

Conditional Monte Carlo Learning for Diffusions I: main methodology and application to backward stochastic differential equations

Lokman Abbas-Turki, G. Pagès, Babacar Diallo

▶ **To cite this version:**

Lokman Abbas-Turki, G. Pagès, Babacar Diallo. Conditional Monte Carlo Learning for Diffusions I: main methodology and application to backward stochastic differential equations. 2020. hal-02959492

HAL Id: hal-02959492
https://hal.science/hal-02959492

Preprint submitted on 6 Oct 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Conditional Monte Carlo Learning for Diffusions I: main methodology and application to backward stochastic differential equations

L. Abbas-Turki*, B. Diallo† and G. Pagès‡

October 6, 2020

Abstract

We present a new algorithm based on a One-layered Nested Monte Carlo (1NMC) to simulate functionals U of a Markov process X. The main originality of the proposed methodology comes from the fact that it provides a recipe to simulate $U_{t\geq s}$ conditionally on X_s. Because of the nested structure that allows a Taylor-like expansion, it is possible to use a very reduced basis for the regression. Although this methodology can be adapted for a large number of situations, we only apply it here for the simulation of Backward Stochastic Differential Equations (BSDEs). The generality and the stability of this algorithm, even in high dimension, make its strength. It is heavier than a straight Monte Carlo (MC) but it is far more accurate to simulate quantities that are almost impossible to simulate with MC. The parallel suitability of 1NMC makes it feasible in a reasonable computing time. This paper explains the main version of this algorithm and provides first results of error estimates. We also give various numerical examples with a dimension equal to 100 that are executed from few seconds to few minutes on one Graphics Processing Unit (GPU).

1 Introduction

Numerous contributions in numerical methods based on Monte Carlo reached recently their limits in dealing with the curse of dimensionality [4]. This paper is a natural progress of an increasing interest in Nested Monte Carlo (NMC) started in [18, 20, 21] and used with regression in [1, 6] and with Multilevel method in [13]. In this contribution, One-layered Nested Monte Carlo (1NMC) is set for only two layers of Monte Carlo that offer the possibility of having two layers of approximation. In contrast to previous works, our methodology is based on a judicious combination between 1NMC and various localized regressions.

*Email: lokmane.abbas_turki@sorbonne-universite.fr. LPSM (UMR 8001), Sorbonne Université, 4 Place Jussieu 75005 Paris, France & CMAP (UMR 7641), École Polytechnique, Route de Saclay 91128 Palaiseau cedex, France.

†Email: babacardiallopro@gmail.com. LaMME (UMR 8071), Université Paris-Saclay, 23 Boulevard de France 91037 Evry, France.

‡Email: gilles.pages@sorbonne-universite.fr. LPSM (UMR 8001), Sorbonne Université, 4 Place Jussieu 75005 Paris, France.

Although we are not the first to propose a learning procedure for BSDEs or option pricing/hedging [7, 11, 19], we are the first to do it using NMC and local regressions instead of a neural network. Thus, we replace standard learning using a global minimization to train a neural network by local regressions that provide very accurate results due to 1NMC. Combining regression with 1NMC makes the local projections accurate, even on a reduced basis, as long as the projected values are sufficiently smooth. Of course, local neural networks can replace local regressions but, as will be shown, the latter are already quite sufficient for the considered examples and have very efficient implementation using batch parallel processing [3].

The presented procedure is called Conditional Monte Carlo Learning for Diffusions (CMCLD) as the regression-based analytical representations are established on the top of a set of outer trajectories of diffusions. If these outer trajectories are erased the whole learning procedure is lost. The conditional learning is the main drawback of this methodology but we are working on making the learning unconditional in future works. The goal of this paper is to lay the foundation of the simplest version of this methodology making possible further extensions of it in Conditional Monte Carlo Learning for Diffusions part II (CMCLDII) [2].

On the top of what is presented in this part I, part II [2] will be dedicated to further error controls and slight modifications that make possible accurate simulation of tail values as well as optimal stopping problems in high/very high dimensions. Part I and II have a common purpose of explaining in depth CMCLD and its benefit either when the dimension is high/very high or when the problem can be formulated as a composition of various functionals of a Markov process.

On a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{0\leq t\leq T}, \mathbb{P})$, the generic example is driven by an $\mathcal{F}_t$-Markov process $(X_t)_{t\in[0,T]}$ taking its values on $\mathbb{R}^{d_1}$. Given the fine time discretization $\mathbb{S} = \{t_0, ..., t_{2^L}\} = \{0, T/2^L, ..., T\}$, let U_s be a functional of X defined for $s \in \mathbb{S}$ by

$$(f) \quad U_s = u_s(X_s) = \mathbb{E}_s\left(\sum_{k=s2^L/T}^{2^L} f(t_k, X_{t_k}, X_{t_{k+1}})\right) = \mathbb{E}\left(\sum_{s\leq t_k\leq T} f(t_k, X_{t_k}, X_{t_{k+1}})\Big|\mathcal{F}_s\right),$$

where $\mathbb{E}_s(\cdot) = \mathbb{E}\left(\cdot\Big|\mathcal{F}_s\right)$, the expectation is always considered under $\mathbb{P}$, each deterministic function $f(t_k, \cdot, \cdot)$ is $\mathcal{B}(\mathbb{R}^{d_1}) \otimes \mathcal{B}(\mathbb{R}^{d_1})$-measurable and assumed to satisfy the square integrability condition $\mathbb{E}(f^2(t_k, X_{t_k}, X_{t_{k+1}})) < +\infty$ with convention $f(t_{2^L}, X_{t_{2^L}}, X_{t_{2^L+1}}) = f(t_{2^L}, X_{t_{2^L}})$. From the Markov assumption, for each $s \in \mathbb{S}$, the $\mathcal{B}(\mathbb{R}^{d_1})$-measurable function u_s is deterministic. The simulation of U is generic to all BSDE examples presented in this paper. As nested simulations involve heavy notations, it is easier to present the whole algorithm for the simulation of U then apply it on specific examples.

When previous contributions target estimations of U_{t_k} for $k = 0, ..., 2^L$ knowing some realization of $\{X_{t_j}\}_{0\leq j\leq k}$ $(m_0 = 1, ..., M_0)$, our purpose is to simulate approximations $\{U^{m_0,m_1}_{t_k,s}\}_{s\geq t_{k+1}}$, with $(m_0 = 1, ..., M_0)$ and $(m_1 = 1, ..., M_1)$, of $\{U_s\}_{s\geq t_{k+1}}$ conditionally on the realization $\{X^{m_0}_{t_j}\}_{0\leq j\leq k}$. This task requires the simulation of a first layer $(X^{m_0})_{m_0=1,...,M_0}$ of trajectories that are kept in the machine's random-access memory, then a second $(X^{m_0,m_1})_{m_1=1,...,M_1}$ unstored layer of trajectories, on the top of the first layer indexed by m_0, only used to learn how should we perform approximations U^{m_0,m_1} called inner layer simulation of the process U.

Although more complex than a regression/MC method (cf. [22, 17]), getting the inner layer simulation provides much more information on the process that has to be simulated.

In particular, it is possible to set the outer layer simulation $U_{t_k}^{m_0}$ of U_{t_k} to be equal to $\frac{1}{M_1}\sum_{m_1=1}^{M_1}\left(f(t_k, X_{t_k}^{m_0}, X_{t_k,t_{k+1}}^{m_0,m_1}) + U_{t_k,t_{k+1}}^{m_0,m_1}\right)$. Knowing the inner layer simulation U^{m_0,m_1}, we can compute quantiles on U or, even more remarkable, can simulate another process $\widetilde{U}$ that satisfies equation $(\widetilde{f})$ (Replace f by $\widetilde{f}$ in equation (f)) with an $\widetilde{f}$ that can be a function of U like for instance $\widetilde{f}(t_k, x, y) = f(t_k, U_{t_k}(x), U_{t_{k+1}}(y))$. Consequently, when sufficient assumptions are satisfied, we can learn how to compute functionals of functionals of X with the same 1NMC. This latter fact makes possible the simulation of Valuation Adjustments [1] as long as one can write them as a composition of functionals then start simulating by the innermost functional till the most outer composition.

The outer simulation U^{m_0} is then defined using the inner one U^{m_0,m_1}, the latter obtained from regressing on the inner trajectories X^{m_0,m_1}. When $U_{t_k}^{m_0}$ is defined for all $k = 0, ..., 2^L$, U^{m_0,m_1} has not to be defined on the whole fine discretization $\mathbb{S} = \{t_0, ..., t_{2^L}\} = \{0, T/2^L, ..., T\}$. We introduce then a coarser discretization set $\mathcal{S}$ that controls the depth of our learning procedure by adding/reducing regression steps needed for sufficiently accurate values of U^{m_0}. It is possible to improve the depth in a parareal fashion [23] which increases further the parallel scalability of the algorithm. Controlling the depth of our learning procedure makes possible the balance between complexity and accuracy. This compromise can be even formulated iteratively as explained in [2].

Having inner and outer simulations allows also to control the bias throughout the whole algorithm. Indeed, for any $s \in [0, T[$ and any $s' \in]s, T]$ one can estimate both parts of equality

$$\mathbb{E}(U_s) = \mathbb{E}\Big(U_{s'} + \sum_{t_{l+1}>s}^{s'} f(t_l, X_{t_l}, X_{t_{l+1}})\Big) \tag{1.1}$$

using $\frac{\sum_{m_0=1}^{M_0} U_s^{m_0}}{M_0}$, $\frac{1}{M_0}\sum_{m_0=1}^{M_0}\left(U_{s'}^{m_0} + \sum_{t_{l+1}>s}^{s'} f(t_l, X_{t_l}^{m_0}, X_{t_{l+1}}^{m_0})\right)$ and their difference that represents the estimated value of the average bias. Controlling the difference is numerically possible using a particular choice of the time step $\overline{s}$, associated to any $s \in \mathcal{S}$, at which we define the terminal condition involved in the inner simulation. To simulate U_s, we can subsequently set a terminal condition $\overline{u}_{s,\overline{s}}^{m_0,\mathcal{S}}(\cdot)$ to be only equal to $f(T, \cdot)$ when $\overline{s} = T$. The choice of $\overline{s}$ and of the terminal condition $\overline{u}_{s,\overline{s}}^{m_0,\mathcal{S}}(\cdot)$ are discussed in the example of Section 2.2, are presented in definitions 3.1 and 3.2 with an upper bound error established in CMCLDII [2]. Combining the tower property in equality (1.1) with the nested structure of our algorithm is very beneficial to stop the bias propagation. Further benefits of this combination on the variance of regressed values is presented in [2].

Starting with a simple application of the generic example, Section 2 introduces the method as well as notations. In Section 3, we define both inner U^{m_0,m_1} and outer U^{m_0} simulations of the process U given in (f) then we adapt it to BSDEs with a Markov forward process. Using similar arguments to the one presented in [6], Section 4 explains the impact of inner regressions on outer simulations. Section 5 shows the robustness of our methodology on highly dimensional problems beyond what is known to be possible in previous contributions.

2 Notations and the methodology applied on a simple example

We introduce in Section 2.1 some needed notations and set the stage for Section 2.2 that explains CMCLD starting from a simple application.

2.1 Notations: 1NMC, stabilized regressions, coarse discretization and δ time operator

1NMC stands for one-layered nested MC and thus involves two layers of Monte Carlo simulation of a Markov process X. Using a sufficiently fine discretization $\mathbb{S} = \{t_0, ..., t_{2^L}\} = \{0, \Delta_t, 2\Delta_t, ..., T\}$ with $\Delta_t = T/2^L$, one simulates M_0 realizations $(X_{t_k}^{m_0})_{k=1,...,2^L}^{m_0=1,...,M_0}$ of the Markov process X starting at a deterministic point $X_0 = x_0 \in \mathbb{R}^{d^1}$ with the following induction

$$X_{t_k}^{m_0} = \mathcal{E}_{t_{k-1}}(X_{t_{k-1}}^{m_0}, \xi_{t_k}^{m_0}), \text{ when } k \geq 1 \text{ and } X_{t_0}^{m_0} = x_0, \tag{2.1}$$

where $(\xi_{t_k}^{m_0})_{k=1,...,2^L}^{m_0=1,...,M_0}$ are independent realizations of an $\mathbb{R}^{d_2}$ random vector ξ and the functions $(\mathcal{E}_{t_k})_{k=0,...,2^L-1} : \mathbb{R}^{d_1+d_2} \to \mathbb{R}^{d_1}$ are Borel-measurable. We use $X_{t_k}^{m_0,1}$, ..., $X_{t_k}^{m_0,d_1}$ to denote the d_1 components of the vector $X_{t_k}^{m_0}$. The sample $(X_{t_k}^{m_0})_{k=1,...,2^L}^{m_0=1,...,M_0}$ stays on the machine memory and is supposed to approximate accurately $(X_t)_{t\in[0,T]}$ in a sense explained in Section 4.

For a decreasing sequence $(s_j)_{j=0,...,2^L}$ that takes its values in the time discretization set $\mathbb{S}$, an extra simulation conditional to the starting $X_{s_j}^{m_0}$ is needed for the learning procedure. Introducing independent realizations $(\xi_{t_j,t_k}^{m_0,m_1})_{k\in\{j,...,2^L\},j\in\{1,...,2^L\}}^{(m_0,m_1)\in\{1,...,M_0\}\times\{1,...,M_1+M_1'\}}$ of the random vector ξ that are also independent from $(\xi_{t_k}^{m_0})_{k=1,...,2^L}^{m_0=1,...,M_0}$, we set for $t_{k-1} \geq s_j$

$$X_{s_j,t_k}^{m_0,m_1} = \mathcal{E}_{t_{k-1}}(X_{s_j,t_{k-1}}^{m_0,m_1}, \xi_{s_j,t_k}^{m_0,m_1}) \text{ and } X_{s_j,s_j}^{m_0,m_1}\Big|_{m_1=1,...,M_1+M_1'} = X_{s_j}^{m_0}. \tag{2.2}$$

We use $X_{s_j,t_k}^{m_0,m_1,1}$, ..., $X_{s_j,t_k}^{m_0,m_1,d_1}$ to denote the d_1 components of the vector $X_{s_j,t_k}^{m_0,m_1}$. For $s_j \leq s_l \leq s_k$, we also introduce the notation $X_{s_j,s_l:s_k}^{m_0,m_1}$ and $\xi_{s_j,s_l:s_k}^{m_0,m_1}$ for respectively $(X_{s_j,s_l}^{m_0,m_1}, X_{s_j,s_l+\Delta_t}^{m_0,m_1}, ..., X_{s_j,s_k-\Delta_t}^{m_0,m_1}, X_{s_j,s_k}^{m_0,m_1})$ and $(\xi_{s_j,s_l}^{m_0,m_1}, \xi_{s_j,s_l+\Delta_t}^{m_0,m_1}, ..., \xi_{s_j,s_k-\Delta}^{m_0,m_1}, \xi_{s_j,s_k}^{m_0,m_1})$.

In (2.2), we simulate $M_1 + M_1'$ conditional realizations of X in order to keep those indexed from $m_1 = M_1 + 1$ to $m_1 = M_1 + M_1'$ for the approximation of regression matrices. Consequently, we make explicit the independence between trajectories used for the estimation of regression matrices and those used in the backward induction. This makes also explicit the difference between the number of trajectories needed for regression matrices approximation and the number of those used for the backward induction. To reduce the complexity of the algorithm and memory occupation, trajectories used for regression matrices can be simulated offline then erased from the memory. Given m_0, if the inner trajectories $\{X^{m_0,m_1}\}_{m_1=1,...,M_1}$ are needed α times in the backward induction, we simulate α independent copies and use each copy once. This reduces further memory occupation as well as any superfluous dependence structure.

For each ordered couple $(j < k)$ of indices that take their values in $\{1, ..., 2^L\}$, we introduce the stabilized regression basis

$$\mathcal{T}_{t_j,t_k,M_1'}^{m_0} : \mathbb{R}^{d_1} \ni x \mapsto {}^t\widetilde{\Gamma}_{t_j,t_k,M_1'}^{m_0}\left(x - X_{t_k}^{m_0}\right) \in \mathbb{R}^{d_1'} \ (d_1' \leq d_1), \tag{2.3}$$

that performs a linear combination of the components of $\left(x - X_{t_k}^{m_0}\right)$ using $\widetilde{\Gamma}_{t_j,t_k,M_1'}^{m_0}$ that contains some eigenvectors from $\Gamma_{t_j,t_k,M_1'}^{m_0}$ obtained with the eigenvalue decomposition

$$\Gamma_{t_j,t_k,M_1'}^{m_0} \Lambda_{t_j,t_k,M_1'}^{m_0} {}^t\Gamma_{t_j,t_k,M_1'}^{m_0} \tag{2.4}$$

of the regression matrix

$$\frac{1}{M_1'} \sum_{m_1=M_1+1}^{M_1+M_1'} \left(X_{t_j,t_k}^{m_0,m_1} - X_{t_k}^{m_0}\right)^t \left(X_{t_j,t_k}^{m_0,m_1} - X_{t_k}^{m_0}\right) \tag{2.5}$$

where ^{t} is the transpose operator.

Once factorization (2.4)=(2.5) is performed, we obtain the diagonal matrix $\Lambda_{t_j,t_k,M_1'}^{m_0} = \text{diag}\left(\left\{\lambda_{t_j,t_k,M_1'}^{m_0,l}\right\}_{l=1,\dots,d_1}\right)$ of decreasing positive eigenvalues. Then, we define $\widetilde{\Lambda}_{t_j,t_k,M_1'}^{m_0} = \text{diag}\left(\left\{\lambda_{t_j,t_k,M_1'}^{m_0,l}\right\}_{l=1,\dots,d_1'}\right)$ as the truncation of $\Lambda_{t_j,t_k,M_1'}^{m_0}$ with d_1' defined by

$$d_1' = \min\left\{k \in \{1,..,d_1''\},\ \sum_{l=1}^{k} \lambda_{t_j,t_k,M_1'}^{m_0,l} \geq p\% \sum_{l=1}^{d_1''} \lambda_{t_j,t_k,M_1'}^{m_0,l}\right\}, \tag{2.6}$$

where $p\% \in [95\%, 100\%]$, d_1'' keeps only eigenvalues that make the regression problem well-conditioned i.e. The ratio $\left.\frac{\lambda_{t_j,t_k,M_1'}^{m_0,l}}{\lambda_{t_j,t_k,M_1'}^{m_0,1}}\right|_{l=1,\dots,d_1''}$ has to be bigger than 10^{-6} in single precision or bigger than 10^{-15} in double precision floating representation [27]. In addition to ensuring a well-conditioned regression problem, equality (2.6) also performs a principal component analysis [27]. At the same time that we set the components of $\widetilde{\Lambda}_{t_j,t_k,M_1'}^{m_0}$, we define the matrix $\widetilde{\Gamma}_{t_j,t_k,M_1'}^{m_0}$ that contains only the eigenvectors in $\Gamma_{t_j,t_k,M_1'}^{m_0}$ that are associated to $\widetilde{\Lambda}_{t_j,t_k,M_1'}^{m_0}$.

The regression with respect to ${}^t\widetilde{\Gamma}_{t_j,t_k,M_1'}^{m_0}\left(X_{t_j,t_k}^{m_0,m_1} - X_{t_k}^{m_0}\right) \in \mathbb{R}^{d_1'}$ with $d_1' \leq d_1$, instead of $\left(X_{t_j,t_k}^{m_0,m_1} - X_{t_k}^{m_0}\right) \in \mathbb{R}^{d_1}$, involves the inversion of the diagonal matrix $\widetilde{\Lambda}_{t_j,t_k,M_1'}^{m_0}$ which replaces the whole regression matrix (2.5). Since $\widetilde{\Lambda}$ is bounded below away from zero, its inverse is bounded and the same for the regression procedure. This stabilizes the computation of the regression estimator whose expression is detailed in Section 3. Besides, since we have a large number of regression matrices, we can batch compute these inversions like explained in [3]. The latter reference presents a recipe to resolve efficiently and accurately large number of small symmetric linear systems on GPUs. In particular, the authors of [3] present a batch parallelization strategy of Cuppen's divide & conquer algorithm (cf. [10]) suited for CMCLD.

As shown in the numerical examples of Section 5, performing a regression with respect to ${}^t\widetilde{\Gamma}_{t_j,t_k,M_1'}^{m_0}\left(X_{t_j,t_k}^{m_0,m_1} - X_{t_k}^{m_0}\right)$ for any couple of fine increments $(t_j < t_k)$ is mostly both unacessary and computationally heavy. Thus, only regressions with respect to ${}^t\widetilde{\Gamma}_{s_j,s_k,M_1'}^{m_0}\left(X_{s_j,s_k}^{m_0,m_1} - X_{s_k}^{m_0}\right)$ are considered, with (s_j, s_k) being a couple that take their values in $\mathcal{S} \times \left(\mathcal{S} \cap [\underline{s_j}, \overline{s_j}]\right)$. The set of the possible values $\mathcal{S}$ is a subset of the fine time

discretization $\mathbb{S}$. The couple $(\underline{s_j}, \overline{s_j}) \in (\mathcal{S} \cap]s_j, T])^2$ sets the minimal and the maximal time increment of the performed regressions conditionally on the realization $X_{s_j}^{m_0}$ of the Markov process X. Given a coarse discretization set $\mathcal{S}$, definitions 3.1 and 3.2 express the value of $(\underline{s}, \overline{s})_{s \in \mathcal{S}}$.

The choice of $\mathcal{S}$ and of $\mathcal{S} \ni s_j \mapsto (\underline{s_j}, \overline{s_j}) \in (\mathcal{S} \cap]s_j, T])^2$ have to be specific to each application. In some examples like the one presented in Section 5.6, $\mathcal{S} = \mathbb{S}$ or $\mathcal{S}$ has to be fine enough because fine discretization is needed for the corresponding BSDE. Also for some situations as the Hamilton-Jacobi-Bellman simulation presented in Section 5.4, one has to cut the bias propagation and set $\overline{s_j} < T$ for some values $s_j \in \mathcal{S}$. Consequently, $\mathcal{S}$ and $\mathcal{S} \ni s \mapsto (\underline{s}, \overline{s}) \in (\mathcal{S} \cap]s, T])^2$ are not predefined but rather add a degree of freedom to increase accuracy and stop the bias propagation. In Conditional Monte Carlo Learning for Diffusions part II, for $i = 0, ..., L - L'$, we define $(s_j)_{j=0,...,2^L}$ to be iteratively equal to $(T - s_j^i)_{j=0,...,2^L}$ starting with a homogeneously distributed sequence where each term is repeated $2^{L-L'}$ times as follows

$$s_j^0 = \left\lceil \frac{j 2^{L'}}{2^L} \right\rceil \frac{T}{2^{L'}} \text{ where } \lceil \cdot \rceil \text{ is the ceiling function.} \tag{2.7}$$

Denoting $\mathcal{S}^i$ the set of values taken by $(T - s_j^i)_{j=0,...,2^L}$, for example $\mathcal{S}^0 = \{0, T/2^{L'}, ..., (2^{L'} - 1)T/2^{L'}, T\}$, it is possible to set a refinement strategy.

Given that $(s_j)_{j \in \{0,...,2^L\}}$ is a decreasing, and not strictly decreasing, sequence of coarse increments, we need to define on $\mathcal{S}$ a new operator $\delta^{\mathcal{S}}$ that associates to each $s \in \mathcal{S}$ the next increment in $\mathcal{S}$. For a fixed index $j \in \{1, ..., 2^L\}$, we define $\delta_{s_j}^{\mathcal{S}}(\cdot)$ on $(s_k)_{k \leq j}$, taken its values in $\mathcal{S} \cap [s_j, \overline{s_j}]$, by

$$\delta_{s_j}^{\mathcal{S}}(s_k) = \min\left(\overline{s_j}, \min\{s \in \mathcal{S}; s_k < s \leq \overline{s_j}\}\right) \tag{2.8}$$

with $\min(\emptyset) = \infty$.

When there is no confusion on the chosen set $\mathcal{S}$, we use δ_{s_j} notation instead of $\delta_{s_j}^{\mathcal{S}}$. When $s_k < \overline{s_j}$, we use $\delta^{\mathcal{S}}$ notation instead of $\delta_{s_j}^{\mathcal{S}}$. When there is no confusion on the chosen set $\mathcal{S}$ and $s_k < \overline{s_j}$, we simplify both indices and use δ instead of $\delta_{s_j}^{\mathcal{S}}$.

This time operator will be largely used and for a given set $\mathcal{S}$ it has the following properties

Pr1. $\underline{s_j} = \delta_{s_j}(s_j) = \delta(s_j)$.

Pr2. As long as $\max(s_{j_1}, s_{j_2}) \leq s_k < \min(\overline{s_{j_1}}, \overline{s_{j_2}})$, $\delta_{s_{j_1}}(s_k) = \delta_{s_{j_2}}(s_k) = \delta(s_k)$.

Pr3. The nth composition of δ_{s_j} denoted $\delta_{s_j}^n(\cdot)$ is equal to $\overline{s_j}$ when $n \geq |\mathcal{S} \cap]s_j, \overline{s_j}]|$ where $|\cdot|$ denotes the cardinal.

2.2 Illustration of coarse and fine approximations for conditional expectations

We consider here the following process

$$U_t = \mathbb{E}\left(f(X_T) \middle| X_t\right),$$

with a deterministic function f. Thus, we assume that there is no path dependence through the sum on the realizations of X as done in (f). In this path-independent

situation for the fixed time set (2.7), it is clear that one can simulate U using 1NMC without any need of regression and thus without using our method. However, we choose to illustrate our method on this simple case and we will see at the end of this section what could be the benefits.

For $j, j' \in \{1, ..., 2^L\}$ and time steps $s_{j'}, s_j \in \mathbb{S} - \{0\}$ with $s_{j'} < s_j$ and for a fixed outer trajectory $(X_{t_k}^{m_0})_{k=0,...,2^L}$, let us assume that we want to simulate one realization of $U_{s_{j'}}$ and of U_{s_j}. A straight way to do it is to draw the inner trajectories $\{X_{s_{j'},T}^{m_0,m_1}\}_{m_1=1,...,M_1}$ and $\{X_{s_j,T}^{m_0,m_1}\}_{m_1=1,...,M_1}$, as in Figure 1, then average with respect to m_1 on the realizations $f(X_{s_{j'},T}^{m_0,m_1})$ and $f(X_{s_j,T}^{m_0,m_1})$ respectively. If s_j and $s_{j'}$ are close to each other in some sense[1], our method makes possible to simulate U_{t_k} for any $t_k \in [s_{j'}, s_j)$ using

$$U_{t_k}^{m_0} = \widetilde{u}_{t_k,T}^{m_0,\mathcal{S}} = \frac{1}{M_1} \sum_{m_1=1}^{M_1} \overline{u}_{s_{j'},s_j}^{m_0,\mathcal{S}}(X_{t_k,s_j}^{m_0,m_1}).$$

Thus, $\overline{u}_{s_{j'},s_j}^{m_0,\mathcal{S}}(x)$ and $X_{t_k,s_j}^{m_0,m_1}$ replace respectively $f(x)$ and $X_{t_k,T}^{m_0,m_1}$ involved in a standard nested simulation that would approximate the m_0 realization of U_{t_k} by $\displaystyle\sum_{m_1=1}^{M_1} \frac{f(X_{t_k,T}^{m_0,m_1})}{M_1}$.

The introduced

- $\overline{u}_{s,s'}^{m_0,\mathcal{S}}(x)$ $(0 \leq s < s' < \overline{s})$, called coarse approximation, is the regression representation of $\mathbb{E}\left(f(X_T)\middle|X_{s'} = x\right)$ computed using the inner trajectories $\{X_{s,s'}^{m_0,m_1}\}_{1 \leq m_1 \leq M_1}$.
- $\widetilde{u}_{s,\overline{s}}^{m_0,\mathcal{S}}$ $(0 \leq s < T)$, called fine approximation, is the Monte Carlo representation of $\mathbb{E}\left(f(X_T)\middle|X_s\right)$ computed by averaging on the regression representations $\{\overline{u}_{s,\underline{s}}^{m_0,\mathcal{S}}(X_{s,\underline{s}}^{m_0,m_1})\}_{1 \leq m_1 \leq M_1}$ with $\underline{s} = \delta(s)$. The index $\overline{s}$ in $\widetilde{u}_{s,\overline{s}}^{m_0,\mathcal{S}}$ provides the depth of the regression learning conditionally on the realization $X_s^{m_0}$.

Below, we explain how $\overline{u}_{s,s'}^{m_0,\mathcal{S}}(x)$ should be computed and how $\overline{s}$ should be set.

First of all, since $s_{j'}$ and s_j are assumed to be "close enough" one can consider the same learning depth $\overline{s_j} = \overline{s_{j'}} = t_{2^L} = T$ and define

$$(\overline{u}_{s_{j'}T}) \;\&\; (\overline{u}_{s_jT}) \quad \overline{u}_{s_{j'},T}^{m_0,\mathcal{S}}(x) = \overline{u}_{s_j,T}^{m_0,\mathcal{S}}(x) = f(x)$$

and if $\underline{s_j} = T$ then

$$(\widetilde{u}_{s_j}) \quad U_{s_j}^{m_0} = \widetilde{u}_{s_j,T}^{m_0,\mathcal{S}} = \frac{1}{M_1} \sum_{m_1=1}^{M_1} \overline{u}_{s_j,T}^{m_0,\mathcal{S}}(X_{s_j,T}^{m_0,m_1}).$$

As will be presented in (3.1), we define

$$(\overline{u}_{s_{j'}s_j}) \quad \overline{u}_{s_{j'},s_j}^{m_0,\mathcal{S}}(x) = \widetilde{u}_{s_j,T}^{m_0,\mathcal{S}} + {}^t\mathcal{T}_{s_{j'},s_j,M_1'}^{m_0}(x) A_{s_{j'},s_j}^{m_0,\mathcal{S}},$$

where the adaptation of (3.3) makes

$$(A_{s_{j'}s_j}^T)\; A_{s_{j'},s_j}^{m_0,\mathcal{S}} = \frac{\left(\widetilde{\Lambda}_{s_{j'},s_j,M_1'}^{m_0}\right)^{-1}}{M_1} \sum_{m_1=1}^{M_1} \mathcal{T}_{s_{j'},s_j,M_1'}^{m_0}(X_{s_{j'},s_j}^{m_0,m_1}) \left[\overline{u}_{s_{j'},T}^{m_0,\mathcal{S}}\left(X_{s_{j'},T}^{m_0,m_1}\right) - \widetilde{u}_{s_j,T}^{m_0,\mathcal{S}}\right].$$

[1] Not necessary an Euclidean distance.

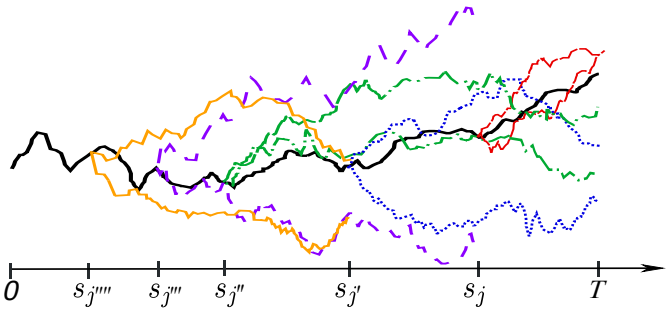


Figure 1: Given the realization of one outer trajectory (bold), we simulate inner trajectories to approximate U_{s_j}, $U_{s_{j'}}$, $U_{s_{j''}}$, $U_{s_{j'''}}$ and $U_{s_{j''''}}$.

If we add a third increment $s_{j''}$ (cf. Figure 1) such that $s_{j''}$, $s_{j'}$ and s_j are close enough, for any $t_k \in [s_{j''}, s_{j'})$ one can set $U_{t_k}^{m_0} = \widetilde{u}_{t_k,T}^{m_0,\mathcal{S}}$. The latter equality requires the definition of $\overline{u}_{s_{j''},s_{j'}}^{m_0,\mathcal{S}}$ which can be obtained from $(\overline{u}_{s_{j''}s_{j'}})$ (replace $s_{j'}$ by $s_{j''}$ and s_j by $s_{j'}$ in $(\overline{u}_{s_{j'}s_j})$) involving $\widetilde{u}_{s_{j'},T}^{m_0,\mathcal{S}}$ and $A_{s_{j''},s_{j'}}^{m_0,\mathcal{S}}$ that can be computed using $(A_{s_{j''}s_{j'}}^{s_j})$. The calculations in $(A_{s_{j''}s_{j'}}^{s_j})$ use $\widetilde{u}_{s_{j'},T}^{m_0,\mathcal{S}}$ and $\overline{u}_{s_{j''},s_j}^{m_0,\mathcal{S}}$ whose expression depends on $\widetilde{u}_{s_j,T}^{m_0,\mathcal{S}}$ and $A_{s_{j''},s_j}^{m_0,\mathcal{S}}$. Finally, $A_{s_{j''},s_j}^{m_0,\mathcal{S}}$ is the regression vector of $\overline{u}_{s_{j''},T}^{m_0,\mathcal{S}}$ around $\widetilde{u}_{s_j,T}^{m_0,\mathcal{S}}$. Subsequently, the computations of $\widetilde{u}_{s_{j''},T}^{m_0,\mathcal{S}}$, $\widetilde{u}_{s_{j'},T}^{m_0,\mathcal{S}}$ and $\widetilde{u}_{s_j,T}^{m_0,\mathcal{S}}$ involve the dependence structure given in (2.9).

$$
\begin{array}{ccccccccc}
\widetilde{u}_{s_{j''},T}^{m_0,\mathcal{S}} & \to & \overline{u}_{s_{j''},s_{j'}}^{m_0,\mathcal{S}} & \to & \widetilde{u}_{s_{j'},T}^{m_0,\mathcal{S}} & \to & \overline{u}_{s_{j'},s_j}^{m_0,\mathcal{S}} & \to & \widetilde{u}_{s_j,T}^{m_0,\mathcal{S}} \to \overline{u}_{s_j,T}^{m_0,\mathcal{S}} = f \\
 & & \searrow & & \uparrow & & \searrow & & \uparrow \\
 & & & & A_{s_{j''},s_{j'}}^{m_0,\mathcal{S}} & & & & A_{s_{j'},s_j}^{m_0,\mathcal{S}} \to \overline{u}_{s_{j'},T}^{m_0,\mathcal{S}} = f \\
 & & & & & \searrow & & & \\
 & & & & & \overline{u}_{s_{j''},s_j}^{m_0,\mathcal{S}} & \to & \widetilde{u}_{s_j,T}^{m_0,\mathcal{S}} & \to \overline{u}_{s_j,T}^{m_0,\mathcal{S}} = f \\
 & & & & & & \searrow & \uparrow & \\
 & & & & & & & A_{s_{j''},s_j}^{m_0,\mathcal{S}} & \to \overline{u}_{s_{j''},T}^{m_0,\mathcal{S}} = f
\end{array}
\tag{2.9}
$$

By adding other increments $s_{j'''}$ and $s_{j''''}$ (cf. Figure 1), it can happen that $s_{j''''}$, $s_{j'''}$ and s_j can no longer be considered close enough to each other. In this situation, a regression on a linear basis around $X_{s_j}^{m_0}$ would not be considered sufficient for inner trajectories that start at $X_{s_{j'''}}^{m_0}$ or $X_{s_{j''''}}^{m_0}$. To deal with this situation, one should introduce $(\underline{s_{j'''}}, \overline{s_{j'''}})$ and $(\underline{s_{j''''}}, \overline{s_{j''''}})$ that set the beginning and the ending of the family of successive regressions. For instance if $\overline{s_{j'''}} = s_j$ and $\underline{s_{j'''}} = s_{j''}$, one starts the backward induction associated to the increment $s_{j'''}$ by the final condition $\overline{u}_{s_{j'''},s_j}^{m_0,\mathcal{S}}(x) = \overline{u}_{s_{j''},s_j}^{m_0,\mathcal{S}}(x)$ instead of $\overline{u}_{s_{j'''},T}^{m_0,\mathcal{S}}(x) = f(x)$ and the dependence tree (2.9) becomes

$$
\begin{array}{ccccccccc}
\widetilde{u}_{s_{j'''},s_{j''}}^{m_0,\mathcal{S}} & \to & \overline{u}_{s_{j'''},s_{j''}}^{m_0,\mathcal{S}} & \to & \widetilde{u}_{s_{j''},s_{j'}}^{m_0,\mathcal{S}} & \to & \overline{u}_{s_{j''},s_{j'}}^{m_0,\mathcal{S}} & \to & \widetilde{u}_{s_{j'},s_j}^{m_0,\mathcal{S}} \to \overline{u}_{s_{j'},s_j}^{m_0,\mathcal{S}} \to \widetilde{u}_{s_j,T}^{m_0,\mathcal{S}} \ldots f \\
 & & \searrow & & \uparrow & & \searrow & & \uparrow \\
 & & & & A_{s_{j'''},s_{j''}}^{m_0,\mathcal{S}} & & & & A_{s_{j''},s_{j'}}^{m_0,\mathcal{S}} \\
 & & & & & \searrow & & & \\
 & & & & & \overline{u}_{s_{j'''},s_{j'}}^{m_0,\mathcal{S}} & \to & \widetilde{u}_{s_{j''},s_j}^{m_0,\mathcal{S}} & \to \overline{u}_{s_{j''},s_j}^{m_0,\mathcal{S}} \to \widetilde{u}_{s_j,T}^{m_0,\mathcal{S}} \ldots f \\
 & & & & & & \searrow & \uparrow & \\
 & & & & & & & A_{s_{j'''},s_{j'}}^{m_0,\mathcal{S}} & \\
 & & & & & & & \searrow & \\
 & & & & & & & \overline{u}_{s_{j'''},s_j}^{m_0,\mathcal{S}} = \overline{u}_{s_{j''},s_j}^{m_0,\mathcal{S}} &
\end{array}
\tag{2.10}
$$

In Figure 1, we also set $\overline{s_{j''''}} = s_{j'}$ as well as $\underline{s_{j''''}} = s_{j'''}$ and the tree (2.10) can be further changed to include the dependency structure induced by $s_{j''''}$. Indeed, we urge the reader to check that (2.10) can be as easily completed as done for (2.9) to include the dependency structure induced by $s_{j''''}$.

Even with the simple example presented in this subsection, one can show the benefit of this method. Indeed, in addition to a fine simulation of U using $\widetilde{u}$, this method defines a set of functions $\overline{u}$ that can be considered as coarse conditional approximation of U. These conditional approximations can be used as forward components of another functional. For instance, given the example presented above and illustrated in Figure 1, the simulation of an m_0 realization of $V_{s_{j''}} = \mathbb{E}\left((U_{s_j} - U_{s_{j'}})_+ \middle| X_{s_{j''}}\right)$ can be done with

$$\widetilde{V}^{m_0}_{s_{j''}} = \frac{1}{M_1}\sum_{m_1=1}^{M_1}\left(\left[\overline{u}^{m_0,\mathcal{S}}_{s_{j''},s_j}(X^{m_0,m_1}_{s_{j''},s_j}) - \overline{u}^{m_0,\mathcal{S}}_{s_{j''},s_{j'}}(X^{m_0,m_1}_{s_{j''},s_{j'}})\right]_+\right).$$

These functions $\overline{u}$ can be also used for risk measures. For example, the conditional value at risk $\mathbb{V}a\mathbb{R}^{\alpha\%}\left[U_{s_j} - U_{s_{j'}} \middle| X_{s_{j''}}\right]$ of level $\alpha\%$ can be computed after sorting the values $\left(\overline{u}^{m_0,\mathcal{S}}_{s_{j''},s_j}(X^{m_0,m_1}_{s_{j''},s_j}) - \overline{u}^{m_0,\mathcal{S}}_{s_{j''},s_{j'}}(X^{m_0,m_1}_{s_{j''},s_{j'}})\right)_{1\le m_1\le M_1}$.

Remark 2.1. *Referring to Figure 1, for any g, when $\mathbb{E}\left(g(U_{s_{j'''}})\middle|X_{s_{j''''}}\right)$, $\mathbb{E}\left(g(U_{s_{j''}})\middle|X_{s_{j''''}}\right)$ and $\mathbb{E}\left(g(U_{s_{j'}})\middle|X_{s_{j''''}}\right)$ are well defined their simulation can be directly performed using $\overline{u}^{m_0,\mathcal{S}}_{s_{j''''},s_{j'''}}$, $\overline{u}^{m_0,\mathcal{S}}_{s_{j''''},s_{j''}}$ or $\overline{u}^{m_0,\mathcal{S}}_{s_{j''''},s_{j'}}$. This is not the case for $\mathbb{E}\left(g(U_{s_j})\middle|X_{s_{j''''}}\right)$ since $\overline{u}^{m_0,\mathcal{S}}_{s_{j''''},s_j}$ were not computed because $\overline{s_{j''''}} = s_{j'} < s_j$.*

The other benefit of our methodology is the possibility to have a parareal alike implementation [23] and thus make the algorithm parallel in time in addition to have it parallel in paths. Indeed, refering to Figure 1, if we associate the final conditions $\overline{u}^{m_0,\mathcal{S}}_{s_{j''},s_{j'}}$ and $\overline{u}^{m_0,\mathcal{S}}_{s_{j'},s_j}$ respectively to each subinterval $[s_{j''}, s_{j'})$ and $[s_{j'}, s_j)$, we can perform concurrent calculations on these intervals.

3 General methodology applied to BSDEs with a Markov forward process

Based on what was presented previously, we detail in Section 3.1 the simulation of approximations of U defined by (f). Section 3.2 illustrates the adaptation of this new method to BSDEs.

3.1 Fine and coarse approximations

Considering the discretization sequence $(s_j)_{j=0,\dots,2^L}$ that takes its values in the set $\mathcal{S} \subset \mathbb{S}$, we use a learning procedure to associate to each scenario m_0 and to each discretization set $\mathcal{S}$ a couple of function families $(\widetilde{u}^{m_0,\mathcal{S}}, \overline{u}^{m_0,\mathcal{S}})$.

Now, for given indices $k, j \in \{1, \dots, 2^L\}$ $(k < j)$ that satisfy $s_j < s_k < \overline{s_j}$, for $x \in \mathbb{R}^{d_1}$ and $s \in \{s_k, s_k + \Delta_t, \dots, \delta(s_k) - \Delta_t\}$, we define two approximation levels: A coarse approximation around $X^{m_0}_{s_k}$ conditionally on $X^{m_0}_{s_j}$ defined by

$$\overline{u}^{m_0,\mathcal{S}}_{s_j,s_k}(x) = \widetilde{u}^{m_0,\mathcal{S}}_{s_k,\overline{s_k}} + {}^t\mathcal{T}^{m_0}_{s_j,s_k,M_1'}(x)B^{m_0,\mathcal{S}}_{s_j,s_k}, \tag{3.1}$$

and a fine approximation at $X_s^{m_0}$ defined by

$$\widetilde{u}_{s,\overline{s_k}}^{m_0,\mathcal{S}} = \frac{1}{M_1}\sum_{m_1=1}^{M_1}\left[\overline{u}_{s_k,\delta(s_k)}^{m_0,\mathcal{S}}(X_{s,\delta(s_k)}^{m_0,m_1}) + \sum_{t_{l+1}>s}^{\delta(s_k)} f(t_l, X_{s,t_l}^{m_0,m_1}, X_{s,t_{l+1}}^{m_0,m_1})\right]. \tag{3.2}$$

The value of $B_{s_j,s_k}^{m_0,\mathcal{S}}$ is given by

$$B_{s_j,s_k}^{m_0,\mathcal{S}} = (\widetilde{\Lambda}_{s_j,s_k,M_1'}^{m_0})^{-1}\frac{1}{M_1}\sum_{m_1=1}^{M_1}\mathcal{B}_{s_j,s_k,M_1'}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}(X_{s_j,s_k:\delta_{s_j}(s_k)}^{m_0,m_1}) \tag{3.3}$$

where $X_{s_j,s_k:\delta_{s_j}(s_k)}^{m_0,m_1} = \left(X_{s_j,s_k}^{m_0,m_1}, X_{s_j,s_k+\Delta_t}^{m_0,m_1}, ..., X_{s_j,\delta_{s_j}(s_k)-\Delta_t}^{m_0,m_1}, X_{s_j,\delta_{s_j}(s_k)}^{m_0,m_1}\right)$ and $\mathcal{B}_{s_j,s_k,M_1'}^{m_0,\mathcal{S},\delta_{s_j}(s_k)} : \Omega \times \mathbb{R}^{d_1(\delta_{s_j}(s_k)-s_k)/\Delta_t} \ni (\omega, x_1, ..., x_{(\delta_{s_j}(s_k)-s_k)/\Delta_t}) \to \Omega \times \mathbb{R}^{d_1'}$ is $\mathcal{F}_{\delta_{s_j}(s_k)} \otimes \mathcal{B}(\mathbb{R}^{d_1(\delta_{s_j}(s_k)-s_k)/\Delta_t})$-measurable and defined by

$$\mathcal{B}_{s_j,s_k,M_1'}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}(x) = \mathcal{T}_{s_j,s_k,M_1'}^{m_0}(x_1)\underbrace{\left[\begin{array}{l}\overline{u}_{s_j,\delta_{s_j}(s_k)}^{m_0,\mathcal{S}}\left(x_{\frac{\delta_{s_j}(s_k)-s_k}{\Delta_t}}\right) - \widetilde{u}_{s_k,\delta_{s_k}(s_k)}^{m_0,\mathcal{S}} \\ + \sum_{l=1}^{\frac{\delta_{s_j}(s_k)-s_k}{\Delta_t}-1} f(t_{k_{s_k}+l}, x_l, x_{l+1})\end{array}\right]}_{\mathbb{B}_{s_j,s_k}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}(x)} \tag{3.4}$$

where $k_{s_k} = s_k/\Delta_t - 1$ and $x = (x_1, ..., x_{(\delta_{s_j}(s_k)-s_k)/\Delta_t})$.

$\mathcal{T}$ involved in (3.1) was already defined in (2.3). Regarding the regression vector $B_{s_j,s_k}^{m_0,\mathcal{S}}$, its value can be seen as an estimation of the vector $a \in \mathbb{R}^{d_1'}$ that minimizes the quadratic error given by

$$\mathbb{E}\left[\mathbb{B}_{s_j,s_k}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}(X_{s_j,s_k:\delta_{s_j}(s_k)}^{m_0,m_1}) - {}^t a\mathcal{T}_{s_j,s_k,M_1'}^{m_0}(X_{s_j,s_k}^{m_0,m_1})\right]^2. \tag{3.5}$$

To complete this inductive interconnected backward definition of $\overline{u}$ and $\widetilde{u}$, we set the final coarse approximation to

$$\overline{u}_{s_j,\overline{s_j}}^{m_0,\mathcal{S}}(x) = \begin{cases} f(T,x) & \text{if } \overline{s_j} = T, \\ \overline{u}_{\underline{s_j},\overline{s_j}}^{m_0,\mathcal{S}}(x) = \overline{u}_{\delta(s_j),\overline{s_j}}^{m_0,\mathcal{S}}(x) & \text{if } \overline{s_j} < T, \end{cases} \tag{3.6}$$

where the values $s_j, \overline{s_j} \in (\mathcal{S} \cap]s_j, T])^2$ with $\underline{s_j} < \overline{s_j}$, expressed in Definition 3.1, delimit the learning depth of the performed regressions conditionally on the realization $X_{s_j}^{m_0}$ of the Markov process X. In numerical examples of Section 5, we see that $\overline{s_j}$ and $\underline{s_j}$ are needed when the bias becomes important because either T is sufficiently big or the variance produced by X is large enough. Otherwise, (3.6) can be replaced by $\overline{u}_{s_j,T}^{m_0,\mathcal{S}}(x) = f(t_{2^L}, x) = f(T, x)$. In CMCLDII [2], we suggest a method of iterative actualization of the couple $\{(\underline{s}, \overline{s})\}_{s\in\mathcal{S}}$ when we refine the discretization set $\mathcal{S}$.

According to equations (3.1), (3.2), (3.3), (3.4) and (3.6), the functions $\overline{u}$ and $\widetilde{u}$ are defined backwardly. When $\widetilde{u}$ is a straight Monte Carlo involving $\overline{u}$, the latter is defined using a regression around a point at which we expressed $\widetilde{u}$. Consequently, $\overline{u}$ can be seen as a conditional first order Taylor expansion around the first layer of trajectories $(X_{t_k}^{m_0})_{k=1,...,2^L}^{m_0=1,...,M_0}$. The term of order zero in this expansion is played by $\widetilde{u}$, where the term ${}^t\mathcal{T}_{s_j,s_k,M_1'}^{m_0}(x)B_{s_j,s_k}^{m_0,\mathcal{S}}$, deduced from the minimization of (3.5), plays the order one.

Remark 3.1. *1. Since we do not want to increase further the algorithm complexity by considering higher order terms, the definition of $\overline{u}$ involves only linear terms for regressions around $X_{s_k}^{m_0}$.*

2. When the dimension d_1 is not too high, it is possible to regress the residual of the first regression on higher order terms. These successive regressions do not increase drastically the complexity when compared to the standard procedure. Nevertheless, as it separates regression with respect to first order terms and regression with respect to higher order terms, it loses orthogonality between first and higher order terms.

3. In case X is a martingale, the linearity simplifies further computations since, for instance, (3.2) can be replaced by

$$\widetilde{u}_{s,\overline{s_k}}^{m_0,\mathcal{S}} = \overline{u}_{s_k,\delta(s_k)}^{m_0,\mathcal{S}}(X_s^{m_0}) + \frac{1}{M_1}\sum_{m_1=1}^{M_1}\sum_{t_{l+1}>s}^{\delta(s_k)} f(t_l, X_{s,t_l}^{m_0,m_1}, X_{s,t_{l+1}}^{m_0,m_1}).$$

Definition 3.1. *Given a discretization set $\mathcal{S} \subset \mathbb{S}$*

- *For any $s \in \mathcal{S}$, $\underline{s}$ is set to be equal to $\delta(s)$ and $\overline{s}$ is set backwardly to be the largest discretization time $u \in \mathcal{S} \cap]s, \overline{\delta(s)}]$ that satisfies*

$$\left| \frac{1}{M_0}\sum_{m_0=1}^{M_0} \left(\widetilde{u}_{s,u}^{m_0,\mathcal{S}} - \widetilde{u}_{\delta(s),\overline{\delta(s)}}^{m_0,\mathcal{S}} - \sum_{t_{l+1}>s}^{\delta(s)} f(t_l, X_{t_l}^{m_0}, X_{t_{l+1}}^{m_0}) \right) \right| < \epsilon_s^{\mathcal{S}} \tag{3.7}$$

  where $\{\epsilon_s^{\mathcal{S}}\}_{s\in\mathcal{S}}$ is a family of positive bias tuning parameters.

- *For $k, j \in \{1, ..., 2^L\}$ $(k < j)$ that satisfy $s_j < s_k \leq \overline{s_j} < t_{2^L} = T$, the simulation $U_{s_j,s_k}^{m_0,m_1}$ of U around $X_{s_k}^{m_0}$ conditionally on $X_{s_j}^{m_0}$ is set to be equal to $\overline{u}_{s_j,s_k}^{m_0,\mathcal{S}}(X_{s_j,s_k}^{m_0,m_1})$ where $\overline{u}$ is given in (3.1) and (3.6).*

- *For $k \in \{1, ..., 2^L\}$ and $s \in \{s_k, s_k + \Delta_t, ..., \delta(s_k) - \Delta_t\} - \{0\}$, the simulation $U_s^{m_0}$ of U at $X_s^{m_0}$ is set to be equal to $\widetilde{u}_{s,\overline{s_k}}^{m_0,\mathcal{S}}$ with $\widetilde{u}$ expressed in (3.2).*

- *The average U_0^{lear} of learned values on U_0 is equal to*

$$U_0^{lear} = \frac{1}{M_0}\sum_{m_0=1}^{M_0} \widetilde{u}_{0,\overline{0}}^{m_0,\mathcal{S}} \tag{3.8}$$

  and the simulated value U_0^{sim} of U_0 is equal to

$$U_0^{sim} = \frac{1}{M_0}\sum_{m_0=1}^{M_0} \left[\widetilde{u}_{\delta(0),\overline{\delta(0)}}^{m_0,\mathcal{S}} + \sum_{t_{l+1}>0}^{\delta(0)} f(t_l, X_{t_l}^{m_0}, X_{t_{l+1}}^{m_0}) \right] \tag{3.9}$$

  with $\widetilde{u}$ expressed in (3.2).

The definition of $\underline{s}$ is straightforward since we take $\underline{s} = \delta(s)$. To set the value of $\overline{s}$, we need to know $\overline{\delta(s)}$ whose value involves the one of $\overline{\delta^2(s)}$ and so on till T. In [2], Section 3.1, we give an example of how to update the values of $(\underline{s}, \overline{s})_{s\in\mathcal{S}}$ when the coarse discretization $\mathcal{S}$ gets finer.

U^{m_0,m_1} can be seen as the inner or second layer approximation of U and U^{m_0} can be seen as the outer or first layer approximation of U. When U^{m_0,m_1} is only defined on $\mathcal{S}$, it is remarkable to see that U^{m_0} is defined on the whole fine discretization set $\mathbb{S}$. At $t = 0$, inner and outer trajectories coincide that yield U_0^{lear} and U_0^{sim} as two possible outer first layer approximations of U_0. For any $\mathcal{S}$, it is natural to have $\epsilon_s^{\mathcal{S}}$ proportional to the value of the estimation $\frac{1}{M_0}\sum_{m_0=1}^{M_0} \widetilde{u}_{s,\overline{s}}^{m_0,\mathcal{S}}$. Used to control the bias, the choice of $\epsilon_s^{\mathcal{S}}$ has also to take into account the confidence interval of the estimator of the left side of inequality (3.7). Based on equality (1.1), (3.7) is quite sufficient in the considered examples to have almost unbiased estimates. CMCLDII [2] introduces a more stringent local bias control that can be used when (3.7) is not sufficient.

3.2 BSDEs with a Markov forward process

In Section 2.2, we saw the implementation of CMCLD on a simple problem and we showed its benefits when one has to simulate functionals of functionals of a Markov process. BSDEs with a Markov forward process are specific functionals of functionals of a Markov process. After [26], BSDEs became very widely studied, especially in the quantitative finance community starting with [12]. Here we adapt CMCLD to the One step forward Dynamic Programming (ODP) scheme for discrete BSDEs

$$(ODP)\quad Y_T = \zeta \text{ and for } k < 2^L \begin{cases} Y_{t_k} = \mathbb{E}_{t_k}[Y_{t_{k+1}} + \Delta_t f(t_k, Y_{t_{k+1}}, Z_{t_k})], \\ Z_{t_k} = \mathbb{E}_{t_k}[Y_{t_{k+1}}(W_{t_{k+1}} - W_{t_k})/\Delta_t]. \end{cases}$$

(ODP) was studied for instance in [14, 22]. We consider $\zeta = f(T, X_T)$ to be some square integrable random variable that depends on X_T. Given a discretization sequence $(s_j)_{j=0,\dots,2^L} \in \mathcal{S}$ and referring to (2.1) and (2.2), the simulation of X involves the increments of an $\mathbb{R}^{d_2}$-Brownian motion W with $\xi_{t_k}^{m_0} = W_{t_k}^{m_0} - W_{t_{k-1}}^{m_0}$ and $\xi_{s_j,t_k}^{m_0,m_1} = W_{s_j,t_k}^{m_0,m_1} - W_{s_j,t_{k-1}}^{m_0,m_1}$ where $W^1, \dots, W^{M_0}$ are independent realizations of W with

$$W_{s_j,t_k}^{m_0,m_1} = W_{s_j,t_{k-1}}^{m_0,m_1} + \Delta W_{s_j,t_k}^{m_0,m_1} \text{ and } \quad W_{s_j,s_j}^{m_0,m_1}\Big|_{m_1=1,\dots,M_1+M_1'} = W_{s_j}^{m_0},$$

$(\Delta W_{s_j,t_k}^{m_0,m_1})_{k\in\{j,\dots,2^L\},j\in\{1,\dots,2^L\}}^{(m_0,m_1)\in\{1,\dots,M_0\}\times\{1,\dots,M_1+M_1'\}}$ are independent Brownian motion increments independent from $W^1, \dots, W^{M_0}$ with $\mathbb{E}([\Delta W_{s_j,t_k}^{m_0,m_1}]^2) = \Delta_t$. As pointed out in Section 2.1, if an inner trajectory $\{X^{m_0,m_1}\}$ is needed several times in the backward induction, we simulate independent copies of it and thus independent copies of ξ^{m_0,m_1} and use each copy once.

For given indices $k, j \in \{1, \dots, 2^L\}$ with $k < j$ that satisfy $s_j < s_k < \overline{s_j}$ and using $\delta_{s_j}(s_k)$ defined in (2.8), we also set $\Delta W_{s_j,s_k,\delta_{s_j}(s_k)}^{m_0,m_1} = W_{s_j,\delta_{s_j}(s_k)}^{m_0,m_1} - W_{s_j,s_k}^{m_0,m_1}$. For each k, the Borel $\mathcal{B}(\mathbb{R})\otimes\mathcal{B}(\mathbb{R}^{d_2})$-measurable driver $f(t_k, \cdot, \cdot)$ is assumed to satisfy Lipschitz condition of Section 4.

Given the discretization set $\mathcal{S}$, one can define two coarse approximations around $X_{s_k}^{m_0}$ conditionally on $X_{s_j}^{m_0}$ given by

$$\overline{y}_{s_j,s_k}^{m_0,\mathcal{S}}(x) = \widetilde{y}_{s_k,\overline{s_k}}^{m_0,\mathcal{S}} + {}^t\mathcal{T}_{s_j,s_k,M_1'}^{m_0}(x) C_{s_j,s_k}^{m_0,\mathcal{S}}, \tag{3.10}$$

$$ {}^{t}\overline{z}^{m_0,\mathcal{S}}_{s_j,s_k}(x) = {}^{t}\widetilde{z}^{m_0,\mathcal{S}}_{s_k,\overline{s_k}} + {}^{t}\mathcal{T}^{m_0}_{s_j,s_k,M_1'}(x) D^{m_0,\mathcal{S}}_{s_j,s_k}, \tag{3.11} $$

as well as two fine approximations at $X^{m_0}_s$, for $s \in \{s_k, s_k + \Delta_t, ..., \delta_{s_j}(s_k) - \Delta_t\}$ with $\Delta_s = \delta_{s_j}(s_k) - s$ and $\Delta_{s_k} = \delta_{s_j}(s_k) - s_k$, given by

$$ \widetilde{y}^{m_0,\mathcal{S}}_{s,\overline{s_k}} = \frac{1}{M_1} \sum_{m_1=1}^{M_1} \begin{bmatrix} \Delta_s f(s_k, \overline{y}^{m_0,\mathcal{S}}_{s_k,\delta_{s_j}(s_k)}(X^{m_0,m_1}_{s,\delta_{s_j}(s_k)}), \widetilde{z}^{m_0,\mathcal{S}}_{s,\overline{s_k}}) \\ + \overline{y}^{m_0,\mathcal{S}}_{s_k,\delta_{s_j}(s_k)}(X^{m_0,m_1}_{s,\delta_{s_j}(s_k)}) \end{bmatrix}, \tag{3.12} $$

$$ \widetilde{z}^{m_0,\mathcal{S}}_{s,\overline{s_k}} = \frac{1}{M_1 \Delta_s} \sum_{m_1=1}^{M_1} \overline{y}^{m_0,\mathcal{S}}_{s_k,\delta_{s_j}(s_k)}(X^{m_0,m_1}_{s,\delta_{s_j}(s_k)}) \left(W^{m_0,m_1}_{s,\delta_{s_j}(s_k)} - W^{m_0}_s \right) \tag{3.13} $$

and we set the final coarse approximation to

$$ \overline{y}^{m_0,\mathcal{S}}_{s_j,\overline{s_j}} = \begin{cases} f(t_{2^L}, X^{m_0,m_1}_{s_j,t_{2^L}}) & \text{if } \overline{s_j} = t_{2^L}, \\ \overline{y}^{m_0,\mathcal{S}}_{\underline{s_j},\overline{s_j}}(X^{m_0,m_1}_{s_j,\overline{s_j}}) = \overline{y}^{m_0,\mathcal{S}}_{\delta_{s_j}(s_j),\overline{s_j}}(X^{m_0,m_1}_{s_j,\overline{s_j}}) & \text{if } \overline{s_j} < t_{2^L}, \end{cases} \tag{3.14} $$

where the values $\underline{s_j}, \overline{s_j} \in (\mathcal{S} \cap]s_j, T])^2$ with $\underline{s_j} < \overline{s_j}$, expressed in Definition 3.2, delimit the learning depth of the performed regressions conditionally on the realization $X^{m_0}_{s_j}$ of the Markov process X.

Since $\mathcal{T}$ was already expressed in (2.3), to complete this inductive interconnected definition of $(\overline{y}, \widetilde{y}, \overline{z}, \widetilde{z})$, we set the vector $C^{m_0,\mathcal{S}}_{s_j,s_k}$ and the matrix $D^{m_0,\mathcal{S}}_{s_j,s_k}$ to be equal to

$$ C^{m_0,\mathcal{S}}_{s_j,s_k} = \frac{(\widetilde{\Lambda}^{m_0}_{s_j,s_k,M_1'})^{-1}}{M_1} \sum_{m_1=1}^{M_1} \mathcal{Y}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}_{s_j,s_k,M_1'}(X^{m_0,m_1}_{s_j,s_k}, X^{m_0,m_1}_{s_j,\delta_{s_j}(s_k)}), \tag{3.15} $$

$$ D^{m_0,\mathcal{S}}_{s_j,s_k} = \frac{(\widetilde{\Lambda}^{m_0}_{s_j,s_k,M_1'})^{-1}}{M_1} \sum_{m_1=1}^{M_1} \mathcal{Z}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}_{s_j,s_k,M_1'}(X^{m_0,m_1}_{s_j,s_k}, \Delta W^{m_0,m_1}_{s_j,s_k,\delta_{s_j}(s_k)}, X^{m_0,m_1}_{s_j,\delta_{s_j}(s_k)}), \tag{3.16} $$

with $\mathcal{Y}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}_{s_j,s_k,M_1'}(x',x) = \mathcal{T}^{m_0}_{s_j,s_k,M_1'}(x') \mathbb{Y}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}_{s_j,s_k}(x',x)$ is $\mathcal{F}_{\delta_{s_j}(s_k)} \otimes \mathcal{B}(\mathbb{R}^{2d_1})$-measurable and $\mathcal{Z}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}_{s_j,s_k,M_1'}(x',w,x) = \mathcal{T}^{m_0}_{s_j,s_k,M_1'}(x')^t \mathbb{Z}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}_{s_j,s_k}(w,x)$ is a vector function measurable with respect to $\mathcal{F}_{\delta_{s_j}(s_k)} \otimes \mathcal{B}(\mathbb{R}^{2d_1+d_2})$, where

$$ \mathbb{Y}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}_{s_j,s_k}(x',x) = \begin{bmatrix} \Delta_{s_k} f(s_k, \overline{y}^{m_0,\mathcal{S}}_{s_j,\delta_{s_j}(s_k)}(x), \overline{z}^{m_0,\mathcal{S}}_{s_j,s_k}(x')) \\ + \overline{y}^{m_0,\mathcal{S}}_{s_j,\delta_{s_j}(s_k)}(x) - \widetilde{y}^{m_0,\mathcal{S}}_{s_k,\overline{s_k}} \end{bmatrix} \tag{3.17} $$

and

$$ \mathbb{Z}^{m_0,\mathcal{S},\delta_{s_j}(s_k)}_{s_j,s_k}(w,x) = \overline{y}^{m_0,\mathcal{S}}_{s_j,\delta_{s_j}(s_k)}(x) \frac{w}{\Delta_{s_k}} - \widetilde{z}^{m_0,\mathcal{S}}_{s_k,\overline{s_k}}. \tag{3.18} $$

From equations above, one can associate quadratic minimization problems to $C^{m_0,\mathcal{S}}_{s_j,s_k}$ and to $D^{m_0,\mathcal{S}}_{s_j,s_k}$, as done for $B^{m_0,\mathcal{S}}_{s_j,s_k}$ in (3.5). In the same fashion as in Definition 3.1, we define the double layer approximations (Y^{m_0}, Z^{m_0}) and $(Y^{m_0,m_1}, Z^{m_0,m_1})$ of functionals Y and Z.

Definition 3.2. *Given a discretization set* $\mathcal{S} \subset \mathbb{S}$

- *For any* $s \in \mathcal{S}$, $\underline{s}$ *is set to be equal to* $\delta(s)$ *and* $\overline{s}$ *is set backwardly to be the largest discretization time* $u \in \mathcal{S} \cap]s, \overline{\delta(s)}]$ *that satisfies*

$$\left| \frac{1}{M_0} \sum_{m_0=1}^{M_0} \left(\widetilde{y}_{s,u}^{m_0,\mathcal{S}} - \widetilde{y}_{\delta(s),\overline{\delta(s)}}^{m_0,\mathcal{S}} - (\delta(s) - s) f(s, \widetilde{y}_{\delta(s),\overline{\delta(s)}}^{m_0,\mathcal{S}}, \widetilde{z}_{s,\overline{s}}^{m_0,\mathcal{S}}) \right) \right| < \epsilon_s^{\mathcal{S}} \tag{3.19}$$

  where $\{\epsilon_s^{\mathcal{S}}\}_{s \in \mathcal{S}}$ *is a family of positive bias tuning parameters.*

- *For* $k, j \in \{1, ..., 2^L\}$ *with* $k < j$ *that satisfy* $s_j < s_k \leq \overline{s_j} < t_{2^L} = T$, *the simulation* $Y_{s_j,s_k}^{m_0,m_1}$ *and* $Z_{s_j,s_k}^{m_0,m_1}$ *of* Y *and* Z *respectively around* $X_{s_k}^{m_0}$ *conditionally on* $X_{s_j}^{m_0}$ *are set to be equal to* $\overline{y}_{s_j,s_k}^{m_0,\mathcal{S}}(X_{s_j,s_k}^{m_0,m_1})$ *and* $\overline{z}_{s_j,s_k}^{m_0,\mathcal{S}}(X_{s_j,s_k}^{m_0,m_1})$ *where* $\overline{y}$ *and* $\overline{z}$ *are given in* (3.10), (3.11) *and* (3.14).

- *For* $k \in \{1, ..., 2^L\}$ *and* $s \in \{s_k, s_k + \Delta_t, ..., \delta_{s_k}(s_k) - \Delta_t\} - \{0\}$, *the simulation* $Y_s^{m_0}$ *and* $Z_s^{m_0}$ *of* Y *and* Z *respectively at* $X_s^{m_0}$ *are set to be equal to* $\widetilde{y}_{s,\overline{s_k}}^{m_0,\mathcal{S}}$ *and to* $\widetilde{z}_{s,\delta_{s_k}(s_k)}^{m_0,\mathcal{S}}$ *with* $\widetilde{y}$ *and* $\widetilde{z}$ *expressed in* (3.12) *and* (3.13).

- *The average* Y_0^{lear} *and* Z_0^{lear} *of learned values on* Y_0 *and* Z_0 *are respectively equal to*

$$Y_0^{lear} = \frac{1}{M_0} \sum_{m_0=1}^{M_0} \widetilde{y}_{0,\overline{0}}^{m_0,\mathcal{S}}, \quad Z_0^{lear} = \frac{1}{M_0} \sum_{m_0=1}^{M_0} \widetilde{z}_{0,\overline{0}}^{m_0,\mathcal{S}} \tag{3.20}$$

  and the simulated values Y_0^{sim} *and* Z_0^{sim} *of* Y_0 *and* Z_0 *are respectively equal to*

$$\begin{aligned} Y_0^{sim} &= \frac{1}{M_0} \sum_{m_0=1}^{M_0} \left[\delta(0) f\left(\delta(0), \widetilde{y}_{\delta(0),\overline{\delta(0)}}^{m_0,\mathcal{S}}, Z_0^{sim}\right) + \widetilde{y}_{\delta(0),\overline{\delta(0)}}^{m_0,\mathcal{S}} \right], \\ Z_0^{sim} &= \sum_{m_0=1}^{M_0} \widetilde{y}_{\delta(0),\overline{\delta(0)}}^{m_0,\mathcal{S}} \frac{W_{\delta(0)}^{m_0}}{\delta(0) M_0}. \end{aligned} \tag{3.21}$$

Y^{m_0,m_1} and Z^{m_0,m_1} are the inner or second layer approximation of Y and Z respectively. Y^{m_0} and Z^{m_0} are the outer or first layer approximation of Y and Z respectively. Similar to what was presented for the generic example in Section 3.1, as inner and outer trajectories coincide at $t = 0$, we get two possible outer first layer approximations of Y_0 and Z_0. It is also remarkable to have Y^{m_0} and Z^{m_0} defined on the whole fine discretization set $\mathbb{S}$ despite Y^{m_0,m_1} and Z^{m_0,m_1} are only defined on $\mathcal{S}$. With BSDEs, it is possible to replace (3.19), using rather an MDP scheme (cf. [16]), by

$$\left| \frac{1}{M_0} \sum_{m_0=1}^{M_0} \left(\widetilde{y}_{s,u}^{m_0,\mathcal{S}} - \widetilde{y}_{\delta(r),\overline{\delta(r)}}^{m_0,\mathcal{S}} - \sum_{\theta \in \mathcal{S}, \theta = s}^{r} (\delta(\theta) - \theta) f(s, \widetilde{y}_{\delta(\theta),\overline{\delta(\theta)}}^{m_0,\mathcal{S}}, \widetilde{z}_{\theta,\overline{\theta}}^{m_0,\mathcal{S}}) \right) \right| < \epsilon_s^{\mathcal{S}}, \tag{3.22}$$

for a chosen $r \in \mathcal{S} \cap [s, u[$. Both conditions (3.19) and (3.22) involve mainly the approximation of Y since using criteria on the approximation of Z would involve very large number of trajectories which makes it impracticable.

4 Regression-based NMC and bias propagation

Before presenting the main elements, we point out that we have intentionally considered only discrete functionals of a Markov process. The approximation due to discretization of the continuous version of BSDEs is not studied and we refer to [5, 14, 22] among others that quantify well the resulting error. Moreover, we also consider the discretized version of the Markov process introduced in (2.1) and (2.2) where

$$\mathcal{E}_{t_k}(x,\xi) = x + \Delta_t b(t_k,x) + \sigma(t_k,x)\xi \tag{4.1}$$

with the usual (cf. [25]) Lipschitz continuity condition on the coefficients $b(t,x)$ and $\sigma(t,x)$ uniformly with respect to $t \in [0,T]$. Similar to what was considered in Section 3.2, the noise ξ is given by increments of a vector of independent Brownian motions i.e. $\xi_{t_k}^{m_0} = W_{t_k}^{m_0} - W_{t_{k-1}}^{m_0}$ and $\xi_{s_j,t_k}^{m_0,m_1} = W_{s_j,t_k}^{m_0,m_1} - W_{s_j,t_{k-1}}^{m_0,m_1}$. (4.1) can be read as an Euler scheme of a stochastic differential equation that admits a strong solution. In this paper, when the discretization is needed, we assume that L is sufficiently large to neglect the discretization error of the forward process X.

Given two arbitrary square integrable random variables χ_1 and χ_2, consider $\{\overline{\chi_3}^{m_1}\}_{m_1=1}^{M_1}$ to be the empirical regression of χ_1 with respect to χ_2, the authors of [6] established an upper bound error of the regression-based NMC estimator $\frac{1}{M_1}\sum_{m_1=1}^{M_1}\phi(\overline{\chi_3}^{m_1})$ of $\mathbb{E}(\phi(\mathbb{E}(\chi_1|\chi_2)))$ once we know the representation error $\kappa = \mathbb{E}(\chi_1|\chi_2) - {}^tR\mathcal{B}(\chi_2)$ induced by the projection of $\mathbb{E}(\chi_1|\chi_2)$ on the basis $\mathcal{B}(\chi_2)$. The fine approximations $\widetilde{u}$ and $\widetilde{y}$ presented earlier were computed by averaging on the empirical regressions $\overline{u}$ and $\overline{y}$. It is then interesting to see how to control the error of the fine approximations through the representation error like in [6].

First, for $s_j < s_k < \overline{s_j}$ and Borel measurable Θ function of $(X_{s_j,s_k:\delta(s_k)}^{m_0,m_1})$ with $\Theta(X_{s_j,s_k:\delta(s_k)}^{m_0,m_1})$ integrable, we denote $\mathbb{E}_{s_j,s_k}^{m_0,x}$, $\overline{\mathbb{E}}_{s_j,s_k}^{m_0,x}$ and $\widehat{\mathbb{E}}_{s_j,s_k}^{m_0,x}$ the operators defined by

$$\mathbb{E}_{s_j,s_k}^{m_0,x}(\Theta(X_{s_j,s_k:\delta(s_k)}^{m_0,m_1})) = \mathbb{E}_{s_j}^{m_0}\left(\Theta(X_{s_j,s_k:\delta(s_k)}^{m_0,m_1})|X_{s_j,s_k}^{m_0,m_1} = x\right),$$

$$\begin{aligned}&\overline{\mathbb{E}}_{s_j,s_k}^{m_0,x}(\Theta(X_{s_j,s_k:\delta(s_k)}^{m_0,m_1}))\\ &\quad = {}^t\mathcal{T}_{s_j,s_k}^{m_0}(x - X_{s_k}^{m_0})\frac{\left(\overline{\Lambda}_{s_j,s_k}^{m_0}\right)^{-1}}{M_1}\sum_{m_1=1}^{M_1}\left[\mathcal{T}_{s_j,s_k}^{m_0}\left(X_{s_j,s_k}^{m_0,m_1} - X_{s_k}^{m_0}\right)\Theta(X_{s_j,s_k:\delta(s_k)}^{m_0,m_1})\right]\end{aligned}$$

and

$$\widehat{\mathbb{E}}_{s_j,s_k}^{m_0,x}(\Theta(X_{s_j,s_k:\delta(s_k)}^{m_0,m_1})) = {}^t\mathcal{T}_{s_j,s_k}^{m_0}(x - X_{s_k}^{m_0})R_{s_j,s_k}^{m_0}\left[\Theta(X_{s_j,s_k:\delta(s_k)}^{m_0,m_1})\right]$$

with

$$R_{s_j,s_k}^{m_0}\left[\Theta(X_{s_j,s_k:\delta(s_k)}^{m_0,m_1})\right] \in \underset{r\in\mathbb{R}^{d_1'}}{\operatorname{argmin}}\,\mathbb{E}_{s_j}^{m_0}\left(\left[\begin{array}{c}\mathbb{E}_{s_j}^{m_0}\left(\Theta(X_{s_j,s_k:\delta(s_k)}^{m_0,m_1})|X_{s_j,s_k}^{m_0,m_1}\right)\\ -{}^t\mathcal{T}_{s_j,s_k}^{m_0}(X_{s_j,s_k}^{m_0,m_1} - X_{s_k}^{m_0})r\end{array}\right]^2\right).$$

When $\mathbb{E}_{s_j}$ is the conditional expectation knowing $X_{s_j}^{m_0}$, $\mathbb{E}_{s_j}^{m_0}$ is the conditional expectation knowing the trajectory of X^{m_0} starting from $X_{s_j}^{m_0}$. $\mathbb{E}_{s_j}^{m_0}$ is used as the regression

basis depends on X^{m_0}. For a given $s_j \in \mathcal{S}$, in contrast to expressions presented in sections 2.2 and 3.2, we simplify the presentation here and we omit to center the regressions around $\widetilde{u}^{m_0,\mathcal{S}}_{\delta(s_j),\overline{\delta(s_j)}}$ or $\widetilde{y}^{m_0,\mathcal{S}}_{\delta(s_j),\overline{\delta(s_j)}}$. Consequently, the value of $\widetilde{u}^{m_0,\mathcal{S}}_{s_j,\overline{s_j}}$ and $\widetilde{y}^{m_0,\mathcal{S}}_{s_j,\overline{s_j}}$ are obtained through respectively averaging on $\overline{u}^{m_0,\mathcal{S}}_{s_j,\delta(s_j)}(x) = \overline{\mathbb{E}}^{m_0,x}_{s_j,\delta(s_j)}(\Theta^u(X^{m_0,m_1}_{s_j,\delta(s_j):\delta^2(s_j)}))$ and on $\overline{y}^{m_0,\mathcal{S}}_{s_j,\delta(s_j)}(x) = \overline{\mathbb{E}}^{m_0,x}_{s_j,\delta(s_j)}(\Theta^y(X^{m_0,m_1}_{s_j,\delta(s_j):\delta^2(s_j)}))$, where

$$\Theta^u(X^{m_0,m_1}_{s_j,\delta(s_j):\delta^2(s_j)}) = \overline{u}^{m_0,\mathcal{S}}_{s_j,\delta^2(s_j)}(X^{m_0,m_1}_{s_j,\delta^2(s_j)}) + \sum_{t_l \geq \delta(s_j)}^{\delta^2(s_j)} f(t_l, X^{m_0,m_1}_{s_j,t_l}, X^{m_0,m_1}_{s_j,t_{l+1}}),$$

$$\begin{aligned}\Theta^y(X^{m_0,m_1}_{s_j,\delta(s_j):\delta^2(s_j)}) = \overline{y}^{m_0,\mathcal{S}}_{s_j,\delta^2(s_j)}(X^{m_0,m_1}_{s_j,\delta^2(s_j)}) \\ +\Delta_{\delta(s_j)} f(\delta(s_j), \overline{y}^{m_0,\mathcal{S}}_{s_j,\delta^2(s_j)}(X^{m_0,m_1}_{s_j,\delta^2(s_j)}), \overline{z}^{m_0,\mathcal{S}}_{s_j,\delta(s_j)}(X^{m_0,m_1}_{s_j,\delta(s_j)})).\end{aligned}$$

We assume Lipschitz condition uniformly in time of the driver f involved in (ODP) with respect to its Y and Z coordinates. Although this condition is not necessary to obtain good numerical results in Section 5, it is required to apply Theorem 2 of [6] (cf. Assumption F2 in [6]) that yields the following asymptotical result.

Proposition 4.1. *Given that assumptions A1, A2 and A3 of [6] are fulfilled and that the driver involved in (ODP) is $[f]_{Lip}$-Lipschitz we have the following asymptotical inequality*

$$(\widetilde{\rho} - \rho)^2 \leq [\rho]_{Lip} \mathbb{E}^{m_0}_{s_j}(\kappa^2(X^{m_0,m_1}_{s_j,\delta(s_j)})) + O_p(1/M_1) \tag{4.2}$$

as $M_1 \longrightarrow \infty$ where $(\widetilde{\rho}, \rho, [\rho]_{Lip}, \kappa)$ is either equal to $(\widetilde{\rho}^u, \rho^u, [\rho]^u_{Lip}, \kappa^u)$ for (f) or equal to $(\widetilde{\rho}^y, \rho^y, [\rho]^y_{Lip}, \kappa^y)$ for (ODP) with $\widetilde{\rho}^u = \widetilde{u}^{m_0,\mathcal{S}}_{s_j,\overline{s_j}}$, $\widetilde{\rho}^y = \widetilde{y}^{m_0,\mathcal{S}}_{s_j,\overline{s_j}}$,

$$\rho^u = \mathbb{E}^{m_0}_{s_j}\left(\mathbb{E}^{m_0,X^{m_0,m_1}_{s_j,\delta(s_j)}}_{s_j,\delta(s_j)}\left[\Theta^u(X^{m_0,m_1}_{s_j,\delta(s_j):\delta^2(s_j)})\right] + \sum_{t_l \geq s_j}^{\delta(s_j)} f(t_l, X^{m_0,m_1}_{s_j,t_l}, X^{m_0,m_1}_{s_j,t_{l+1}})\right),$$

$$\begin{aligned}\rho^y = \mathbb{E}^{m_0}_{s_j}\Bigg(\mathbb{E}^{m_0,X^{m_0,m_1}_{s_j,\delta(s_j)}}_{s_j,\delta(s_j)}\left[\Theta^y(X^{m_0,m_1}_{s_j,\delta(s_j):\delta^2(s_j)})\right] \\ +\Delta_{s_j} f\left(s_j, \mathbb{E}^{m_0,X^{m_0,m_1}_{s_j,\delta(s_j)}}_{s_j,\delta(s_j)}\left[\Theta^y(X^{m_0,m_1}_{s_j,\delta(s_j):\delta^2(s_j)})\right], \widetilde{z}^{m_0,\mathcal{S}}_{s_j,\overline{s_j}}\right)\Bigg),\end{aligned}$$

$[\rho]^u_{Lip} = 1$, $[\rho]^y_{Lip} = 1 + \Delta_{s_j}[f]_{Lip}$ *and*

$$\kappa^u(x) = \mathbb{E}^{m_0,x}_{s_j,\delta(s_j)}\left[\Theta^u(X^{m_0,m_1}_{s_j,\delta(s_j):\delta^2(s_j)})\right] - \widehat{\mathbb{E}}^{m_0,x}_{s_j,\delta(s_j)}\left[\Theta^u(X^{m_0,m_1}_{s_j,\delta(s_j):\delta^2(s_j)})\right],$$

$$\kappa^y(x) = \mathbb{E}^{m_0,x}_{s_j,\delta(s_j)}\left[\Theta^y(X^{m_0,m_1}_{s_j,\delta(s_j):\delta^2(s_j)})\right] - \widehat{\mathbb{E}}^{m_0,x}_{s_j,\delta(s_j)}\left[\Theta^y(X^{m_0,m_1}_{s_j,\delta(s_j):\delta^2(s_j)})\right].$$

Proposition 4.1 results from Theorem 2 and Remark 2 of [6]; we expressed $[\rho]_{Lip}$ associated to each problem and we replaced $\mathbb{E}$ by $\mathbb{E}^{m_0}_{s_j}$ as the regression basis depends on X^{m_0}. Assumptions A1, A2 and A3 of [6] are standard assumptions for regressions

(cf. [28]). Considering the regression basis presented in Section 2.1 with $\mathbb{E}(|X_t|^2) < \infty$ for any $t \in [0,T]$, these assumptions are fulfilled if: *i*) the conditional variance of each regressed quantity is integrable and bounded from below by $v_0 > 0$, *ii*) the regression value is unbiased and *iii*) each component of the regression basis as well as κ (denoted M in [6]) admit a finite fourth moment. When the latter moment assumption *iii* is needed to establish error control and can be modified using truncation (cf. [15]), the further *i*&*ii* are sufficient to ensure the existence and uniqueness of the regressed representation.

In Proposition 4.1, we provided a control on fine approximations $\widetilde{u}$ and $\widetilde{y}$. In Proposition 4.2, we rather focus on coarse approximations and decompose the conditional mean square error $\mathbb{E}^{m_0}_{s_j}\Big([\overline{u}^{m_0,\mathcal{S}}_{s_j,s_k}(X^{m_0,m_1}_{s_j,s_k}) - u_{s_k}(X^{m_0,m_1}_{s_j,s_k})]^2\Big)$ into a bias term $\mathcal{W}$, a variance term $\mathcal{V}$ and a regression error term $\mathcal{R}$.

Proposition 4.2. *Assuming i and iii introduced above, for $s_j < s < s_k$ taking their values in the discretization set $\mathcal{S}$, we define*

$$\mathcal{W}^{m_0,\mathcal{S}}_{s_j,s_k}(x) = \mathbb{E}^{m_0}_{s_j}\Big(\overline{u}^{m_0,\mathcal{S}}_{s_j,s_k}(x) - \boldsymbol{u}^{m_0,\mathcal{S}}_{s_j,s_k}(x)\Big),$$

$$\mathcal{R}^{m_0,\mathcal{S}}_{s_j,s_k}(x) = \mathbb{E}^{m_0}_{s_j}\Big(\boldsymbol{u}^{m_0,\mathcal{S}}_{s_j,s_k}(x) - u_{s_k}(x)\Big),$$

$$\mathcal{V}^{m_0,\mathcal{S}}_{s_j,s_k}(x) = Var^{m_0}_{s_j}\Big(\overline{u}^{m_0,\mathcal{S}}_{s_j,s_k}(x)\Big),$$

with

$$\boldsymbol{u}^{m_0,\mathcal{S}}_{s_j,s_k}(x) = \overline{\mathbb{E}}^{m_0,x}_{s_j,s_k}\left(u_{\delta(s_k)}(X^{m_0,m_1}_{s_j,\delta(s_k)}) + \sum_{t_{l+1}>s_k}^{\delta(s_k)} f(t_l, X^{m_0,m_1}_{s_j,t_l}, X^{m_0,m_1}_{s_j,t_{l+1}})\right)$$

then

$$\begin{aligned}\mathbb{E}^{m_0}_{s_j}\Big([\overline{u}^{m_0,\mathcal{S}}_{s_j,s_k}(X^{m_0,m_1}_{s_j,s_k}) - u_{s_k}(X^{m_0,m_1}_{s_j,s_k})]^2\Big) &= \mathbb{E}^{m_0}_{s_j}(\mathcal{V}^{m_0,\mathcal{S}}_{s_j,s_k}(X^{m_0,m_1}_{s_j,s_k}))\\ &\quad+\mathbb{E}^{m_0}_{s_j}\Big([\mathcal{R}^{m_0,\mathcal{S}}_{s_j,s_k}(X^{m_0,m_1}_{s_j,s_k}) + \mathcal{W}^{m_0,\mathcal{S}}_{s_j,s_k}(X^{m_0,m_1}_{s_j,s_k})]^2\Big)\end{aligned}$$

and there exists a positive constant $\mathcal{K}^{m_0}_{1,s_j,s_k}$ depending on the regression basis such that

$$\mathbb{E}^{m_0}_{s_j}\Big([\mathcal{W}^{m_0,\mathcal{S}}_{s_j,s_k}(X^{m_0,m_1}_{s_j,s_k})]^2\Big) \leq \mathcal{K}^{m_0}_{1,s_j,s_k}\mathbb{E}^{m_0}_{s_j}\Big([\overline{u}^{m_0,\mathcal{S}}_{s_j,\delta(s_k)}(X^{m_0,m_1}_{s_j,\delta(s_k)}) - u_{\delta(s_k)}(X^{m_0,m_1}_{s_j,\delta(s_k)})]^2\Big).$$

Proof. As we simulate several independent copies of X^{m_0,m_1}, we make sure that the approximations $\overline{u}$ are independent from X^{m_0,m_1} conditionally on X^{m_0}. Then, the expansion of $\mathbb{E}^{m_0}_{s_j}\Big([\overline{u}^{m_0,\mathcal{S}}_{s_j,s_k}(X^{m_0,m_1}_{s_j,s_k}) - u_{s_k}(X^{m_0,m_1}_{s_j,s_k})]^2\Big)$ is gotten when we notice that

$$u_{s_k}(X^{m_0,m_1}_{s_j,s_k}) = \mathbb{E}^{m_0,X^{m_0,m_1}_{s_j,s_k}}_{s_j,s_k}\left(u_{\delta(s_k)}(X^{m_0,m_1}_{s_j,\delta(s_k)}) + \sum_{t_{l+1}>s_k}^{\delta(s_k)} f(t_l, X^{m_0,m_1}_{s_j,t_l}, X^{m_0,m_1}_{s_j,t_{l+1}})\right).$$

An expression for the constant $\mathcal{K}^{m_0}_{1,s_j,s_k}$ can be obtained after expanding $\mathbb{E}^{m_0}_{s_j}\Big([\mathcal{W}^{m_0,\mathcal{S}}_{s_j,s_k}(X^{m_0,m_1}_{s_j,s_k})]^2\Big)$ using

$$\overline{u}^{m_0,\mathcal{S}}_{s_j,s_k}(x) = \overline{\mathbb{E}}^{m_0,x}_{s_j,s_k}\left(\overline{u}^{m_0,\mathcal{S}}_{s_j,\delta(s_k)}(X^{m_0,m_1}_{s_j,\delta(s_k)}) + \sum_{t_{l+1}>s_k}^{\delta(s_k)} f(t_l, X^{m_0,m_1}_{s_j,t_l}, X^{m_0,m_1}_{s_j,t_{l+1}})\right).$$

□

Finally, we should point out that one could establish a similar result for (ODP). Indeed, for instance, using the following coarse discretization to approximate (ODP)

$$\begin{cases} \hat{Y}_{s_k} &= \mathbb{E}_{s_k}\left(\widetilde{f}_{s_k}(\hat{Y}_{\delta(s_k)}, \hat{Z}_{s_k})\right), \\ \hat{Z}_{s_k} &= \frac{1}{\Delta_{s_k}}\mathbb{E}_{s_k}\left(\hat{Y}_{\delta(s_k)}(W_{\delta(s_k)} - W_{s_k})\right), \end{cases} \tag{4.3}$$

with $\widetilde{f}_{s_k}(y,z) = y + \Delta_{s_k} f_{s_k}(y,z)$, the bias is then controlled as follows

$$\mathbb{E}^{m_0}_{s_j}\left([\widetilde{\mathcal{W}}^{m_0,\mathcal{S}}_{s_j,s_k}(X^{m_0,m_1}_{s_j,s_k})]^2\right) \leq \widetilde{\mathcal{K}}^{m_0}_{1,s_j,s_k}\mathbb{E}^{m_0}_{s_j}\left(\begin{bmatrix} \widetilde{f}_{s_k}(\widehat{Y}_{s_j,\delta(s_k)}(X^{m_0,m_1}_{s_j,\delta(s_k)}), \widehat{Z}_{s_j,s_k}(X^{m_0,m_1}_{s_j,s_k})) \\ -\widetilde{f}_{s_k}(\overline{y}^{m_0,\mathcal{S}}_{s_j,\delta(s_k)}(X^{m_0,m_1}_{s_j,\delta(s_k)}), \overline{z}^{m_0,\mathcal{S}}_{s_j,s_k}(X^{m_0,m_1}_{s_j,s_k})) \end{bmatrix}^2\right),$$

for some positive constant $\widetilde{\mathcal{K}}^{m_0}_{1,s_j,s_k}$ depending on the regression basis. Therefore, the bias upper bound depends heavily on the driver choice.

As shown in Proposition 4.2, the bias $\mathcal{W}$ at time step s_k is controlled by the mean square error at time step $\delta(s_k)$ decomposed into a variance term $\mathcal{V}$, a regression error term $\mathcal{R}$ and a bias term at time step $\delta(s_k)$. Thus, increasing the number of time steps loosens the bias control as it involves more and more terms. In some situations, this accumulation of errors is a source of a significant bias backpropagation. A standard method to reduce this bias backpropagation is to use a coarser set $\mathcal{S}$. In this paper, we rather proposed a new trick to cut this bias backpropagation using the couple $(\underline{s}, \overline{s})$ for any $s \in \mathcal{S}$. Like for the example of Section 5.4, the latter method is quite effective and will be further studied in CMCLDII (cf. [2], Section 4).

5 Some numerical results

In this section we test the presented conditional MC learning procedure on various BSDE examples. The fact that the driver f depends also on X is not a burden to the use of our method. All simulations are run on a laptop that has an Intel i7-7700HQ CPU and a single GeForce GTX 1060 GPU programmed with the CUDA/C application programming interface. We refer the reader to [24] for an introduction to CUDA programming.

The proposed 1NMC simulation combined with local regressions leads to a simulation with batch parallel regressions that can be successfully implemented using the contribution [3]. We had even to extend the work presented in [3] to examples with larger dimensions. It is worth mentioning that the very competitive execution times that we present hereafter are possible not only because of the GPU use but also due to the adaptation of parallelization strategies presented in [3].

5.1 Allen-Cahn equation

We consider (ODP) simulation as presented in Section 3.2, we use the following functions

$$f(t,x,y,z) = y - y^3,$$

$$f(T,x) = \left[2 + \frac{2}{5}|x|^2_{d_1}\right]^{-1}$$

and

$$\mathcal{E}_{t_k}(x,w) = x + \sqrt{2}w, \quad X_{t_0} = 0.$$

Table 1: Numerical simulations for PDE (5.1): $T = 0.3$, $M_0 = 2^4$, $d_1 = 100$, $L = 4$; [Benchmarck solution] $y_b(0,0) = 0.0528$.

M_1	Average learned		Simulated		Runtime in sec. (10^{-3})
	Y_0^{learn}	std	Y_0^{sim}	std	
2^4	0.0454	($\pm$ 0.0093)	0.0455	($\pm$ 0.0073)	13
2^5	0.0513	($\pm$ 0.0011)	0.0517	($\pm$ 0.0008)	23
2^6	0.0523	($\pm$ 0.0004)	0.0518	($\pm$ 0.0006)	56
2^7	0.0526	($\pm$ 0.0003)	0.0515	($\pm$ 0.0001)	119
2^8	0.0525	($\pm$ 0.0002)	0.0517	($\pm$ 0.0002)	227
2^9	0.0527	($\pm$ 0.0002)	0.0515	($\pm$ 0.0002)	414

Table 2: Numerical simulations for PDE (5.1): $T = 1$, $d_1 = 100$, $M_0 = 2^5$, $L = 6$; [Benchmarck solution] $y_b(0,0) = 0.0338$.

M_1	Average learned		Simulated		Runtime in sec.
	Y_0^{learn}	std	Y_0^{sim}	std	
2^5	0.0345	($\pm$ 0.0008)	0.0350	($\pm$ 0.0021)	2
2^6	0.0333	($\pm$ 0.0003)	0.0326	($\pm$ 0.0004)	4
2^7	0.0334	($\pm$ 0.0002)	0.0330	($\pm$ 0.0003)	7
2^8	0.0336	($\pm$ 0.0002)	0.0332	($\pm$ 0.0002)	12
2^9	0.0336	($\pm$ 0.0001)	0.0331	($\pm$ 0.0001)	27

We would like to approximate the solution $y(t,x)$ of the Allen-Cahn PDE defined as follows, $y(T,x) = f(T,x)$,

$$\frac{\partial y}{\partial t}(t,x) + y(t,x) - [y(t,x)]^3 + (\Delta_x y)(t,x) = 0. \tag{5.1}$$

A benchmark approximation $y_b(0,x)$ for the solution $y(0,x)$ of the PDE (5.1) is given in [Section 4.2; [11]].

Table 1 provides the $y(0,0)$ simulated value of equation (5.1), approximated by Y_0^{learn} and Y_0^{sim} expressions, with respect to the number of inner trajectories M_1. The benchmark solution $y_b(0,0)$ is equal to 0.0528 for $T = 0.3$ and $d_1 = 100$. The standard deviation of each expression and the runtime in seconds are also given. We reduce the bias by increasing the number of inner trajectories. Table 1 shows that a relative small number of outer and inner trajectories is sufficient to observe a small variance and bias for both options. In fact, we show that the standard deviation is already acceptable for $M_0 = 2^4$ outer trajectories and the bias is acceptable for $M_1 = 2^6$ inner trajectories with an execution time of 56 millisecond.

The average learned Y_0^{learn} and the simulated Y_0^{sim} values in Table 2 are presented for the longer time horizon $T = 1$. The benchmark solution is equal to 0.0338 for $T = 1$ and $d_1 = 100$. To achieve a similar level of variance and bias, we need larger number of outer and inner trajectories than in Table 1. In fact, for $M_0 = 2^5$ of outer trajectories and $M_1 = 2^6$ of inner trajectories, we obtained an acceptable bias and standard deviation within 4 seconds of execution.

5.2 Multidimensional Burgers-type PDEs

We assume the (ODP) setting presented in Section 3.2, we use the following functions

$$f(t,x,y,z) = \left(y - \frac{2+d_1}{2d_1}\right)\left(\sum_{i=1}^{d_1} z_i\right),$$

$$f(T,x) = \frac{\exp\left(T + \frac{1}{d_1}\sum_{i=1}^{d_1} x_i\right)}{1+\exp\left(T + \frac{1}{d_1}\sum_{i=1}^{d_1} x_i\right)}$$

and

$$\mathcal{E}_{t_k}(x,w) = x + \frac{d_1}{\sqrt{2}}w, \quad X_{t_0} = 0.$$

We simulate the solution $y(t,x)$ of the multidimensional Burgers-type PDE (cf [8], Example 4.6) defined as follows, $y(T,x) = f(T,x)$,

$$\frac{\partial y}{\partial t}(t,x) + \frac{{d_1}^2}{2}(\Delta_x y)(t,x) + \left(y(t,x) - \frac{2+d_1}{2d_1}\right)\left(d_1\sum_{i=1}^{d_1}\frac{\partial y}{\partial x_i}(t,x)\right) = 0. \tag{5.2}$$

PDE (5.2) admits an explicit solution, we refer the reader to [Lemma 4.3, [11]] for more details. The value of the solution $y(0,0)$ is 0.5000 for $T = 0.2$ and $d_1 = 100$.

Table 3: Numerical simulations for PDE (5.2): $T = 0.2$, $d_1 = 100$, $M_0 = 2^6$, $L = 5$; [Explicit solution] $y(0,0) = 0.5000$.

M_1	Average learned		Simulated		Runtime in sec.
	Y_0^{learn}	std	Y_0^{sim}	std	
2^8	0.4785	($\pm$ 0.0428)	0.0517	($\pm$ 0.0431)	7
2^9	0.5113	($\pm$ 0.0450)	0.5108	($\pm$ 0.0450)	16
2^{10}	0.4966	($\pm$ 0.0448)	0.4912	($\pm$ 0.0447)	27
2^{11}	0.5022	($\pm$ 0.0421)	0.5012	($\pm$ 0.0435)	49

Table 4: Numerical simulations for PDE (5.2): $T = 0.2$, $d_1 = 100$, $M_1 = 2^{11}$, $L = 5$; [Explicit solution] $y(0,0) = 0.5000$.

M_0	Average learned		Simulated		Runtime in sec.
	Y_0^{learn}	std	Y_0^{sim}	std	
2^5	0.4953	($\pm$ 0.0618)	0.4941	($\pm$ 0.0615)	24
2^6	0.5022	($\pm$ 0.0424)	0.5013	($\pm$ 0.0435)	49
2^7	0.5079	($\pm$ 0.0346)	0.5066	($\pm$ 0.0342)	103
2^8	0.5158	($\pm$ 0.0221)	0.5151	($\pm$ 0.0221)	194
2^9	0.5023	($\pm$ 0.0164)	0.5029	($\pm$ 0.0164)	408

Table 3 presents the approximations (3.20) and (3.21) of the equation (5.2), with respect to the number of inner trajectories M_1. It shows that the standard deviation

of both results should be reduced by increasing the number of outer trajectories. The number M_0 of outer trajectories in Table 4 is augmented till reaching a sufficiently small standard deviation within less than 7 minutes of execution time.

5.3 Time-dependent reaction-diffusion-type example

Let $\kappa = 0.6$, $\lambda = \frac{1}{\sqrt{d_1}}$, we use the following functions

$$f(t,x,y,z) = \min\left\{1, \left[y - \kappa - 1 - \sin\left(\lambda \sum_{i=1}^{d_1} x_i\right) e^{\frac{\lambda^2 d(t-T)}{2}}\right]\right\},$$

$$f(T,x) = 1 + \kappa + \sin\left(\lambda \sum_{i=1}^{d_1} x_i\right)$$

and

$$\mathcal{E}_{t_k}(x,w) = x + w, \quad X_{t_0} = 0.$$

We simulate the solution $y(t,x)$ of the time dependent reaction-diffusion-type PDE (cf [17], Section 6) defined as follows, $y(T,x) = f(T,x)$,

$$\frac{\partial y}{\partial t}(t,x) + \min\left\{1, \left[y - \kappa - 1 - \sin\left(\lambda \sum_{i=1}^{d_1} x_i\right)\right]\right\} + \frac{1}{2}(\Delta_x y)(t,x) = 0. \tag{5.3}$$

The explicit solution of the PDE (5.3) is given in [Lemma 4.4; [11]].

Table 5 shows the approximated solution of the equation (5.3), calculated by the average learned (3.20) and the simulated (3.21) expressions, with respect to the number of inner trajectories M_1. The standard deviation of each expression and the runtime in milliseconds are also given. The benchmark solution is equal to 1.6000 for $T = 1$, $d_1 = 100$.

Table 5: Numerical simulations for PDE (5.3): $T = 0.5$, $d_1 = 100$, $M_0 = 2^{10}$, $L = 3$; [Benchmark solution] $y_b(0,0) = 1.6000$.

M_1	Average learned		Simulated		Runtime in sec. (10^{-3})
	Y_0^{learn}	std	Y_0^{sim}	std	
2^5	1.8197	($\pm$ 0.0386)	1.7587	($\pm$ 0.0287)	244
2^6	1.7125	($\pm$ 0.0104)	1.6799	($\pm$ 0.0116)	311
2^7	1.6605	($\pm$ 0.0037)	1.6376	($\pm$ 0.0091)	466
2^8	1.6458	($\pm$ 0.0023)	1.6290	($\pm$ 0.0089)	817
2^9	1.6439	($\pm$ 0.0019)	1.6283	($\pm$ 0.0061)	1526

Due in part to small number of time steps ($L = 3$), we obtain very accurate solutions within less than 2 seconds of execution.

5.4 A Hamilton-Jacobi-Bellman (HJB) equation

We assume here the driver to be equal to

$$f(t,x,y,z) = -|z|_{d_1}^2,$$

$$f(T,x) = ln\left(\left[1+|x|_{d_1}^2\right]/2\right)$$

and

$$\mathcal{E}_{t_k}(x,w) = x + \sqrt{2}w, \quad X_{t_0} = 0.$$

We calculate the solution $y(t,x)$ of the HJB equation (cf [9] Section 4.2) defined by $y(T,x) = f(T,x)$,

$$\frac{\partial y}{\partial t}(t,x) + (\Delta_x y)(t,x) - |(\nabla_x y)(t,x)|_{d_1}^2 = 0. \tag{5.4}$$

PDE (5.4) admits a benchmark solution. We refer the reader to [Lemma 4.2; [11]] for more details.

Related to inequality (3.19), Figure 2 shows between two approximations $\frac{1}{M_0}\sum_{m_0=1}^{M_0} \widetilde{y}_{s,\overline{s}}^{m_0,\mathcal{S}}$ and $\frac{1}{M_0}\sum_{m_0=1}^{M_0}\left(\widetilde{y}_{\delta(s),\overline{\delta(s)}}^{m_0,\mathcal{S}} + (\delta(s)-s)f(s,\widetilde{y}_{\delta(s),\overline{\delta(s)}}^{m_0,\mathcal{S}},\widetilde{z}_{s,\overline{s}}^{m_0,\mathcal{S}})\right)$ of the same quantity with respect to the time discretization. The purpose here is to show the benefits of using the couple $(\underline{s},\overline{s}) = (\delta(s),\overline{s})$ to stop the bias backpropagation. On Figure 2 left, we perform the conditional MC procedure taking $\overline{s} = T$ for all $s \in \mathcal{S} = \{0, \frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \frac{4}{8}, \frac{5}{8}, \frac{6}{8}, \frac{7}{8}, 1\}$. On Figure 2 right, we take $\overline{s} = \left(s+\frac{3}{8}\right)\wedge T$ when $s \in \mathcal{S}$. The latter choice of $\overline{s}$ is then clearly justified, it can be even set automatically during the execution of the simulation using inequality (3.19) without any manual intervention.

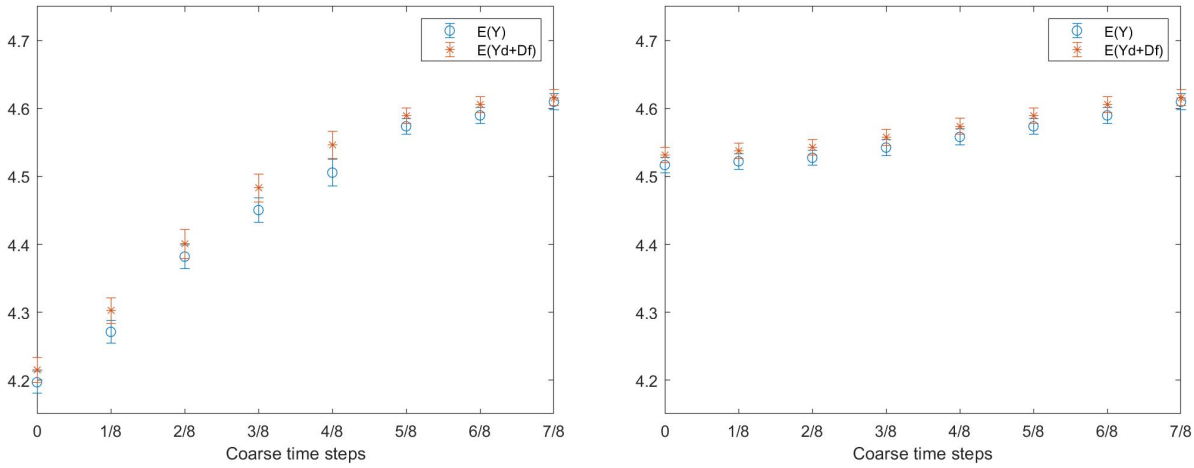


Figure 2: $\frac{1}{M_0}\sum_{m_0=1}^{M_0} \widetilde{y}_{s,\overline{s}}^{m_0,\mathcal{S}}$ vs. $\frac{1}{M_0}\sum_{m_0=1}^{M_0}\left(\widetilde{y}_{\delta(s),\overline{\delta(s)}}^{m_0,\mathcal{S}} + (\delta(s)-s)f(s,\widetilde{y}_{\delta(s),\overline{\delta(s)}}^{m_0,\mathcal{S}},\widetilde{z}_{s,\overline{s}}^{m_0,\mathcal{S}})\right)$ [Left] $\overline{s} = T$, [Right] $\overline{s} = \left(s+\frac{3}{8}\right)\wedge T$: $T = 1$, $d_1 = 100$, $M_0 = 2^7$, $M_1 = 2^{15}$.

Figure 3 shows the convergence of the expressions (3.20) and (3.21) to the benchmark value with respect to the number of inner trajectories. In particular, we observe that both expressions converge to the benchmark solution with a small variance when $M_1 = 2^{17}$.

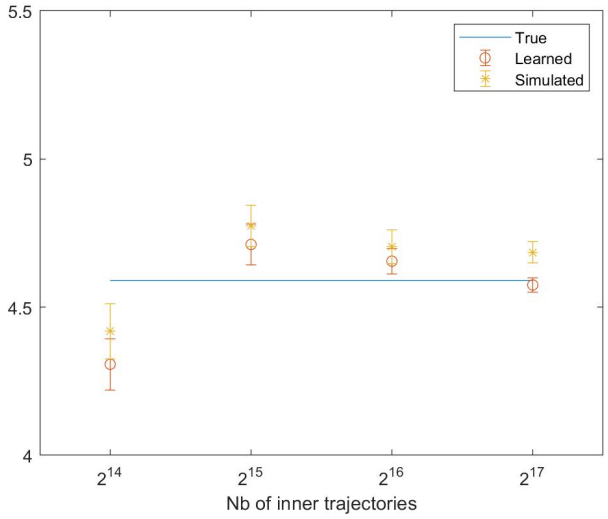


Figure 3: Numerical solution of PDE (5.4) calculated by (3.20) and (3.21) expressions: $T = 1$, $d_1 = 100$, $M_0 = 2^7$, $L = 3$.

5.5 Pricing of European financial derivatives with different interest rates for borrowing and lending

Assuming $\mu = 0.06$, $\sigma = 0.2$, $R^l = 0.04$ and $R^b = 0.06$, we introduce the following functions

$$f(t,x,y,z) = -R^l y - \frac{(\mu - R^l)}{\sigma}\sum_{i=1}^{d_1} z_i + (R^b - R^l)\max\{0, \frac{1}{\sigma}\sum_{i=1}^{d_1} z_i - y\},$$

$$f(T,x) = \max\{\max_{1\le i\le d_1} x_i - 120, 0\} - 2\max\{\max_{1\le i\le d_1} x_i - 150, 0\}$$

and

$$\mathcal{E}_{t_k}(x,w) = x\exp\left((\mu - \frac{\sigma^2}{2})\Delta_t + \sigma w\right), \quad X_{t_0} = 100.$$

Let y defined as the solution of the following PDE, $y(T,x) = f(T,x)$,

$$\frac{\partial y}{\partial t}(t,x) + \frac{\sigma}{2}\sum_{i=1}^{d_1}|x_i|^2\frac{\partial^2 y}{\partial x_i^2}(t,x) \tag{5.5}$$
$$+\max\{R^b\left(\sum_{i=1}^{d_1} x_i\left(\frac{\partial y}{\partial x_i}(t,x)\right) - y(t,x)\right), R^l\left(\sum_{i=1}^{d_1} x_i\left(\frac{\partial y}{\partial x_i}(t,x)\right) - y(t,x)\right)\} = 0.$$

PDE (5.5) has a benchmark solution given in [Section 4.4; [11]]. This benchmark solution is equal to 21.299 for $T = 0.5$ and $d_1 = 100$.

Figure 4 shows the approximation of the solution of PDE (5.5) with respect to the number of inner trajectories. As before, we illustrate the values given by both (3.20) and (3.21) expressions. We show that 2^7 outer trajectories and 2^{11} inner trajectories are sufficient to get an accurate approximation of the solution as the obtained values are in the corridor of the standard deviation of the benchmark solution given in [Section 4.4; [11]]. No bias cut is needed here and $\overline{s_k} = T$ for any $s_k \in \mathcal{S}$. The runtime with 2^7 outer trajectories and 2^{11} inner trajectories is 53 seconds.

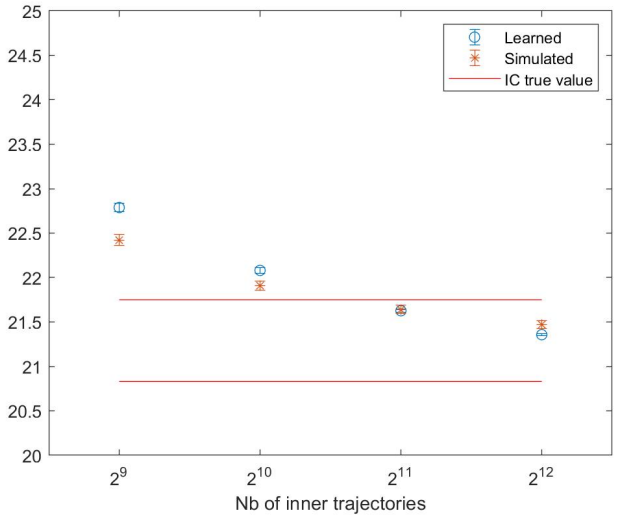


Figure 4: Numerical solution of PDE (5.5) calculated by the average learned and the simulated expressions: $T = 0.5$, $d_1 = 100$, $M_0 = 2^7$, $L = 2$.

5.6 A PDE example with quadratically growing derivatives

Assuming the (ODP) setting presented in Section 3.2, we introduce $\alpha = 0.4$ and $\psi(t,x) = \sin\left(\left[T - t + |x|_{d_1}^2\right]^\alpha\right)$ and we define the driver functions

$$f(t,x,y,z) = |z|_{d_1}^2 - |\nabla_x \psi(t,x)|_{d_1}^2 - \frac{\partial \psi}{\partial t}(t,x) - \frac{1}{2}(\Delta_x \psi)(t,x),$$

$$f(T,x) = \sin\left(|x|_{d_1}^{2\alpha}\right)$$

and

$$\mathcal{E}_{t_k}(x,w) = x + w, \quad X_{t_0} = 0.$$

Let y defined as the solution of the following PDE, $y(T,x) = f(T,x)$,

$$\begin{array}{rcl} \frac{\partial y}{\partial t}(t,x) + |\nabla_x y(t,x)|_{d_1}^2 + \frac{1}{2}(\Delta_x y)(t,x) & = & \frac{\partial \psi}{\partial t}(t,x) \\ & + & |\nabla_x \psi(t,x)|_{d_1}^2 + \frac{1}{2}(\Delta_x \psi)(t,x). \end{array} \tag{5.6}$$

Straight use of Itô's Lemma shows that PDE (5.6) has an explicit solution $y(t,x) = \psi(t,x)$, we refer the reader to [Section 5; [16]] for more details.

Considering equality (3.12), the coarse approximation $\overline{y}^{m_0,\mathcal{S}}_{\frac{248}{256},\frac{250}{256}}(x)$ of $y\left(\frac{250}{256},x\right)$, learned using $\left\{X^{m_0,m_1}_{\frac{248}{256},\frac{250}{256}}\right\}_{1\le m_1\le M_1}$, can replace the coarse approximation $\overline{y}^{m_0,\mathcal{S}}_{\frac{249}{256},\frac{250}{256}}(x)$ of $y\left(\frac{250}{256},x\right)$, learned using $\left\{X^{m_0,m_1}_{\frac{249}{256},\frac{250}{256}}\right\}_{1\le m_1\le M_1}$. The same can be said for $y\left(\frac{254}{256},x\right)$ that can be approximated by $\overline{y}^{m_0,\mathcal{S}}_{\frac{252}{256},\frac{254}{256}}(x)$ instead of $\overline{y}^{m_0,\mathcal{S}}_{\frac{253}{256},\frac{254}{256}}(x)$. In Figure 5 [Left], we then evaluate replacing $\overline{y}^{m_0,\mathcal{S}}_{\frac{249}{256},\frac{250}{256}}\left(X^{m_0,m_1}_{\frac{249}{256},\frac{250}{256}}\right)$ by $\overline{y}^{m_0,\mathcal{S}}_{\frac{248}{256},\frac{250}{256}}\left(X^{m_0,m_1}_{\frac{249}{256},\frac{250}{256}}\right)$ used for the approximation of $y\left(\frac{249}{256},X^{m_0}_{\frac{249}{256}}\right)$. In Figure 5 [Right], the distribution of $y\left(\frac{253}{256},X^{m_0}_{\frac{253}{256}}\right)$ is either simulated using $Y^{m_0}_{\frac{253}{256}}$ that involves $\overline{y}^{m_0,\mathcal{S}}_{\frac{253}{256},\frac{254}{256}}$ or $\frac{1}{M_1}\sum_{m_1=1}^{M_1}\left(\overline{y}^{m_0,\mathcal{S}}_{\frac{252}{256},\frac{254}{256}}\left(X^{m_0,m_1}_{\frac{253}{256},\frac{254}{256}}\right) + \Delta_s f\left(\frac{253}{256},\overline{y}^{m_0,\mathcal{S}}_{\frac{252}{256},\frac{254}{256}}\left(X^{m_0,m_1}_{\frac{253}{256},\frac{254}{256}}\right),\widetilde{z}^{m_0,\mathcal{S}}_{\frac{253}{256},\frac{254}{256}}\right)\right)$.

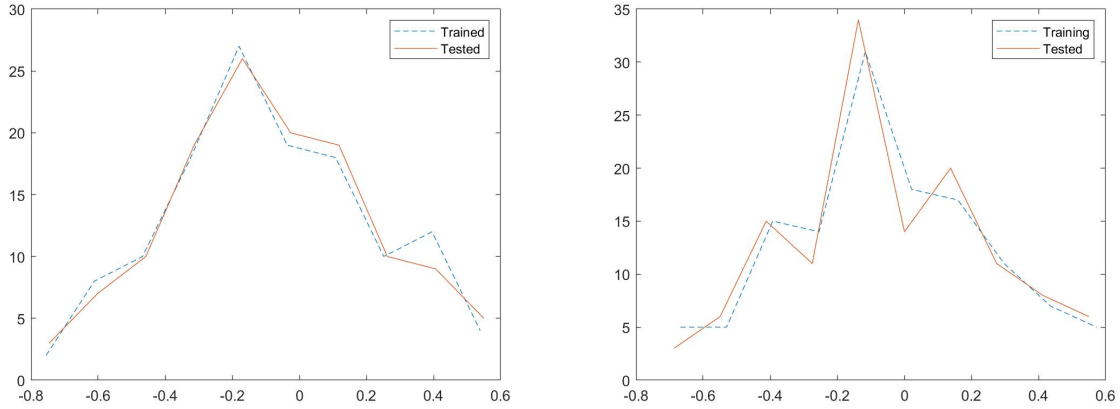


Figure 5: [Left] Distribution of $Y^{m_0}_{\frac{249}{256}}$ called "Trained" vs. its "Tested" counterpart used for the expression $\frac{1}{M_1}\sum_{m_1=1}^{M_1}\left(\overline{y}^{m_0,\mathcal{S}}_{\frac{248}{256},\frac{250}{256}}\left(X^{m_0,m_1}_{\frac{249}{256},\frac{250}{256}}\right)+\Delta_s f\left(\frac{249}{256},\overline{y}^{m_0,\mathcal{S}}_{\frac{248}{256},\frac{250}{256}}\left(X^{m_0,m_1}_{\frac{249}{256},\frac{250}{256}}\right),\overline{z}^{m_0,\mathcal{S}}_{\frac{249}{256},\frac{250}{256}}\right)\right)$ [Right] Distribution of $Y^{m_0}_{\frac{253}{256}}$ called "Trained" vs. its "Tested" counterpart used for the expression $\frac{1}{M_1}\sum_{m_1=1}^{M_1}\left(\overline{y}^{m_0,\mathcal{S}}_{\frac{252}{256},\frac{254}{256}}\left(X^{m_0,m_1}_{\frac{253}{256},\frac{254}{256}}\right)+\Delta_s f\left(\frac{253}{256},\overline{y}^{m_0,\mathcal{S}}_{\frac{252}{256},\frac{254}{256}}\left(X^{m_0,m_1}_{\frac{253}{256},\frac{254}{256}}\right),\overline{z}^{m_0,\mathcal{S}}_{\frac{253}{256},\frac{254}{256}}\right)\right)$: $T=1$, $d=100$, $M_0=2^7$, $M_1=2^{12}$, $L=8$, $\Delta_s=\Delta_t=2^{-8}$.

The obtained distributions are quite similar which strengthen the possibility of using (3.12) not only for $s=s_k$ but for various intermediary discretization times $s\in\{s_k, s_k+\Delta_t, ..., \delta_{s_j}(s_k)-\Delta_t\}$ (cf. CMCLDII [2]).

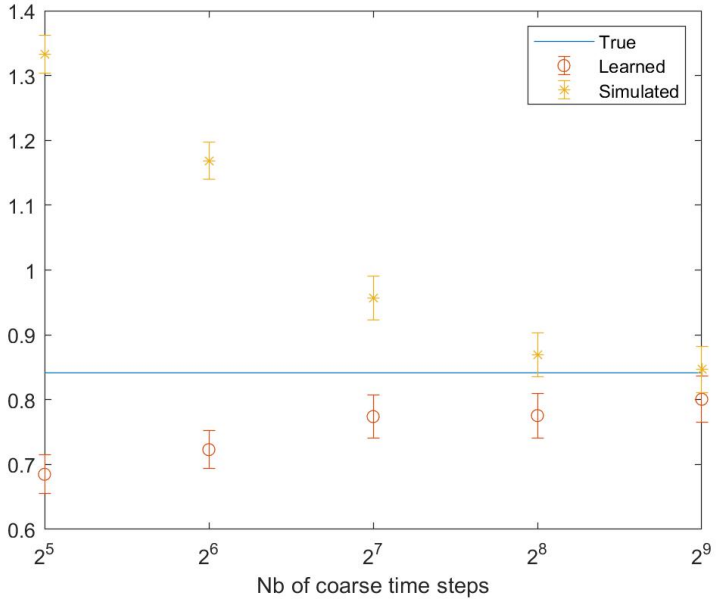


Figure 6: Numerical solution of PDE (5.6) calculated by expressions (3.20) and (3.21): $T=1$, $d_1=100$, $M_0=2^7$, $M_1=2^7$.

Figure 6 shows the numerical solution $y(0,0)$ of (5.6) with respect to various choices of coarse time steps. $L = 8$ is sufficient to discretize the problem when the time horizon $T = 1$. The latter simulation is achieved in 620 seconds of runtime.

References

[1] Abbas-Turki, L. A., Crépey, S. and Diallo, B. (2018). XVA principles, nested Monte Carlo strategies, and GPU optimizations. *International Journal of Theoretical and Applied Finance.* **21(06)**.

[2] Abbas-Turki, L. A., Diallo, B., Pagès, G. (2020) Conditional Monte Carlo Learning for Diffusions II: extended methodology and application to risk measures and early stopping problems. *Preprint on HAL.*

[3] Abbas-Turki, L. A. and Graillat, S. (2017). Resolving small random symmetric linear systems on graphics processing units. *The Journal of Supercomputing.* **73(4)**, 1360–1386.

[4] Bellman, R. (2010). *Dynamic programming*, Princeton Landmarks in Mathematics. Princeton University Press, Princeton, NJ. Reprint of the 1957 edition, With a new introduction by Stuart Dreyfus.

[5] Bouchard, B. and Touzi, N. (2004). Discrete time approximation and Monte Carlo simulation of backward stochastic differential equations. *Stochastic Processes and their Applications.* **111(2)** 175–206.

[6] Broadie, M., Du, Y. and Moallemi, C. C. (2015). Risk estimation via regression. *Operations Research.* **63(5)** 1077–1097.

[7] Buehler, H., Gonon, L., Teichmann, J. and Wood, B. (2019). Deep hedging. *Quantitative Finance.* **19(8)** 1271–1291.

[8] Chassagneux, J.-F. (2014). Linear multistep schemes for BSDEs. *SIAM J. Numer. Anal.* **52(6)** 2815–2836.

[9] Chassagneux, J.-F., and Richou A. (2016). Numerical simulation of quadratic BSDEs. *Ann. Appl. Probab.* **26(1)** 262–304.

[10] Cuppen, J. J. M. (1980). A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.* **36** 177–195.

[11] E, W., Han, J. and Jentzen, A. (2018). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics.* **5(4)** 349–380.

[12] El Karoui, N., Peng, S. and Quenez, M. C. (1997). Backward stochastic differential equations in Finance. *Mathematical Finance.* **7(1)** 349–380.

[13] Giorgi, D., Lemaire, V. and Pagès, G. (2020). Weak Error for Nested Multilevel Monte Carlo. *Methodology and Computing in Applied Probability.* **22** 1325–1348.

[14] Gobet, E. and Labart, C. (2007). Error expansion for the discretization of backward stochastic differential equations. *Stochastic Processes and their Applications.* **117(7)** 803–829.

[15] Gobet, E., Lemor, J. P. and Warin, X. (2005). A regression-based Monte Carlo method to solve backward stochastic differential equations *The Annals of Applied Probability.* **15(3)** 2172–2202.

[16] Gobet, E. and Turkedjiev, P. (2016). Linear regression MDP scheme for discrete backward stochastic differential equations under general conditions. *Mathematics of Computation.* **85** 1359–1391.

[17] Gobet, E. and Turkedjiev, P. (2017). Adaptive importance sampling in least-squares Monte Carlo algorithms for backward stochastic differential equations. *Stochastic Process. Appl.* **127(4)** 1171–1203.

[18] Gordy, M. B. and Juneja, S. (2010). Nested Simulation in Portfolio Risk Measurement. *Management Science.* **56(10)** 1833–1848.

[19] Goudenège, L., Molent, A. and Zanette, A. (2019). Variance Reduction Applied to Machine Learning for Pricing Bermudan/American Options in High Dimension. *Preprint.* https://arxiv.org/abs/1903.11275

[20] Lee, S.-H. (1998). *Monte Carlo Computation of Conditional Expectation Quantiles*, Ph.D. thesis, Stanford University.

[21] Lee, S.-H. and Glynn, P. W. (2003). Computing the distribution function of a conditional expectation via Monte Carlo: Discrete conditioning spaces. *ACM Transactions on Modeling and Computer Simulation.* **13(3)** 238–258.

[22] Lemor, J. P., Gobet, E. and Warin, X. (2006). Rate of convergence of an empirical regression method for solving generalized backward stochastic differential equations. *Bernoulli.* **12(5)** 889–916.

[23] Lions, J.-L., Maday, Y. and Turinici, G. (2015). A "parareal" in time discretization of PDE's. *Comptes Rendus de l'Académie des Sciences.* **Série I. 332(7)** 661–668.

[24] NVIDIA (2017). *Cuda C PROGRAMMING GUIDE.*

[25] Pagès, G. (2018). *Numerical probability: an introduction with applications to finance*, Universitext, Springer.

[26] Pardoux, E. and Peng, S. (1990). Adapted solution of a backward stochastic differential equation. *Systems & Control Letters.* **14** 55–61.

[27] Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (2002). *Numerical Recipes in C++: The Art of Scientific Computing*, Cambridge University Press.

[28] White, H. (2001). *Asymptotic theory for econometricians.* Academic Press.