Deep backward multistep schemes for nonlinear PDEs and approximation error analysis

Maximilien Germain, Huyen Pham, Xavier Warin

Deep backward multistep schemes for nonlinear PDEs and approximation error analysis *

Maximilien Germain † Huyên Pham ‡ Xavier Warin §

June 1, 2020

Abstract

We develop multistep machine learning schemes for solving nonlinear partial differential equations (PDEs) in high dimension. The method is based on probabilistic representation of PDEs by backward stochastic differential equations (BSDEs) and its iterated time discretization. In the case of semilinear PDEs, our algorithm estimates simultaneously by backward induction the solution and its gradient by neural networks through sequential minimizations of suitable quadratic loss functions that are performed by stochastic gradient descent. The approach is extended to the more challenging case of fully nonlinear PDEs, and we propose different approximations of the Hessian of the solution to the PDE, i.e., the Γ-component of the BSDE, by combining Malliavin weights and neural networks. Extensive numerical tests are carried out with various examples of semilinear PDEs including viscous Burgers equation and examples of fully nonlinear PDEs like Hamilton-Jacobi-Bellman equations arising in portfolio selection problems with stochastic volatilities, or Monge-Ampère equations in dimension up to 15. The performance and accuracy of our numerical results are compared with some other recent machine learning algorithms in the literature, see [HJE17], [HPW20], [BEJ19], [Bec+19] and [PWG19].

Furthermore, we provide a rigorous approximation error analysis of the deep backward multistep scheme as well as the deep splitting method for semilinear PDEs, which yields convergence rate in terms of the number of neurons for shallow neural networks.

Key words: Nonlinear PDEs, Backward SDEs, neural networks, numerical approximation, multistep schemes, Malliavin weights, error estimates.

MSC Classification: 60H35, 65C20, 65M12.

*This work is supported by FiME, Laboratoire de Finance des Marchés de l'Energie, and the "Finance and Sustainable Development" EDF - CACIB Chair.

†EDF R&D, LPSM, Université de Paris mgermain at lpsm.paris

‡LPSM, Université de Paris, FiME, CREST ENSAE pham at lpsm.paris

§EDF R&D, FiME xavier.warin at edf.fr

Contents

1 Introduction **2**

2 BSDE Machine learning schemes for semilinear PDEs **4**
2.1 Neural networks . . . 4
2.2 Existing schemes . . . 7
2.3 Deep backward multi-step scheme (MDBDP) . . . 9

3 Extension to fully non linear PDEs: second order deep backward multistep scheme **10**

4 Convergence analysis in the semilinear case **13**
4.1 Convergence of the MDBDP scheme . . . 14
4.2 Convergence of the DS scheme . . . 15

5 Proof of the main theoretical results **16**
5.1 Proof of Theorem 4.1 . . . 17
5.2 Proof of Proposition 4.1 . . . 19
5.3 Proof of Theorem 4.2 . . . 21
5.4 Proof of Proposition 4.2 . . . 23

6 Numerical study **25**
6.1 Semilinear PDEs . . . 25
6.1.1 PDE with bounded solution and simple structure . . . 25
6.1.2 PDE with unbounded solution and more complex structure . . . 27
6.1.3 Viscous Burgers equation . . . 29
6.2 Fully nonlinear PDEs . . . 31
6.2.1 PDE with bounded solution and simple structure . . . 31
6.2.2 Monge-Ampère equation . . . 34
6.2.3 Portfolio selection . . . 35
6.3 Results synthesis . . . 40

1 Introduction

Let us consider the nonlinear parabolic partial differential equation (PDE) of the form

$$\begin{cases} \partial_t u + \mu \cdot D_x u + \frac{1}{2}\,\mathrm{Tr}(\sigma\sigma^\intercal D_x^2 u) \;=\; f(\cdot,\cdot,u,\sigma^\intercal D_x u) & \text{on } [0,T)\times\mathbb{R}^d \\ u(T,\cdot) \;=\; g & \text{on } \mathbb{R}^d, \end{cases} \tag{1.1}$$

with μ, σ functions defined on $[0,T]\times\mathbb{R}^d$, valued respectively in $\mathbb{R}^d$, and $\mathbb{M}^d$ (the set of $d\times d$ matrices), a nonlinear generator function f defined on $[0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^d$, and a terminal function g defined on $\mathbb{R}^d$. Here, the operators D_x, D_x^2 refer respectively to the first and second order spatial derivatives, the symbol . denotes the scalar product, and $^\intercal$ is the transpose of vector or matrix.

A major challenge in the numerical resolution of such semilinear PDEs is the so-called "curse of dimensionality" making unfeasible the standard discretization of the state space in dimension greater than 3. Probabilistic mesh-free methods based on the Backward Stochastic Differential Equation (BSDE) representation of semilinear PDEs through the nonlinear Feynman-Kac formula were developed in [Zha04], [BT04], [GLW05], [LGW06] to overcome this obstacle. These schemes are successfully applied upon dimension 6 or 7, nevertheless, their use of regression methods implies a dimension dependence through the number of required basis functions. Let us also mention recent probabilistic approach relying on (i) branching method, see [HL+19] and [War17], and (ii) on multilevel Picard methods, developed in [E+18] and [Hut+18] with algorithms based on Picard iterations, multi-level techniques and automatic differentiation. These methods permit to handle some high dimensional PDEs with non linearity in u and its gradient $D_x u$, with convergence results as well as numerous numerical examples showing their efficiency in high dimension.

An even more challenging problem is to design numerical schemes for high dimensional fully nonlinear PDEs of the form

$$\begin{cases} \partial_t u + \mu \cdot D_x u + \frac{1}{2}\,\mathrm{Tr}(\sigma\sigma^\intercal D_x^2 u) \;=\; F(\cdot,\cdot,u,D_x u, D_x^2 u) & \text{on } [0,T)\times\mathbb{R}^d \\ u(T,\cdot) \;=\; g & \text{on } \mathbb{R}^d. \end{cases} \tag{1.2}$$

In this case, the nonlinearity of the PDE also carries on the second order derivative through the function F, which is now defined on the larger space $[0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^d\times\mathbb{S}^d$ with $\mathbb{S}^d$ the set of symmetric $d\times d$ matrices.

Several methods are available to solve fully nonlinear equations mostly in low or moderate dimension. Let us mention among others:

- Deterministic methods based on finite-difference and finite-element: we refer to [FGN13] for a review of this extensive literature.
- Max-plus methods for Hamilton-Jacobi-Bellman equations, see e.g. [McE07], [AGL08].
- An effective scheme developed in [FTW11], using some regression techniques, has been shown to be convergent under some ellipticity conditions. Due to the use of basis functions, this scheme does not permit to solve PDE in dimension greater than 5.
- A scheme based on nesting Monte Carlo has been recently proposed in [War18]. It seems to be effective in very high dimension for not too long maturities and small non linearities. However, no results of convergence is given.

Over the last few years, machine learning methods have emerged since the pioneering papers by [HJE17] and [SS17], and have shown their efficiency for solving high-dimensional nonlinear PDEs by means of neural networks approximation. The work [HJE17] introduces a global machine learning resolution technique for the BSDE approach. The solution is represented by one feedforward neural network by time step, whose parameters are chosen as solutions of a single global optimization problem. It allows to solve PDEs in high dimension and a convergence study of Deep BSDE is conducted in [HL18]. Variants with a single neural network or Long Short Term Memory (LSTM) networks are extensively tested in [CWNMW19], alongside a fixed point machine learning method relying on branching processes. Extension to fully nonlinear PDEs has been proposed by [BEJ19] based on the second order backward stochastic differential equations (2BSDE) representation of [Che+07] and global deep neural networks minimizing a terminal objective function, but no test on real fully nonlinear case is given. The Deep Galerkin method of [SS17] proposes another global meshfree method with a random sampling of time and space points inside a bounded domain. Their authors propose a convergence study without rate of the scheme in Lebesgue spaces for semilinear equations on bounded domains. Neural networks with a specific structure especially designed to represent Hamilton Jacobi Bellman equations solutions are also derived in [DLM19].

A different point of view is proposed by [HPW20] with convergence results in L^2 for solving semilinear PDEs, where the solution and its gradient are estimated simultaneously by backward induction through the minimization of sequential loss functions. Similar idea also appears in [VSS18] for linear PDEs. At the cost of solving multiple optimization problems, the Deep Backward scheme of [HPW20] verifies better stability and accuracy properties than the global method in [HJE17], as illustrated on several test cases. The recent paper [Bec+19] also introduces machine learning schemes based on local loss functions, called Deep Splitting method which estimates the PDE solution through backward explicit local optimization problems relying on a neural network regression method for the computation of conditional expectations. Finally, we mention our paper [PWG19], which combines ideas in [HPW20] and [Bec+19] to propose a neural networks-based scheme for fully nonlinear PDEs where the Hessian is approximated by automatic differentiation of the gradient computed at the previous step.

In this paper, we propose machine learning schemes that use multistep methods introduced in [BD07] and [GT14] (see also [GT16], [Tur15]). The idea is to rely on the whole previously computed values of the discretized processes in the backward computations of the approximation. According to these works, it leads to a better propagation of regression errors. We shall adapt this approach to the deep backward scheme (DBDP) of [HPW20], leading to the so-called deep backward multi-step scheme (MDBDP). This can be viewed as a machine learning version of the Multi-step Forward Dynamic Programming method studied by [GT14]. However, instead of solving at each time step two regression problems, our approach allows to consider only a single minimization as in the DBDP scheme. Compared to the latter, the multi-step consideration is expected to provide better accuracy by reducing the propagation of errors in the backward induction. Our main theoretical contribution is a detailed study of the approximation error of MDBDP scheme. Furthermore, by relying on recent approximation results for shallow neural networks in [Bac17], and by deriving estimates for Lipschitz constants of shallow neural networks and its gradient, we can obtain a rate of convergence of our scheme in terms on the number of neurons. The arguments can be adapted to show the convergence of the deep splitting (DS) method in [Bec+19], and we show in particular that the error rate for MDBDP improves the rate of DBDP and DS schemes.

Next, we extend this deep multistep backward scheme to the fully nonlinear PDE case where the main difficulty is to provide en efficient approximation of the Hessian, or equivalently of the Γ-component of the BSDE. We first give a multistep version of the scheme in [PWG19]. Then, we combine ideas from the Malliavin regression scheme of [GT16] together with the monotone scheme from [FTW11], in order to approximate the Hessian by neural networks. This can be done with Malliavin weights of order one or two, hence corresponding

to two versions of the algorithm, and it leads to more stable approximation of the Hessian than the algorithm in [PWG19].

We provide numerous and various numerical tests of our proposed algorithms both for semilinear and fully nonlinear PDEs, and we compare our results with the cited machine learning schemes. Our examples include viscous Burgers equations, fully nonlinear Hamilton-Jacobi-Bellman equations arising from portfolio selection problems, and Monge-Ampère equations up to dimension 15.

The outline of the paper is organized as follows. In Section 2, we give a brief reminder on neural networks and notably on a specific class of shallow network functions considered in [Bac17] that yields an approximation result with rate of convergence for Lipschitz functions. We also review recent machine learning schemes for the numerical resolution of semilinear PDEs. We then describe in detail the deep backward multi-step scheme. We then extend in Section 3 our algorithm to the case of fully nonlinear PDEs. We state in Section 4 the convergence of the deep backward multi-step and splitting schemes, while Section 5 is devoted to the proof of these results. Finally, we present all the numerical results in Section 6, and we conclude with some discussion about the pros and cons of the different tested algorithms. All the codes of the implemented algorithms in this paper are available at: `https://github.com/MaxGermain/MultistepBSDE`.

2 BSDE Machine learning schemes for semilinear PDEs

In this section, we review recent numerical schemes, and present a new scheme for the resolution of the semilinear PDE (1.1) by approximations in the class of neural networks and relying on probabilistic representation of the solution to the PDE.

2.1 Neural networks

We denote by

$$\mathcal{L}^{\rho}_{d_1,d_2} = \Big\{\phi : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} : \exists\, (\mathcal{W},\beta) \in \mathbb{R}^{d_2\times d_1}\times\mathbb{R}^{d_2},\ \phi(x) \ = \ \rho(\mathcal{W}x+\beta)\ \Big\},$$

the set of layer functions with input dimension d_1, output dimension d_2, and activation function $\rho : \mathbb{R} \to \mathbb{R}$. Here, the activation is applied component-wise, i.e., $\rho(x_1,\dots,x_{d_2}) = \big(\rho(x_1),\dots,\rho(x_{d_2})\big)$, to the affine map $x \in \mathbb{R}^{d_1} \mapsto \mathcal{W}x+\beta \in \mathbb{R}^{d_2}$, with a matrix $\mathcal{W}$ called weight, and vector β called bias. Standard examples of activation functions are the sigmoid, the ReLu, the Elu, tanh. When ρ is the identity function, we simply write $\mathcal{L}_{d_1,d_2}$.

We then define

$$\mathcal{N}^{\rho}_{d_0,d',\ell,m} = \Big\{\varphi : \mathbb{R}^{d_0} \to \mathbb{R}^{d'} : \exists\phi_0 \in \mathcal{L}^{\rho}_{d_0,m},\ \exists\phi_i \in \mathcal{L}^{\rho}_{m,m}, i=1,\dots,\ell-1,\ \exists\phi_\ell \in \mathcal{L}_{m,d'},$$
$$\varphi \ = \ \phi_\ell\circ\phi_{\ell-1}\circ\cdots\circ\phi_0\Big\},$$

as the set of feedforward (or artificial) neural networks with input layer dimension d_0, output layer dimension d', and ℓ hidden layers with m neurons (or units). These numbers d_0, d', ℓ, m, and the activation function ρ, form the architecture of the network. When $\ell = 1$, one usually refers to shallow neural networks, as opposed to deep neural networks which have several hidden layers. In the sequel, we shall mostly work with the case $d_0 = d$ (dimension of the state variable x) and $d' = 1$ (dimension of value function u) or $d' = d$ (dimension of gradient function $D_x u$).

A given network function $\varphi \in \mathcal{N}^{\rho}_{d_0,d',\ell,m}$ is determined by the weight/bias parameters $\theta = (\mathcal{W}_0,\beta_0,\dots,\mathcal{W}_\ell,\beta_\ell)$ defining the layer functions $\phi_0\dots,\phi_\ell$, and we shall sometimes write $\varphi = \varphi_\theta$. The set of parameters is denoted by Θ. When there are no constraints as in the above definition and practically the case in numerical implementation, $\Theta = \mathbb{R}^M$, where $M = m(d_0+1)+m(m+1)(\ell-1)+d'(m+1)$ is the number of parameters. For theoretical analysis, we may consider sparsity-inducing norms on the parameters as in [Bac17] for an architecture with a single hidden layer $\ell = 1$ and ReLu_α activation function, i.e., $\rho(x) = (x_+)^\alpha$, for some integer α, with $x_+ = \max(x,0)$. In this case, we denote by $\mathcal{N}^{\alpha,R,\gamma}_{d,m,d'}$, for $\gamma, R \geq 1$, the set of network functions $\varphi_\theta : \mathbb{R}^d \to \mathbb{R}^{d'}$, with parameters $\theta = (\mathcal{W}_0,\beta_0,\mathcal{W}_1,\beta_1) \in \mathbb{R}^{m\times d}\times\mathbb{R}^m\times\mathbb{R}^{d'\times m}\times\mathbb{R}^{d'}$ satisfying row by row

$$|(\mathcal{W}_0^i,\beta_0^i/R)|_2 \ = \ \frac{1}{R},\ \ i\in[\![1,m]\!],\ \text{and}\ |(\mathcal{W}_1^k,\beta_1^k)|_1 \ \leq\ \gamma,\ \ k\in[\![1,d']\!], \tag{2.1}$$

where $|.|_1$, $|.|_2$ and $|.|_\infty$ are respectively the ℓ_1, ℓ_2 and ℓ_∞ norms in Euclidian spaces. We simply denote as usual $|.|$ for the norm on $\mathbb{R}$.

In the sequel, we shall mainly focus on the cases $\alpha = 1$ or 2, and $d' = 1$ or d. Notice that a network function $\varphi \in \mathcal{N}^{\alpha,R,\gamma}_{d,m,d'}$, for $\alpha = 2$, is C^1. We state an elementary Lemma about the growth and Lipschitz properties on network functions in theses classes.

Lemma 2.1. *(1) Let $\phi \in \mathcal{N}_{d,m,d'}^{1,R,\gamma}$. Then*

$$|\phi(x)|_2 \le \sqrt{d'}|\phi(x)|_\infty \le \sqrt{dd'}\gamma \max\Big(1, \frac{|x|_2}{R}\Big), \quad x \in \mathbb{R}^d, \tag{2.2}$$

$$|\phi(x) - \phi(x')|_2 \le \sqrt{d'}\,|\phi(x) - \phi(x')|_\infty \le \sqrt{d'}\frac{\gamma}{R}|x - x'|_2, \qquad x, x' \in \mathbb{R}^d. \tag{2.3}$$

(2) Let $\varphi \in \mathcal{N}_{d,m,1}^{2,R,\gamma}$. Then,

$$|\varphi(x)| \le \gamma\Big(1 + 2\max\big[1, \frac{|x|_2^2}{R^2}\big]\Big), \qquad x \in \mathbb{R}^d, \tag{2.4}$$

$$|D_x\varphi(x)|_2 \le \sqrt{d}|D_x\varphi(x)|_\infty \le 2d\frac{\gamma}{R}\max\Big(1, \frac{|x|_2}{R}\Big), \qquad x \in \mathbb{R}^d, \tag{2.5}$$

$$\big|D_x\varphi(x) - D_x\varphi(x')\big|_\infty \le 2\frac{\gamma}{R^2}|x - x'|_2, \qquad x, x' \in \mathbb{R}^d. \tag{2.6}$$

Proof. (1) By definition of $\phi = \phi_\theta \in \mathcal{N}_{d,m,d'}^{1,R,\gamma}$, with parameter $\theta = (\mathcal{W}_0, \beta_0, \mathcal{W}_1, \beta_1)$, the k-th component of the $\mathbb{R}^{d'}$-valued function ϕ is equal to

$$\phi^k(x) = \sum_{i=1}^m \mathcal{W}_1^{ki}\Big(\sum_{j=1}^d \mathcal{W}_0^{ij}x_j + \beta_0^i\Big)_+ + \beta_1^k, \quad x \in \mathbb{R}^d,\ k \in [\![1, d']\!].$$

We deduce that

$$\begin{aligned}
\big|\phi^k(x)\big| &\le \sum_{i=1}^m |\mathcal{W}_1^{ki}|\Big[|x|_2\sum_{j=1}^d |\mathcal{W}_0^{ij}| + |\beta_0^i|\Big] + |\beta_1^k| \\
&\le \max(|x|_2, R)\sum_{i=1}^m |\mathcal{W}_1^{ki}||(\mathcal{W}_0^i, \beta_0^i/R)|_1 + |\beta_1^k| \\
&\le \sqrt{d}\max(|x|_2, R)\sum_{i=1}^m |\mathcal{W}_1^{1i}||(\mathcal{W}_0^i, \beta_0^i/R)|_2 + |\beta_1^k| \le \sqrt{d}\max\Big(1, \frac{|x|_2}{R}\Big)\sum_{i=1}^m |\mathcal{W}_1^{1i}| + |\beta_1^k| \\
&\le \sqrt{d}\max\Big(1, \frac{|x|_2}{R}\Big)|(\mathcal{W}_1^k, \beta_1^k)|_1 \le \sqrt{d}\gamma\max\Big(1, \frac{|x|_2}{R}\Big),
\end{aligned}$$

where we used the fact that $|.|_1 \le \sqrt{d}|.|_2$ in $\mathbb{R}^d$, and the condition (2.1) on θ. We get the required growth condition (2.2) by recalling that $|\cdot|_2 \le \sqrt{d'}|\cdot|_\infty$ in $\mathbb{R}^{d'}$.

On the other hand, since ReLu function is 1-Lipschitz, we have for all $x, x' \in \mathbb{R}^d$,

$$\begin{aligned}
|\phi^k(x) - \phi^k(x')| &\le \sum_{i=1}^m |\mathcal{W}_1^{ki}|\sum_{j=1}^d |\mathcal{W}_0^{ij}||x^j - x'^j| \\
&\le \sum_{i=1}^m |\mathcal{W}_1^{ki}||\mathcal{W}_0^i|_2|x - x'|_2 \le \frac{1}{R}|\mathcal{W}_1^k|_1|x - x'|_2 \le \frac{\gamma}{R}|x - x'|_2,
\end{aligned}$$

by Cauchy-Schwarz inequality, and the condition (2.1) on θ. This proves the required Lipschitz property (2.3) for ϕ.

(2) By definition of $\varphi = \varphi_\theta \in \mathcal{N}_{d,m,1}^{2,R,\gamma}$, with $\theta = (\mathcal{W}_0, \beta_0, \mathcal{W}_1, \beta_1)$, we have

$$\varphi(x) = \sum_{i=1}^m \mathcal{W}_1^{1i}\Big|\Big(\sum_{j=1}^d \mathcal{W}_0^{ij}x_j + \beta_0^i\Big)_+\Big|^2 + \beta_1, \quad x \in \mathbb{R}^d, \tag{2.7}$$

and thus by Cauchy-Schwarz inequality

$$\begin{aligned}
|\varphi(x)| &\le 2\sum_{i=1}^m |\mathcal{W}_1^{1i}|\Big[|x|_2^2\sum_{j=1}^d |\mathcal{W}_0^{ij}|^2 + |\beta_0^i|^2\Big] + |\beta_1| \\
&\le 2\max(|x|_2^2, R^2)\sum_{i=1}^m |\mathcal{W}_1^{1i}||(\mathcal{W}_0^i, \beta_0^i/R)|_2^2 + |\beta_1|.
\end{aligned}$$

From the condition (2.1) on θ, this shows the required quadratic growth condition on φ. Next, from (2.7), we derive

$$\partial_{x_j}\varphi(x) = 2\sum_{i=1}^{m}\mathcal{W}_1^{1i}\mathcal{W}_0^{ij}\big(\sum_{j=1}^{d}\mathcal{W}_0^{ij}x_j+\beta_0^i\big)_+, \quad j=1,\dots,d, \tag{2.8}$$

from which we deduce that

$$\begin{aligned}\big|D_x\varphi(x)\big|_\infty &\le 2\sum_{i=1}^{m}|\mathcal{W}_1^{1i}||\mathcal{W}_0^i|_2\Big[|x|_2\sum_{j=1}^{d}|\mathcal{W}_0^{ij}|+|\beta_0^i|\Big]\\ &\le 2\max(|x|_2,R)\sum_{i=1}^{m}|\mathcal{W}_1^{1i}||\mathcal{W}_0^i|_2|(\mathcal{W}_0^i,\beta_0^i/R)|_1\\ &\le 2\sqrt{d}\max(|x|_2,R)\sum_{i=1}^{m}|\mathcal{W}_1^{1i}||\mathcal{W}_0^i|_2|(\mathcal{W}_0^i,\beta_0^i/R)|_2.\end{aligned}$$

From the condition (2.1) on θ, this shows the required linear growth condition on $D_x\varphi$.

Finally, from (2.8), and since ReLu function is 1-Lipschitz, we see by Cauchy-Schwarz inequality that for all $x,x'\in\mathbb{R}^d$,

$$\big|D_x\varphi(x)-D_x\varphi(x')\big|_\infty \le 2\sum_{i=1}^{m}|\mathcal{W}_1^{1i}||W_0^i|_2|W_0^i|_1|x-x'|_2,$$

which ends the proof by using again the condition (2.1) on θ. □

Remark 2.1. Relation (2.3)in Lemma 2.1 means that any $\phi\in\mathcal{N}_{d,m,d}^{1,R,\gamma}$ is Lipschitz on $\mathbb{R}^d$ with Lipschitz constant $[\phi]_L\le C(d)\gamma/R$, for some constant $C(d)$ that depends only on the input dimension d (but not on the number of neurons). Relation (2.5) shows in particular that any $\varphi\in\mathcal{N}_{d,m,1}^{2,R,\gamma}$ is locally Lipschitz, and

$$[\varphi]_{L,R} := \sup_{x,x'\in B_2(R),x\neq x'}\frac{|\varphi(x)-\varphi(x')|}{|x-x'|_2} \le C(d)\frac{\gamma}{R},$$

Here, $B_2(R)$ denotes the ball $B_2(R):=\{x\in\mathbb{R}^d:|x|_2\le R\}$. Moreover, relation (2.6) means that $D_x\varphi$ is Lipschitz continuous on $\mathbb{R}^d$ with Lipschitz constant $[D_x\varphi]_L\le C(d)\frac{\gamma}{R^2}$. □

We recall the fundamental results of [HSW89]-[HSW90] that justify the use of neural networks as function approximators.

Universal approximation theorem. The space $\bigcup_{m=1}^{\infty}\mathcal{N}_{d_0,d',\ell,m}^{\rho}$ is dense in $L^2(\nu)$, the set of measurable functions $h:\mathbb{R}^{d_0}\to\mathbb{R}^{d'}$ s.t. $\int|h(x)|_2^2\nu(dx)<\infty$, for any finite measure ν on $\mathbb{R}^{d_0}$, whenever ρ is continuous and non-constant.

This universal approximation theorem does not provide any rate of convergence, nor reveals even in theory how to achieve a given accuracy for a fixed number of neurons. There are few results in the literature that prove precise rates of convergence for approximation with deep neural networks, and most of them focus on single hidden layer. We mention the recent approximation theorem for (locally) Lipschitz continuous functions, which results from Proposition 6 combined with Proposition 1 (for $\alpha=1$) or Section 2.5 (for $\alpha=2$), in [Bac17], and motivates the introduction of the set of network functions $\mathcal{N}_{d,m,d'}^{\alpha,R,\gamma}$.

Approximation of Lipschitz continuous functions with finitely many neurons. For any locally Lipchitz continuous function $h:\mathbb{R}^d\to\mathbb{R}^{d'}$, s.t. $[h]_{L,R}\le\eta$, there exists $\varphi\in\mathcal{N}_{d,m,d'}^{\alpha,R,\gamma}$, for some γ larger than a constant depending only on d, such that

$$\sqrt{\int_{x\in B_2(R)}|h(x)-\varphi(x)|_2^2\,\rho(\mathrm{d}x)} \le C(d,d')\Big(\eta\Big(\frac{\gamma}{\eta}\Big)^{-\frac{1}{\alpha+(d-1)/2}}\ln\Big(\frac{\gamma}{\eta}\Big)+\gamma m^{-d_\alpha}\Big), \tag{2.9}$$

with $d_\alpha=(d+3)/(2d)$ for $\alpha=1$, and $d_\alpha=1/2$ for $\alpha=2$, and where $C(d,d')$ is a constant that depends only on d,d', and ρ is a probability measure on $\mathbb{R}^d$.

2.2 Existing schemes

We recall recent machine learning schemes that will serve as benchmarks for our new scheme described in the next section. All these schemes rely on Backward Stochastic Differential Equation (BSDE) representation of the solution to the PDE, and differ according to the formulation of the time discretization of the BSDE

For this purpose, let us introduce the diffusion process $\mathcal{X}$ in $\mathbb{R}^d$ associated to the linear part of the differential operator in the PDE (1.1), namely:

$$\mathcal{X}_t = \mathcal{X}_0 + \int_0^t \mu(s, \mathcal{X}_s)\mathrm{d}s + \int_0^t \sigma(s, \mathcal{X}_s)\mathrm{d}W_s, \quad 0 \le t \le T, \tag{2.10}$$

where W is a d-dimensional standard Brownian motion on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$ equipped with a filtration $\mathbb{F} = (\mathcal{F}_t)_t$, and $\mathcal{X}_0$ is an $\mathcal{F}_0$-measurable random variable valued in $\mathbb{R}^d$. Recall from [PP90] that the solution u to the PDE (1.1) admits a probabilistic representation in terms of the BSDE:

$$Y_t = g(\mathcal{X}_T) - \int_t^T f(s, \mathcal{X}_s, Y_s, Z_s)\mathrm{d}s - \int_t^T Z_s.\mathrm{d}W_s, \quad 0 \le t \le T, \tag{2.11}$$

via the Feynman-Kac formula $Y_t = u(t, \mathcal{X}_t)$, $0 \le t \le T$. When u is a smooth function, this BSDE representation is directly obtained by Itô's formula applied to $u(t, \mathcal{X}_t)$, and we have $Z_t = \sigma(t, \mathcal{X}_t)^\intercal D_x u(t, \mathcal{X}_t)$, $0 \le t \le T$.

Let π be a subdivision $\{t_0 = 0 < t_1 < \cdots < t_N = T\}$ with modulus $|\pi| := \sup_i \Delta t_i$, $\Delta t_i := t_{i+1} - t_i$, satisfying $|\pi| = O\left(\frac{1}{N}\right)$, and consider the Euler-Maruyama discretization $(X_i)_{i=0,\dots,N}$ defined by

$$X_i = \mathcal{X}_0 + \sum_{j=0}^{i-1} \mu(t_j, X_j)\Delta t_j + \sum_{j=0}^{i-1} \sigma(t_j, X_j)\Delta W_j,$$

where $\Delta W_j := W_{t_{j+1}} - W_{t_j}$, $j = 0, \dots, N$. When the diffusion process $\mathcal{X}$ cannot be directly simulated, we shall rely on the simulated paths of $(X_i)_i$ that act as training data in the setting of machine learning, and thus our training set can be chosen as large as desired.

The time discretization of the BSDE (2.11) can be written in backward induction as

$$Y_i^\pi = Y_{i+1}^\pi - f(t_i, X_i, Y_i^\pi, Z_i^\pi)\Delta t_i - Z_i^\pi.\Delta W_i, \quad i = 0, \dots, N-1, \tag{2.12}$$

which also reads as conditional expectation formulae

$$\left\{ \begin{array}{rcl} Y_i^\pi & = & \mathbb{E}_i\Big[Y_{i+1}^\pi - f(t_i, X_i, Y_i^\pi, Z_i^\pi)\Delta t_i\Big] \\ Z_i^\pi & = & \mathbb{E}_i\Big[\frac{\Delta W_i}{\Delta t_i} Y_{i+1}^\pi\Big], \qquad i = 0, \dots, N-1, \end{array} \right. \tag{2.13}$$

where $\mathbb{E}_i$ denotes the conditional expectation w.r.t. $\mathcal{F}_{t_i}$. Alternatively, by iterating relations (2.12) together with the terminal relation $Y_N^\pi = g(X_N)$, we have

$$Y_i^\pi = g(X_N) - \sum_{j=i}^{N-1} \big[f(t_j, X_j, Y_j^\pi, Z_j^\pi)\Delta t_j + Z_j^\pi.\Delta W_j\big], \quad i = 0, \dots, N-1. \tag{2.14}$$

• Deep BSDE scheme [HJE17].

The idea of the method is to treat the backward equation (2.12) as a forward equation by approximating the initial condition Y_0 and the Z component at each time by networks functions of the X process, so as to match the terminal condition. More precisely, the problem is to minimize over network functions $\mathcal{U}_0 : \mathbb{R}^d \to \mathbb{R}$, and sequences of network functions $\boldsymbol{\mathcal{Z}} = (\mathcal{Z}_i)_i$, $\mathcal{Z}_i : \mathbb{R}^d \to \mathbb{R}^d$, $i = 0, \dots, N-1$, the global quadratic loss function

$$J_G(\mathcal{U}_0, \boldsymbol{\mathcal{Z}}) = \mathbb{E}\Big|Y_N^{\mathcal{U}_0, \boldsymbol{\mathcal{Z}}} - g(X_N)\Big|^2,$$

where $(Y_i^{\mathcal{U}_0, \boldsymbol{\mathcal{Z}}})_i$ is defined by forward induction as

$$Y_{i+1}^{\mathcal{U}_0, \boldsymbol{\mathcal{Z}}} = Y_i^{\mathcal{U}_0, \boldsymbol{\mathcal{Z}}} + f(t_i, X_i, Y_i^{\mathcal{U}_0, \boldsymbol{\mathcal{Z}}}, \mathcal{Z}_i(X_i))\Delta t_i + \mathcal{Z}_i(X_i).\Delta W_i, \quad i = 0, \dots, N-1,$$

starting from $Y_0^{\mathcal{U}_0, \boldsymbol{\mathcal{Z}}} = \mathcal{U}_0(\mathcal{X}_0)$. The output of this scheme, for the solution $(\widehat{\mathcal{U}}_0, \widehat{\boldsymbol{\mathcal{Z}}})$ to this global minimization problem, provides an approximation $\widehat{\mathcal{U}}_0$ of the solution $u(0, .)$ to the PDE at time 0, and approximations $Y_i^{\widehat{\mathcal{U}}_0, \widehat{\boldsymbol{\mathcal{Z}}}}$ of the solution to the PDE (1.1) at times t_i evaluated at $\mathcal{X}_{t_i}$, i.e., of $Y_{t_i} = u(t_i, \mathcal{X}_{t_i})$, $i = 0, \dots, N$.

• Deep Backward Dynamic Programming (DBDP) [HPW20].

The method relies on the backward dynamic programming relation (2.12) arising from the time discretization of the BSDE, and learns simultaneously at each time step t_i the pair (Y_{t_i}, Z_{t_i}) with neural networks trained with the forward process X and the Brownian motion W. The scheme has two versions:

1. *DBDP1.* Starting from $\widehat{\mathcal{U}}_N^{(1)} = g$, proceed by backward induction for $i = N-1, \dots, 0$, by minimizing over network functions $\mathcal{U}_i : \mathbb{R}^d \to \mathbb{R}$, and $\mathcal{Z}_i : \mathbb{R}^d \to \mathbb{R}^d$ the local quadratic loss function

$$J_i^{(B1)}(\mathcal{U}_i, \mathcal{Z}_i) = \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}^{(1)}(X_{i+1}) - \mathcal{U}_i(X_i) - f(t_i, X_i, \mathcal{U}_i(X_i), \mathcal{Z}_i(X_i))\Delta t_i - \mathcal{Z}_i(X_i).\Delta W_i\Big|^2,$$

and update $(\widehat{\mathcal{U}}_i^{(1)}, \widehat{\mathcal{Z}}_i^{(1)})$ as the solution to this local minimization problem.

2. *DBDP2.* Starting from $\widehat{\mathcal{U}}_N^{(2)} = g$, proceed by backward induction for $i = N-1, \dots, 0$, by minimizing over C^1 network functions $\mathcal{U}_i : \mathbb{R}^d \to \mathbb{R}$ the local quadratic loss function

$$J_i^{(B2)}(\mathcal{U}_i) = \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \mathcal{U}_i(X_i) - f(t_i, X_i, \mathcal{U}_i(X_i), \sigma(t_i, X_i)^\intercal D_x\mathcal{U}_i(X_i))\Delta t_i - D_x\mathcal{U}_i(X_i)^\intercal\sigma(t_i, X_i)\Delta W_i\Big|^2,$$

where $D_x\mathcal{U}_i$ is the automatic differentiation of the network function $\mathcal{U}_i$. Update $\widehat{\mathcal{U}}_i^{(2)}$ as the solution to this local minimization problem, and set $\widehat{\mathcal{Z}}_i^{(2)} = \sigma^\intercal(t_i, .)D_x\mathcal{U}_i^{(2)}$.

The output of DBDP provides an approximation $(\widehat{\mathcal{U}}_i, \widehat{\mathcal{Z}}_i)$ of the solution $u(t_i, .)$ and its gradient $\sigma^\intercal(t_i, .)D_x u(t_i, .)$ to the PDE (1.1) at times t_i, $i = 0, \dots, N-1$. The approximation error has been analyzed in [HPW20].

Remark 2.2. A regression-based machine learning scheme in the spirit of regression-based Monte-Carlo methods ([BT04], [GLW05]) for approximating condition expectations in the time discretization (2.13) of the BSDE, can be formulated as follows: starting from $\hat{\mathcal{U}}_N = g$, proceed by backward induction for $i = N-1, \dots, 0$, in two regression problems:

(a) Minimize over network functions $\mathcal{Z}_i : \mathbb{R}^d \to \mathbb{R}^d$

$$J_i^{r,Z}(\mathcal{Z}_i) = \mathbb{E}\Big|\frac{\Delta W_i}{\Delta t_i}\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \mathcal{Z}_i(X_i)\Big|^2$$

and update $\widehat{\mathcal{Z}}_i$ as the solution to this minimization problem

(b) Minimize over network functions $\mathcal{U}_i : \mathbb{R}^d \to \mathbb{R}$

$$J_i^{r,Y}(\mathcal{U}_i) = \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \mathcal{U}_i(X_i) - f(t_i, X_i, \mathcal{U}_i(X_i), \widehat{\mathcal{Z}}_i(X_i))\Delta t_i\Big|^2$$

and update $\widehat{\mathcal{U}}_i$ as the solution to this minimization problem.

Compared to these regression-based schemes, the DBDP scheme approximates simultaneously the pair component (Y, Z) via the minimization of the loss functions $J_i^{(B1)}(\mathcal{U}_i, \mathcal{Z}_i)$ (or $J_i^{(B2)}(\mathcal{U}_i)$ for the second version), $i = N-1, \dots, 0$. One advantage of this latter approach is that the accuracy of the DBDP scheme can be tested when computing at each time step the infimum of loss function, which should be equal to zero for the exact solution (up to the time discretization). In contrast, the infimum of the loss functions in the regression-based schemes is not known for the exact solution as it corresponds in theory to the residual of L^2-projection, and thus the accuracy of the scheme cannot be tested directly in-sample. □

• Deep Splitting (DS) scheme [Bec+19].

This method also proceeds by backward induction as follows:

- Minimize over C^1 network functions $\mathcal{U}_N : \mathbb{R}^d \to \mathbb{R}$ the terminal loss function

$$J_N^S(\mathcal{U}_N) = \mathbb{E}\Big|g(X_N) - \mathcal{U}_N(X_N)\Big|^2,$$

and denote by $\widehat{\mathcal{U}}_N$ as the solution to this minimization problem. If g is C^1, we can choose directly $\widehat{\mathcal{U}}_N = g$.

- For $i = N-1, \dots, 0$, minimize over C^1 network functions $\mathcal{U}_i : \mathbb{R}^d \to \mathbb{R}$ the loss function

$$J_i^S(\mathcal{U}_i) = \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \mathcal{U}_i(X_i) - f(t_i, X_{i+1}, \widehat{\mathcal{U}}_{i+1}(X_{i+1}), \sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1}))\Delta t_i\Big|^2, \tag{2.15}$$

and update $\widehat{\mathcal{U}}_i$ as the solution to this minimization problem. Here D_x refers again to the automatic differentiation operator for network functions.

The DS scheme combines ideas of the DBDP2 and regression-based schemes where the current regression-approximation on Z is replaced by the automatic differentiation of the network function computed at the previous step. The current approximation of Y is then computed by a regression network-based scheme. In Section 4, we shall analyze the approximation error of the DS scheme, i.e., a bound for the difference between $\widehat{\mathcal{U}}_i$ and $u(t_i,.)$, $i=0,\dots,N-1$.

2.3 Deep backward multi-step scheme (MDBDP)

The starting point of the Multi-step Deep Backward Dynamic Programming (MDBDP) scheme is the iterated representation (2.14) for the time discretization of the BSDE. This backward scheme is described as follows: for $i=N-1,\dots,0$, minimize over network functions $\mathcal{U}_i:\mathbb{R}^d\to\mathbb{R}$, and $\mathcal{Z}_i:\mathbb{R}^d\to\mathbb{R}^d$ the loss function

$$J_i^{MB}(\mathcal{U}_i,\mathcal{Z}_i) = \mathbb{E}\Big|g(X_N) - \sum_{j=i+1}^{N-1} f(t_j,X_j,\widehat{\mathcal{U}}_j(X_j),\widehat{\mathcal{Z}}_j(X_j))\Delta t_j - \sum_{j=i+1}^{N-1}\widehat{\mathcal{Z}}_j(X_j).\Delta W_j \\ - \mathcal{U}_i(X_i) - f(t_i,X_i,\mathcal{U}_i(X_i),\mathcal{Z}_i(X_i))\Delta t_i - \mathcal{Z}_i(X_i).\Delta W_i\Big|^2 \tag{2.16}$$

and update $(\widehat{\mathcal{U}}_i,\widehat{\mathcal{Z}}_i)$ as the solution to this minimization problem. This output provides an approximation $(\widehat{\mathcal{U}}_i,\widehat{\mathcal{Z}}_i)$ of the solution $u(t_i,.)$ to the PDE (1.1) at times t_i, $i=0,\dots,N-1$. This approximation error will be analyzed in Section 4.

MDBDP is a machine learning version of the Multi-step Forward Dynamic Programming method studied by [BD07] and [GT14]. Instead of solving at each time step two regression problems, our approach allows to consider only a single minimization as in the DBDP scheme. Compared to the latter, the multi-step consideration is expected to provide better accuracy by reducing the propagation of errors in the backward induction.

In the numerical implementation, the expectation defining the loss function J_i^{MB} in (2.16) is replaced by an empirical average leading to the so-called *generalization* (or estimation) error, largely studied in the statistical community, see [Gy02], and more recently [Hur+18], [BJK19] and the references therein. Moreover, recalling the parametrization $(\mathcal{U}^\theta,\mathcal{Z}^\theta)$ of neural network functions in $\mathcal{N}^\rho_{d,1,\ell,m}\times\mathcal{N}^\rho_{d,d,\ell,m}$, the minimization of the empirical average is amenable to stochastic gradient descent (SGD) extensively used in machine learning. More precisely, given a fixed time step $i=N-1,\dots,0$, at each iteration of the SGD, we pick a sample $(X_j^k,\Delta W_j^k)_{j=i,\dots,N}$ of the Euler process and increment of Brownian motion $(X_j,\Delta W_j)_j$, $k=1,\dots,K$, of mini-batch size K, and consider the empirical loss function:

$$\mathbb{J}_i^K(\theta) = \frac{1}{K}\sum_{k=1}^{K}\Big|g(X_N^k) - \sum_{j=i+1}^{N-1} f(t_j,X_j^k,\widehat{\mathcal{U}}_j(X_j^k),\widehat{\mathcal{Z}}_j(X_j^k))\Delta t_j - \sum_{j=i+1}^{N-1}\widehat{\mathcal{Z}}_j(X_j^k).\Delta W_j^k \\ - \mathcal{U}^\theta(X_i^k) - f(t_i,X_i^k,\mathcal{U}^\theta(X_i^k),\mathcal{Z}^\theta(X_i^k))\Delta t_i - \mathcal{Z}^\theta(X_i^k).\Delta W_i^k\Big|^2, \tag{2.17}$$

where $\widehat{\mathcal{U}}_j=\mathcal{U}_j^{\hat\theta_j}$, $\widehat{\mathcal{Z}}_j=\mathcal{Z}_j^{\hat\theta_j}$, and $\hat\theta_j$ is the resulting parameter from the SGD obtained at dates $j=i+1,\dots,N-1$. In practice, the number of iterations for SGD at the initial backward induction time $N-1$ should be large enough so as to learn accurately the value function $u(t_{N-1},.)$ and its gradient $D_xu(t_{N-1},.)$ via $\widehat{\mathcal{U}}^{\hat\theta_{N-1}}$ and $\widehat{\mathcal{Z}}^{\hat\theta_{N-1}}$. However, it is then expected (by continuity in time of the value function) that $(\widehat{\mathcal{U}}_j,\widehat{\mathcal{Z}}_j)$ does not vary a lot from $j=i+1$ to i, which means that at time i, one can design the SGD with initialization parameter equal to the resulting parameter from the previous SGD at time $i+1$, and then use few iterations to obtain accurate values of $\widehat{\mathcal{U}}_i$ and $\widehat{\mathcal{Z}}_i$. This observation allows to reduce significantly the computational time in (M)DBDP scheme when applying sequentially N SGD. The stochastic gradient descent algorithm for computing an approximate minimizer of the loss function induces the so-called *optimization* error, which has been extensively studied in the stochastic algorithm and machine learning communities, see [BM], [BF11], [CB18], and more recently [BJK19], and the references therein.

Remark 2.3. The MDBDP requires at each time step i to keep in memory the previously computed network functions $\widehat{\mathcal{U}}_j$, and $\widehat{\mathcal{Z}}_j$, $j=i+1,\dots,N$, and to sample the process X from j until the terminal time N. This is computationally expensive, especially when the horizon T, and so the number N of time steps is very large. To overcome this issue in practice, one can split the time interval $[0,T]$ in two, apply the DBDP scheme on the interval $[T/2,T]$, which yields an approximation $\widehat{\mathcal{U}}_{N/2}$ of the value function at time $T/2$, and then use the MDBDP scheme on $[0,T/2]$ starting from the terminal condition $\widehat{\mathcal{U}}_{N/2}$. □

Algorithm 1: MDBDP scheme.

Data: Initial parameter $\hat{\theta}_N$. A sequence of number of iterations $(S_i)_{i=0,\dots,N-1}$
for $i = N-1, \dots, 0$ **do**
  Initial parameter $\theta_i \leftarrow \hat{\theta}_{i+1}$
  Set $s = 1$
  while $s \leq S_i$ **do**
    Pick a sample of $(X_j, \Delta W_j)_{j=i,\dots,N}$ of mini-batch size K
    Compute the gradient $\nabla \mathbb{J}_i^K(\theta)$ of $\mathbb{J}_i^K(\theta)$ defined in (2.17)
    Update $\theta_i \leftarrow \theta_i - \eta \nabla \mathbb{J}_i^K(\theta_i)$ with η learning rate
    $s \leftarrow s + 1$
  end
  Return $\hat{\theta}_i \leftarrow \theta_i$, $\widehat{\mathcal{U}}_i = \mathcal{U}^{\hat{\theta}_i}$, $\widehat{\mathcal{Z}}_i = \mathcal{Z}^{\hat{\theta}_i}$ `/* Update parameter, function and derivative */`
end

3 Extension to fully non linear PDEs: second order deep backward multistep scheme

In this section, we consider fully nonlinear PDEs in the form (1.2), and discuss how the MDBDP scheme can be extended for their numerical resolution in this more challenging case. Since the function F contains the dependence both on the gradient $D_x u$ and the Hessian $D_x^2 u$, we can shift the linear differential operator (left hand side) of the PDE (1.2) into the function F. However, in practice, this linear differential operator associated to a diffusion process $\mathcal{X}$ is used for training simulations in SGD of machine learning schemes. While the choice of μ does not really matter, the choice of σ is more delicate, and can be viewed as a parameter for state space exploration. In the sequel, we assume w.l.o.g. that $\mu = 0$, and σ is a constant invertible matrix.

Assuming that the solution u to the PDE (1.2) is smooth C^2, and denoting by (Y, Z, Γ) the triple of $\mathbb{F}$-adapted processes valued in $\mathbb{R} \times \mathbb{R}^d \times \mathbb{S}^d$, defined by

$$Y_t \;=\; u(t, \mathcal{X}_t), \quad Z_t \;=\; D_x u(t, \mathcal{X}_t), \quad \Gamma_t \;=\; D_x^2 u(t, \mathcal{X}_t), \quad 0 \leq t \leq T,$$

a direct application of Itô's formula to $u(t, \mathcal{X}_t)$, yields that (Y, Z, Γ) satisfies the backward equation

$$Y_t = \; g(X_T) - \int_t^T F(s, \mathcal{X}_s, Y_s, Z_s, \Gamma_s) \mathrm{d}s - \int_t^T Z_s^\intercal \sigma(s, \mathcal{X}_s) \mathrm{d}W_s, \quad 0 \leq t \leq T. \tag{3.1}$$

Compared to the case of semi-linear PDE, the key point is the approximation/learning of the Hessian matrix $D_x^2 u$, hence of the Γ-component of the BSDE (3.1). A basic idea in line of the schemes described in the previous section would consist in approximating the solution u and its gradient $D_x u$ by network functions $\mathcal{U}$ and $\mathcal{Z}$, and then Hessian $D_x^2 u$ by the automatic differentiation $D_x \mathcal{Z}$ of the network function $\mathcal{Z}$, by a learning approach relying on the time discretization of the BSDE (3.1). It turns out that such method approximates poorly Γ inducing instability of the scheme: indeed, while the unique pair solution (Y, Z) to the BSDE (2.11) completely characterizes the solution to the semi-linear PDE and its gradient, the relation (3.1) does not allow to characterize directly the triple (Y, Z, Γ). Instead, we need a suitable probabilistic representation of the Γ-component for learning accurately the Hessian function $D_x^2 u$.

We start from the training simulations of the forward $(X_i)_i$ on the grid $\pi = \{t_i, i = 0, \dots, N\}$, and assume for simplicity that the grid is uniform, i.e., $t_i = i|\pi|$, $|\pi| = T/N$. Notice that $X_i = \mathcal{X}_{t_i}$, and is equal to

$$X_i = \; \mathcal{X}_0 + \sigma W_{t_i}, \quad i = 0, \dots, N.$$

The approximation of the value function u and its gradient $D_x u$ is learnt simultaneously on the grid π as described in the previous section but requires in addition a preliminary approximation of the Hessian $D_x^2 u$ in the fully non linear case. This will be performed by regression-based machine learning scheme on a subgrid $\hat{\pi} \subset \pi$, which allows to reduce the computational time of the algorithm. We propose three versions of second order MDBDP based on different representations of the Hessian function, and assuming that g is smooth C^2. For the second and the third one, we need to introduce a subgrid $\hat{\pi} = \{t_{\hat{\kappa}\ell}, \ell = 0, \dots, \hat{N}\} \subset \pi$, of modulus $|\hat{\pi}| = \hat{\kappa}|\pi|$, for some $\hat{\kappa} \in \mathbb{N}^*$, with $N = \hat{\kappa}\hat{N}$.

1. Following the methodology introduced in [PWG19], the current Γ-component at step i can be estimated by automatic differentiation of the Z-component at the previous step while the other Γ-components are estimated by automatic differentiation of their associated Z-components:

$$\Gamma_i \;\simeq\; D_x Z_{i+1}, \quad \Gamma_j \;\simeq\; D_x Z_j, \; j > i.$$

2. The time discretization of (3.1) on the grid $\hat{\pi}$, where $(Y_\ell^{\hat{\pi}}, Z_\ell^{\hat{\pi}}, \Gamma_\ell^{\hat{\pi}})$ denotes an approximation of the triple $\big(u(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}), D_x u(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}), D_x^2 u(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell})\big)$, $\ell = 0, \ldots, \hat{N}$, leads to the standard representation formula for the Z component (see (2.13)):

$$Z_\ell^{\hat{\pi}} = \mathbb{E}_{\hat{\kappa}\ell}\Big[Y_{\ell+1}^{\hat{\pi}} \hat{H}_\ell^1\Big], \quad \ell = 0, \ldots, \hat{N}-1,$$

(recall that $\mathbb{E}_{\hat{\kappa}\ell}$ denotes the conditional expectation w.r.t. $\mathcal{F}_{t_{\hat{\kappa}\ell}}$), with the Malliavin weight of order one:

$$\hat{H}_\ell^1 = (\sigma^\intercal)^{-1} \frac{\hat{\Delta} W_\ell}{|\hat{\pi}|}, \quad \hat{\Delta} W_\ell := W_{t_{\hat{\kappa}(\ell+1)}} - W_{t_{\hat{\kappa}\ell}}.$$

By direct differentiation, we then obtain an approximation of the Γ-component as

$$\Gamma_\ell^{\hat{\pi}} \simeq \mathbb{E}_{\hat{\kappa}\ell}\Big[D_x u(t_{\hat{\kappa}(\ell+1)}, X_{\hat{\kappa}(\ell+1)}) \hat{H}_\ell^1\Big].$$

Moreover, by introducing the antithetic variable

$$\hat{X}_{\hat{\kappa}(\ell+1)} = X_{\hat{\kappa}\ell} - \sigma \hat{\Delta} W_\ell,$$

we then propose the following regression estimator of $D_x^2 u$ on the grid $\hat{\pi}$ with

$$\begin{cases} \hat{\Gamma}^{(1)}(t_{\hat{\kappa}\hat{N}}, X_{\hat{\kappa}\hat{N}}) &= D^2 g(X_{\hat{\kappa}\hat{N}}) \\ \hat{\Gamma}^{(1)}(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}) &= \mathbb{E}_{\hat{\kappa}\ell}\Big[\frac{D_x u(t_{\hat{\kappa}(\ell+1)}, X_{\hat{\kappa}(\ell+1)}) - D_x u(t_{\hat{\kappa}(\ell+1)}, \hat{X}_{\hat{\kappa}(\ell+1)})}{2} \hat{H}_\ell^1\Big], \quad \ell = 0, \ldots, \hat{N}-1. \end{cases}$$

3. Alternatively, the time discretization of (3.1) on $\hat{\pi}$ yields the iterated conditional expectation relation:

$$Y_\ell^{\hat{\pi}} = \mathbb{E}_{\hat{\kappa}\ell}\Big[g(X_{\hat{\kappa}\hat{N}}) - |\hat{\pi}| \sum_{m=\ell}^{\hat{N}-1} F(t_{\hat{\kappa}m}, X_{\hat{\kappa}m}, Y_m^{\hat{\pi}}, Z_m^{\hat{\pi}}, \Gamma_m^{\hat{\pi}})\Big], \quad \ell = 0, \ldots, \hat{N},$$

By (double) integration by parts, and using Malliavin weights on the Gaussian vector X, we obtain a multi-step approximation of the Γ-component as

$$\Gamma_\ell^{\hat{\pi}} \simeq \mathbb{E}_{\hat{\kappa}\ell}\Big[g(X_{\hat{\kappa}\hat{N}}) \hat{H}_{\ell,\hat{N}}^2 - |\hat{\pi}| \sum_{m=\ell+1}^{\hat{N}-1} F(t_{\hat{\kappa}m}, X_{\hat{\kappa}m}, Y_m^{\hat{\pi}}, Z_m^{\hat{\pi}}, \Gamma_m^{\hat{\pi}}) \hat{H}_{\ell,m}^2\Big], \quad \ell = 0, \ldots, \hat{N},$$

where

$$\hat{H}_{\ell,m}^2 = (\sigma^\intercal)^{-1} \frac{\hat{\Delta} W_\ell^m (\hat{\Delta} W_\ell^m)^\intercal - (m-\ell)|\hat{\pi}| I_d}{(m-\ell)^2 |\hat{\pi}|^2} \sigma^{-1}, \quad \hat{\Delta} W_\ell^m := W_{t_{\hat{\kappa}m}} - W_{t_{\hat{\kappa}\ell}}.$$

By introducing again the antithetic variables

$$\hat{X}_{\hat{\kappa}m} = X_{\hat{\kappa}\ell} - \sigma \hat{\Delta} W_\ell^m, \quad m = \ell+1, \ldots, \hat{N},$$

we then propose another regression estimator of $D_x^2 u$ on the grid $\hat{\pi}$ with

$$\begin{aligned} \hat{\Gamma}^{(2)}(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}) = \mathbb{E}_{\hat{\kappa}\ell}\Big[& \frac{g(X_{\hat{\kappa}\hat{N}}) + g(\hat{X}_{\hat{\kappa}\hat{N}})}{2} \hat{H}_{\ell,\hat{N}}^2 \\ & - \frac{|\hat{\pi}|}{2} \sum_{m=\ell+1}^{\hat{N}-1} \Big(F\big(t_{\hat{\kappa}m}, X_{\hat{\kappa}m}, u(t_{\hat{\kappa}m}, X_{\hat{\kappa}m}), D_x u(t_{\hat{\kappa}m}, X_{\hat{\kappa}m}), \hat{\Gamma}^{(2)}(t_{\hat{\kappa}m}, X_{\hat{\kappa}m})\big) \\ & \quad + F\big(t_{\hat{\kappa}m}, \hat{X}_{\hat{\kappa}m}, u(t_{\hat{\kappa}m}, \hat{X}_{\hat{\kappa}m}), D_x u(t_{\hat{\kappa}m}, \hat{X}_{\hat{\kappa}m}), \hat{\Gamma}^{(2)}(t_{\hat{\kappa}m}, \hat{X}_{\hat{\kappa}m})\big) \\ & \quad - 2F\big(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}, u(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}), D_x u(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}), \hat{\Gamma}^{(2)}(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell})\big)\Big) \hat{H}_{\ell,m}^2\Big], \end{aligned}$$

for $\ell = 0, \ldots, N-1$, and $\hat{\Gamma}^{(2)}(t_{\hat{\kappa}\hat{N}}, X_{\hat{\kappa}\hat{N}}) = D^2 g(X_{\hat{\kappa}\hat{N}})$. The correction term $-2F$ evaluated at time $t_{\hat{\kappa}\ell}$ in $\hat{\Gamma}^{(2)}(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell})$ does not add bias since

$$\mathbb{E}_{\hat{\kappa}\ell}\Big[F\big(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}, u(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}), D_x u(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}), \hat{\Gamma}^{(2)}(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell})\big) \hat{H}_{\ell,m}^2\Big] = 0,$$

for all $m = \ell+1, \ldots, \hat{N}-1$, and by Taylor expansion of F at second order, we see that it allows together with the antithetic variable to control the variance when the time step goes to zero. Similar idea was used in [War18].

Remark 3.1. In the case where the function g has some regularity property, one can avoid the integration by parts at the terminal data component in the above expression of $\hat{\Gamma}^{(2)}$. For example, when g is C^1, $\frac{g(X_{\hat{\kappa}\hat{N}})+g(\hat{X}_{\hat{\kappa}\hat{N}})}{2}\hat{H}^2_{\ell,\hat{N}}$ is alternatively replaced in $\hat{\Gamma}^{(2)}$ expression by $(Dg(X_{\hat{\kappa}\hat{N}}) - Dg(\hat{X}_{\hat{\kappa}\hat{N}}))\hat{H}^1_{\ell,\hat{N}}$, while when it is C^2 it is replaced by $D^2g(X_{\hat{\kappa}\hat{N}})$. □

We can now describe the three versions of second order MDBDP schemes for the numerical resolution of the fully nonlinear PDE (1.2).

Algorithm 2: Second order Explicit Multi-step DBDP (2EMDBDP)

for $i = N-1, \dots, 0$ **do**

If $i = N-1$, update $\widehat{\Gamma}_i = D^2g$, otherwise $\widehat{\Gamma}_i = D_x\widehat{\mathcal{Z}}_{i+1}$, $\widehat{\Gamma}_j = D_x\widehat{\mathcal{Z}}_j$, $j \in [\![i+1, N-1]\!]$, `/* Update Hessian */`

Minimize over network functions $\mathcal{U} : \mathbb{R}^d \to \mathbb{R}$, and $\mathcal{Z} : \mathbb{R}^d \to \mathbb{R}^d$ the loss function at time t_i:

$$J_i^{MB}(\mathcal{U}, \mathcal{Z}) = \mathbb{E}\Big|g(X_N) - |\pi| \sum_{j=i+1}^{N-1} F(t_j, X_j, \widehat{\mathcal{U}}_j(X_j), \widehat{\mathcal{Z}}_j(X_j), \hat{\Gamma}_j(X_j)) - \sum_{j=i+1}^{N-1} \widehat{\mathcal{Z}}_j(X_j)^\intercal \sigma \Delta W_j$$
$$- \mathcal{U}(X_i) - |\pi| F(t_i, X_i, \mathcal{U}(X_i), \mathcal{Z}(X_i), \hat{\Gamma}_i(X_{i+1})) - \mathcal{Z}(X_i)^\intercal \sigma \Delta W_i\Big|^2.$$

Update $(\widehat{\mathcal{U}}_i, \widehat{\mathcal{Z}}_i)$ as the solution to this minimization problem `/* Update the function and its derivative */`

end

Algorithm 3: Second order Multi-step DBDP (2MDBDP)

for $\ell = \hat{N}, \dots, 0$ **do**

If $\ell = \hat{N}$, update $\widehat{\Gamma}_\ell = D^2g$, otherwise minimize over network functions $\Gamma : \mathbb{R}^d \to \mathbb{S}^d$ the loss function

$$\mathcal{J}_\ell^{1,M}(\Gamma) = \mathbb{E}\Big|\Gamma(X_{\hat{\kappa}\ell}) - \frac{\widehat{\mathcal{Z}}_{\hat{\kappa}(\ell+1)}(X_{\hat{\kappa}(\ell+1)}) - \widehat{\mathcal{Z}}_{\hat{\kappa}(\ell+1)}, \hat{X}_{\hat{\kappa}(\ell+1)})}{2}\hat{H}^1_\ell\Big|^2.$$

Update $\hat{\Gamma}_\ell$ the solution to this minimization problem `/* Update Hessian */`

for $k = \hat{\kappa} - 1, \dots, 0$ **do**

Minimize over network functions $\mathcal{U} : \mathbb{R}^d \to \mathbb{R}$, and $\mathcal{Z} : \mathbb{R}^d \to \mathbb{R}^d$ the loss function at time t_i, $i = (\ell-1)\hat{\kappa} + k$:

$$J_i^{MB}(\mathcal{U}, \mathcal{Z}) = \mathbb{E}\Big|g(X_N) - |\pi| \sum_{j=i+1}^{N-1} F(t_j, X_j, \widehat{\mathcal{U}}_j(X_j), \widehat{\mathcal{Z}}_j(X_j), \hat{\Gamma}_\ell(X_j)) - \sum_{j=i+1}^{N-1} \widehat{\mathcal{Z}}_j(X_j)^\intercal \sigma \Delta W_j$$
$$- \mathcal{U}(X_i) - |\pi| F(t_i, X_i, \mathcal{U}(X_i), \mathcal{Z}(X_i), \hat{\Gamma}_\ell(X_i)) - \mathcal{Z}(X_i)^\intercal \sigma \Delta W_i\Big|^2.$$

Update $(\widehat{\mathcal{U}}_i, \widehat{\mathcal{Z}}_i)$ as the solution to this minimization problem `/* Update the function and its derivative */`

end

end

Algorithm 4: Second order Multi-step Malliavin DBDP (2M^{2}DBDP)

for $\ell = \hat{N}, \dots, 0$ **do**

If $\ell = \hat{N}$, update $\widehat{\Gamma}_\ell = D^2 g$, otherwise minimize over network functions $\Gamma : \mathbb{R}^d \to \mathbb{S}^d$ the loss function

$$\begin{aligned}\mathcal{J}_\ell^{2,M}(\Gamma) = \mathbb{E}\Big|\Gamma(X_{\hat{\kappa}\ell}) - \frac{D^2 g(X_{\hat{\kappa}\hat{N}}) + D^2 g(\hat{X}_{\hat{\kappa}\hat{N}})}{2} \\ + \frac{|\hat{\pi}|}{2}\sum_{m=\ell+1}^{\hat{N}-1}\Big(F\big(t_{\hat{\kappa}m}, X_{\hat{\kappa}m}, \widehat{\mathcal{U}}_{\hat{\kappa}m}(X_{\hat{\kappa}m}), \widehat{\mathcal{Z}}_{\hat{\kappa}m}(X_{\hat{\kappa}m}), \hat{\Gamma}_m(X_{\hat{\kappa}m})\big) \\ + F\big(t_{\hat{\kappa}m}, \hat{X}_{\hat{\kappa}m}, \widehat{\mathcal{U}}_{\hat{\kappa}m}(\hat{X}_{\hat{\kappa}m}), \widehat{\mathcal{Z}}_{\hat{\kappa}m}(\hat{X}_{\hat{\kappa}m}), \widehat{\Gamma}_m(\hat{X}_{\hat{\kappa}m})\big) \\ - 2F\big(t_{\hat{\kappa}\ell}, \hat{X}_{\hat{\kappa}\ell}, \widehat{\mathcal{U}}_{\hat{\kappa}\ell}(\hat{X}_{\hat{\kappa}\ell}), \widehat{\mathcal{Z}}_{\hat{\kappa}\ell}(\hat{X}_{\hat{\kappa}\ell}), \widehat{\Gamma}_\ell(\hat{X}_{\hat{\kappa}\ell})\big)\Big)\hat{H}^2_{\ell,m}\Big|^2.\end{aligned}$$

Update $\widehat{\Gamma}_\ell$ the solution to this minimization problem `/* Update Hessian */`

for $k = \hat{\kappa} - 1, \dots, 0$ **do**

Minimize over network functions $\mathcal{U} : \mathbb{R}^d \to \mathbb{R}$, and $\mathcal{Z} : \mathbb{R}^d \to \mathbb{R}^d$ the loss function at time t_i, $i = (\ell-1)\hat{\kappa} + k$:

$$\begin{aligned}J_i^{MB}(\mathcal{U}, \mathcal{Z}) = \mathbb{E}\Big|g(X_N) - |\pi|\sum_{j=i+1}^{N-1} F(t_j, X_j, \widehat{\mathcal{U}}_j(X_j), \widehat{\mathcal{Z}}_j(X_j), \widehat{\Gamma}_\ell(X_j)) - \sum_{j=i+1}^{N-1}\widehat{\mathcal{Z}}_j(X_j)^\intercal \sigma \Delta W_j \\ - \mathcal{U}(X_i) - |\pi| F(t_i, X_i, \mathcal{U}(X_i), \mathcal{Z}(X_i), \widehat{\Gamma}_\ell(X_i)) - \mathcal{Z}(X_i)^\intercal \sigma \Delta W_i\Big|^2.\end{aligned}$$

Update $(\widehat{\mathcal{U}}_i, \widehat{\mathcal{Z}}_i)$ as the solution to this minimization problem `/* Update the function and its derivative */`

end

end

The above proposed schemes are in backward iteration, and involve one optimization at each step. Moreover, as the computation of Γ requires a further derivation for Algorithms 3 and 4, we may expect that the additional propagation error varies according to $\frac{|\pi|}{|\hat{\pi}|} = \frac{1}{\hat{\kappa}}$, and thus the convergence of the scheme when $\hat{\kappa}$ is large. We postpone the convergence analysis of second order MDBDP for further investigation. In the numerical implementation, as detailed in Algorithm 1, the expectation in the loss functions $\mathcal{J}_\ell^{1,M}$ (resp. $\mathcal{J}_\ell^{2,M}$), and J_i^{MB} are replaced by empirical average and the minimization over network functions is performed by stochastic gradient descent.

4 Convergence analysis in the semilinear case

This section is devoted to the approximation error analysis and rate of convergence of the deep backward multistep (MDBDP) and deep splitting (DS) schemes described in Section 2.

We make the following standard assumptions on the coefficients of the forward-backward equation associated to semilinear PDE (1.1).

Assumption 4.1. *(i) $\mathcal{X}_0$ is square-integrable: $\mathcal{X}_0 \in L^2(\mathcal{F}_0, \mathbb{R}^d)$.*

(ii) The functions μ and σ are Lipschitz in $x \in \mathbb{R}^d$, uniformly in $t \in [0,T]$.

(iii) The generator function f is $1/2$-Hölder continuous in time and Lipschitz continuous in all other variables: $\exists\, [f]_L > 0$ such that for all (t,x,y,z) and $(t',x',y',z') \in [0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^d$,

$$|f(t,x,y,z) - f(t',x',y',z')| \le [f]_L\big(|t-t'|^{1/2} + |x-x'|_2 + |y-y'| + |z-z'|_2\big).$$

Moreover,

$$\sup_{t\in[0,T]}|f(t,0,0,0)| < \infty.$$

(iv) The function g satisfies a linear growth condition.

Assumption 4.1 guarantees the existence and uniqueness of an adapted solution $(\mathcal{X}, Y, Z)$ to the forward-backward equation (2.10)-(2.11), satisfying

$$\mathbb{E}\Big[\sup_{0\le t\le T}|\mathcal{X}_t|_2^2+\sup_{0\le t\le T}|Y_t|^2+\int_0^T|Z_t|_2dt\Big]<\infty.$$

Given the time grid $\pi=\{t_i: i=0,\dots,N\}$, let us introduce the so-called L^2-regularity of Z:

$$\varepsilon^Z(\pi):=\mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}}|Z_t-\bar Z_{t_i}|_2^2dt\Big],\quad\text{with }\bar Z_{t_i}:=\frac{1}{\Delta t_i}\mathbb{E}_i\Big[\int_{t_i}^{t_{i+1}}Z_tdt\Big].$$

Since $\bar Z$ is a L^2-projection of Z, we know that $\varepsilon^Z(\pi)$ converges to zero when $|\pi|$ goes to zero. Moreover, as shown in [Zha04], when the terminal condition g is also Lipschitz, we have

$$\varepsilon^Z(\pi)=O(|\pi|).$$

Here, the standard notation $O(|\pi|)$ means that $\limsup_{|\pi|\to0}|\pi|^{-1}O(|\pi|)<\infty$.

4.1 Convergence of the MDBDP scheme

We fix a class of functions $\mathcal{N}$ and $\mathcal{N}'$ for the approximations of the value function and its gradient, and recall that $(\widehat{\mathcal{U}}_i,\widehat{\mathcal{Z}}_i)$ denotes the output of the MDBDP scheme at times t_i, $i=0,\dots,N$, resulting from the minimization of the loss function in (2.16). The class $\mathcal{N}$ is typically the class of neural networks $\mathcal{N}^{\rho}_{d,1,\ell,m}$, or more specifically $\mathcal{N}^{1,\gamma,R}_{d,m,1}$, while the class $\mathcal{N}'$ may be the class of neural networks $\mathcal{N}^{\rho}_{d,d,\ell,m}$ or $\mathcal{N}^{1,\gamma,R}_{d,m,d}$.

Let us define (implicitly) the process

$$\begin{cases} V_i &= \mathbb{E}_i\Big[g(X_N)+f\big(t_i,X_i,V_i,\overline{\widehat{Z}_i}\big)\Delta t_i+\sum_{j=i+1}^{N-1}f\big(t_j,X_j,\widehat{\mathcal{U}}_j(X_j),\widehat{\mathcal{Z}}_j(X_j)\big)\Delta t_j\Big],\\ \overline{\widehat{Z}}_i &= \mathbb{E}_i\Big[\frac{g(X_N)\Delta W_i}{\Delta t_i}+\sum_{j=i+1}^{N-1}f\big(t_j,X_j,\widehat{\mathcal{U}}_j(X_j),\widehat{\mathcal{Z}}_j(X_j)\big)\frac{\Delta W_i\Delta t_j}{\Delta t_i}\Big],\quad i=0,\dots,N,\end{cases}\tag{4.1}$$

and notice by the Markov property of the discretized forward process $(X_i)_i$ that

$$V_i=v_i(X_i),\qquad\overline{\widehat{Z}_i}=\widehat{z}_i(X_i),\quad i=0,\dots,N,$$

for some deterministic functions $v_i,\widehat{z}_i$. Let us then introduce

$$\varepsilon_i^y:=\inf_{\mathcal{U}\in\mathcal{N}}\mathbb{E}\big|v_i(X_i)-\mathcal{U}(X_i)\big|^2,\qquad\varepsilon_i^z:=\inf_{\mathcal{Z}\in\mathcal{N}'}\mathbb{E}\big|\widehat{z}_i(X_i)-\mathcal{Z}(X_i)\big|_2^2,$$

for $i=0,\dots,N-1$, which represent the L^2-approximation errors of the functions $v_i,\widehat{z}_i$ in the class of neural networks $\mathcal{N}$ and $\mathcal{N}'$.

Theorem 4.1 (Approximation error of MDPBD)**.** *Under Assumption 4.1, there exists a constant $C>0$ (depending only on the data of the problem μ,σ,f,g,d,T) such that*

$$\begin{aligned}&\sup_{i\in[\![0,N]\!]}\mathbb{E}\big|Y_{t_i}-\widehat{\mathcal{U}}_i(X_i)\big|^2+\mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}}\big|Z_s-\widehat{\mathcal{Z}}_i(X_i)\big|_2^2\,\mathrm{d}s\Big]\\&\le C\Big(\mathbb{E}\big|g(\mathcal{X}_T)-g(X_N)\big|^2+|\pi|+\varepsilon^Z(\pi)+\sum_{j=0}^{N-1}(\varepsilon_j^y+\Delta t_j\varepsilon_j^z)\Big).\end{aligned}\tag{4.2}$$

Remark 4.1. The upper bound in (4.2) for the approximation error of the MDBDP consists of four terms. The first three terms correspond to the time discretization of BSDE, similarly as in [Zha04], [BT04], [GLW05], namely (i) the strong approximation of the terminal condition (depending on the forward scheme and the terminal data g), and converging to zero, as $|\pi|$ goes to zero, with a rate $|\pi|$ when g is Lipschitz, (ii) the strong approximation of the forward Euler scheme, and the L^2-regularity of Y, which gives a convergence of order $|\pi|$, (iii) the L^2-regularity of Z. Finally, the last term is the approximation error by the chosen class of functions. Note that the approximation error $\sum_{j=0}^{N-1}(\varepsilon_j^y+\Delta t_j\varepsilon_j^z)$ in (4.2) is better than the one for the DBDP scheme derived in [HPW20], with an order $\sum_{j=0}^{N-1}(N\varepsilon_j^y+\varepsilon_j^z)$. □

We next study the rate of convergence for the approximation error of the MDBDP scheme, and need to specify the class of network functions $\mathcal{N}$ and $\mathcal{N}'$.

Proposition 4.1 (Rate of convergence of MDBDP). *Let Assumption 4.1 hold, and assume that $\mathcal{X}_0 \in L^{2+\delta}(\mathcal{F}_0, \mathbb{R}^d)$, for some $\delta > 0$, and g is Lipschitz. Then, for $\mathcal{N} = \mathcal{N}_{d,m,1}^{1,\gamma,R}$, and $\mathcal{N}' = \mathcal{N}_{d,m,d}^{1,\gamma,R}$, we have*

$$\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i(X_i)\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \big|Z_s - \widehat{\mathcal{Z}}_i(X_i)\big|_2^2 \,\mathrm{d}s\Big] \;=\; O(1/N),$$

for a choice of m, R, γ of order

$$m \;=\; O\big(N^{d+\epsilon}\big), \quad R \;=\; O\big(N^{\frac{d+1}{2}+\epsilon}\big), \quad \gamma \;=\; O(R),$$

for any $\epsilon > 0$. Here, the constants in the $O(\cdot)$ term depend only on the data $\mu, \sigma, f, g, d, T, \mathcal{X}_0$.

Remark 4.2. The rate of convergence in Proposition 4.2 can be expressed in terms of the number of neurons m, and says that

$$\big\|u(0,\mathcal{X}_0) - \widehat{\mathcal{U}}_0(\mathcal{X}_0)\big\|_2 \;:=\; \sqrt{\mathbb{E}\big|u(0,\mathcal{X}_0) - \widehat{\mathcal{U}}_0(\mathcal{X}_0)\big|^2} \;=\; O\big(m^{-\frac{1}{2d}+\epsilon}\big).$$

This exponent $1/(2d)$, which is decreasing with the input dimension d, does not overcome in theory the so-called curse of dimensionality. This is due to the use of the approximation result in [Bac17] for locally Lipschitz functions with shallow neural networks, and could be probably improved by considering a class of multilayer neural networks. However, to the best of our knowledge, we could not find in the literature approximation results providing a rate of convergence for class of network functions that allow in our context to overcome the curse of dimensionality. □

4.2 Convergence of the DS scheme

We consider an architecture $\mathcal{N}_{d,m,1}^{2,R,\gamma}$ of the neural network for the approximation of the value function at time t_i, and $\widehat{\mathcal{U}}_i \in \mathcal{N}_{d,m,1}^{2,R,\gamma}$ denotes the output of the DS scheme, for $i = 0, \ldots, N$.

Let us define the process

$$V_i \;=\; \mathbb{E}_i\Big[\widehat{\mathcal{U}}_{i+1}(X_{i+1}) + f\big(t_i, X_i, \mathbb{E}_i[\widehat{\mathcal{U}}_{i+1}(X_{i+1})], \mathbb{E}_i[\sigma(t_i,X_i)^\intercal D_x[\widehat{\mathcal{U}}_{i+1}(X_{i+1})]\big)\Delta t_i\Big], \tag{4.3}$$

for $i = 0, \ldots, N-1$, and $V_N = \widehat{\mathcal{U}}_N(X_N)$. By the Markov property of $(X_i)_i$, we have $V_i = v_i(X_i)$, for some deterministic functions $v_i : \mathbb{R}^d \to \mathbb{R}$, $i = 0, \ldots, N-1$, and we introduce

$$\varepsilon_i^{m,R,\gamma} := \begin{cases} \inf_{\mathcal{U}\in\mathcal{N}_{d,m,1}^{2,R,\gamma}} \mathbb{E}\big|v_i(X_i) - \mathcal{U}(X_i)\big|^2, & i = 0, \ldots, N-1, \\ \inf_{\mathcal{U}\in\mathcal{N}_{d,m,1}^{2,R,\gamma}} \mathbb{E}\big|g(X_N) - \mathcal{U}(X_N)\big|^2 \;=\; \mathbb{E}\big|g(X_N) - \widehat{\mathcal{U}}_N(X_N)\big|^2, & i = N. \end{cases}$$

the L^2-approximation error of v_i, $i = 0, \ldots, N-1$, and g in the class of neural networks $\mathcal{N}_{d,m}^{+,R,\gamma}$.

Theorem 4.2 (Approximation error of DS). *Let Assumption 4.1 hold, and assume that $\mathcal{X}_0 \in L^4(\mathcal{F}_0, \mathbb{R}^d)$. Then, there exists a constant $C > 0$ (depending only on $\mu, \sigma, f, g, d, T, \mathcal{X}_0$) such that*

$$\begin{aligned}\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i(X_i)\big|^2 \le C\Big(&\mathbb{E}\big|g(X_N) - g(\mathcal{X}_T)\big|^2 + |\pi| + \varepsilon^Z(\pi) \\ &+ \;\max\big[1, \frac{\gamma^2}{R^2}\big]|\pi| + \varepsilon_N^{m,R,\gamma} + N\sum_{i=0}^{N-1}\varepsilon_i^{m,R,\gamma}\Big).\end{aligned} \tag{4.4}$$

Remark 4.3. We retrieve a similar error as in the analysis of the second version DBDP2 of the deep backward dynamic programming scheme derived in [HPW20]. Notice that when g is C^1, one can choose to initialize the DS scheme with $\widehat{\mathcal{U}}_N = g$, and the term $\varepsilon_N^{m,R,\gamma}$ is removed in (4.4). □

We next study the rate of convergence for the approximation error of the DS scheme, and make the additional Lipschitz assumptions on f and g.

Assumption 4.2. *(i) The function $f_\sigma(t,x,y,z) := f(t,x,y,\sigma(t,x)^\intercal z)$ is Lipschitz in z, i.e., there exists a constant $[f_\sigma]_L > 0$ such that for all $t \in [0,T]$, $x \in \mathbb{R}^d$, $y \in \mathbb{R}$, $z, z' \in \mathbb{R}^d$,*

$$|f(t,x,y,\sigma(t,x)^\intercal z) - f(t,x,y,\sigma(t,x)^\intercal z')| \leq [f_\sigma]_L |z - z'|_2.$$

(ii) g is Lipschitz continuous on $\mathbb{R}^d$.

Remark 4.4. The Lipschitz condition in Assumption 4.2 (i) combined with the Lipschitz property of f, σ in Assumption 4.1 implies that there exists some constant $K_{f,\sigma}$ such that for all $t \in [0,T]$, $x, x' \in \mathbb{R}^d$, $y, y' \in \mathbb{R}$, $z, z' \in \mathbb{R}^d$,

$$|f(t,x,y,\sigma(t,x)^\intercal z) - f(t,x',y',\sigma(t,x')^\intercal z')| \leq K_{f,\sigma}\big(|x-x'|_2(1+|z|_2) + |y-y'| + |z-z'|_2\big). \tag{4.5}$$

This is clearly satisfied when σ is bounded under Assumption 4.1. □

Proposition 4.2 (Rate of convergence of DS)**.** *Let Assumptions 4.1 and 4.2 hold, and assume that $\mathcal{X}_0 \in L^{4+\delta}(\mathcal{F}_0, \mathbb{R}^d)$, for some $\delta > 0$. Then, we have*

$$\sup_{i \in [\![0,N]\!]} \mathbb{E}\Big|Y_{t_i} - \widehat{\mathcal{U}}_i(X_i)\Big|^2 = O(1/N),$$

for a choice of m, R, γ in the class of networks functions $\mathcal{N}_{d,m,1}^{2,\gamma,R}$ of order

$$m = O\big(N^{\frac{3(d+5)}{2}+\epsilon}\big), \quad R = O\big(N^{\frac{3(d+3)}{4}+\epsilon}\big), \quad \gamma = O(R),$$

for any $\epsilon > 0$.

Remark 4.5. The rate of convergence in Proposition 4.2 can be expressed in terms of the number of neurons m, and says that

$$\big\|u(0,\mathcal{X}_0) - \widehat{\mathcal{U}}_0(\mathcal{X}_0)\big\|_2 = O\big(m^{-\frac{1}{3(d+5)}+\epsilon}\big).$$

In comparison with the rate obtained in Proposition 4.1, we see that $1/3(d+5) < 1/(2d)$, which means that the MDBDP has a better rate of convergence than the DS (and DBDP) scheme in terms of the number m of neurons in shallow networks, whatever the dimension d. □

5 Proof of the main theoretical results

We shall often use some classical inequalities that we recall:

Young inequality. For all $(a,b) \in \mathbb{R}^2$, $\beta > 0$,

$$(1-\beta)a^2 + \big(1 - \frac{1}{\beta}\big)b^2 \leq (a+b)^2 \leq (1+\beta)a^2 + \big(1+\frac{1}{\beta}\big)b^2.$$

Discrete Gronwall lemma. Let $(u_n, v_n, h_n)_n$ be positive sequences satisfying for all $n \in \mathbb{N}$

$$u_n \leq (1+h_n)u_{n+1} + v_n, \quad \forall n \in \mathbb{N}.$$

Then, we have for all $N \in \mathbb{N}^*$

$$\sup_{i \in [\![0,N]\!]} u_i \leq \exp\Big(\sum_{i=0}^{N-1} h_i\Big)\Big(u_N + \sum_{i=0}^{N-1} v_i\Big).$$

In particular, when $h_i = \beta \Delta t_i$, with $\beta > 0$, $\Delta t_i = O(1/N)$, there exists C independent of N s.t.

$$\sup_{i \in [\![0,N]\!]} u_i \leq C\Big(u_N + \sum_{i=0}^{N-1} v_i\Big), \quad \forall N \in \mathbb{N}^*.$$

5.1 Proof of Theorem 4.1

Let us introduce the processes $(\bar{V}_i, Z_i)_i$ arising from the time discretization of the BSDE (2.11), and defined by the *implicit* backward Euler scheme:

$$\left\{ \begin{array}{rcl} \bar{V}_i & = & \mathbb{E}_i\Big[\bar{V}_{i+1} + f\big(t_i, X_i, \bar{V}_i, Z_i\big)\Delta t_i\Big] \\ Z_i & = & \mathbb{E}_i\Big[\bar{V}_{i+1}\frac{\Delta W_i}{\Delta t_i}\Big], \quad i = 0, \ldots, N-1, \end{array} \right. \tag{5.1}$$

starting from $\bar{V}_N = g(X_N)$. We recall from [Zha04] (see also [BT04] or [GLW05]) the time discretization error:

$$\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \bar{V}_i\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \big|Z_s - Z_i\big|_2^2\,\mathrm{d}s\Big] \le C\Big(\mathbb{E}\big|g(\mathcal{X}_T) - g(X_N)\big|^2 + |\pi| + \varepsilon^Z(\pi)\Big), \tag{5.2}$$

for some constant C depending only on the coefficients satisfying Assumption 4.1.

Let us introduce the auxiliary process

$$\hat{V}_i = \mathbb{E}_i\Big[g(X_N) + \sum_{j=i}^{N-1} f\big(t_j, X_j, \widehat{\mathcal{U}}_j(X_j), \widehat{\mathcal{Z}}_j(X_j)\big)\Delta t_j\Big], \quad i = 0, \ldots, N, \tag{5.3}$$

and notice by the tower property of conditional expectations that we have the recursive relations:

$$\hat{V}_i = \mathbb{E}_i\Big[\hat{V}_{i+1} + f\big(t_i, X_i, \widehat{\mathcal{U}}_i(X_i), \widehat{\mathcal{Z}}_i(X_i)\big)\Delta t_i\Big], \quad i = 0, \ldots, N-1. \tag{5.4}$$

Observe also that $\overline{\widehat{Z}}_i$ defined in (4.1) satisfies

$$\overline{\widehat{Z}}_i = \mathbb{E}_i\Big[\hat{V}_{i+1}\frac{\Delta W_i}{\Delta t_i}\Big], \quad i = 0, \ldots, N-1. \tag{5.5}$$

We now decompose the approximation error, for $i \in [\![0, N-1]\!]$, into

$$\begin{aligned} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i(X_i)\big|^2 \le\ & 4\Big(\mathbb{E}\big|Y_{t_i} - \bar{V}_i\big|^2 + \mathbb{E}\big|\bar{V}_i - \hat{V}_i\big|^2 + \mathbb{E}\big|\hat{V}_i - V_i\big|^2 + \mathbb{E}\big|V_i - \hat{\mathcal{U}}_i(X_i)\big|^2\Big) \\ =:\ & 4(I_i^1 + I_i^2 + I_i^3 + I_i^4), \end{aligned} \tag{5.6}$$

and analyze each of these contributions terms. In the sequel, C denotes a generic constant independent of π that may vary from line to line, and depending only on the coefficients satisfying Assumption 4.1. Notice that the first contribution term is the time discretization error for BSDE given by (5.2), and we shall study the three other terms in the following steps.

Step 1. Fix $i \in [\![0, N-1]\!]$. From the definition (4.1) of V_i and by the martingale representation theorem, there exists a square integrable process $\{\widehat{Z}_s, t_i \le s \le T\}$ such that

$$g(X_N) + f\big(t_i, X_i, V_i, \overline{\widehat{Z}_i}\big)\Delta t_i + \sum_{j=i+1}^{N-1} f\big(t_j, X_j, \widehat{\mathcal{U}}_j(X_j), \widehat{\mathcal{Z}}_j(X_j)\big)\Delta t_j = V_i + \int_{t_i}^{t_N} \widehat{Z}_s \mathrm{d}W_s. \tag{5.7}$$

From the definition (4.1) of $\overline{\widehat{Z}_i}$, and by Itô isometry, we then have

$$\overline{\widehat{Z}_i} = \frac{\mathbb{E}_i\big[\int_{t_i}^{t_{i+1}} \widehat{Z}_s \mathrm{d}s\big]}{\Delta t_i}, \ \text{ i.e. } \ \mathbb{E}_i\Big[\int_{t_i}^{t_{i+1}} \big(\widehat{Z}_s - \overline{\widehat{Z}_i}\big)\mathrm{d}s\Big] = 0. \tag{5.8}$$

Plugging (5.7) into (2.16), we see that the loss function of the MDBDP scheme can be rewritten as

$$\begin{aligned} \mathcal{J}_i^{MB}(\mathcal{U}_i, \mathcal{Z}_i) =\ & \mathbb{E}\Big|V_i - \mathcal{U}_i(X_i) + \Delta t_i\big[f\big(t_i, X_i, \mathcal{U}_i(X_i), \mathcal{Z}_i(X_i)\big) - f\big(t_i, X_i, V_i, \overline{\widehat{Z}_i}\big)\big] \\ & + \sum_{j=i+1}^{N-1}\int_{t_j}^{t_{j+1}} \big[\widehat{Z}_s - \widehat{\mathcal{Z}}_j(X_j)\big]\mathrm{d}W_s + \int_{t_i}^{t_{i+1}} \big[\widehat{Z}_s - \mathcal{Z}_i(X_i)\big]\mathrm{d}W_s\Big|^2 \\ =\ & \widetilde{\mathcal{J}}_i^{MB}(\mathcal{U}_i, \mathcal{Z}_i) + \mathbb{E}\Big[\sum_{j=i}^{N-1}\int_{t_j}^{t_{j+1}} \big|\widehat{Z}_s - \overline{\widehat{Z}_j}\big|_2^2 \mathrm{d}s\Big] + \sum_{j=i+1}^{N-1} \Delta t_j \mathbb{E}\big|\overline{\widehat{Z}_j} - \widehat{\mathcal{Z}}_j(X_j)\big|_2^2, \end{aligned} \tag{5.9}$$

where we use (5.8), and

$$\widetilde{\mathcal{J}}_i^{MB}(\mathcal{U}_i,\mathcal{Z}_i) := \mathbb{E}\Big|V_i - \mathcal{U}_i(X_i) + \Delta t_i\big[f\big(t_i,X_i,\mathcal{U}_i(X_i),\mathcal{Z}_i(X_i)\big) - f\big(t_i,X_i,V_i,\overline{\widehat{Z}_i}\big)\big]\Big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_i} - \mathcal{Z}_i(X_i)\big|_2^2.$$

It is clear by Lipschitz continuity of f in Assumption 4.1 that

$$\widetilde{\mathcal{J}}_i^{MB}(\mathcal{U}_i,\mathcal{Z}_i) \le C\Big(\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_i} - \mathcal{Z}_i(X_i)\big|_2^2\Big). \tag{5.10}$$

On the other hand, by Young inequality with $\beta \in (0,1)$, we have

$$\begin{aligned}
\widetilde{\mathcal{J}}_i^{MB}(\mathcal{U}_i,\mathcal{Z}_i) &\ge (1-\beta)\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2 + \Big(1-\frac{1}{\beta}\Big)|\Delta t_i|^2\mathbb{E}\big|f\big(t_i,X_i,\mathcal{U}_i(X_i),\mathcal{Z}_i(X_i)\big) - f\big(t_i,X_i,V_i,\overline{\widehat{Z}_i}\big)\big|^2 \\
&\quad + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_i} - \mathcal{Z}_i(X_i)\big|_2^2 \\
&\ge (1-\beta)\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2 - \frac{2[f]_L^2}{\beta}|\Delta t_i|^2\Big(\mathbb{E}\big|\mathcal{U}_i(X_i) - V_i\big|^2 + \mathbb{E}\big|\mathcal{Z}_i(X_i) - \overline{\widehat{Z}_i}\big|_2^2\Big) \\
&\quad + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_i} - \mathcal{Z}_i(X_i)\big|_2^2 \\
&\ge \Big(1 - \big(4[f]_L^2 + \frac{1}{2}\big)\Delta t_i\Big)\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2 + \frac{1}{2}\Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_i} - \mathcal{Z}_i(X_i)\big|_2^2,
\end{aligned} \tag{5.11}$$

where we use the Lipschitz continuity of f in the second inequality, and choose explicitly $\beta = 4[f]_L^2\Delta t_i$ (for Δt_i small enough) in the last one. By applying inequality (5.11) to $(\mathcal{U}_i,\mathcal{Z}_i) = (\widehat{\mathcal{U}}_i,\widehat{\mathcal{Z}}_i)$, which is a minimizer of $\tilde{\mathcal{J}}_i^{MB}$ by (5.9), and combining with (5.10), this yields for Δt_i small enough and for all network functions $\mathcal{U}_i$, $\mathcal{Z}_i$:

$$\mathbb{E}\big|V_i - \widehat{\mathcal{U}}_i(X_i)\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_i} - \widehat{\mathcal{Z}}_i(X_i)\big|_2^2 \le C\Big(\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_i} - \mathcal{Z}_i(X_i)\big|_2^2\Big).$$

By minimizing over $\mathcal{U}_i,\mathcal{Z}_i$ in the right hand side, we get the approximation error by neural networks of the regressed functions V_i, $\overline{\widehat{Z}_i}$:

$$\mathbb{E}\big|V_i - \widehat{\mathcal{U}}_i(X_i)\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_i} - \widehat{\mathcal{Z}}_i(X_i)\big|_2^2 \le C(\varepsilon_i^y + \Delta t_i\varepsilon_i^z). \tag{5.12}$$

Step 2. From the expression of V_i and $\hat{V}_i$ in (4.1), (5.3), and by Lipschitz continuity of f, we have

$$\begin{aligned}
\mathbb{E}\big|\hat{V}_i - V_i\big|^2 &= \Delta t_i^2\mathbb{E}\Big|\mathbb{E}_i\big[f\big(t_i,X_i,\widehat{\mathcal{U}}_i(X_i),\widehat{\mathcal{Z}}_i(X_i)\big) - f\big(t_i,X_i,V_i,\overline{\widehat{Z}_i}\big)\big]\Big|^2 \\
&\le 2[f]_L^2|\Delta t_i|^2\Big(\mathbb{E}\big|V_i - \widehat{\mathcal{U}}_i(X_i)\big|^2 + \mathbb{E}\big|\overline{\widehat{Z}_i} - \widehat{\mathcal{Z}}_i(X_i)\big|_2^2\Big) \\
&\le C\Delta t_i(\varepsilon_i^y + \Delta t_i\varepsilon_i^z), \qquad i = 0,\dots,N,
\end{aligned} \tag{5.13}$$

where we use (5.12) in the last inequality.

Step 3. From the recursive expressions of $\bar{V}_i$, $\hat{V}_i$ in (5.1), (5.4), and by applying Young, Cauchy-Schwarz inequalities, together with the Lipschitz continuity of f, we get for $\beta \in (0,1)$:

$$\begin{aligned}
\mathbb{E}\big|\bar{V}_i - \hat{V}_i\big|^2 &\le (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\bar{V}_{i+1} - \hat{V}_{i+1}\big]\Big|^2 + 2[f]_L^2\Big(1+\frac{1}{\beta}\Big)|\Delta t_i|^2\Big(\mathbb{E}\big|\bar{V}_i - \widehat{\mathcal{U}}_i(X_i)\big|^2 + \mathbb{E}\big|Z_i - \widehat{\mathcal{Z}}_i(X_i)\big|_2^2\Big) \\
&\le (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\bar{V}_{i+1} - \hat{V}_{i+1}\big]\Big|^2 + 2[f]_L^2\Big(1+\frac{1}{\beta}\Big)|\Delta t_i|^2\big(3\mathbb{E}|\bar{V}_i - \hat{V}_i|^2 + 2\mathbb{E}\big|Z_i - \overline{\widehat{Z}_i}\big|_2^2\big) \\
&\quad + 2[f]_L^2\Big(1+\frac{1}{\beta}\Big)|\Delta t_i|^2\Big(3\mathbb{E}|\hat{V}_i - V_i|^2 + 3\mathbb{E}|V_i - \widehat{\mathcal{U}}_i(X_i)|^2 + 2\mathbb{E}\big|\overline{\widehat{Z}_i} - \widehat{\mathcal{Z}}_i(X_i)\big|_2^2\Big) \\
&\le (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\bar{V}_{i+1} - \hat{V}_{i+1}\big]\Big|^2 + (1+\beta)\frac{2[f]_L^2|\Delta t_i|^2}{\beta}\big(3\mathbb{E}|\bar{V}_i - \hat{V}_i|^2 + 2\mathbb{E}\big|Z_i - \overline{\widehat{Z}_i}\big|_2^2\big) \\
&\quad + C[f]_L^2\Big(1+\frac{1}{\beta}\Big)\Delta t_i(\varepsilon_i^y + \Delta t_i\varepsilon_i^z),
\end{aligned} \tag{5.14}$$

where we use (5.12) and (5.13) in the last inequality. Moreover, by (5.1) and (5.5), we have

$$\begin{aligned}
\Delta t_i\big(Z_i - \overline{\widehat{Z}_i}\big) &= \mathbb{E}_i\Big[\Delta W_i\big(\bar{V}_{i+1} - \hat{V}_{i+1}\big)\Big] \\
&= \mathbb{E}_i\Big[\Delta W_i\Big(\bar{V}_{i+1} - \hat{V}_{i+1} - \mathbb{E}_i\big[\bar{V}_{i+1} - \hat{V}_{i+1}\big]\Big)\Big],
\end{aligned}$$

and thus by Cauchy-Schwarz inequality

$$\Delta t_i \mathbb{E}\big|Z_i - \overline{\widehat{Z}}_i\big|_2^2 \le\ d\Big(\mathbb{E}\big|\bar{V}_{i+1} - \hat{V}_{i+1}\big|^2 - \mathbb{E}\Big|\mathbb{E}_i\big[\bar{V}_{i+1} - \hat{V}_{i+1}\big]\Big|^2\Big). \tag{5.15}$$

Plugging into (5.14), and choosing $\beta = 4d[f]_L^2 \Delta t_i$, gives

$$(1 - C\Delta t_i)\mathbb{E}\big|\bar{V}_i - \hat{V}_i\big|^2 \le\ (1 + C\Delta t_i)\mathbb{E}\big|\bar{V}_{i+1} - \hat{V}_{i+1}\big|^2 + (1 + C\Delta t_i)\big(\varepsilon_i^y + \Delta t_i \varepsilon_i^z\big)$$

By discrete Gronwall lemma, and recalling that $\bar{V}_N = \hat{V}_N\ (= g(X_N))$, we then obtain

$$\sup_{i \in [\![0,N]\!]} \mathbb{E}\big|\bar{V}_i - \hat{V}_i\big|^2 \le\ C \sum_{i=0}^{N-1} \big(\varepsilon_i^y + \Delta t_i \varepsilon_i^z\big). \tag{5.16}$$

The required bound for the approximation error on Y follows by plugging (5.2), (5.12), (5.13), and (5.16) into (5.6).

Step 4. We decompose the approximation error for the Z component into three terms

$$\begin{aligned} &\mathbb{E}\Big[\sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} \big|Z_s - \widehat{\mathcal{Z}}_i(X_i)\big|_2^2\, \mathrm{d}s\Big] \\ &\le\ 3 \sum_{i=0}^{N-1} \Big(\mathbb{E}\Big[\int_{t_i}^{t_{i+1}} \big|Z_s - Z_i\big|_2^2\, \mathrm{d}s\Big] + \Delta t_i \mathbb{E}\big|Z_i - \overline{\widehat{Z}}_i\big|_2^2 + \Delta t_i \mathbb{E}\big|\overline{\widehat{Z}}_i - \widehat{\mathcal{Z}}_i(X_i)\big|_2^2\Big). \end{aligned} \tag{5.17}$$

By summing the inequality (5.15) (recalling that $\bar{V}_N = \hat{V}_N$), and using (5.14), we have for $\beta \in (0,1)$

$$\begin{aligned} &\sum_{i=0}^{N-1} \Delta t_i \mathbb{E}|Z_i - \overline{\widehat{Z}}_i|^2 \\ &\le\ d \sum_{i=0}^{N-1} \Big(\mathbb{E}\big|\bar{V}_i - \hat{V}_i\big|^2 - \mathbb{E}\big|\mathbb{E}_i\big[\bar{V}_{i+1} - \hat{V}_{i+1}\big]\big|^2\Big) \\ &\le\ d \sum_{i=0}^{N-1} \Big(\beta \mathbb{E}\Big|\mathbb{E}_i\big[\bar{V}_{i+1} - \hat{V}_{i+1}\big]\Big|^2 + \big(1 + \frac{1}{\beta}\big)\big(2[f]_L^2 |\Delta t_i|^2\big)\big(3\mathbb{E}|\bar{V}_i - \hat{V}_i|^2 + 2\mathbb{E}\big|Z_i - \overline{\widehat{Z}_i}\big|_2^2\big) \\ &\qquad +\ C[f]_L^2 \big(1 + \frac{1}{\beta}\big)\Delta t_i (\varepsilon_i^y + \Delta t_i \varepsilon_i^z)\Big) \\ &\le\ d \sum_{i=0}^{N-1} \Big(\frac{8d[f]_L^2 \Delta t_i}{1 - 8d[f]_L^2 \Delta t_i} \mathbb{E}\big|\mathbb{E}_i\big[\bar{V}_{i+1} - \hat{V}_{i+1}\big]\big|^2 + \frac{3}{4d} \Delta t_i \mathbb{E}|\bar{V}_i - \hat{V}_i|^2\ +\ \frac{C}{8d}(\varepsilon_i^y + \Delta t_i \varepsilon_i^z)\Big) \\ &\qquad +\ \frac{1}{2} \sum_{i=0}^{N-1} \Delta t_i \mathbb{E}\big|Z_i - \overline{\widehat{Z}}_i\big|_2^2, \end{aligned} \tag{5.18}$$

by choosing explicitely $\beta = \frac{8d[f]_L^2 \Delta t_i}{1 - 8d[f]_L^2 \Delta t_i} = O(\Delta t_i)$ for Δt_i small enough. Plugging (5.2), (5.12), (5.16), and (5.18) (using Jensen inequality) into (5.17), this proves the required bound for the approximation error on Z, and completes the proof. □

5.2 Proof of Proposition 4.1

In the sequel, C denotes a generic constant independent of $|\pi|, \gamma, R, m$, possible depending on the coefficients $\mu, \sigma, f, g, T, \mathcal{X}_0$, that may vary from line to line. We write sometimes the constant $C(d)$ to stress the dependence on d.

Step 1. Fix $i \in [\![0, N-1]\!]$. Let us show that the functions $v_i, \widehat{z}_i$ defined in (4.1) are Lipschitz. For $x \in \mathbb{R}^d$, we define by induction the processes X_j^x, $j = i, \dots, N$,

$$X_{j+1}^x := X_j^x + \mu(t_j, X_j^x)\Delta t_j + \sigma(t_j, X_j^x)\Delta W_j, \quad j = i, \dots, N-1,\ X_i^x\ =\ x,$$

so that

$$\begin{cases} v_i(x)\ =\ \mathbb{E}\Big[g(X_N^x) + f(t_i, x, v_i(x), \widehat{z}_i(x))\Delta t_i + \sum_{j=i+1}^{N-1} f\big(t_j, X_j^x, \widehat{\mathcal{U}}_j(X_j^x), \widehat{\mathcal{Z}}_j(X_j^x)\big)\Delta t_j\Big], \\ \Delta t_i\ \widehat{z}_i(x)\ =\ \mathbb{E}\Big[g(X_N^x)\Delta W_i + \sum_{j=i+1}^{N-1} f\big(t_j, X_j^x, \widehat{\mathcal{U}}_j(X_j^x), \widehat{\mathcal{Z}}_j(X_j^x)\big)\Delta W_i \Delta t_j\Big]. \end{cases} \tag{5.19}$$

From the Lipschitz condition on g in Assumption 4.2, on f in Assumption 4.1, and by Cauchy-Schwarz inequality, we then have for all $x, x' \in \mathbb{R}^d$,

$$\begin{aligned}
&\Delta t_i |\widehat{z}_i(x) - \widehat{z}_i(x')| \\
&\leq [g]_{L} \sqrt{d\Delta t_i} \|X_N^x - X_N^{x'}\|_2 \\
&\quad + [f]_{L} \sqrt{d\Delta t_i} \sum_{j=i+1}^{N-1} \Big[\|X_j^x - X_j^{x'}\|_2 + \big\|\widehat{\mathcal{U}}_j(X_j^x) - \widehat{\mathcal{U}}_j(X_j^{x'})\big\|_2 + \big\|\widehat{\mathcal{Z}}_j(X_j^x) - \widehat{\mathcal{Z}}_j(X_j^{x'})\big\|_2 \Big] \Delta t_j,
\end{aligned}$$

where $\|X\|_{2p} := \big(\mathbb{E}|X|_2^{2p}\big)^{\frac{1}{2p}}$ denotes the L^{2p}-norm, $p \geq 1$. Now, by the Lipschitz property (2.3) of $\widehat{\mathcal{U}}_j \in \mathcal{N}_{d,m,1}^{1,\gamma,R}$ and $\widehat{\mathcal{Z}}_j \in \mathcal{N}_{d,m,d}^{1,\gamma,R}$, we have

$$\big\|\widehat{\mathcal{U}}_j(X_j^x) - \widehat{\mathcal{U}}_j(X_j^{x'})\big\|_2 + \big\|\widehat{\mathcal{Z}}_j(X_j^x) - \widehat{\mathcal{Z}}_j(X_j^{x'})\big\|_2 \leq C(d)\frac{\gamma}{R}\|X_j^x - X_j^{x'}\|_2. \tag{5.20}$$

From the standard estimate $\|X_j^x - X_j^{x'}\|_2 \leq C|x - x'|_2$, $j = i, \ldots, N$, it follows that

$$\Delta t_i |\widehat{z}_i(x) - \widehat{z}_i(x')| \leq C(d)\sqrt{\Delta t_i}\Big(1 + \frac{\gamma}{R}\Big)|x - x'|_2, \quad x, x' \in \mathbb{R}^d. \tag{5.21}$$

Back to (5.19), we have similarly

$$\begin{aligned}
|v_i(x) - v_i(x')| \leq\ & [g]_{L}\|X_N^x - X_N^{x'}\|_2 + \Delta t_i [f]_{L}\big(|x - x'|_2 + |v_i(x) - v_i(x')| + |\widehat{z}_i(x) - \widehat{z}_i(x')|\big) \\
&+ [f]_{L} \sum_{j=i+1}^{N-1} \Big[\|X_j^x - X_j^{x'}\|_2 + \big\|\widehat{\mathcal{U}}_j(X_j^x) - \widehat{\mathcal{U}}_j(X_j^{x'})\big\|_2 + \big\|\widehat{\mathcal{Z}}_j(X_j^x) - \widehat{\mathcal{Z}}_j(X_j^{x'})\big\|_2 \Big] \Delta t_j,
\end{aligned}$$

and so for Δt_i small enough, and by using (5.20), (5.21):

$$\begin{aligned}
|v_i(x) - v_i(x')| &\leq C\Delta t_i\big(|x - x'|_2 + |\widehat{z}_i(x) - \widehat{z}_i(x')|\big) \\
&\quad + C\Big[\|X_N^x - X_N^{x'}\|_2 + \Big(1 + \frac{\gamma}{R}\Big) \sum_{j=i+1}^{N-1} \|X_j^x - X_j^{x'}\|_2 \Delta t_j\Big] \\
&\leq C(d)\Big(1 + \frac{\gamma}{R}\Big)|x - x'|_2, \quad x, x' \in \mathbb{R}^d.
\end{aligned} \tag{5.22}$$

On the other hand, back to the expression (5.19), and from the linear growth conditions on f, g, together with Cauchy-Schwarz inequality, we have

$$\begin{aligned}
\Delta t_i |\widehat{z}_i(x)| &\leq C\sqrt{\Delta t_i}\Big[1 + \|X_N^x\|_2 + \sum_{j=i+1}^{N-1} \Big(1 + \|X_j^x\|_2 + \big\|\widehat{\mathcal{U}}_j(X_j^x)\big\|_2 + \big\|\widehat{\mathcal{Z}}_j(X_j^x)\big\|_2\Big)\Delta t_j\Big] \\
&\leq C(d)\gamma\sqrt{\Delta t_i}\Big[1 + \|X_N^x\|_2 + \sum_{j=i+1}^{N-1} \Big(1 + \|X_j^x\|_2\Big)\Delta t_j\Big] \\
&\leq C(d)\gamma\sqrt{\Delta t_i}(1 + |x|_2), \quad x \in \mathbb{R}^d,
\end{aligned}$$

from the growth condition (2.2) on $\widehat{\mathcal{U}}_i \in \mathcal{N}_{d,m,1}^{1,\gamma,R}$, $\widehat{\mathcal{Z}}_i \in \mathcal{N}_{d,m,d}^{1,\gamma,R}$, and the standard estimate $\|X_j^x\|_{2p} \leq C(1 + |x|_2)$. We then get similarly for v_i:

$$\begin{aligned}
|v_i(x)| \leq\ & C\Big[1 + \|X_N^x\|_2 + \Delta t_i\Big(1 + \|X_j^x\|_2 + |v_i(x)| + |\widehat{z}_i(x)|\Big) \\
&+ \sum_{j=i+1}^{N-1} \Big(1 + \|X_j^x\|_2 + \big\|\widehat{\mathcal{U}}_j(X_j^x)\big\|_2 + \big\|\widehat{\mathcal{Z}}_j(X_j^x)\big\|_2\Big)\Delta t_j\Big],
\end{aligned}$$

and so for Δt_i small enough

$$|v_i(x)| \leq C(d)\gamma(1 + |x|_2), \quad x \in \mathbb{R}^d. \tag{5.23}$$

Step 2. From the approximation result in (2.9) for Lipschitz functions, localization of X_i on $B_2(R) = \{x \in \mathbb{R}^d : |x|_2 \leq R\}$, Hölder inequality, and Bienaymé–Chebyshev inequality, the approximation error of v_i, $i =$

$0, \dots, N-1$, by the class of networks $\mathcal{N}_{d,m,1}^{1,R,\gamma}$, is bounded by

$$\begin{aligned}
\sqrt{\varepsilon_i^y} &= \inf_{\mathcal{U}\in\mathcal{N}_{d,m,1}^{1,R,\gamma}} \big\| v_i(X_i) - \mathcal{U}(X_i) \big\|_2 \\
&\le \inf_{\mathcal{U}\in\mathcal{N}_{d,m,1}^{1,R,\gamma}} \Big\| \big(v_i(X_i) - \mathcal{U}(X_i)\big)\mathbf{1}_{X_i\in B_2(R)} \big\|_2 + \Big\| \big(v_i(X_i) - \widehat{\mathcal{U}}_i(X_i)\big)\mathbf{1}_{|X_i|_2\ge R} \Big\|_2 \\
&\le C\Big([v_i]_{L,R} \Big(\frac{\gamma}{[v_i]_{L,R}} \Big)^{-2/(d+1)} \log\Big(\frac{\gamma}{[v_i]_{L,R}} \Big) + \gamma m^{-\frac{d+3}{2d}} \Big) \\
&\quad + \frac{\big(\|v_i(X_i)\|_{2+\delta} + \|\widehat{\mathcal{U}}_i(X_i)\|_{2+\delta} \big) \|X_i\|_p^p}{R^p},
\end{aligned} \tag{5.24}$$

for any $\delta > 0$, and $p \ge 1$. Now, from the growth condition (2.2) for any function in $\mathcal{N}_{d,m,1}^{1,\gamma,R}$, and the growth condition (5.23), we have

$$\|\widehat{\mathcal{U}}_i(X_i)\|_4 + \|v_i(X_i)\|_4 \le C(d)\gamma\Big(1 + \|X_i\|_4\Big).$$

Recalling the standard estimate $\|X_i\|_{2p} \le C(1 + \|\mathcal{X}_0\|_{2p})$, $i = 0, \dots, N$, we then have for R large enough, $\gamma/R = O(1)$, and using the fact that $[v_i]_{L,R} \le C(d)\gamma/R$ from (5.22):

$$\varepsilon_i^y \le C(d, \mathcal{X}_0, p)\Big\{ \Big(R^{-2/(d+1)} \log(R) + R m^{-\frac{d+3}{2d}} \Big)^2 + \frac{1}{R^{2p-2}} \Big\}, \tag{5.25}$$

for some constant $C(d, \mathcal{X}_0, p)$ independent of N, R, m, and this holds for any $p \ge 1$. Similarly, we obtain the bound for the approximation error of $\widehat{z}_i$, $i = 0, \dots, N-1$, for $\gamma/R = O(1)$, by using now the fact that $\sqrt{\Delta t_i}[\widehat{z}_i]_{L,R} \le C(d)\gamma/R$ from (5.21):

$$\begin{aligned}
\Delta t_i \varepsilon_i^z &\le C\Big(\sqrt{\Delta t_i}[\widehat{z}_i]_{L,R} \Big(\frac{\gamma}{[\widehat{z}_i]_{L,R}} \Big)^{-2(d+1)} \log\Big(\frac{\gamma}{[\widehat{z}_i]_{L,R}} \Big) + \gamma\sqrt{\Delta t_i} m^{-\frac{d+3}{2d}} \Big)^2 \\
&\quad + \frac{\Delta t_i \big(\|\widehat{z}_i(X_i)\|_{2+\delta} + \|\widehat{\mathcal{U}}_i(X_i)\|_{2+\delta} \big)^2 \|X_i\|_p^{2p}}{R^{2p}} \\
&\le C(d, \mathcal{X}_0, p)\Big\{ \Big((R\sqrt{\Delta t_i})^{-2/(d+1)} \log(R\sqrt{\Delta t_i}) + R\sqrt{\Delta t_i} m^{-\frac{d+3}{2d}} \Big)^2 + \frac{\Delta t_i}{R^{2p-2}} \Big\}.
\end{aligned} \tag{5.26}$$

Plugging estimates (5.25)-(5.26) into (4.2), and recalling that $\mathbb{E}|g(\mathcal{X}_T) - g(X_N)|^2 + \varepsilon^Z(\pi) = O(|\pi|) = O(1/N)$ when g is Lipschitz, the approximation error in (Y, Z) is bounded by

$$\begin{aligned}
&\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i(X_i)\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} \big|Z_s - \widehat{\mathcal{Z}}_i(X_i)\big|_2^2 \, \mathrm{d}s \Big] \\
&\le C(d, \mathcal{X}_0, p)\Big[\frac{1}{N} + N\Big(R^{-2/(d+1)} \log(R) + R m^{-\frac{d+3}{2d}} \Big)^2 + \frac{N}{R^{2p-2}} \Big],
\end{aligned}$$

for any $p \ge 1$. Therefore, to achieve a rate of convergence of order $1/N$ for the left hand side of the above inequality, it suffices to choose R and m s.t.

$$N R^{-4/(d+1)} |\log(R)|^2 = O(1/N), \quad \text{and} \quad N R^2 m^{-\frac{d+3}{d}} = O(1/N),$$

hence for example with $R = O\big(N^{\frac{d+1}{2}+\epsilon}\big)$, and $m = O\big(N^{d+2\epsilon}\big)$, for any $\epsilon > 0$, and then take $p = (d+3)/(d+1)$ so that $N/R^{2p-2} = O(1/N)$. □

5.3 Proof of Theorem 4.2

Let us introduce the *explicit* backward Euler scheme of the BSDE (2.11):

$$\left\{ \begin{array}{rcl} \bar{V}_i &=& \mathbb{E}_i\Big[\bar{V}_{i+1} + f\big(t_i, X_i, \bar{V}_{i+1}, Z_i\big)\Delta t_i \Big] \\ Z_i &=& \mathbb{E}_i\Big[\bar{V}_{i+1} \frac{\Delta W_i}{\Delta t_i} \Big], \quad i = 0, \dots, N-1, \end{array} \right. \tag{5.27}$$

starting from $\bar{V}_N = g(X_N)$, and which is also known to converge with the same time discretization error (5.2) than the implicit backward scheme.

We decompose the approximation error into three terms:

$$\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i(X_i)\big|^2 \le 3\Big(\mathbb{E}\big|Y_{t_i} - \bar{V}_i\big|^2 + \mathbb{E}\big|\bar{V}_i - V_i\big|^2 + \mathbb{E}\big|V_i - \widehat{\mathcal{U}}_i(X_i)\big|^2\Big). \tag{5.28}$$

The first term is the classical time discretization error, and the rest of the proof is devoted to the analysis of the second and third terms.

Step 1. Fix $i \in [\![0, N-1]\!]$. By definition of V_i in (4.3) and the martingale representation theorem, there exists a square integrable process $\{\widehat{Z}_s, t_i \le s \le t_{i+1}\}$ such that

$$\widehat{\mathcal{U}}_{i+1}(X_{i+1}) + f\big(t_i, X_i, \mathbb{E}_i[\widehat{\mathcal{U}}_{i+1}(X_{i+1})], \mathbb{E}_i[\sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1})]\big)\Delta t_i = V_i + \int_{t_i}^{t_{i+1}} \widehat{Z}_s \, \mathrm{d}W_s.$$

It follows that the quadratic loss function of the DS scheme in (2.15) can be written as

$$\begin{aligned} J_i^S(\mathcal{U}_i) &:= \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \mathcal{U}_i(X_i) + f\big(t_i, X_{i+1}, \widehat{\mathcal{U}}_{i+1}(X_{i+1}), \sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1})\big)\Delta t_i\Big|^2 \\ &= \tilde{J}_i^S(\mathcal{U}_i) + \mathbb{E}\Big[\int_{t_i}^{t_{i+1}} |\widehat{Z}_s|_2^2 \mathrm{d}s\Big], \end{aligned} \tag{5.29}$$

where

$$\begin{aligned} \tilde{J}_i^S(\mathcal{U}_i) &:= \mathbb{E}\Big|V_i - \mathcal{U}_i(X_i) + \Delta f_i \Delta t_i\Big|^2 \\ \text{with} \quad \Delta f_i &:= f\big(t_i, X_{i+1}, \widehat{\mathcal{U}}_{i+1}(X_{i+1}), \sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1})\big) \\ &\quad - f\big(t_i, X_i, \mathbb{E}_i[\widehat{\mathcal{U}}_{i+1}(X_{i+1})], \mathbb{E}_i[\sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1})]\big). \end{aligned}$$

A direct application of Young inequality in the form $(a+b)^2 \ge \frac{1}{2}a^2 - b^2$ leads to

$$\tilde{J}_i^S(\mathcal{U}_i) + |\Delta t_i|^2\mathbb{E}\big|\Delta f_i\big|^2 \ge \frac{1}{2}\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2. \tag{5.30}$$

On the other hand, by Lipschitz continuity of f, we have

$$\begin{aligned} \tilde{J}_i^S(\mathcal{U}_i) + |\Delta t_i|^2\mathbb{E}\big|\Delta f_i\big|^2 &\le 2\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2 + 3|\Delta t_i|^2\mathbb{E}\big|\Delta f_i\big|^2 \\ &\le 2\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2 + 9|\Delta t_i|^2[f]_L^2\mathbb{E}|X_{i+1} - X_i|_2^2 \\ &\quad + 9|\Delta t_i|^2[f]_L^2\mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \mathbb{E}_i[\widehat{\mathcal{U}}_{i+1}(X_{i+1})]\Big|^2 \\ &\quad + 9|\Delta t_i|^2[f]_L^2\mathbb{E}\Big|\sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \mathbb{E}_i\big[\sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1})\big]\Big|_2^2 \\ &\le 2\,\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2 + 9|\Delta t_i|^2[f]_L^2\mathbb{E}|X_{i+1} - X_i|_2^2 \\ &\quad + 9|\Delta t_i|^2[f]_L^2\mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \widehat{\mathcal{U}}_{i+1}(X_i)\Big|^2 \\ &\quad + 9|\Delta t_i|^2[f]_L^2\mathbb{E}\Big[|\sigma(t_i, X_i)|_2^2\mathbb{E}_i\big|D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - D_x\widehat{\mathcal{U}}_{i+1}(X_i)\big|_2^2\Big], \end{aligned} \tag{5.31}$$

where we use the definition of conditional expectation $\mathbb{E}_i[.]$, and the tower property of conditional expectation in the last inequality. Recall from Lemma 2.1 that the network function $\widehat{\mathcal{U}}_{i+1} \in \mathcal{N}_{d,m,1}^{2,\gamma,R}$ is locally Lipschitz on $\mathbb{R}^d$. Actually, from (2.5), we have

$$\big|\widehat{\mathcal{U}}_{i+1}(x) - \widehat{\mathcal{U}}_{i+1}(x')\big| \le 2d\frac{\gamma}{R}\max\big(1, \frac{|x|_2 + |x'|_2}{R}\big)|x - x'|_2, \quad \forall x, x' \in \mathbb{R}^d. \tag{5.32}$$

By Cauchy-Schwarz inequality, we then have

$$\begin{aligned} \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \widehat{\mathcal{U}}_{i+1}(X_i)\Big|^2 &\le C(d)\frac{\gamma^2}{R^2}\Big(1 + \frac{\|X_{i+1}\|_4^2 + \|X_i\|_4^2}{R^2}\Big)\big\|X_{i+1} - X_i\big\|_4^2 \\ &\le C(d)\frac{\gamma^2}{R^2}\Big(1 + \big\|\mathcal{X}_0\big\|_4^2\Big)^2\Delta t_i \end{aligned}$$

for Δt_i small enough, $R \ge 1$, and we used again the standard estimate: $\|X_i\|_{2p} \le C(1 + \|\mathcal{X}_0\|_{2p})$, $\|X_{i+1} - X_i\|_{2p} \le C(1 + \|\mathcal{X}_0\|_{2p})\sqrt{\Delta t_i}$, for $p \ge 1$. By using also the Lipschitz condition on $D_x\widehat{\mathcal{U}}_{i+1}$ in $\mathcal{N}_{d,m,1}^{2,\gamma,R}$ (see Remark 2.1), and plugging into (5.31), we then get

$$\tilde{J}_i^S(\mathcal{U}_i) + |\Delta t_i|^2\mathbb{E}\big|\Delta f_i\big|^2 \le 2\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2 + C(d)\max\big[1, \frac{\gamma^2}{R^2}\big]\Big(1 + \big\|\mathcal{X}_0\big\|_4^2\Big)^2|\Delta t_i|^3. \tag{5.33}$$

By applying inequality (5.30) to $\mathcal{U}_i = \widehat{\mathcal{U}}_i$, which is a minimizer of $\tilde{\mathcal{J}}_i^S$ by (5.29), and combining with (5.33), this yields for all network functions $\mathcal{U}_i$ in $\mathcal{N}_{d,m}^{+,\gamma,R}$:

$$\mathbb{E}\big|V_i - \widehat{\mathcal{U}}_i(X_i)\big|^2 \leq C\Big(\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2 + (1 + \|\mathcal{X}_0\|_4^2)^2|\Delta t_i|^3 \max\big[1, \frac{\gamma^2}{R^2}\big]\Big),$$

and thus by minimizing over $\mathcal{U}_i$ in the right hand side

$$\mathbb{E}\big|V_i - \widehat{\mathcal{U}}_i(X_i)\big|^2 \leq C\Big(\varepsilon_i^{m,\gamma,R} + (1 + \|\mathcal{X}_0\|_4^2)^2|\Delta t_i|^3 \max\big[1, \frac{\gamma^2}{R^2}\big]\Big). \tag{5.34}$$

Step 2. From the expressions of V_i, and $\bar{V}_i$ in (4.3) and (5.27), and by applying Young, Cauchy-Schwarz inequalities, we get with $\beta \in (0,1)$

$$\begin{aligned}
\mathbb{E}\big|\bar{V}_i - V_i\big|^2 \leq\ & (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{V}_{i+1}\big]\Big|^2 \\
&+ \big(1 + \frac{1}{\beta}\big)|\Delta t_i|^2\mathbb{E}\Big|f\big(t_i, X_i, \mathbb{E}_i[\widehat{\mathcal{U}}_{i+1}(X_{i+1})], \mathbb{E}_i[\sigma(t_i,X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1})]\big) \\
&\qquad - f\big(t_i, X_i, \bar{V}_{i+1}, Z_i\big)\Big|^2 \\
\leq\ & (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{V}_{i+1}\big]\Big|^2 \\
&+ 2[f]_L^2\big(1 + \frac{1}{\beta}\big)|\Delta t_i|^2\Big(\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{V}_{i+1}\big|^2 + \mathbb{E}\Big|\mathbb{E}_i\big[\sigma(t_i,X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1})\big] - Z_i\Big|_2^2\Big)
\end{aligned} \tag{5.35}$$

Now, recalling the expression of Z_i in (5.27), and by a standard integration by parts argument (see e.g. Lemma 2.1 in [FTW11]), we have

$$\begin{aligned}
\mathbb{E}_i\big[\sigma(t_i,X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1})\big] - Z_i =\ & \mathbb{E}_i\Big[\big(\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{V}_{i+1}\big)\frac{\Delta W_i}{\Delta t_i}\Big] \\
=\ & \mathbb{E}_i\Big[\Big(\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{V}_{i+1} - \mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{V}_{i+1}\big]\Big)\frac{\Delta W_i}{\Delta t_i}\Big].
\end{aligned}$$

By plugging into (5.35), we then obtain by Cauchy-Schwarz inequality

$$\begin{aligned}
\mathbb{E}\big|\bar{V}_i - V_i\big|^2 \leq\ & (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{V}_{i+1}\big]\Big|^2 + 2[f]_L^2(1+\beta)\frac{|\Delta t_i|^2}{\beta}\Big\{\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{V}_{i+1}\big|^2 \\
&+ \frac{d}{\Delta t_i}\Big[\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{V}_{i+1}\big|^2 - \mathbb{E}\Big|\mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{V}_{i+1}\big]\Big|^2\Big]\Big\} \\
\leq\ & (1 + C\Delta t_i)\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{V}_{i+1}\big|^2,
\end{aligned} \tag{5.36}$$

by choosing explicitly $\beta = 2d[f]_L^2\Delta t_i$ for Δt_i small enough. By using again Young inequality on the r.h.s. of (5.36), and since $\Delta t_i = O(1/N)$, we then get

$$\mathbb{E}\big|\bar{V}_i - V_i\big|^2 \leq (1 + C\Delta t_i)\mathbb{E}\big|\bar{V}_{i+1} - V_{i+1}\big|^2 + CN\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - V_{i+1}\big|^2.$$

By discrete Gronwall lemma, and recalling that $\bar{V}_N = g(X_N)$, $V_N = \widehat{\mathcal{U}}_N(X_N)$, we deduce with (5.34) that

$$\sup_{i\in[\![0,N]\!]}\mathbb{E}\big|\bar{V}_i - V_i\big|^2 \leq C\varepsilon_N^{m,\gamma,R} + CN\sum_{i=1}^{N-1}\Big(\varepsilon_i^{m,\gamma,R} + (1 + \|\mathcal{X}_0\|_4^2)^2|\Delta t_i|^3\max\big[1, \frac{\gamma^2}{R^2}\big]\Big). \tag{5.37}$$

The required bound (4.4) for the approximation error on Y follows by plugging (5.2), (5.34) and (5.37) into (5.28). □

5.4 Proof of Proposition 4.2

Step 1. In the sequel, C denotes a generic constant independent of $|\pi|, \gamma, R, m$, possible depending on the coefficients μ, σ, f, T, that may vary from line to line. We write sometimes the constant $C(d)$ to stress the dependence on d. Fix $i \in [\![0, N-1]\!]$. Let us show that the function v_i defined via $V_i = v_i(X_i)$ (see (4.3)) is locally Lipschitz.

Define $X_{i+1}^x := x + \mu(t_i,x)\Delta t_i + \sigma(t_i,x)\Delta W_i$, and let $x, x' \in \mathbb{R}^d$. Then, from (4.3), we write

$$v_i(x) - v_i(x') = \mathbb{E}\Big[\widehat{\mathcal{U}}_{i+1}(X_{i+1}^x) - \widehat{\mathcal{U}}_{i+1}(X_{i+1}^{x'}) + \Delta t_i\Delta f_i\Big], \tag{5.38}$$

where

$$\Delta f_i := f\big(t_i, x, \mathbb{E}[\widehat{\mathcal{U}}_{i+1}(X^x_{i+1})], \mathbb{E}[\sigma(t_i,x)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X^x_{i+1})]\big) - f\big(t_i, x', \mathbb{E}[\widehat{\mathcal{U}}_{i+1}(X^{x'}_{i+1})], \mathbb{E}[\sigma(t_i,x')^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X^{x'}_{i+1})]\big).$$

From the Lipschitz condition on f in Assumption 4.1 and 4.2 (see relation (4.5)), we then have

$$\begin{aligned}|\Delta f_i| \le\ & K_{f,\sigma}\Big\{|x-x'|_2\big(1+\mathbb{E}|D_x\widehat{\mathcal{U}}_{i+1}(X^x_{i+1})|\big) + \mathbb{E}|\widehat{\mathcal{U}}_{i+1}(X^x_{i+1}) - \widehat{\mathcal{U}}_{i+1}(X^{x'}_{i+1})| \\ & +\ \mathbb{E}\big|D_x\widehat{\mathcal{U}}_{i+1}(X^x_{i+1}) - D_x\widehat{\mathcal{U}}_{i+1}(X^{x'}_{i+1})\big|_2\Big\}. \end{aligned} \tag{5.39}$$

From (5.32), and by Cauchy-Schwarz inequality, we have

$$\begin{aligned}\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}(X^x_{i+1}) - \widehat{\mathcal{U}}_{i+1}(X^{x'}_{i+1})\big| &\le\ 2d\frac{\gamma}{R}\Big(1+\frac{\|X^x_{i+1}\|_2 + \|X^{x'}_{i+1}\|_2}{R}\Big)\big\|X^x_{i+1} - X^{x'}_{i+1}\big\|_2 \\ &\le\ C(d)\frac{\gamma}{R}\Big(1+\frac{|x|_2+|x'|_2}{R}\Big)|x-x'|_2 \end{aligned} \tag{5.40}$$

for Δt_i small enough, where we used the standard estimate: $\|X^x_{i+1}\|_{2p} \le |x|_2(1+C\sqrt{\Delta t_i}) + C\sqrt{\Delta t_i}$, $\|X^x_{i+1} - X^{x'}_{i+1}\|_{2p} \le (1+C\sqrt{\Delta t_i})|x-x'|_2$. From the Lipschitz property of $D_x\widehat{\mathcal{U}}_{i+1}$ (see Remark 2.1), we have

$$\mathbb{E}\big|D_x\widehat{\mathcal{U}}_{i+1}(X^x_{i+1}) - D_x\widehat{\mathcal{U}}_{i+1}(X^{x'}_{i+1})\big|_2 \le\ C(d)\frac{\gamma}{R^2}|x-x'|_2. \tag{5.41}$$

Moreover, from (2.5), we have

$$\big\|D_x\widehat{\mathcal{U}}_{i+1}(X^x_{i+1})\big\|_2 \le\ 2d\frac{\gamma}{R}\Big(1+\frac{\|X^x_{i+1}\|_2}{R}\Big) \le\ C(d)\frac{\gamma}{R}\Big(1+\frac{|x|_2}{R}\Big). \tag{5.42}$$

It follows from (5.38), (5.39), (5.40), (5.41) and (5.42) that for $R \ge 1$, $\Delta t_i \le 1$:

$$|v_i(x) - v_i(x')| \le\ C(d)\Big[1+\frac{\gamma}{R}\Big(1+\frac{|x|_2+|x'|_2}{R}\Big)\Big]|x-x'|_2, \quad x, x' \in \mathbb{R}^d,$$

which shows that v_i is locally Lipschitz. In the next step, we shall take γ/R of order 1 (with respect to the modulus $|\pi|$ of the time discretization), and so $[v_i]_{L,R} \le C(d)\gamma/R$.

On the other hand, back to the expression of v_i in (4.3), from the growth linear condition of σ, f in Assumption 4.1, and the growth conditions in Lemma 2.1, we have for all $x \in \mathbb{R}^d$:

$$\begin{aligned}|v_i(x)| &\le\ (1+C\Delta t_i)\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}(X^x_{i+1})\big| + C\Delta t_i(1+|x|_2)\Big(1+\mathbb{E}\big|D_x\widehat{\mathcal{U}}_{i+1}(X^x_{i+1})\big|_2\Big) \\ &\le\ 3\gamma(1+C\Delta t_i)\Big(1+\frac{\|X^x_{i+1}\|_2^2}{R^2}\Big) + 2d\frac{\gamma}{R}C\Delta t_i(1+|x|_2)\Big(1+\frac{\|X^x_{i+1}\|_2}{R}\Big) \\ &\le\ C(d)\gamma\big(1+\frac{|x|_2^2}{R^2}\big), \end{aligned} \tag{5.43}$$

for Δt_i small enough, where we used again the estimate $\|X^x_{i+1}\|_2 \le |x|_2(1+C\sqrt{\Delta t_i}) + C\sqrt{\Delta t_i}$.

Step 2. Similarly as in (5.24), by using the approximation result in (2.9) for locally Lipschitz continuous, we see that the approximation error of v_i, $i = 0, \ldots, N-1$, by the class of networks $\mathcal{N}^{2,R,\gamma}_{d,m,1}$, is bounded by

$$\begin{aligned}\sqrt{\varepsilon_i^{r,\gamma,R}} &= \inf_{\mathcal{U}\in\mathcal{N}^{2,R,\gamma}_{d,m,1}} \big\|v_i(X_i) - \mathcal{U}(X_i)\big\|_2 \\ &\le\ C\Big([v_i]_{L,R}\Big(\frac{\gamma}{[v_i]_{L,R}}\Big)^{-2/(d+3)}\log\Big(\frac{\gamma}{[v_i]_{L,R}}\Big) + \gamma m^{-1/2}\Big) \\ &\quad +\ \frac{\big(\|v_i(X_i)\|_{2+\delta} + \big\|\widehat{\mathcal{U}}_i(X_i)\big\|_{2+\delta}\big)\|X_i\|_p^p}{R^p}, \end{aligned}$$

for any $\delta > 0$, and $p \ge 1$. Now, from the growth condition (2.4) for any function in $\mathcal{N}^{2,\gamma,R}_{d,m,1}$, and the growth condition (5.43), we have

$$\big\|\widehat{\mathcal{U}}_i(X_i)\big\|_{2+\delta} + \big\|v_i(X_i)\big\|_{2+\delta} \le\ C(d)\gamma\Big(1+\frac{\|X_i\|^2_{4+2\delta}}{R^2}\Big).$$

Recalling the standard estimate $\|X_i\|_{2p} \le C(1 + \|\mathcal{X}_0\|_{2p})$, $i = 0, \dots, N$, we then have for R large enough, $\gamma/R = O(1)$, and using the fact that $[v_i]_{L,R} \le C(d)\gamma/R$ from *Step 1*:

$$\varepsilon_i^{r,\gamma,R} \le C(d, \mathcal{X}_0, p)\Big\{\Big(R^{-2/(d+3)}\log(R) + Rm^{-1/2}\Big)^2 + \frac{1}{R^{2p-2}}\Big\}, \tag{5.44}$$

for some constant $C(d, \mathcal{X}_0)$ independent of N, R, m, and this holds for any $p \ge 1$. Since g is Lipschitz under Assumption (4.2), we notice that estimate (5.44) on the network approximation error also holds for $i = N$ by taking $\gamma/R = [g]_L$. Plugging this estimate into (4.4), and recalling that $\mathbb{E}|g(\mathcal{X}_T) - g(X_N)|^2 + \varepsilon^Z(\pi) = O(|\pi|) = O(1/N)$ when g is Lipschitz, we get

$$\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i(X_i)\big|^2 \le C(d, \mathcal{X}_0)\Big[\frac{1}{N} + N^2\Big(R^{-2/(d+3)}\log(R) + Rm^{-1/2}\Big)^2 + \frac{N^2}{R^{2p-2}}\Big].$$

Therefore, to achieve a rate of convergence of order $1/N$ for $\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i(X_i)\big|^2$, it suffices to choose R and m s.t.

$$N^2 R^{-4/(d+3)}|\log(R)|^2 = O(1/N), \quad \text{and} \quad N^2R^2m^{-1} = O(1/N),$$

hence for example with $R = O\big(N^{\frac{3(d+3)}{4}+\epsilon}\big)$, and $m = O\big(N^{\frac{3(d+5)}{2}+2\epsilon}\big)$, for any $\epsilon > 0$, and then take $p = (d+5)/(d+3)$ so that $N^2/R^{2p-2} = O(1/N)$. □

6 Numerical study

We test our different algorithms and the cited ones in this paper on various examples and by varying the state space dimension. If not stated otherwise, we choose the maturity $T = 1$. In each example we use tanh as activation function, and an architecture composed of 2 hidden layers with $d + 10$ neurons. We apply Adam gradient descent [KB14] with a decreasing learning rate, using the Tensorflow library [Aba+16]. Each numerical experiment is conducted using a node composed of 2 Intel® Xeon® Gold 5122 Processors, 192 Go of RAM, and 2 GPU nVidia® Tesla® V100 16Go. We use a batch size of 1000.

6.1 Semilinear PDEs

We first consider examples from [HPW20] to compare its Deep Backward Dynamic Programming methods with the Deep Splitting and the Deep Multistep methods, and then examples coming from the well-known viscous Burgers equation. The three first lines of the tables below are taken from [HPW20]. For each test, the two best results are highlighted in boldface. We use 5000 gradient descent iterations by time step except 20000 for the projection of the final condition. The execution of the multistep algorithm approximately takes between 8000 s. and 16000 s. (depending on the dimension) for a resolution with $N = 120$.

6.1.1 PDE with bounded solution and simple structure

We take the parameters

$$\begin{cases} \mu = \frac{0.2}{d},\ \sigma = \frac{I_d}{\sqrt{d}},\ \overline{x} = \sum_{i=1}^d x_i \\ f(x,y,z) = \Big(\cos(\overline{x})\Big(e^{\frac{T-t}{2}} + \frac{1}{2}\Big) + 0.2\sin(\overline{x})\Big)e^{\frac{T-t}{2}} - \frac{1}{2}(\sin(\overline{x})\cos(\overline{x})e^{T-t})^2 \\ 1 + \frac{1}{2d}(y(1_d \cdot z))^2. \end{cases} \tag{6.1}$$

so that the PDE solution is given by

$$u(t,x) = \cos(\overline{x})\exp\left(\frac{T-t}{2}\right).$$

We first provide a test with a larger maturity $T = 2$, in dimension $d = 1$. The results are reported in Figure 1, while Figure 2 (resp. Figure 3) plots the graphs of the neural network approximations with the Deep splitting (resp. Multistep DBDP) scheme. It is observed in this example that the multistep scheme gives similar precision as the DBDP scheme and outperforms the DS scheme, while the DBSDE scheme is not convergent due to the high number $N = 240$ of time steps.

	Averaged value	Standard deviation	Relative error (%)
[HPW20] (DBDP1)	**1.4633**	**0.0143**	**0.36**
[HPW20] (DBDP2)	1.4388	**0.0135**	2.04
[HJE17] (DBSDE)	NC	NC	NC
[Bec+19] (DS)	1.4968	0.0367	1.91
MDBDP	**1.4626**	0.020	**0.38**

Figure 1: Estimate of $u(0, x_0)$ in the case (6.1), where $d = 1, x_0 = 1, T = 2$ with $N = 240$ time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 1.4686938.

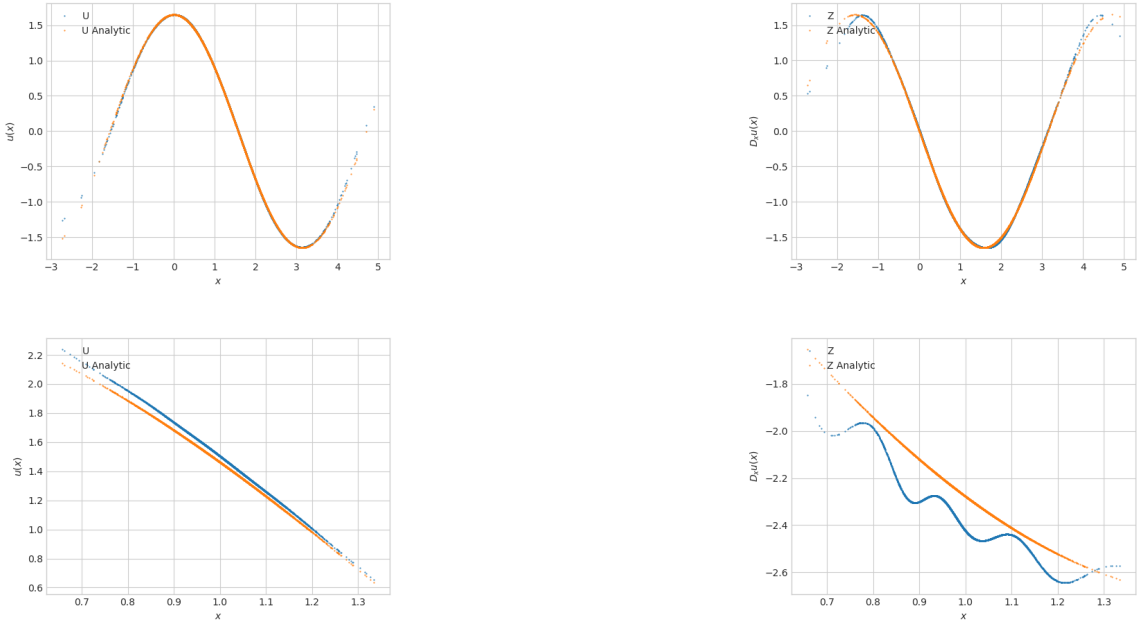

Figure 2: Estimates of u and $D_x u$ using DS scheme in the case (6.1), $T = 2$. We take $d = 1$, $x_0 = 1$, at the top $t = 1.$, and at the bottom $t = 0.0083$.

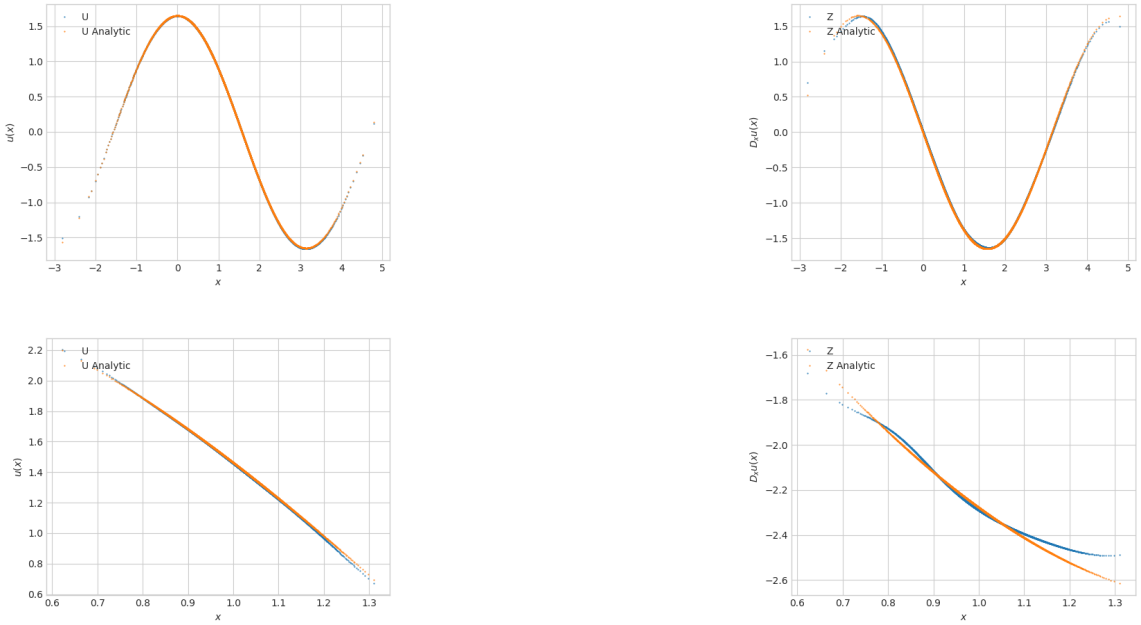

Figure 3: Estimates of u and $D_x u$ in the case (6.1) using MDBDP scheme, $T = 2$. We take $d = 1$, $x_0 = 1$, at the top $t = 1$, and at the bottom $t = 0.0083$.

Next, we fix $T = 1$, and increase the dimension d. The results are reported in Figure 4 for $d = 5$, in Figure 5 for $d = 10$, in Figure 6 for $d = 20$, and in Figure 7 for $d = 50$. It is observed that all the schemes DBDP, DBSDE and MDBDP provide quite accurate results with comparable precision, and largely outperforms the DS scheme.

	Averaged value	Standard deviation	Relative error (%)
[HPW20] (DBDP1)	0.4637	0.0043	0.85
[HPW20] (DBDP2)	0.4634	**0.0014**	0.92
[HJE17] (DBSDE)	**0.4656**	0.0035	**0.44**
[Bec+19] (DS)	0.4790	0.0169	2.42
MDBDP	**0.4649**	**0.0006**	**0.59**

Figure 4: Estimate of $u(0, x_0)$ in the case (6.1), where $d = 5, x_0 = 1\ \mathbb{1}_5, T = 1$ with 120 time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 0.46768.

	Averaged value	Standard deviation	Relative error (%)
[HPW20] (DBDP1)	- 1.3895	0.0015	0.44
[HPW20] (DBDP2)	- 1.3913	**0.0006**	0.57
[HJE17] (DBSDE)	**- 1.3880**	0.0016	**0.33**
[Bec+19] (DS)	- 1.4097	0.0173	1.90
MDBDP	**-1.3887**	**0.0006**	**0.38**

Figure 5: Estimate of $u(0, x_0)$ in the case (6.1), where $d = 10, x_0 = 1\ \mathbb{1}_{10}, T = 1$ with 120 time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is -1.383395.

	Averaged value	Standard deviation	Relative error (%)
[HPW20] (DBDP1)	0.6760	0.0027	0.47
[HPW20] (DBDP2)	**0.6710**	0.0056	**0.27**
[HJE17] (DBSDE)	0.6869	**0.0024**	2.09
[Bec+19] (DS)	0.6944	0.0201	3.21
MDBDP	**0.6744**	**0.0005**	**0.24**

Figure 6: Estimate of $u(0, x_0)$ in the case (6.1), where $d = 20, x_0 = 1\ \mathbb{1}_{20}, T = 1$ with 120 time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 0.6728135.

	Averaged value	Standard deviation	Relative error (%)
[HPW20] (DBDP1)	**1.5903**	**0.0063**	**0.04**
[HPW20] (DBDP2)	1.5876	0.0068	0.21
[HJE17] (DBSDE)	1.5830	0.0361	0.50
[Bec+19] (DS)	1.6485	0.0140	3.62
MDBDP	**1.5924**	**0.0005**	**0.09**

Figure 7: Estimate of $u(0, x_0)$ in the case (6.1), where $d = 50, x_0 = 1\ \mathbb{1}_{50}, T = 1$ with 120 time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 1.5909.

6.1.2 PDE with unbounded solution and more complex structure

We take the parameters

$$\begin{cases} \mu = 0, \sigma = \frac{I_d}{\sqrt{d}} \\ f(x, y, z) = k(x) + \frac{y}{\sqrt{d}}(1_d \cdot z) + \frac{y^2}{2} \end{cases} \tag{6.2}$$

with a function k so that the PDE solution is given by

$$u(t, x) = \frac{T - t}{d} \sum_{i=1}^{d} (\sin(x_i) 1_{x_i < 0} + x_i 1_{x_i \geq 0}) + \cos\Big(\sum_{i=1}^{d} i x_i \Big).$$

We start with tests in dimension $d = 1$. The results are reported in Figure 8, and graphs of the neural network approximations with the Deep splitting (resp. Multistep DBDP) scheme are plotted in Figure 9 (resp.

Figure 10). Similar conclusion can be drawn as in the previous section with bounded solution and simple structure.

	Averaged value	Standard deviation	Relative error (%)
[HPW20] (DBDP1)	1.3720	0.0030	0.41
[HPW20] (DBDP2)	**1.3736**	0.0022	**0.29**
[HJE17] (DBSDE)	1.3724	**0.0005**	0.38
[Bec+19] (DS)	1.3630	0.0079	1.06
MDBDP	**1.3735**	**0.0003**	**0.30**

Figure 8: Estimate of $u(0, x_0)$ in the case (6.2), where $d = 1, x_0 = 0.5$ with 120 time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 1.3776.

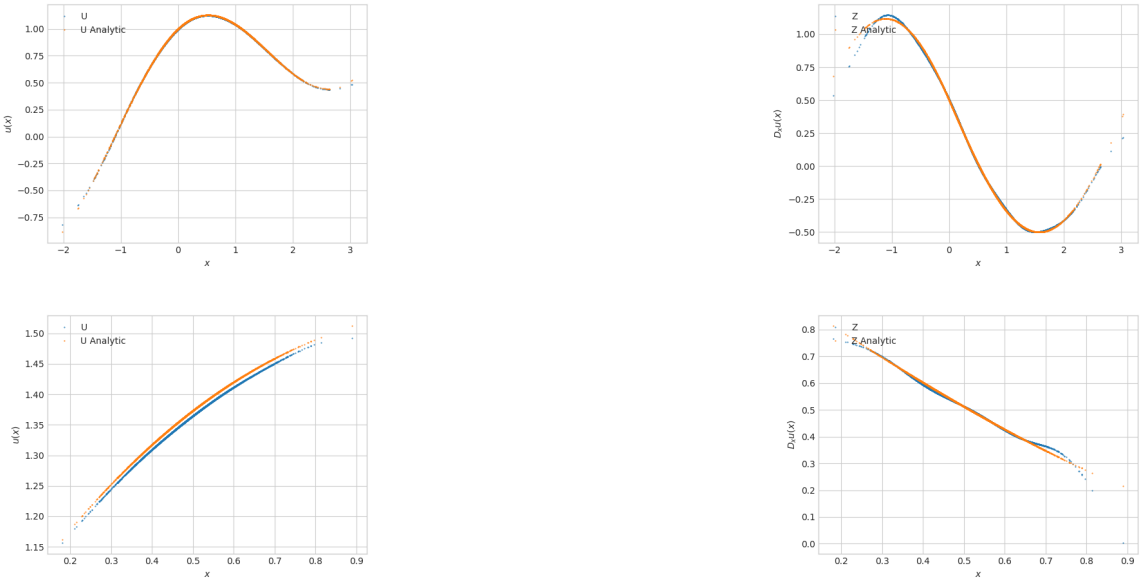

Figure 9: Estimates of u and $D_x u$ using DS scheme in the case (6.2). We take $d = 1$, $x_0 = 0.5$, at the top $t = 0.5$, and at the bottom $t = 0.0083$.

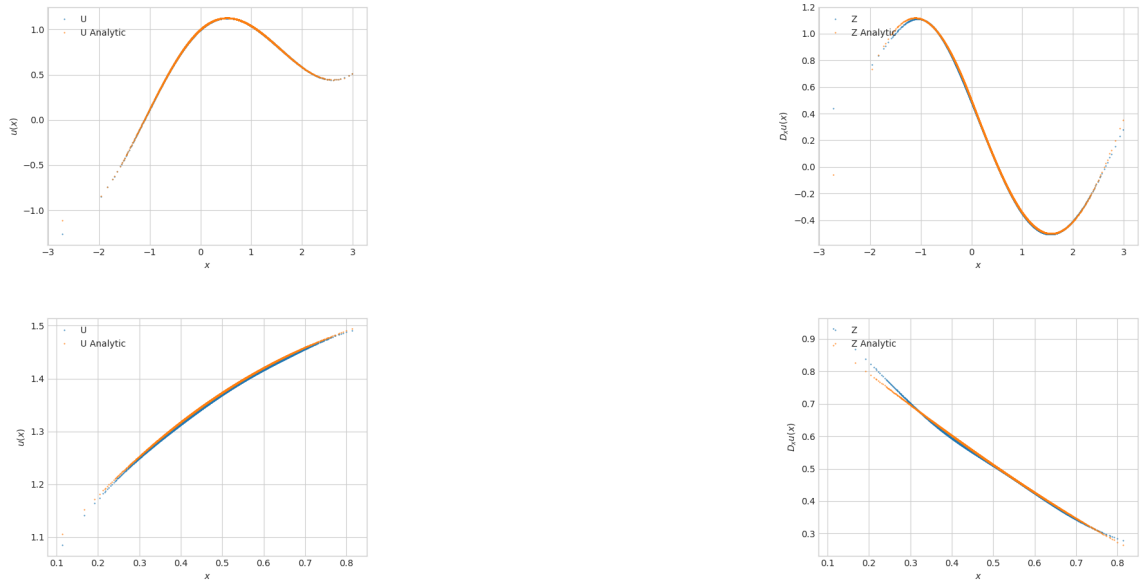

Figure 10: Estimates of u and $D_x u$ in the case (6.2) using MDBDP scheme. We take $d = 1$, $x_0 = 0.5$, at the top $t = 0.5$, and at the bottom $t = 0.0083$.

We next increase the dimension up to $d = 8$, and report the results in the following figures. The accuracy is not so good as in the previous section with simple structure of the solution, but we notice that the MDBDP scheme yields the best performance in dimension $d = 8$ (above dimension $d = 10$, all the schemes do not give good approximation results).

	Averaged value	Standard deviation	Relative error (%)
[HPW20] (DBDP1)	**0.5715**	0.0038	**0.14**
[HPW20] (DBDP2)	**0.5708**	0.0024	**0.02**
[HJE17] (DBSDE)	**0.5715**	**0.0006**	**0.14**
[Bec+19] (DS)	0.5680	0.0124	0.47
MDBDP	0.5721	**0.0005**	0.37

Figure 11: Estimate of $u(0, x_0)$ in the case (6.2), where $d = 2, x_0 = 0.5\ \mathbb{1}_2$ with 120 time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 0.5707.

	Averaged value	Standard deviation	Relative error (%)
[HPW20] (DBDP1)	**0.8666**	0.0130	**1.21**
[HPW20] (DBDP2)	0.8365	**0.0045**	4.64
[HJE17] (DBSDE)	NC	NC	NC
[Bec+19] (DS)	0.6846	0.0576	21.96
MDBDP	**0.8675**	**0.0008**	**1.08**

Figure 12: Estimate of $u(0, x_0)$ in the case (6.2), where $d = 5, x_0 = 0.5\ \mathbb{1}_5$ with 120 time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 0.8772.

	Averaged value	Standard deviation	Relative error (%)
[HPW20] (DBDP1)	**1.1694**	0.0254	**0.78**
[HPW20] (DBDP2)	1.0758	**0.0078**	7.28
[HJE17] (DBSDE)	NC	NC	NC
[Bec+19] (DS)	1.2283	**0.0113**	5.86
MDBDP	**1.1654**	0.0379	**0.47**

Figure 13: Estimate of $u(0, x_0)$ in the case (6.2), where $d = 8, x_0 = 0.5\ \mathbb{1}_8$ with 120 time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 1.1603.

6.1.3 Viscous Burgers equation

The basic viscous Burgers' equation is a semilinear PDE in dimension one written as

$$\begin{cases} \partial_t u + \frac{\sigma^2}{2}\partial^2_{xx} u \;=\; u\partial_x u, \quad \text{on } [0,T)\times\mathbb{R} \\ u(T,.) \;=\; g. \end{cases} \tag{6.3}$$

An (quasi)-explicit solution, obtained by Hopf-Cole transformation, is given by:

$$u(t,x) = \; -\sigma^2 \partial_x \log\Big[\frac{1}{2\pi\sigma^2(T-t)^2}\int_{\mathbb{R}} \exp\Big(-\frac{(x-x')^2}{2\sigma^2(T-t)} - \frac{1}{\sigma^2}\int_0^{x'} g(x'')\,\mathrm{d}x''\Big)\,\mathrm{d}x'\Big].$$

We take the terminal condition $g(x) = \cos(x)$ so that an analytic expression of the solution is

$$u(t,x) = \; \frac{\int_{\mathbb{R}} \frac{x-x'}{T-t}\exp\Big(-\frac{(x-x')^2}{2\sigma^2(T-t)} - \frac{\sin(x')}{\sigma^2}\Big)\,\mathrm{d}x'}{\int_{\mathbb{R}} \exp\Big(-\frac{(x-x')^2}{2\sigma^2(T-t)} - \frac{\sin(x')}{\sigma^2}\Big)\,\mathrm{d}x'}.$$

This analytic function is estimated through Monte-Carlo simulation, and we report the results of the different schemes in Figure 14, while Figure 15 (resp. Figure 16) plots the graphs of the neural network approximations with the Deep splitting (resp. Multistep DBDP) scheme. All the schemes provide very good results both in terms of relative error and standard deviation, and it is observed that the DS scheme achieves the smallest relative error.

	Averaged value	Standard deviation	Relative error (%)
[HPW20] (DBDP1)	0.43489	0.00174	0.78
[HPW20] (DBDP2)	0.43432	0.00154	0.65
[HJE17] (DBSDE)	**0.43356**	**0.00010**	**0.48**
[Bec+19] (DS)	**0.43040**	0.00296	**0.26**
MDBDP	0.43501	**0.00083**	0.81

Figure 14: Estimate of $u(0, x_0)$ in the case of Burgers equation (6.3), where $d = 1, x_0 = 1$. with 120 time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution, estimated with Monte-Carlo simulation (10^6 samples) is 0.4315.

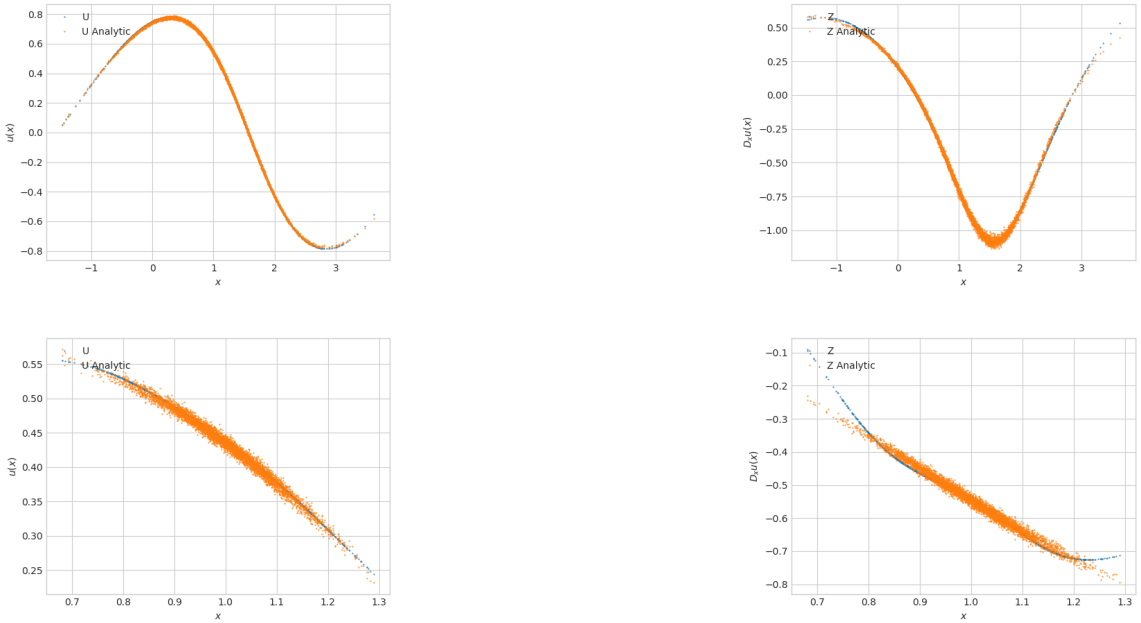

Figure 15: Estimates of u and $D_x u$ in the case of Burgers equation (6.3) using DS scheme. We take $d = 1$, $x_0 = 1$, at the top $t = 0.5$, and at the bottom $t = 0.0083$.

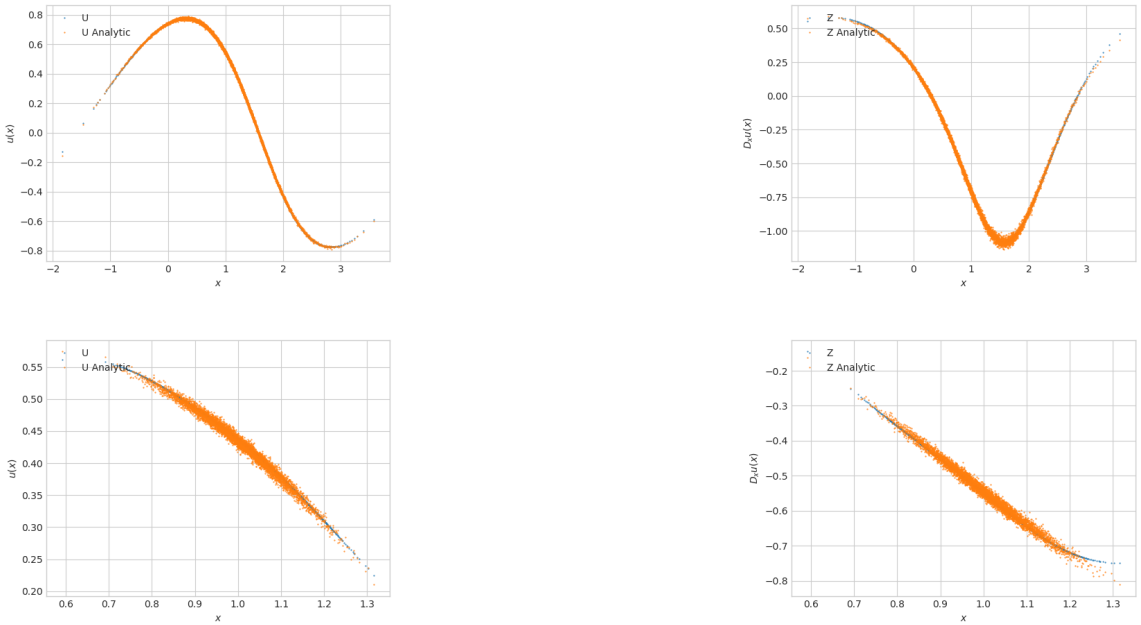

Figure 16: Estimates of u and $D_x u$ in the case of Burgers equation (6.3) using MDBDP. scheme We take $d = 1$, $x_0 = 1$, at the top $t = 0.5$, and at the bottom $t = 0.0083$.

6.2 Fully nonlinear PDEs

We consider examples from [PWG19] that we compare with Algorithms 2 (2EMDBDP), 3 (2MDBDP), and 4 (2M^{2}DBDP) designed in this paper. Notice that some comparison tests with the 2DBSDE scheme [BEJ19] have been already done in [PWG19]. For a resolution with $N = 120, \hat{N} = 30$, the execution of our multitep algorithms takes between 10000 s. and 30000 s. (depending on the dimension) with a number of gradient descent iterations fixed at 4000 at each time step except 80000 at the first one.

6.2.1 PDE with bounded solution and simple structure

We take the parameters

$$\mu = 0,\ \sigma = \frac{I_d}{\sqrt{d}},\ \overline{x} = \sum_{i=1}^{d} x_i$$

and solve the nonlinear PDE

$$\partial_t u = -\frac{1}{2}\cos(\overline{x})\exp\Big(\frac{T-t}{2}\Big) + \cos(\overline{x})^2\exp(T-t) + \frac{u\,\mathrm{Tr}(D_x^2 u)}{d}, \tag{6.4}$$

so that its solution is given by

$$u(t,x) = \cos(\overline{x})\exp\Big(\frac{T-t}{2}\Big).$$

For the time discretization we take $N = 120$, $\hat{N} = 30$. We first start in dimension $d = 1$ with results given in Figure 17, and graphs of $x \mapsto u$, D_x and $D_x^2 u$ in Figures 20, 21, and 22 for the different algorithms.

	Averaged value	Standard deviation	Relative error (%)
[PWG19]	1.53606	0.02795	4.59
2EMDBDP	1.61976	0.22369	10.29
2MDBDP	**1.49930**	**0.00561**	**2.08**
2M^{2}DBDP	1.50165	0.00676	2.24

Figure 17: Estimate of $u(0, x_0)$ in the case (6.4), where $d = 1, x_0 = 1$ with $N = 120$, $\hat{N} = 30$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 1.468693.

Next, we study the impact of the choice of the parameter $\hat{N}$ on the approximation results, and by increasing the dimension d.

	$\hat{N}$	Averaged value	Standard deviation	Relative error (%)
[PWG19]		0.21790	0.09972	71.74
2EMDBDP		0.65697	0.93289	14.8
2MDBDP	12	0.81601	0.00480	5.83
2MDBDP	30	**0.77699**	**0.00571**	**0.76**
2MDBDP	60	0.80684	0.02153	4.64
2M^{2}DBDP	12	**0.77691**	**0.00456**	**0.76**
2M^{2}DBDP	30	0.78722	0.01187	2.09
2M^{2}DBDP	60	0.826206	0.05405	7.15

Figure 18: Estimate of $u(0, x_0)$ in the case (6.4), where $d = 5, x_0 = 1_5$ with $N = 120$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 0.771074.

	Averaged value	Standard deviation	Relative error (%)
[PWG19]	1.94670	0.05342	185.35
2EMDBDP	-1.21034	0.08323	46.93
2MDBDP	-3.50869	0.57553	53.83
2M^{2}DBDP	NC	NC	NC

Figure 19: Estimate of $u(0, x_0)$ in the case (6.4), where $d = 10, x_0 = 1_{10}$ with $N = 120$, $\hat{N} = 30$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is -2.280833.

It is worth mentioning that the 2M^{2}DBDP algorithm also diverges in dimension 10 when $\hat{N} = 40$ or $\hat{N} = 20$. The other algorithms converge but not to the right solution.

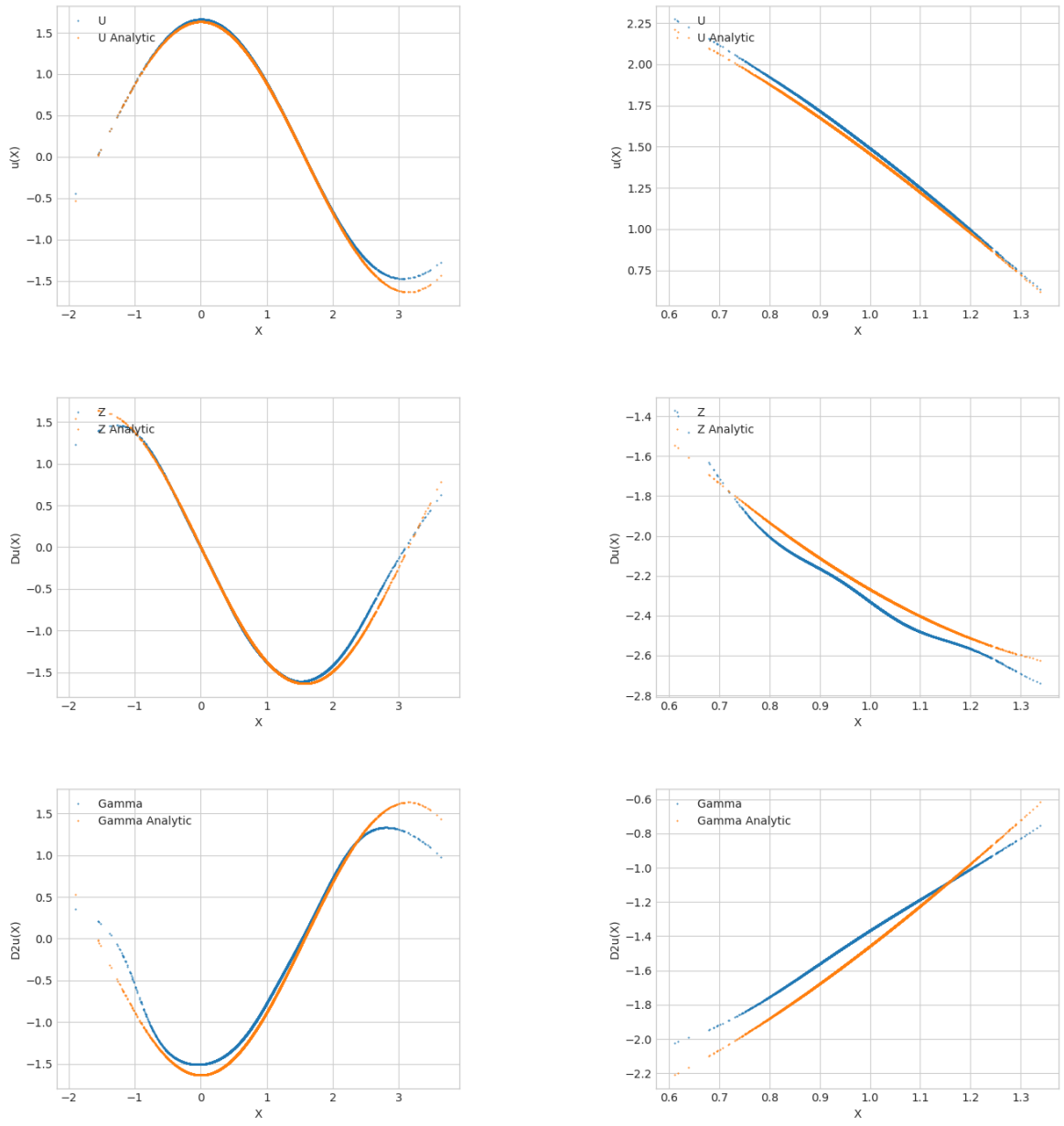

Figure 20: Estimates of u, $D_x u$, $D_x^2 u$ using 2MDBDP in the case (6.4) for $d = 1$ with $N = 120$, $\hat{N} = 30$. We take $x_0 = 1.$, at the left $t = 0.5126$, and at the right $t = 0.0084$.

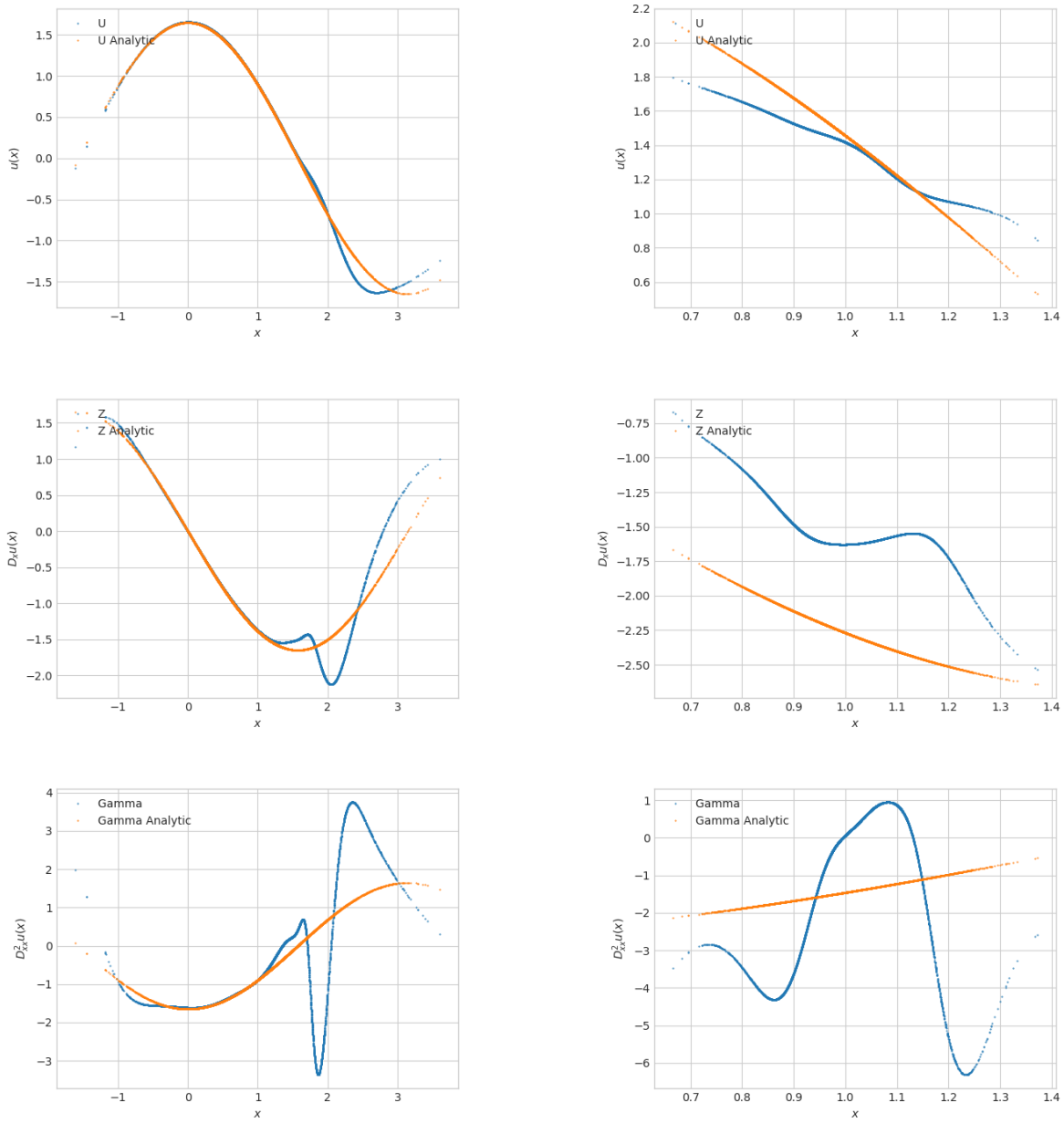

Figure 21: Estimates of u, $D_x u$, $D_x^2 u$ using 2EMDBDP in the case (6.4) for $d = 1$ with $N = 120$. We take $x_0 = 1.$, at the left $t = 0.5042$, and at the right $t = 0.0084$.

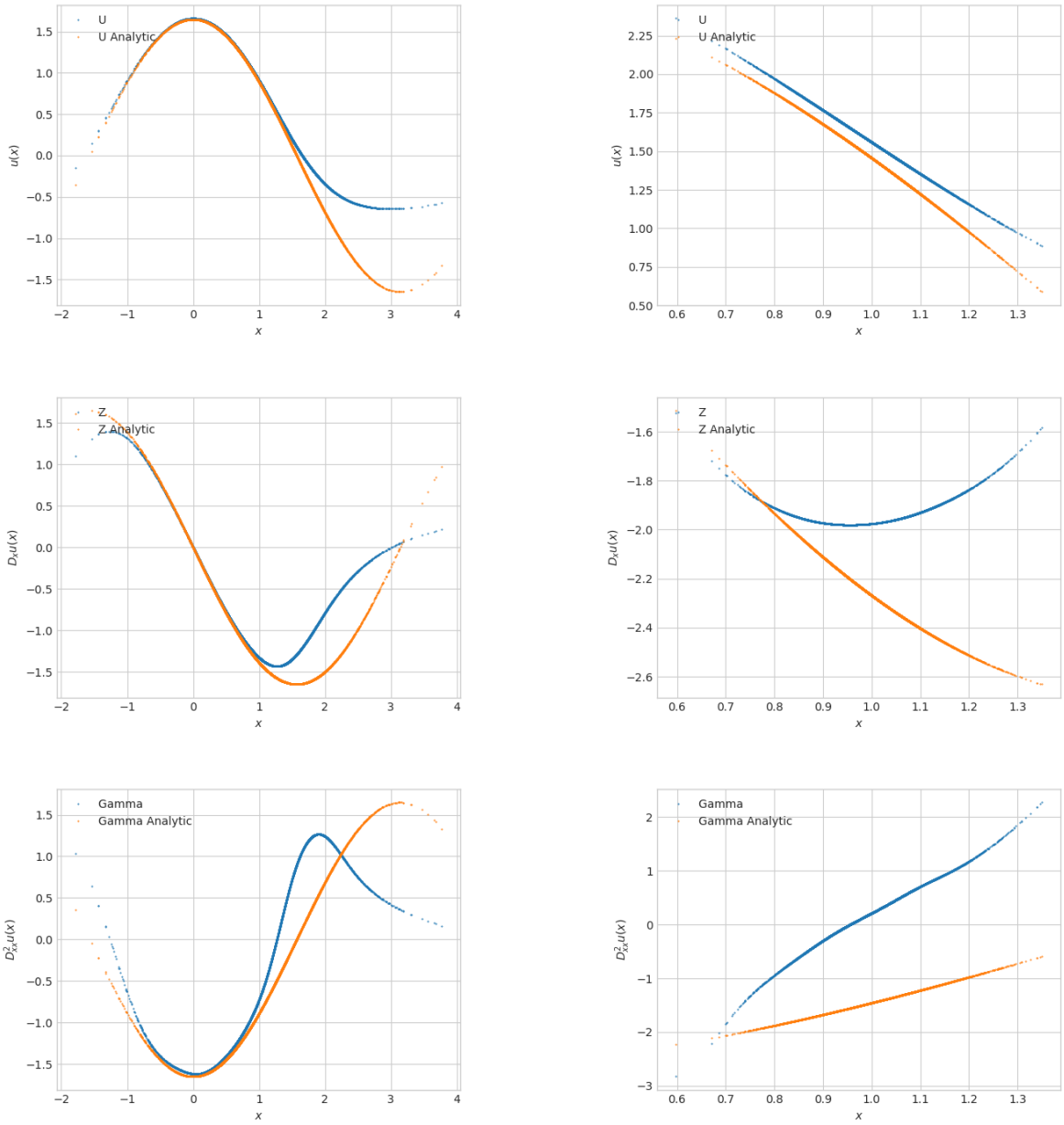

Figure 22: Estimates of u, $D_x u$, $D_x^2 u$ using [PWG19] in the case (6.4) for $d = 1$ with $N = 120$. We take $x_0 = 1.$, at the left $t = 0.5042$, and at the right $t = 0.0084$.

This example highlights the requirement for a multistep method with adaptive time step for the estimation of the second order derivative as far as both the 2EMDBDP method and the algorithm from [PWG19] do not converge anymore when the dimension increases. Even in dimension one, as shown in figures 21 and 22, instabilities occur in the Hessian estimation, before propagating to the gradient and the solution itself. With $\hat{N}$ smaller than N, this behavior seems corrected for small dimensions but appears anyway when the dimension becomes high.

6.2.2 Monge-Ampère equation

We consider another example of fully nonlinear PDE, namely the parabolic Monge-Ampère equation:

$$\begin{cases} \partial_t u + \det(D_x^2 u) &= f(x), \quad (t,x) \in [0,T] \times \mathbb{R}^d, \\ u(T,x) &= g(x), \end{cases} \tag{6.5}$$

where $\det(D_x^2 u)$ is the determinant of the Hessian matrix $D_x^2 u$.

We test our algorithms by choosing a C^2 function g, then compute $G = \det(D^2 g)$, and set $f := G - 1$. Then,

by construction, the function

$$u(t,x) = \ g(x) + T - t,$$

is solution to the Monge-Ampère (MA) equation (6.5). We choose $g(x) = \cos(\sum_{i=1}^d x_i/\sqrt{d})$. For the numerical implementation, we rewrite the MA equation as

$$\partial_t u + \frac{1}{8}\Delta u = \ F(x, D_x^2 u), \quad \text{ on } [0,T) \times \mathbb{R}^d,$$

(Δu is the Laplacian of u), with $F(x,\Gamma) = \frac{1}{8}\mathrm{Tr}(\Gamma) - \det(\Gamma) + f(x)$, and train with the forward process

$$X_{k+1} = X_0 + \frac{1}{2}W_k, \quad k = 0, \dots, N, \ X_0 = 1_d.$$

	$\hat{N}$	Averaged value	Standard deviation	Relative error (%)
[PWG19]		0.37901	0.00312	0.97
2EMDBDP		0.376989	**0.000272**	1.50
2MDBDP	30	0.391464	0.000448	2.28
2MDBDP	60	**0.383953**	0.000417	**0.32**
2MDBDP	120	0.37984	0.000276	0.75
2M^{2}DBDP	30	NC	NC	NC

Figure 23: Estimate of $u(0, 1_5)$ on the Monge Ampere problem (6.5) ($d = 5$) and $N = 120$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 0.38272712.

	$\hat{N}$	Averaged value	Standard deviation	Relative error (%)
[PWG19]		0.25276	0.00235	1.17
2EMDBDP		**0.25523**	0.000478	**0.20**
2MDBDP	30	0.23622	0.000659	7.64
2MDBDP	60	0.24516	0.004383	4.14
2MDBDP	120	0.25310	**0.00046**	1.03
2M^{2}DBDP	30	NC	NC	NC

Figure 24: Estimate of $u(0, 1_{15})$ on the Monge Ampere problem (6.5) ($d = 15$) and $N = 120$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 0.25575373.

This example shows that for medium dimension $d = 5$, Algorithm 3 improves significantly the approximation result compared to the other algorithms when choosing a suitable adaptive step $\hat{N}$ for the computation of the Hessian, while for higher dimension $d = 15$, the multistep extension of the scheme in [PWG19] already gives quite accurate result.

6.2.3 Portfolio selection

We consider a portfolio selection problem formulated as follows. There are n risky assets of uncorrelated price process $P = (P^1, \dots, P^n)$ with dynamics governed by

$$\mathrm{d}P_t^i = \ P_t^i \sigma(V_t^i)\big[\lambda_i(V_t^i)\mathrm{d}t + \mathrm{d}W_t^i\big], \quad i = 1, \dots, n,$$

where $W = (W^1, \dots, W^n)$ is a n-dimensional Brownian motion, $\lambda = (\lambda^1, \dots, \lambda^n)$ is the market price of risk of the assets, σ is a positive function (e.g. $\sigma(v) = e^v$ corresponding to the Scott model), and $V = (V^1, \dots, V^n)$ is the volatility factor modeled by an Ornstein-Uhlenbeck (O.U.) process

$$\mathrm{d}V_t^i = \ \kappa_i[\theta_i - V_t^i]\mathrm{d}t + \nu_i \mathrm{d}B_t^i, \quad i = 1, \dots, n,$$

with $\kappa_i, \theta_i, \nu_i > 0$, and $B = (B^1, \dots, B^n)$ a n-dimensional Brownian motion, s.t. $d < W^i, B^j > = \delta_{ij}\rho_{ij}dt$, with $\rho_i := \rho_{ii} \in (-1, 1)$. An agent can invest at any time an amount $\alpha_t = (\alpha_t^1, \dots, \alpha_t^n)$ in the stocks, which generates a wealth process $\mathcal{X} = \mathcal{X}^\alpha$ governed by

$$\mathrm{d}\mathcal{X}_t = \ \sum_{i=1}^n \alpha_t^i \sigma(V_t^i)\big[\lambda_i(V_t^i)\mathrm{d}t + \mathrm{d}W_t^i\big].$$

The objective of the agent is to maximize her expected utility from terminal wealth:

$$\mathbb{E}\big[U(\mathcal{X}_T^\alpha)\big] \quad \leftarrow \quad \text{maximize over } \alpha$$

It is well-known that the solution to this problem can be characterized by the dynamic programming method (see e.g. [Pha09]), which leads to the Hamilton-Jacobi-Bellman for the value function on $[0,T)\times\mathbb{R}\times\mathbb{R}^n$:

$$\begin{cases} \partial_t u + \sum_{i=1}^n \big[\kappa_i(\theta_i - v_i)\partial_{v_i} u + \frac{1}{2}\nu_i^2 \partial^2_{v_i} u\big] \;=\; \frac{1}{2}R(v)\frac{(\partial_{\mathrm{x}} u)^2}{\partial^2_{\mathrm{xx}} u} + \sum_{i=1}^n \big[\rho_i\lambda_i(v_i)\nu_i \frac{\partial_{\mathrm{x}} u \partial^2_{\mathrm{x}v_i} u}{\partial^2_{\mathrm{xx}} u} + \frac{1}{2}\rho_i^2\nu_i^2 \frac{(\partial^2_{\mathrm{x}v_i} u)^2}{\partial^2_{\mathrm{xx}} u}\big] \\ u(T,\mathrm{x},v) \;=\; U(\mathrm{x}), \qquad \mathrm{x}\in\mathbb{R},\; v\in\mathbb{R}^n, \end{cases}$$

with a Sharpe ratio $R(v) := |\lambda(v)|^2$, for $v = (v_1,\ldots,v_n) \in (0,\infty)^n$. The optimal portfolio strategy is then given in feedback form by $\alpha_t^* = \hat{a}(t, \mathcal{X}_t^*, V_t)$, where $\hat{a} = (\hat{a}_1,\ldots,\hat{a}_n)$ is given by

$$\hat{a}_i(t,\mathrm{x},v) = \; -\frac{1}{\sigma(v_i)}\Big(\lambda_i(v_i)\frac{\partial_{\mathrm{x}} u}{\partial^2_{\mathrm{xx}} u} + \rho_i\nu_i \frac{\partial^2_{\mathrm{x}v_i} u}{\partial^2_{\mathrm{xx}} u}\Big), \quad (t,\mathrm{x},v=(v_1,\ldots,v_n)) \in [0,T)\times\mathbb{R}\times\mathbb{R}^n,$$

for $i = 1,\ldots,n$.

We shall test this example when the utility function U is of exponential form: $U(x) = -\exp(-\eta x)$, with $\eta > 0$, and under different cases for which explicit solutions are available:

(1) *Merton problem.* This corresponds to a degenerate case where the factor V, hence the volatility σ and the risk premium λ are constant. We rewrite the Bellman equation as

$$\partial_t u + |\lambda|\partial_x u + \frac{1}{2}\partial^2_{xx} u = \; \frac{1}{2}|\lambda|^2\frac{(\partial_x u)^2}{\partial^2_{xx} u} + |\lambda|\partial_x u + \frac{1}{2}\partial^2_{xx} u, \tag{6.6}$$

and train our algorithms with the forward process

$$X_{k+1} = \; X_k + |\lambda|\Delta t_k + \Delta W_k, \quad k = 0,\ldots,N,\; X_0 \;=\; x_0.$$

Recall that the explicit solution is given by

$$u(t,x) = \; e^{-(T-t)\frac{|\lambda|^2}{2}} U(x).$$

(2) *One risky asset:* $n = 1$. In this case, we rewrite the Bellman equation as

$$\begin{aligned} & \partial_t u + \lambda(\theta)\partial_{\mathrm{x}} u + \frac{1}{2}\partial^2_{\mathrm{xx}} u + \frac{1}{2}\nu^2\partial^2_{vv} u \\ = \;& \frac{1}{2}|\lambda(v)|^2\frac{(\partial_{\mathrm{x}} u)^2}{\partial^2_{\mathrm{xx}} u} + \big[\rho\lambda(v)\gamma\frac{\partial_{\mathrm{x}} u\partial^2_{\mathrm{x}v} u}{\partial^2_{\mathrm{xx}} u} + \frac{1}{2}\rho^2\gamma^2\frac{(\partial^2_{\mathrm{x}v} u)^2}{\partial^2_{\mathrm{xx}} u}\big] + \lambda(\theta)\partial_{\mathrm{x}} u + \frac{1}{2}\partial^2_{\mathrm{xx}} u - \kappa(\theta - v)\partial_v u, \end{aligned}$$

for $(t,\mathrm{x},v) \in [0,T)\times\mathbb{R}\times\mathbb{R}$, and train our algorithms with the forward process

$$\begin{aligned} \mathcal{X}_{k+1} &= \; \mathcal{X}_k + \lambda(\theta)\Delta t_k + \Delta W_k, \quad k = 0,\ldots,N-1,\; \mathcal{X}_0 \;=\; \mathrm{x}_0 \\ V_{k+1} &= \; V_k + \nu\Delta B_k, \qquad k = 0,\ldots,N-1,\; V_0 \;=\; \theta, \end{aligned}$$

with $\mathrm{d} < W, B > = \rho\mathrm{d}t$. We test our algorithm with $\lambda(v) = \lambda v$, $\lambda > 0$, for which we have an explicit solution: $u(t,\mathrm{x},v) = U(\mathrm{x})w(t,v)$ with

$$w(t,v) = \; \exp\big(-\phi(t)\frac{v^2}{2} - \psi(t)v - \chi(t)\big), \quad (t,v)\in[0,T]\times\mathbb{R},$$

where (ϕ,ψ,χ) are solutions of the Riccati system of ODEs:

$$\begin{aligned} \dot{\phi} - 2\bar{\kappa}\phi - \nu^2(1-\rho^2)\phi^2 + \lambda^2 &= \; 0, \quad \phi(T) \;=\; 0, \\ \dot{\psi} - (\bar{\kappa} + \nu^2(1-\rho^2)\phi)\psi + \kappa\theta\phi &= \; 0, \quad \psi(T) \;=\; 0, \\ \dot{\chi} + \kappa\theta\psi - \frac{\nu^2}{2}(-\phi + (1-\rho^2)\psi^2) &= \; 0, \quad \chi(T) \;=\; 0, \end{aligned}$$

with $\bar\kappa = \kappa + \rho\nu\lambda$, and explicitly given by (see e.g. Appendix in [SZ99])

$$\begin{aligned}
\phi(t) =&\ \lambda^2 \frac{\sinh(\hat\kappa(T-t))}{\hat\kappa\cosh(\hat\kappa(T-t)) + \bar\kappa\sinh(\hat\kappa(T-t))}\\
\psi(t) =&\ \lambda^2 \frac{\kappa\theta}{\hat\kappa}\frac{\cosh(\hat\kappa(T-t)) - 1}{\hat\kappa\cosh(\hat\kappa(T-t)) + \bar\kappa\sinh(\hat\kappa(T-t))}\\
\chi(t) =&\ \frac{1}{2(1-\rho^2)}\ln\big[\cosh(\hat\kappa(T-t)) + \frac{\bar\kappa}{\hat\kappa}\sinh(\hat\kappa(T-t))\big] - \frac{1}{2(1-\rho^2)}\bar\kappa(T-t)\\
&-\ \lambda^2\frac{(\kappa\theta)^2}{\hat\kappa^2}\Big[\frac{\sinh(\hat\kappa(T-t))}{\hat\kappa\cosh(\hat\kappa(T-t)) + \bar\kappa\sinh(\hat\kappa(T-t))} - (T-t)\Big]\\
&-\ \lambda^2\frac{(\kappa\theta)^2\bar\kappa}{\hat\kappa^3}\frac{\cosh(\hat\kappa(T-t)) - 1}{\hat\kappa\cosh(\hat\kappa(T-t)) + \bar\kappa\sinh(\hat\kappa(T-t))},
\end{aligned}$$

with $\hat\kappa = \sqrt{\kappa^2 + 2\rho\nu\lambda\kappa + \nu^2\lambda^2}$.

(3) *No leverage effect, i.e.,* $\rho_i = 0$, $i = 1,\dots,n$. In this case, we rewrite the Bellman equation as

$$\begin{aligned}
&\partial_t u + \big[\sum_{i=1}^n \lambda_i(\theta)\big]\partial_{\mathrm{x}} u + \frac{1}{2}\partial^2_{\mathrm{xx}} u + \frac{1}{2}\sum_{i=1}^n \nu_i^2\partial^2_{v_i} u\\
=&\ \frac{1}{2}|\lambda(v)|^2\frac{(\partial_{\mathrm{x}} u)^2}{\partial^2_{\mathrm{xx}} u} + \big[\sum_{i=1}^n \lambda_i(\theta)\big]\partial_{\mathrm{x}} u + \frac{1}{2}\partial^2_{\mathrm{xx}} u - \sum_{i=1}^n \kappa_i(\theta_i - v_i)\partial_{v_i} u,
\end{aligned}$$

and train with the forward process

$$\begin{aligned}
\mathcal{X}_{k+1} =&\ \mathcal{X}_k + \sum_{i=1}^n \lambda_i(\theta_i)\Delta t_k + \Delta W_k, \quad k = 0,\dots,N-1,\ \ X_0 \ =\ x_0\\
V^i_{k+1} =&\ V^i_k + \nu_i\Delta B^i_k, \qquad k = 0,\dots,N-1,\ \ V^i_0 \ =\ \theta_i,
\end{aligned}$$

with $< W, B^i >= 0$. We test our algorithm with $\lambda_i(v) = \lambda_i v_i$, $\lambda_i > 0$, $i = 1,\dots,n$, $v = (v_1,\dots,v_n)$, for which we have an explicit solution $u(t,\mathrm{x},v) = U(\mathrm{x})w(t,v)$ with

$$\begin{aligned}
w(t,v) =&\ \exp\Big(-\sum_{i=1}^n\big[\phi_i(t)\frac{v_i^2}{2} + \psi_i(t)v_i + \chi_i(t)\big]\Big), \quad (t,v)\in[0,T]\times\mathbb{R}^n,\\
\phi_i(t) =&\ \lambda_i^2\frac{\sinh(\hat\kappa_i(T-t))}{\kappa_i\sinh(\hat\kappa_i(T-t)) + \hat\kappa_i\cosh(\hat\kappa_i(T-t))}\\
\psi_i(t) =&\ \lambda_i^2\frac{\kappa_i\theta_i}{\hat\kappa_i}\frac{\cosh(\hat\kappa_i(T-t)) - 1}{\kappa_i\sinh(\hat\kappa_i(T-t)) + \hat\kappa_i\cosh(\hat\kappa_i(T-t))}\\
\chi_i(t) =&\ \frac{1}{2}\ln\big[\cosh(\hat\kappa_i(T-t)) + \frac{\kappa_i}{\hat\kappa_i}\sinh(\hat\kappa_i(T-t))\big] - \frac{1}{2}\kappa_i(T-t)\\
&-\ \lambda_i^2\frac{(\kappa_i\theta_i)^2}{\hat\kappa_i^2}\Big[\frac{\sinh(\hat\kappa_i(T-t))}{\hat\kappa_i\cosh(\hat\kappa_i(T-t)) + \kappa_i\sinh(\hat\kappa_i(T-t))} - (T-t)\Big]\\
&-\ \lambda^2\frac{(\kappa_i\theta_i)^2\kappa_i}{\hat\kappa_i^3}\frac{\cosh(\hat\kappa_i(T-t)) - 1}{\hat\kappa_i\cosh(\hat\kappa_i(T-t)) + \kappa_i\sinh(\hat\kappa_i(T-t))},
\end{aligned}$$

with $\hat\kappa_i = \sqrt{\kappa_i^2 + \nu_i^2\lambda_i^2}$.

Merton Problem. We take $\eta = 0.5$, $\lambda = 0.6$, $N = 120$, $\hat N = 30$. We plot in Figure 26 the neural networks approximation of $u, D_x u, D_x^2 u$, and the feedback control $\hat a$ (for one asset) computed from our different algorithms, together with their analytic values (in orange). As also reported in the estimates of Figure 25, the multistep algorithms improve significantly the results obtained in [PWG19], where the estimation of the Hessian is not really accurate (see blue curve in Figure 26).

	Averaged value	Standard deviation	Relative error (%)
[PWG19]	-0.50510	0.00393	0.30
2EMDBDP	**-0.50673**	**0.000193**	**0.022**
2MDBDP	-0.50647	0.000329	0.030
2M^{2}DBDP	-0.50644	0.000219	0.035

Figure 25: Estimate of $u(0,1.)$ in the Merton problem (6.6) with $N = 120$, $\hat N = 30$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is -0.50662.

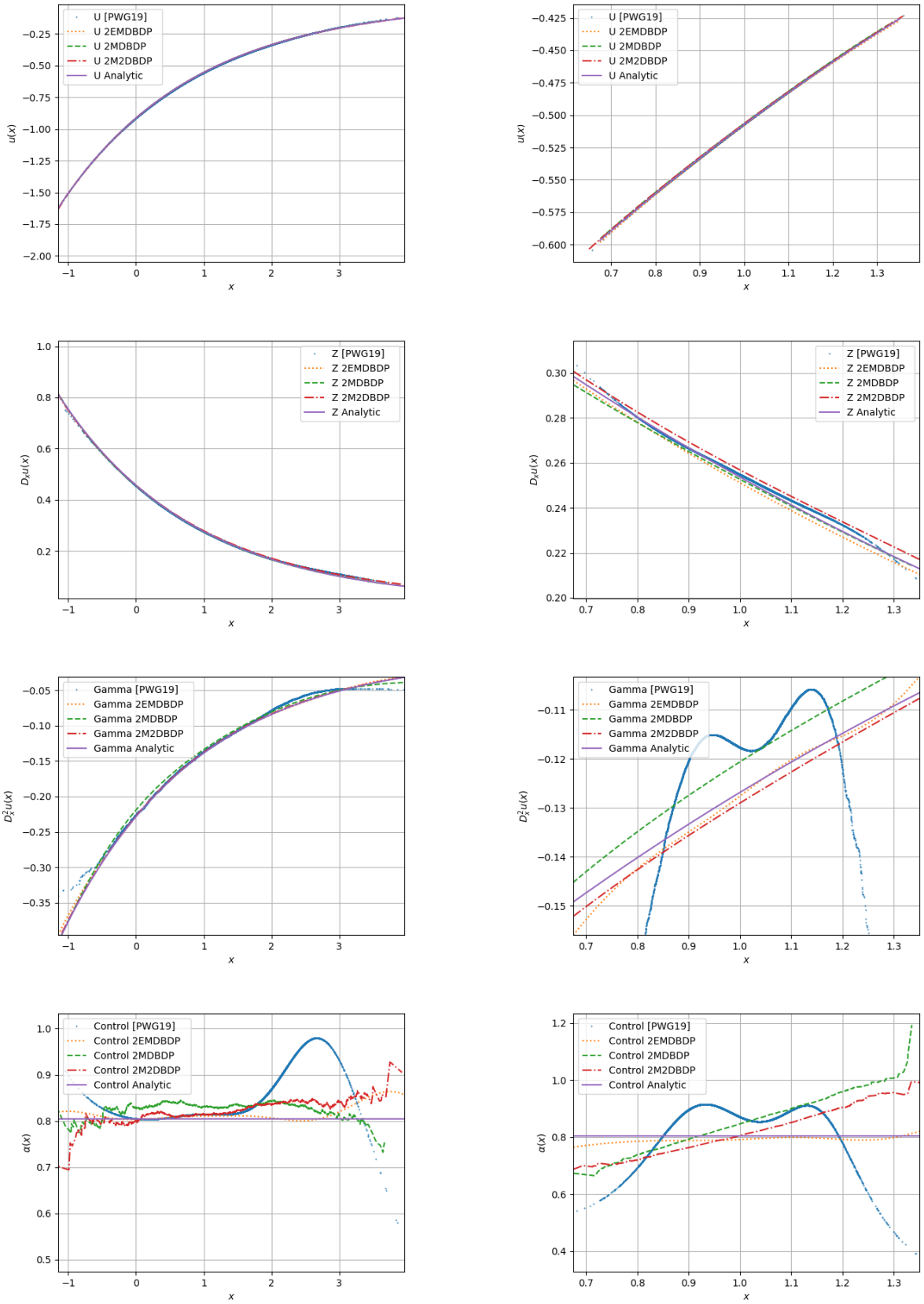

Figure 26: Estimates of u, $D_x u$, $D_x^2 u$ and of the optimal control α on the Merton problem (6.6) with $N = 120$, $\hat{N} = 30$. We take $x_0 = 1.$, at the left $t = 0.5042$, and at the right $t = 0.0084$.

One asset $n = 1$ in Scott volatility model. We take $\eta = 0.5$, $\lambda = 1.5$, $\theta = 0.4$, $\nu = 0.4$, $\kappa = 1$, $\rho = -0.7$. For all tests we choose $N = 120$, $\hat{N} = 30$ and $\sigma(v) = e^v$. We report in Figure 27 the relative error between the neural networks approximation of $u, D_x u, D_x^2 u$ computed from our different algorithms and their analytic

values. It turns out that the multistep extension of [PWG19], namely 2EMDBDP scheme, yields a very accurate approximation result, much better than the other algorithms, with also a reduction of the standard deviation.

	Averaged value	Standard deviation	Relative error (%)
[PWG19]	-0.53327	0.00619	0.53
2EMDBDP	**-0.53613**	**0.000447**	**0.007**
2MDBDP	-0.53772	0.000463	0.304
2M^{2}DBDP	-0.53205	0.000501	0.755

Figure 27: Estimate of $u(0, 1, \theta)$ on the One Asset problem with stochastic volatility ($d = 2$) and $N = 120$, $\hat{N} = 30$. Average and standard deviation observed over 10 independent runs are reported. The exact solution is -0.53609477.

No Leverage in Scott model. In the case with one asset we take $\eta = 0.5$, $\lambda = 1.5$, $\theta = 0.4$, $\nu = 0.2$, $\kappa = 1$. For all tests we choose $N = 120$, $\hat{N} = 30$ and $\sigma(v) = e^v$. We report in Figure 28 the relative error between the neural networks approximation of $u, D_x u, D_x^2 u$ computed from our different algorithms and their analytic values. All the algorithms yield quite accurate results, but compared to the case with correlation in Figure 27, it appears here that the best performance in terms of precision is achieved by Algorithm 2M^{2}DBDP.

	Averaged value	Standard deviation	Relative error (%)
[PWG19]	-0.50160	0.00594	0.007
2EMDBDP	-0.50400	0.00229	0.485
2MDBDP	-0.50149	**0.00024**	0.015
2M^{2}DBDP	**-0.50157**	0.00036	**0.001**

Figure 28: Estimate of $u(0, 1, \theta)$, with 120 time steps on the No Leverage problem (6.7) with 1 asset ($d = 2$) and $N = 120$, $\hat{N} = 30$. Average and standard deviation observed over 10 independent runs are reported. The exact solution is -0.501566.

In the case with four assets we take $\eta = 0.5$, $\lambda = \big(1.5 \quad 1.1 \quad 2. \quad 0.8\big)$, $\theta = \big(0.1 \quad 0.2 \quad 0.3 \quad 0.4\big)$, $\nu = \big(0.2 \quad 0.15 \quad 0.25 \quad 0.31\big)$, $\kappa = \big(1. \quad 0.8 \quad 1.1 \quad 1.3\big)$. The results are reported in Figure 29. We observe that the algorithm in [PWG19] provides a not so accurate outcome, while its multistep version (2EMDBDP scheme) divides by 10 the relative error and the standard deviation.

	Averaged value	Standard deviation	Relative error (%)
[PWG19]	-0.45119	0.00507	2.13
2EMDBDP	**-0.440709**	**0.00051**	**0.239**
2MDBDP	-0.437963	0.00098	0.861
2M^{2}DBDP	-0.448307	0.00566	1.481

Figure 29: Estimate of $u(0, 1, \theta)$, with 120 time steps on the No Leverage problem (6.7) with 4 assets ($d = 5$) and $N = 120$, $\hat{N} = 30$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is -0.44176462.

Finally, in the case with nine assets, we take $\eta = 0.5$, $\lambda = \big(1.5 \quad 1.1 \quad 2. \quad 0.8 \quad 0.5 \quad 1.7 \quad 0.9 \quad 1. \quad 0.9\big)$, $\theta = \big(0.1 \quad 0.2 \quad 0.3 \quad 0.4 \quad 0.25 \quad 0.15 \quad 0.18 \quad 0.08 \quad 0.91\big)$, $\nu = \big(0.2 \quad 0.15 \quad 0.25 \quad 0.31 \quad 0.4 \quad 0.35 \quad 0.22 \quad 0.4 \quad 0.15\big)$, $\kappa = \big(1. \quad 0.8 \quad 1.1 \quad 1.3 \quad 0.95 \quad 0.99 \quad 1.02 \quad 1.06 \quad 1.6\big)$. The results are reported in Figure 30. The approximation is less accurate than in lower dimension, but we observe again that compared to one-step scheme in [PWG19], the multistep versions improve significantly the error with the best performance in precision obtained here by the 2MDBDP scheme.

	$\hat{N}$	Averaged value	Standard deviation	Relative error (%)
[PWG19]		-0.30150	0.03475	9.60
2EMDBDP		-0.266311	**0.00283**	3.19
2MDBDP	30	-0.289786	0.00559	5.34
2MDBDP	60	-0.285491	0.00948	3.78
2MDBDP	120	**-0.283000**	0.01129	**2.87**
2M^{2}DBDP	30	NC	NC	NC

Figure 30: Estimate of $u(0, 1, \theta)$, with 120 time steps on the No Leverage problem with 9 assets ($d = 10$) and $N = 120$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is -0.27509173.

6.3 Results synthesis

From the results of the different algorithms that we have tested on various examples, we make the following remarks. The local machine learning schemes (DBDP, DS and their extensions) are more stable and converge in more cases than the global deep learning scheme (DBSDE) as they are not limited by the number of time steps required in the time discretization. In general, the DBDP schemes yield better approximation results than the DS scheme with the notable exception of the Burgers equation in dimension 1. For unbounded and complex structure of the solution to semilinear PDEs, the multistep version of the DBDP scheme allows to gain in precision and standard deviation at least in medium dimension. When dealing with fully nonlinear PDEs, we observe that multistep schemes improve significantly the approximation error, and permit to achieve accurate results that are not attainable by one-step scheme. Among the three multistep algorithms proposed in this paper, one cannot draw a conclusion about the dominance of one over the others, as the performance depends on the examples and dimension, although it seems that Algorithm 2M^{2}DBDP with Malliavin weights of second order often diverges in high dimension. Anyway, the computational cost of multistep schemes is rather high compared to one-step scheme. Finally, we point out that the choice of the adaptive grid for the computation of the Hessian with Malliavin weights is a delicate point in multistep schemes, and would need a theoretical study that is postponed for future work.

References

[Aba+16] M. Abadi et al. "TensorFlow: A system for large-scale machine learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283. URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[AGL08] M. Akian, S. Gaubert, and A. Lakhoua. "The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis," in: *SIAM Journal on Control and Optimization* 47.2 (2008), pp. 817–848.

[Bac17] F. Bach. "Breaking the Curse of Dimensionality with Convex Neural Networks". In: *Journal of Machine Learning Research* 18.19 (2017), pp. 1–53. URL: http://jmlr.org/papers/v18/14-546.html.

[BD07] C. Bender and R. Denk. "A forward scheme for backward SDEs". In: *Stochastic Processes and their Applications* 117.12 (2007), pp. 1793 –1812. ISSN: 0304-4149. DOI: https://doi.org/10.1016/j.spa.2007.03.005. URL: http://www.sciencedirect.com/science/article/pii/S0304414907000476.

[Bec+19] C. Beck et al. "Deep splitting method for parabolic PDEs". In: *arXiv:1907.03452* (July 2019).

[BEJ19] C. Beck, W. E, and A. Jentzen. "Machine Learning Approximation Algorithms for High-Dimensional Fully Nonlinear Partial Differential Equations and Second-order Backward Stochastic Differential Equations". In: *J. Nonlinear Sci.* 29.4 (Aug. 2019), pp. 1563–1619. ISSN: 1432-1467. DOI: 10.1007/s00332-018-9525-3. URL: https://doi.org/10.1007/s00332-018-9525-3.

[BF11] B. Bercu and J.C. Fort. "Generic stochastic gradient methods". In: *Wiley Encyclopedia of Operations Research and Management Science*. 2011, pp. 1–8.

[BJK19] C. Beck, A. Jentzen, and B. Kuckuck. "Full error analysis for the training of deep neural networks". In: *arXiv:1910.00121v2* (2019).

[BM] F. Bach and E. Moulines. "Non-strongly-convex smooth stochastic approximation with convergence rate O(1/n)." In: *Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS'13*, pp. 773–781.

[BT04] B. Bouchard and N. Touzi. "Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations". In: *Stochastic Process. Appl.* 111.2 (2004), pp. 175 – 206. ISSN: 0304-4149. DOI: https://doi.org/10.1016/j.spa.2004.01.001. URL: http://www.sciencedirect.com/science/article/pii/S0304414904000031.

[CB18] L. Chizat and F. Bach. "On the Global Convergence of Gradient Descent for Over-parameterized Models using Optimal Transport". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2018).

[Che+07] P. Cheridito et al. "Second-order backward stochastic differential equations and fully nonlinear parabolic PDEs". In: *Comm. Pure Appl. Math.* 60.7 (July 2007), pp. 1081–1110. ISSN: 0010-3640. DOI: 10.1002/cpa.20168.

[CWNMW19] Q. Chan-Wai-Nam, J. Mikael, and X. Warin. "Machine Learning for Semi Linear PDEs". In: *J. Sci. Comput.* (Feb. 2019). DOI: 10.1007/s10915-019-00908-3.

[DLM19] J. Darbon, G. Langlois, and T. Meng. "Overcoming the curse of dimensionality for some Hamilton–Jacobi partial differential equations via neural network architectures". In: *arXiv:1910.09045* (Oct. 2019).

[FGN13] X. Feng, R. Glowinski, and M. Nellan. "Recent developments in numerical methods for fully nonlinear second order partial differential equations". In: *SIAM Review* 55.2 (2013), pp. 205–267.

[FTW11] A. Fahim, N. Touzi, and X. Warin. "A probabilistic numerical method for fully nonlinear parabolic PDEs". In: *Ann. Appl. Probab.* 21.4 (Aug. 2011), pp. 1322–1364. DOI: 10.1214/10-AAP723. URL: https://doi.org/10.1214/10-AAP723.

[GLW05] E. Gobet, J-P. Lemor, and X. Warin. "A regression-based Monte Carlo method to solve backward stochastic differential equations". In: *Ann. Appl. Probab.* 15.3 (Aug. 2005), pp. 2172–2202. DOI: 10.1214/105051605000000412. URL: https://doi.org/10.1214/105051605000000412.

[GT14] E. Gobet and P. Turkedjiev. "Linear regression MDP scheme for discrete backward stochastic differential equations under general conditions". In: *Math. Comp.* 85 (Mar. 2014). DOI: 10.1090/mcom/3013.

[GT16] E. Gobet and P. Turkedjiev. "Approximation of backward stochastic differential equations using Malliavin weights and least-squares regression". In: *Bernoulli* 22.1 (Feb. 2016), pp. 530–562. DOI: 10.3150/14-BEJ667. URL: https://doi.org/10.3150/14-BEJ667.

[Gy02] L. Györfi et al. *A distribution-free theory of nonparametric regression*. Springer Series in Statistics, Springer-Verlag, 2002.

[HJE17] J. Han, A. Jentzen, and W. E. "Solving high-dimensional partial differential equations using deep learning". In: *Proc. Natl. Acad. Sci.* 115 (July 2017). DOI: 10.1073/pnas.1718942115.

[HL+19] P. Henry-Labordère et al. "Branching diffusion representation of semilinear PDEs and Monte Carlo approximation". In: *Ann. Inst. H. Poincaré Probab. Statist.* 55.1 (Feb. 2019), pp. 184–210. DOI: 10.1214/17-AIHP880. URL: https://doi.org/10.1214/17-AIHP880.

[HL18] J. Han and J. Long. "Convergence of the Deep BSDE Method for Coupled FBSDEs". In: *arXiv:1811.01165* (Nov. 2018).

[HPW20] C. Huré, H. Pham, and X. Warin. "Deep backward schemes for high-dimensional nonlinear PDEs". In: *Mathematics of Computation* 89.324 (2020), pp. 1547–1580.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural Netw.* 2.5 (July 1989), pp. 359–366. ISSN: 0893-6080.

[HSW90] K. Hornik, M. Stinchcombe, and H. White. "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks". In: *Neural Network* 3(5) (1990), pp. 551–560.

[Hur+18] C. Huré et al. "Deep neural networks algorithms for stochastic control problems on finite horizon, part I: convergence analysis". In: *arXiv:1812.04300* (2018).

[Hut+18] M. Hutzenthaler et al. "Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations". In: *arXiv:1807.01212* (2018).

[KB14] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. 2014. URL: http://arxiv.org/abs/1412.6980.

[LGW06] J-P. Lemor, E. Gobet, and X. Warin. "Rate of convergence of an empirical regression method for solving generalized backward stochastic differential equations". In: *Bernoulli* 12.5 (Oct. 2006), pp. 889–916. DOI: 10.3150/bj/1161614951. URL: https://doi.org/10.3150/bj/1161614951.

[McE07] W. McEneaney. "A curse of dimensionality free numerical method for solution of certain HJB PDEs". In: *SIAM Journal on Control and Optimization* 46.4 (2007), pp. 1239–1276.

[Pha09] H. Pham. *Continuous-time Stochastic Control and Optimization with Financial Applications*. Vol. 61. SMAP. Springer, 2009.

[PP90] E. Pardoux and S. Peng. "Adapted solution of a backward stochastic differential equation". In: *Systems & Control Letters* 14.1 (1990), pp. 55 –61. ISSN: 0167-6911. DOI: https://doi.org/10.1016/0167-6911(90)90082-6. URL: http://www.sciencedirect.com/science/article/pii/0167691190900826.

[PWG19] H. Pham, X. Warin, and M. Germain. "Neural networks-based backward scheme for fully nonlinear PDEs". In: *arXiv:1908.00412* (2019).

[SS17] J. Sirignano and K. Spiliopoulos. "DGM: A deep learning algorithm for solving partial differential equations". In: *J. Computational Phys.* 375 (Aug. 2017). DOI: 10.1016/j.jcp.2018.08.029.

[SZ99] R. Schöbel and J. Zhu. "Stochastic volatility with an Ornstein-Uhlenbeck process and extension". In: *Review of Finance* 3.1 (1999), pp. 23–46.

[Tur15] P. Turkedjiev. "Two algorithms for the discrete time approximation of Markovian backward stochastic differential equations under local conditions". In: *Electron. J. Probab.* 20 (2015), 49 pp. DOI: 10.1214/EJP.v20-3022. URL: https://doi.org/10.1214/EJP.v20-3022.

[VSS18] M. Sabate Vidales, D. Siska, and L. Szpruch. "Unbiased deep solvers for parametric PDEs". In: *arXiv:1810.05094v2* (2018).

[War17] X. Warin. "Variations on branching methods for non linear PDEs". In: *arXiv:1701.07660* (Jan. 2017). DOI: 10.13140/RG.2.2.10311.39846.

[War18] X. Warin. "Nesting Monte Carlo for high-dimensional non-linear PDEs". In: *arXiv:1804.08432* (2018).

[Zha04] J. Zhang. "A numerical scheme for BSDEs". In: *Ann. Appl. Probab.* 14.1 (Feb. 2004), pp. 459–488. DOI: 10.1214/aoap/1075828058. URL: https://doi.org/10.1214/aoap/1075828058.

[E+18] W. E et al. "On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations". In: *to appear in Journal of Scientific Computing* (2018).