{cases} \mathcal{M}_n^{-\frac{2L}{d}} & \text{if } \gamma < \gamma^\star, \\ \mathcal{M}_n^{-\frac{2L}{d}} \log(M_n) & \text{if } \gamma = \gamma^\star, \\ \mathcal{M}_n^{-\log(2)/\gamma} & \text{if } \gamma > \gamma^\star. \end{cases} \tag{3.22}$$

A better convergence rate is thus obtained for γ small enough, smaller than the threshold $\gamma^\star := \frac{d\log(2)}{2L} \approx 0.347\frac{d}{L}$ related to the smoothness of f and the dimension of Y. Comparing (3.18) and (3.22), observe that SALT-EG yields an improved convergence rate w.r.t. the total simulation effort $\mathcal{M}_n$, compared to SALT-PG.

Let us briefly compare the above accuracy result with the one obtained using a naive Monte-Carlo approach (see Section 2.5). From Proposition 2.1 and (3.17), we have $\xi(K) = K^{-2L/d}$ and $M = \lceil 2^d(K+1)K^{2L/d} \rceil$, which gives

$$\mathbb{E}\left(\left[f_{K,M}^{\text{Naive MC}}(Y) - f(Y) \right]^2 \right) \leq C M^{-\frac{2L}{d+2L}}.$$

The above rate is obviously worse than $M^{-\frac{2L}{d}}$ in (3.22).

Application to the computation of $\mathbb{E}(f(Y))$. Applying **Algorithm 2** of Section 2.4 using the algorithm SALT-EG to the current setting (see Equality (2.10)) with $M = \mathcal{M}_n$ extra simulations (which only doubles the total number of simulations), we get an accuracy in the evaluation of $\mathbb{E}(f(Y))$ controlled by (see Inequality (2.11))

$$\sqrt{\mathbb{E}\left(|I_n^{(2)} - \mathbb{E}(f(Y))|^2 \right)} \leq C \mathcal{M}_n^{-\frac{L}{d}-\frac{1}{2}}. \tag{3.23}$$

Of course, this is a very significant improvement compared to the naive Monte-Carlo algorithm for which the above estimate is simply $C\mathcal{M}_n^{-\frac{1}{2}}$.

Remind that for the computation of $\alpha_0^\star = \mathbb{E}(f(Y))$ when f is a smooth function, the optimal error using $\mathcal{M}_n$ random evaluations of f is $\mathcal{M}_n^{-\frac{L}{d}-\frac{1}{2}}$

(see [1] or [2] for a review). Our SALT-EG achieves this optimal rate. We mention that in [1], the authors provide a different optimal algorithm in the case where Y is uniformly distributed on $(-1,1)^d$.

Nevertheless, not only our SALT-EG algorithm is rate-optimal for the computation of $\mathbb{E}(f(Y))$, but additionally we do compute all the coefficients sequence $(\alpha^\star_{\mathbf{k}})_{\mathbf{k}\in\mathbb{N}^d}$ and not only $\alpha^\star_0$, which is a significant improvement.

3.2. Legendre polynomials

If Y is now uniformly distributed on the cube $(-1,1)^d$, a similar analysis can be derived using Legendre polynomials instead of Chebyshev ones. For related properties and approximation results, we refer to [11, Sections 1.4 and 2.2] and [9, Inequality (5.8.11), Section 5.8.2]. The main difference is that the orthonormalized polynomials are no more bounded independently of the degree. Indeed, we can easily check that in dimension 1, the orthonormalized Legendre polynomial L_k with degree k is uniformly bounded by $\sqrt{k+1} := c_k$. In dimension d, $L_{\mathbf{k}}(x) = L_{k_1}(x_1)\dots L_{k_d}(x_d)$ is bounded by $c_{\mathbf{k}} = \sqrt{(k_1+1)\dots(k_d+1)}$. As for (3.16) with SALT-PG, we take $K_n = (n+1)^d$ (projecting on Legendre polynomials with degree at most n in each variable): then, for a function f with L-derivatives that are square integrable on $(-1,1)^d$, the rate in (3.17) is unchanged. Moreover, following Corollary 2.2, we take $M_n = \lceil 2\sum_{\mathbf{k}:|\mathbf{k}|\le n} c^2_{\mathbf{k}}\rceil = 2\big(\frac{(n+1)(n+2)}{2}\big)^d$, which gives $\mathcal{M}_n \geqslant_c n^{2d+1}$. The quadratic error of the algorithm is then

$$\mathcal{E}_n \le C\mathcal{M}_n^{-\frac{2L}{2d+1}}. \tag{3.24}$$

Similar computations with the SALT-EG version leads to $\mathcal{E}_n \le_c \mathcal{M}_n^{-\frac{L}{d}}$. The application to the computation of $\mathbb{E}(f(Y))$ can be analyzed similarly, we leave the details to the reader.

Observe that the unboundedness of Legendre polynomials has a negative effect on the convergence rate which is similar to doubling the dimension: at least from theoretical viewpoint, the values of $(c_k)_k$ in **(H2)** are an important concern of our algorithm.

3.3. Other distributions and approximations

We mention further extensions that we do not fully detail here. Our objective is rather to give, for the reader convenience, alternative ways to explore, that depend much on the examples to handle.

Obtaining sparse polynomial approximations (i.e. a reduced size of the vector space while maintaining the same accuracy) is possible for functions f belonging to Korobov spaces, see for instance [19]. Some transformations of the distribution of Y are also proposed to reduce to that case.

In the case of other distributions (in particular supported on non compact sets of $\mathbb{R}^d$), we can use damped polynomials (of Laguerre or Hermite type, the damping ensures that **(H2)** is met) when we have to handle exponential or Gaussian distributions for Y. Alternatively, one can reduce to one of the previous cases by a change of variable $Y = \varphi(Z)$ (and then compute the L_2 projection of $f \circ \varphi$ w.r.t. the probability measure $\mathbb{P} \circ Z^{-1}$).

It is also standard to use a one-to-one mapping $\varphi : (-1,1)^d \mapsto \mathbb{R}^d$, to reduce to the cube; for related discussions on φ, see [9, Section 2.7]. Besides, we argue that the cube case (like $(-1,1)^d$ or $(0,1)^d$) is quite frequent in probabilistic applications, since usually a simulation algorithm maps a r.v. uniformly distributed on a cube into the target random variable Y.

If the distribution of $Z = \varphi^{-1}(Y)$ has a bounded density $p^Z(.)$ on $(-1,1)^d$, then, for any g,

$$\begin{aligned}\mathbb{E}(g^2(Y)) &= \int_{(-1,1)^d} g^2(\varphi(z)) p_Z(z) dz \\ &\leq |p^Z|_\infty \pi^d \int_{(-1,1)^d} g^2(\varphi(z)) \frac{dz}{\pi^d \sqrt{(1-z_1^2)\dots(1-z_d^2)}},\end{aligned}$$

meaning that the $L_2(\mathbb{P} \circ Y^{-1})$-norm is controlled (up to a constant) by the L_2-norm w.r.t. Chebyshev weights. Hence, computing the approximation of $f \circ \varphi$ in the latter norm is sufficient to derive a L_2 approximation of f w.r.t. the law of Y.

So far, we have mainly mentioned global polynomial approximations which take well advantage of global smoothness properties, but the use of local approximations (local polynomials for instance) may be relevant in practice. The combination with sparse grids [8] may be a source of significant improvement as well.

4. Numerical experiments

We present numerical results performed when the distribution of Y has the density (3.13), in dimension $d = 1$ and $d = 2$. We use the Chebyshev

Table 1: Variance of $f(Y)$ in each case (Tables 2-3-4-5-6)

Table 1	Table 2	Table 3	Table 4	Table 5
0.134	0.0727	0.144	0.0338	0.337

Table 2: Results in dimension $d = 1$ for $f(x) = (1 - x^2)^4$ using SALT-PG

n	K_n	$\mathcal{M}_n$	$\mathcal{E}_n$	Alg. 2 Error	IF
10	11	264	3.667×10^{-5}	7.305×10^{-4}	6.05×10^{1}
20	21	924	2.374×10^{-12}	9.936×10^{-8}	2.38×10^{5}
30	31	1984	2.914×10^{-17}	2.375×10^{-10}	6.78×10^{7}
40	41	3444	1.399×10^{-20}	3.950×10^{-12}	3.10×10^{9}
50	51	5304	2.210×10^{-22}	4.001×10^{-13}	2.46×10^{10}
70	71	10224	3.294×10^{-25}	1.112×10^{-14}	6.38×10^{11}
100	100	20604	3.458×10^{-28}	2.539×10^{-16}	1.97×10^{13}

polynomials as exposed in Section 3.1, with various SALT-steps n and different strategies for increasing the approximation space (either PG or EG). In the results reported below (Tables 2-3-4-5-6), we indicate:

- The size of the vector space K_n at the final step n: for PG it is equal to $K_n = (n+1)^d$, and for EG it is given by $K_0 = 1$, $K_{n+1} = \max(K_n + 1, \lfloor K_n e^{\gamma} \rfloor)$ for a parameter γ specified later.
- The total amount of simulations $\mathcal{M}_n = \sum_{k=1}^{n} M_k$ used up to step n.
- The quadratic error $\mathcal{E}_n$, which is computed using the formula (2.5) as an empirical average of $\big[f_n(Y) - f(Y)\big]^2$. The integration over Y is made with $\mathcal{M}_n$ extra independent simulations, and the one over f_n with 10 independent runs of SALT.
- The `Alg.2 Error` is defined by the half-width of the 95%-confidence interval using the Algorithm 2 for the computation of $\mathbb{E}(f(Y))$, i.e. using the SALT-approximation as a control variate, see the estimator (2.10); thus we set `Alg.2 Error` $= 1.96\sqrt{\frac{\mathbb{V}\mathrm{ar}(I_n^{(2)})}{\mathcal{M}_n}}$.

Table 3: Results in dimension $d = 1$ for $f(x) = (x_+)^2$

n	SALT	K_n	$\mathcal{M}_n$	$\mathcal{E}_n$	Alg. 2 Error	IF
10	PG	11	264	9.430×10^{-4}	3.704×10^{-3}	8,78
	EG ($\gamma = 0.05$)	11	264	9.812×10^{-4}	3.779×10^{-3}	8,60
20	PG	21	924	1.539×10^{-4}	7.999×10^{-4}	21,7
	EG ($\gamma = 0.05$)	21	924	1.253×10^{-4}	7.217×10^{-4}	24,1
30	PG	31	1984	4.426×10^{-5}	2.927×10^{-4}	40,5
	EG ($\gamma = 0.05$)	32	1988	4.741×10^{-5}	3.027×10^{-4}	39,1
40	PG	41	3444	1.508×10^{-5}	1.297×10^{-4}	69,4
	EG ($\gamma = 0.05$)	53	3712	8.086×10^{-6}	9.147×10^{-5}	94,8
50	PG	51	5304	7.485×10^{-6}	7.118×10^{-5}	102
	EG ($\gamma = 0.05$)	87	6532	1.868×10^{-6}	3.315×10^{-5}	197

Table 4: Results in dimension $d = 1$ for $f(x) = (\sin(10x))_+$

n	SALT	K_n	$\mathcal{M}_n$	$\mathcal{E}_n$	Alg.2 Error	IF
10	PG	11	264	3.107×10^{-1}	6.724×10^{-2}	0,68
	EG ($\gamma = 0.05$)	11	264	2.761×10^{-1}	6.338×10^{-2}	0,72
20	PG	21	924	2.193×10^{-2}	9.543×10^{-3}	2,57
	EG ($\gamma = 0.05$)	21	924	2.500×10^{-2}	1.019×10^{-2}	2,40
30	PG	31	1984	5.812×10^{-3}	3.355×10^{-3}	4,98
	EG ($\gamma = 0.05$)	32	1988	6.261×10^{-3}	3.478×10^{-3}	4,80
40	PG	41	3444	1.898×10^{-3}	1.455×10^{-3}	8,72
	EG ($\gamma = 0.05$)	53	3712	1.283×10^{-3}	1.115×10^{-3}	11,0
50	PG	51	5304	1.367×10^{-3}	9.951×10^{-4}	10,3
	EG ($\gamma = 0.05$)	87	6532	2.411×10^{-4}	3.765×10^{-4}	24,5

- The Improvement Factor (IF) is defined by the ratio between the half-width of the 95%-confidence interval of the standard Monte-Carlo method using the same number of simulations $\mathcal{M}_n$, and `Alg.2 Error`, whence

$$\text{IF} = 1.96\sqrt{\frac{\mathbb{V}\text{ar}(f(Y))}{\mathcal{M}_n}}/\texttt{Alg.2 Error} = \sqrt{\frac{\mathbb{V}\text{ar}(f(Y))}{\mathbb{V}\text{ar}(I_n^{(2)})}}.$$

  Restated in terms of numerical efficiency, obtaining for instance IF $=$ 2 is equivalent to a requirement of $2^2 = 4$ times fewer simulations for the same accuracy. The variances $\mathbb{V}\text{ar}(f(Y))$ have been computed numerically, their values are reported in Table 1.

We first illustrate the phenomenon of geometric convergence stated in Corollary 2.1, by taking the function f as a polynomial (see Tables 2 and 5). As soon as the size K_n is large enough so that the approximation space includes f, we have $r_{K_n}(Y) = 0$ and the estimates of Corollary 2.2 lead to a geometric convergence of the error to 0. This is clearly observed in Tables 2 and 5.

Second, when the function f is in none of the approximation spaces, the convergence is presumably not geometric. In Table 3, we take $f(x) = (x_+)^2$, for which the regularity may be considered equal to $L = 2$, yielding $\gamma^* \approx 0.173$ (see definition (3.21)). We experiment both strategies PG and EG ($\gamma = 0.05 < \gamma^*$) for increasing the size of approximation space. PG and EG coincide for n small since γ is rather small, and they differ only for n large. Observe that the accuracy is similar after $n = 50$ steps with PG and $n = 40$ steps with EG. The advantage of EG here is that we have used a reduced number of simulations, since by increasing faster K_n, we increase faster M_n and the overall computational effort is smaller. In Table 4 where $f(x) = (\sin(10x))_+$, the observation is similar. We have experienced larger γ, with analogous features, but there is no clear picture to us of which values of γ are better.

Third, we observe that the convergence rate seemingly worsens as the regularity of the function f deteriorates, see Table 4 with $f(x) = (\sin(10x))_+$ which is only once (almost-everywhere) differentiable with large derivative (due to the factor 10) and Table 6 with $f(x, y) = (x + y)_+$. SALT still enables to speed-up the computation of $\mathbb{E}(f(Y))$ but the improvement is less significant compared to smooth cases. This is coherent with the previous theoretical analysis, in particular that leading to the estimate (3.23) which illustrates the effect of regularity on the convergence order.

Although the significance of the improvement varies from one example to another, the SALT appears as an interesting scheme for L_2-approximation and variance reduction, since the tuning of its parameters can be made in a robust way, leading to convergence in quite general situations (see Corollary 2.2).

Table 5: Results in dimension $d = 2$ for $f(x, y) = (1 - x^2)^4(1 - y^2)^4$ using SALT-PG

n	K_n	$\mathcal{M}_n$	$\mathcal{E}_n$	Alg.2 Error	IF
10	121	4048	1.529×10^{-5}	1.186×10^{-4}	4.77×10^{1}
20	441	26488	6.604×10^{-14}	3.090×10^{-9}	7.16×10^{5}
30	961	83328	9.212×10^{-18}	2.061×10^{-11}	6.06×10^{6}
40	1681	190568	3.656×10^{-20}	8.585×10^{-13}	9.61×10^{8}

Table 6: Results in dimension $d = 2$ for $f(x, y) = (x + y)_+$, using SALT-PG

n	K_n	$\mathcal{M}_n$	$\mathcal{E}_n$	Alg.2 Error	IF
10	121	4048	1.294×10^{-2}	3.487×10^{-3}	5,13
20	441	26488	1.274×10^{-3}	3.782×10^{-4}	18,5
30	961	83328	3.437×10^{-4}	1.258×10^{-4}	31,3
40	1681	190568	1.569×10^{-4}	5.625×10^{-5}	46,3

Appendix A. Proof of Lemma 2.1

Because of independence between steps,

$$\mathbb{E}^{n-1}(\beta_k^{n,M_n}) = \mathbb{E}(f(Y)\phi_k(Y)) - \sum_{l=0}^{K_{n-1}} \alpha_l^{n-1,M_{n-1}} \mathbb{E}(\phi_l(Y)\phi_k(Y)) = \alpha_k^\star - \alpha_k^{n-1,M_{n-1}},$$

thus $\mathbb{E}^{n-1}(\alpha_k^{n,M_n}) = \alpha_k^{n-1,M_{n-1}} + \alpha_k^\star - \alpha_k^{n-1,M_{n-1}} = \alpha_k^\star$. We now turn to the variance analysis. By bounding the variance by the second moment and using **(H2)**, we obtain

$$\mathbb{V}\mathrm{ar}^{n-1}(\alpha_k^{n,M_n}) = \frac{1}{M_n} \mathbb{V}\mathrm{ar}^{n-1}\left[(f(Y) - \sum_{l=0}^{K_{n-1}} \alpha_l^{n-1,M_{n-1}} \phi_l(Y))\phi_k(Y) \right]$$

$$\leq \frac{c_k^2}{M_n}\mathbb{E}^{n-1}\left[(f(Y)-\sum_{l=0}^{K_{n-1}}\alpha_l^{n-1,M_{n-1}}\phi_l(Y))^2\right]$$
$$= \frac{c_k^2}{M_n}\left(\mathbb{E}[r^2_{K_{n-1}}(Y)]+\sum_{l=0}^{K_{n-1}}(\alpha_l^{n-1,M_{n-1}}-\alpha_l^\star)^2\right)$$

using (1.3) and **(H2)** at the last equality. □

References

[1] E. Atanassov, I. Dimov, A new Monte Carlo method for calculating integrals of smooth functions, Monte Carlo Methods Appl. 5 (1999) 149–167.

[2] E. Atanassov, I. Dimov, What Monte Carlo models can do and cannot do efficiently?, Applied Mathematical Modelling 32 (2008) 1477–1500.

[3] K. Baggerly, D. Cox, R. Picard, Exponential convergence of adaptive importance sampling for Markov chains, J. Appl. Prob. 37 (2000) 342–358.

[4] T. Ben Zineb, E. Gobet, Preliminary control variates to improve empirical regression methods, Monte-Carlo methods and Applications 19 (2013) 331–354.

[5] C. Bender, J. Steiner, Least-squares Monte Carlo for BSDEs, in: R. Carmona, P. Del Moral, P. Hu, N. Oudjane (Eds.), Numerical Methods in Finance, Series: Springer Proceedings in Mathematics, Vol. 12, 2012, pp. 257–289.

[6] T. Booth, Exponential convergence for Monte Carlo particle transport?, Trans. Amer. Nucl. Soc. 50 (1985) 267–268.

[7] B. Bouchard, X. Warin, Monte-Carlo valuation of American options: facts and new algorithms to improve existing methods., in: R. Carmona, P. Del Moral, P. Hu, N. Oudjane (Eds.), Numerical Methods in Finance, Series: Springer Proceedings in Mathematics, Vol. 12, 2012, pp. 215–255.

[8] H.J. Bungartz, M. Griebel, Sparse grids., Acta Numerica 13 (2004) 147–269.

[9] C. Canuto, M. Hussaini, A. Quarteroni, T. Zang, Spectral methods: fundamentals in single domains, Springer-Verlag, New York, 2006.

[10] D. Egloff, Monte Carlo algorithms for optimal stopping and statistical learning, Ann. Appl. Probab. 15 (2005) 1396–1432.

[11] D. Funaro, Polynomial approximation of differential equations, volume 8 of *Lecture Notes in Physics. New Series m: Monographs*, Springer-Verlag, Berlin, 1992.

[12] E. Gobet, C. Labart, Solving BSDE with adaptive control variate, SIAM Numerical Analysis 48 (2010) 257–277.

[13] E. Gobet, J.P. Lemor, X. Warin, A regression-based Monte Carlo method to solve backward stochastic differential equations, Annals of Applied Probability 15 (2005) 2172–2202.

[14] E. Gobet, S. Maire, Sequential control variates for functionals of Markov processes, SIAM Journal on Numerical Analysis 43 (2005) 1256–1275.

[15] E. Gobet, P. Turkedjiev, Linear regression MDP scheme for discrete backward stochastic differential equations under general conditions, In revision for Mathematics of Computation (2013).

[16] O. Le Maître, O. Knio, Spectral Methods for Uncertainty Quantification. With Applications to Computational Fluid Dynamics, Scientific Computation, Springer, 2010.

[17] J.P. Lemor, E. Gobet, X. Warin, Rate of convergence of an empirical regression method for solving generalized backward stochastic differential equations, Bernoulli 12 (2006) 889–916.

[18] F. Longstaff, E. Schwartz, Valuing American options by simulation: A simple least squares approach, The Review of Financial Studies 14 (2001) 113–147.

[19] S. Maire, An iterative computation of approximations on Korobov-like spaces, J. Comput. Appl. Math. 157 (2003) 261–281.

[20] S. Maire, Reducing variance using iterated control variates, The Journal of Statistical Computation and Simulation 73 (2003) 1–29.