Machine learning for stochastic control and partial differential equations in high dimension

Maximilien Germain

Université Paris Cité

École Doctorale de Sciences Mathématiques de Paris Centre ED386
Laboratoire de Probabilités, Statistique et Modélisation

Méthodes d'apprentissage automatique pour la résolution de problèmes de contrôle stochastique et d'équations aux dérivées partielles en grande dimension

---

Machine learning for stochastic control and partial differential equations in high dimension

---

Par Maximilien Germain

Thèse de doctorat de Mathématiques Appliquées

Dirigée par Huyên Pham

Présentée et soutenue publiquement le 20/06/2022

Devant un jury composé de :

Jean-François Chassagneux	Professeur, Université Paris Cité	Examinateur
Arnulf Jentzen	Professeur, University of Münster	Rapporteur
Nadia Oudjane	Ingénieur chercheur, EDF R&D	Examinatrice
Huyên Pham	Professeur, Université Paris Cité	Directeur de thèse
Christoph Reisinger	Professeur, Oxford University	Rapporteur
Agnès Sulem	Directeur de recherche, INRIA Paris	Examinatrice
Xavier Warin	Ingénieur chercheur, EDF R&D	Encadrant industriel, membre invité
Hasnaa Zidani	Professeur, INSA Rouen	Présidente du jury

Titre : Méthodes d'apprentissage automatique pour la résolution de problèmes de contrôle stochastique et d'équations aux dérivées partielles en grande dimension

Résumé : Cette thèse étudie plusieurs schémas numériques d'apprentissage automatique pour la résolution d'Équations aux Dérivées Partielles non-linéaires (EDPs) et du contrôle à champ moyen en dimension modérée ou grande. Elle est divisée en deux parties.

La première partie est consacrée à la résolution d'EDPs paraboliques non-linéaires. Nous décrivons un schéma multistep par réseaux de neurones qui améliore les méthodes existantes et nous étudions son erreur d'approximation ainsi que celle de schémas existants dans le cas semilinéaire où l'équation est linéaire par rapport à la dérivée seconde de la solution. En utilisant des réseaux de neurones lipschitziens de type GroupSort, nous sommes capables de relier l'erreur au nombre de neurones et de couches du réseau utilisé pour l'approximation. Nous développons également des schémas one-step et multistep pour le cas plus délicat des EDPs complétement non-linéaires. Toutes les méthodes sont testées sur des exemples numériques.

La seconde partie de ce travail est dédiée au contrôle à champ moyen et aux équations de McKean-Vlasov. Nous prouvons par des arguments probabilistes une vitesse de convergence pour l'approximation en dimension finie d'une EDP sur l'espace de Wasserstein. Nous utilisons alors des réseaux de neurones symétriques DeepSet pour résoudre des EDPs symétriques en grande dimension. Ainsi nous sommes capables d'approcher la solution de problèmes de contrôle à champ moyen à partir de leurs conditions d'optimalité. Nous considérons ensuite le contrôle à champ moyen avec des contraintes d'état probabilistes. Pour cela, nous représentons le problème par un problème auxiliaire sans contraintes qui peut être résolu par une variante d'un schéma existant d'apprentissage profond.

Mots-clefs : EDPs non-linéaires, réseaux de neurones, contrôle stochastique, contrôle à champ moyen, équation maîtresse, contraintes d'état probabilistes, approximation numérique, DeepSet, GroupSort

Title: Machine learning for stochastic control and partial differential equations in high dimension

Abstract: This thesis studies several machine learning numerical schemes to solve nonlinear PDEs and mean-field control in moderate to high dimension and is divided in two parts.

The first part focuses on the resolution of parabolic nonlinear PDEs. We describe a multistep neural network scheme which improves existing methods from the literature. One of our contributions is the study of its approximation error together with the ones of existing methods in the semilinear case where the equation is linear with respect to the second order derivative. By using Lipschitz GroupSort neural networks, we are able to link the error to the number of layers and neurons of the approximating network. We also develop one-step and multistep schemes in the more challenging case of fully nonlinear PDEs, based on Malliavin weights and automatic differentiation. All the numerical schemes are tested on numerical examples to demonstrate their relevance.

The second part of this work is dedicated to mean-field control and McKean-Vlasov equations. We prove by probabilistic arguments a rate of convergence for the finite dimensional approximation of a PDE on the Wasserstein space. We then use symmetric DeepSet neural networks to solve symmetric PDEs in high dimension. Hence we are able to approximate numerically mean-field control problems by solving their optimality conditions in the form of a Master Bellman PDE in

infinite dimension. We then consider mean-field control with probabilistic state constraints on the law of the controlled state. We represent the problem by an auxiliary unconstrained problem with exact penalisation which can be solved by the modification of an existing brute force deep learning scheme.

Keywords: nonlinear PDEs, neural networks, stochastic control, mean-field control, master equation, probabilistic state constraints, numerical approximation, DeepSet, GroupSort.

List of the six papers being part of this thesis

- H. Pham, X. Warin, and M. Germain. “Neural networks-based backward scheme for fully nonlinear PDEs”. In: *SN Partial Differential Equations and Applications* 2, 16 (2021).
- M. Germain, H. Pham, and X. Warin. “Approximation Error Analysis of Some Deep Backward Schemes for Nonlinear PDEs”. In: *SIAM Journal on Scientific Computing 44.1* (2022), A28–A56.
- M. Germain, M. Laurière, H. Pham, X. Warin. “DeepSets and their derivative networks for solving symmetric PDEs”. In: *Journal of Scientific Computing* 91, 63 (2022).
- M. Germain, H. Pham, and X. Warin. “Neural networks based algorithms for stochastic control and PDEs in finance”, to appear in Machine Learning And Data Sciences For Financial Markets: A Guide To Contemporary Practices. Ed. by A. Capponi and C.A. Lehalle. Cambridge University Press, 2022. Chap. New Frontiers for Stochastic Control in Finance.
- M. Germain, H. Pham, X. Warin. “Rate of convergence for particle approximation of PDEs in Wasserstein space”, to appear in *Journal of Applied Probability* 59.4 (2022).
- M. Germain, H. Pham, and X. Warin. “A level-set approach to the control of state-constrained McKean-Vlasov equations: application to renewable energy storage and portfolio selection". In:arXiv:2112.11059, submitted to *Numerical Algebra, Control and Optimization*, special issue *Stochastic Analysis, Mathematical Finance, and Related Fields.*

Remerciements

Tout d'abord merci à toi Huyên pour ton temps lors de nos nombreuses discussions au tableau ou sur zoom pendant les confinements. Merci pour ta patience et ton aide dans le déroulement de cette thèse. Tu m'as permis d'explorer de très beaux sujets. Je te remercie également Xavier. Tu m'as appris de très nombreuses choses à propos des méthodes numériques probabilistes et tu m'as beaucoup apporté. Merci pour tes nombreuses relectures de mes notes de recherche et ton aide cruciale dans les tests numériques. Vous m'avez tous les deux permis de découvrir et de construire ce très beau sujet de thèse. Je remercie mes rapporteurs Arnulf Jentzen et Christoph Reisinger d'avoir consacré du temps à l'évaluation de mon manuscrit. Vos travaux m'ont particulièrement inspiré. Je souhaite également remercier tous les membres de mon jury, Jean-François Chassagneux, Nadia Oudjane, Agnès Sulem et Hasnaa Zidani pour leur temps et leur implication dans ma soutenance de thèse.

Merci à tous les collègues d'EDF aux côtés de qui j'ai travaillé pendant ces trois années, notamment le meilleur tuteur de stage au monde, Joseph, qui m'a très vite expliqué la signification de tous les sigles désignant les clients de R33, et m'a aussi initié aux joies de Tensorflow. Merci à Edouard et à Clémence qui m'ont permis de réaliser cette thèse dans leur très belle et brillante équipe. Merci à Nadia pour nos échanges pendant le stage et la thèse, mais aussi pour avoir accepté de faire partie de mon jury. Je salue mes illustres co-bureaux Thomas et Matteo. Merci à Nathalie et Amina du laboratoire qui ont toujours été très efficaces pour répondre à mes questions et demandes.

J'ai pu rencontrer de très nombreux doctorants et amis au cours de ce parcours. Je souhaite toutes et tous les remercier, en espérant n'oublier personne. D'abord mes prédécesseurs à EDF : Laura, ma future co-bureau semble-t-il, spécialiste des jeux à champ moyen mais aussi du poney, Carl qui a fait doubler la consommation d'électricité de l'équipe à cause de son PC de gamer aux touches multicolores et Emma en route j'espère pour une future médaille Fields. Ma co-bureau Margaux qui a brillement publié à ICML et passionne ses étudiants de l'ENSTA qui restent jusqu'à 12h05 dans la salle de cours au lieu de se précipiter au Magnan. Les spécialistes de l'optimisation (stochastique?) et des flexibilités du système électrique de demain Adrien et Maxime que je croise parfois en m'égarant dans le couloir de R36. Mes frères de thèse : William qui m'a beaucoup aidé à me préparer lorsque que nous passions de multiples entretiens d'embauches (et futur co-bureau également), Enzo, le roi du contrôle rough, et également Médéric et Côme qui m'ont précédé dans le bureau Serpentard. Guillaume qui a préféré le climat de Chatou à celui de Saclay pour faire sa thèse et qui vient régulièrement en pélérinage sur le platal pour la journée des doctorants. Je remercie mes co-bureaux Junchao, Yiyang, Houzhi mais aussi les voisins des bureaux adjacents : Sylvain que j'ai retrouvé sous le soleil d'Evian et Fabio qui ne manque jamais de nous inviter à déguster de bons petits plats agrémentés d'une bonne bouteille de vin. Je n'oublie pas les petits nouveaux, Mohamed et Nathan, les prochaines stars du contrôle stochastique et des modèles de prix. Merci aux nouveaux co-bureaux Nisrine et Hoang Dung. Je salue les voisins Mohan et Sothea. Je remercie Thomas, le roi du pétrôle, qui malgré ses tentatives répétées n'aura pas réussi à me faire déménager à Londres. Merci à Christophe, Antoine, Quentin et Adèle pour nos réunions café ou celles à l'ambassade autour de gougères et de grands crus. Merci à Grégoire dont je n'ai pas encore testé le jeu vidéo mais promis quand il sera disponible sur Android je m'y mettrai. Je salue mes anciens camarades ENSTA/MVA Omar, Gilles, et Othmane avec qui nous avons enchaîné montagne de projets et TP en tout genre et qui ont à l'unanimité continué en thèse, quelle idée !

Merci à Elena et Tiziano qui m'ont permis de découvrir la recherche en mathématiques. Peut-être qu'un jour notre article sera enfin accepté, qui sait ☺ ? Je remercie également mes enseignants de l'ENSTA, notamment Francesco Russo, Fréderic Jean, Pierre Carpentier, et Hasnaa

Zidani qui m'ont donné envie de commencer cette thèse grâce à leurs excellents cours et leurs conseils vis à vis de mon orientation. Merci aussi à Francesco d'avoir pensé à moi pour donner des travaux dirigés aux élèves de première année de l'ENSTA.

Toute cette aventure doctorale trouve également son origine dans les cours suivis en classes préparatoires au lycée Louis-le-Grand, en particulier ceux de MM. Thouard, Rochet, Logeais, Pommellet, Turiel, et Mme Malaprade. Au lycée Saint-Jean je remercie également MM. Agneray, Delbecq, et MMes Claisse, Pietrowiak qui ont permis à mon goût pour les maths et la physique de s'exprimer. Merci également à M. Borelle et MMes Borelle, Sprimont.

Enfin, je remercie énormément ma famille, mes parents, mes soeurs et mon frère, pour leur soutien et sans qui rien n'aurait été possible. Merci beaucoup à Clémence qui a su me remonter le moral quand j'étais bloqué dans mes tentatives de démonstrations et qui me permet chaque jour d'aller de l'avant. Je crois qu'à cause de moi elle n'a plus envie de faire une thèse.

Contents

Chapter 1

Introduction

This thesis is divided in two parts. The first one studies numerical methods for solving non-linear parabolic Partial Differential Equations (PDEs). These equations typically arise in the context of dynamic programming for stochastic control, which is our main motivation. Our schemes use deep learning in order to approximate PDE solutions. We describe a new multistep scheme for the resolution of semilinear PDEs and realize a convergence analysis of the method and other existing ones from the literature. We show that both the theoretical approximation error and the empirical numerical error are lower in comparison to existing methods. In particular, thanks to recent results on Lipschitz GroupSort neural networks, we are able to relate the scheme error to the network architecture, that is the number of neurons and layers. When the PDEs are fully nonlinear, we describe new schemes, both one-step and multistep ones, able to tackle this more difficult case. We provide numerical tests to demonstrate the relevance of our algorithms.

The second part of the thesis is dedicated to mean-field control. We first prove the rate of convergence of a finite dimensional approximation to an equation on the Wasserstein space of probability measures. We rely on a linearization argument together with a Girsanov change of measure in order to do that. These equations come for instance from the Master Bellman equation of mean-field control. We then use symmetric DeepSets neural networks to solve symmetric PDEs such as the ones coming from the approximation of master Bellman equations. The respect of the problem's symmetry in the scheme itself allows us to solve high dimensional problems. Eventually we consider mean-field control with state constraints. We provide a representation result of the original problem by another one without constraints. This representation allows us to numerically solve this problem.

1.1 Approximation of non-linear PDEs

1.1.1 From stochastic control to PDEs and their numerical resolution

In applied mathematics, especially in financial mathematics, stochastic control is a powerful tool to design efficient strategies for an agent whose dynamics is subject to randomness. Typical examples include among others the quadratic hedging of financial derivatives [Pha00], the hedging of gas storage [War12], or portfolio allocation [ZL00]. Other applications are described and studied in the book [Pha09].

A general form for such problems is given by:

$$\inf_{\alpha} \mathbb{E}\Big[\int_0^T f(s, X_s^\alpha, \alpha_s)\ \mathrm{d}s + g(X_T^\alpha) \Big]$$
$$X_t^\alpha = X_0 + \int_0^t b(s, X_s^\alpha, \alpha_s)\ \mathrm{d}s + \int_0^t \sigma(s, X_s^\alpha, \alpha_s)\ \mathrm{d}W_s,\ t \geq 0$$

where α is some control process with values in $\mathbb{R}^q$, W a d-dimensional Brownian motion, and functions $f : [0,T] \times \mathbb{R}^d \times \mathbb{R}^q \mapsto \mathbb{R}$, $b : [0,T] \times \mathbb{R}^d \times \mathbb{R}^q \mapsto \mathbb{R}^d$, $\sigma : [0,T] \times \mathbb{R}^d \times \mathbb{R}^q \mapsto \mathbb{R}^{d\times d}$. The forward process X is a diffusion process, solution to a controlled Stochastic Differential Equation (SDE). Introducing the value function

$$v(t,x) = \inf_\alpha \mathbb{E}\Big[\int_t^T f(s, X_s^{t,x,\alpha}, \alpha_s)\ \mathrm{d}s + g(X_T^{t,x,\alpha})\Big]$$
$$X_s^{t,x,\alpha} = x + \int_t^s b(u, X_u^{t,x,\alpha}, \alpha_u)\ \mathrm{d}u + \int_t^s \sigma(u, X_u^{t,x,\alpha}, \alpha_u)\ \mathrm{d}W_u,\ s \geq t,$$

it is well known that it solves in the viscosity solution sense the Hamilton-Jacobi-Bellman (HJB) equation

$$\begin{cases} \partial_t v(t,x) + \inf_a \{f(t,x,a) + b(t,x,a)\ D_x v(t,x) + \frac{1}{2}\operatorname{Tr}(\sigma\sigma^\top(t,x,a) D_x^2 v(t,x))\} = 0 \\ v(T,x) = g(x). \end{cases}$$

This parabolic PDE is a special case of fully nonlinear PDE which take the form

$$\begin{cases} \partial_t u(t,x) + F(t,x,u(t,x), D_x u(t,x), D_x^2 u(t,x)) = 0 \\ u(T,x) = g(x), \end{cases} \tag{1.1.1}$$

for a function $F : [0,T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^{d\times d} \mapsto \mathbb{R}$. When the volatility σ is uncontrolled, that is $D_a \sigma = 0$, we can take the volatility part outside of the infimum and the HJB equation becomes linear in the second order derivative $D_x^2 v$:

$$\begin{cases} \partial_t v(t,x) + \frac{1}{2}\operatorname{Tr}(\sigma\sigma^\top(t,x) D_x^2 v(t,x)) + \inf_a\{f(t,x,a) + b(t,x,a)\ D_x v(t,x)\} = 0 \\ v(T,x) = g(x), \end{cases}$$

which is a special case of the semilinear PDE

$$\begin{cases} \partial_t \tilde{u}(t,x) + \mu(t,x)\cdot D_x \tilde{u} + \frac{1}{2}\operatorname{Tr}(\sigma\sigma^\top(t,x) D_x^2 \tilde{u}(t,x)) + \tilde{F}(t,x,\tilde{u}(t,x), \sigma(t,x) D_x \tilde{u}(t,x)) = 0 \\ \tilde{u}(T,x) = g(x), \end{cases} \tag{1.1.2}$$

for a function $\tilde{F} : [0,T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \mapsto \mathbb{R}$. The numerical resolution of this equation is much easier than for the previous one and some of our theoretical results will only apply to this case. This motivates our interest for the resolution of semilinear and fully nonlinear PDEs.

In the semilinear case, the numerical methods we focus on rely on the link between Backward Stochastic Differential Equations (BSDEs) and PDEs to provide numerical methods. With the forward diffusion process X having the generator $\mathcal{L}\phi = \mu(t,x)\cdot D_x\phi(t,x) + \frac{1}{2}\operatorname{Tr}(\sigma\sigma^\top(t,x) D_x^2\phi(t,x))$, that is

$$X_t = \mathcal{X}_0 + \int_0^t \mu(s, X_s)\ \mathrm{d}s + \int_0^t \sigma(s, X_s)\ \mathrm{d}W_s, \tag{1.1.3}$$

the nonlinear Feynman-Kac formula [PP90] links the solution $\tilde{u}$ to (1.1.2) to the adapted solution (Y,Z) to the BSDE:

$$Y_t = g(X_T) + \int_t^T \tilde{F}(s, X_s, Y_s, Z_s)\ \mathrm{d}s - \int_t^T Z_s\ \mathrm{d}W_s, \tag{1.1.4}$$

through $Y_t = \tilde{u}(t, X_t)$ and when $\tilde{u}$ is smooth, $Z_t = \sigma(t,x) D_x \tilde{u}(t, X_t)$. Solving the BSDE amounts to solve the PDE along trajectories of the forward process.

We describe the standard approach to solve this BSDE, that is to generate adapted discrete time processes converging to the discrete time values of the BSDE solutions. Discretizing in time on $t_k := \frac{kT}{N}$ by an Euler scheme for the forward process (1.1.3) and an backward explicit Euler scheme for the backward component (1.1.4) one can define between two consecutive time steps the discrete time processes

$$\begin{aligned} X_{i+1} &= X_i + \mu(t_i, X_i)\ \Delta t + \sigma(t_i, X_i)\ \Delta W_i, \\ Y_{i+1} &= Y_i - \tilde{F}(t_i, X_i, Y_{i+1}, Z_i)\Delta t + Z_i.\Delta W_i,\ i = 0, \cdots, N-1, \end{aligned}$$

with $X_0 = \mathcal{X}_0$, $\Delta t = \frac{T}{N}$ and $\Delta W_i = W_{t_{i+1}} - W_{t_i}$ which yields by adaptedness of Y and Z

$$\begin{cases} Z_i = \mathbb{E}[Y_{i+1}\frac{\Delta W_i}{\Delta t} \mid \mathcal{F}_{t_i}] \\ Y_i = \mathbb{E}[Y_{i+1} + \tilde{F}(t_i, X_i, Y_{i+1}, Z_i)\Delta t \mid \mathcal{F}_{t_i}],\ i = 0, \cdots, N-1. \end{cases} \tag{1.1.5}$$

This representation is the starting point of several numerical backward schemes [Zha04; BT04; GLW05; LGW06] which start from the terminal condition $Y_N = g(X_T)$ and recursively compute Y_i, Z_i by computing the previous conditional expectations. A variant uses an implicit discretization for Y, which requires to solve a fixed point by Picard iteration. By the Markovian structure of the equations, these conditional expectations can be written as measurable functions of the discretized forward process X that is exists functions u_i, z_i such that

$$\begin{cases} Y_i = u_i(X_i) \\ Z_i = z_i(X_i),\ i = 0, \cdots, N-1. \end{cases}$$

The idea is to approximate these functions, which gives at the same time an approximation to the BSDE and the PDE solutions. This is done in a fully implementable way in [GLW05; LGW06] by choosing a function basis $\Psi_1, \cdots, \Psi_n : \mathbb{R}^d \to \mathbb{R}$ and another function basis $\Phi_1, \cdots, \Phi_n : \mathbb{R}^d \to \mathbb{R}^d$ on which the conditional expectation is estimated by regression Monte-Carlo. The method first samples N_s trajectories $(X^k)_{k=1,\cdots,N_s}$ coming from N_s independent Brownian motions $(W^k)_{k=1,\cdots,N_s}$. Then one initializes $Y_N^k = g(X_N^k)$ and, recursively, at time step i, given Y_{i+1}^k one solves the ordinary least squares optimization problems

$$\begin{aligned} &\inf_{\beta_{i,1},\cdots,\beta_{i,n}} \frac{1}{N_s}\sum_{k=1}^{N_s}\Big|\sum_{j=1}^{n}\beta_{i,j}\Phi_j(X_i^k) - Y_{i+1}^k\frac{\Delta W_i^k}{\Delta t}\Big|^2 \\ &\inf_{\alpha_{i,1},\cdots,\alpha_{i,n}} \frac{1}{N_s}\sum_{k=1}^{N_s}\Big|\sum_{j=1}^{n}\alpha_{i,j}\Psi_j(X_i^k) - Y_{i+1}^k - \tilde{F}(t_i, X_i^k, Y_{i+1}^k, Z_i^k)\Delta t\Big|^2, \end{aligned} \tag{1.1.6}$$

where the arginf are respectively called $\beta_{i,1}^*, \cdots, \beta_{i,n}^*$ and $\alpha_{i,1}^*, \cdots, \alpha_{i,n}^*$. In that case Z_i^k is defined by $Z_i^k := \sum_{j=1}^n \beta_{i,j}^*\Phi_j(X_i^k)$ and Y_i^k is given by $Y_i^k := \sum_{j=1}^n \alpha_{i,j}^*\Psi_j(X_i^k)$. Under technical assumptions, it can be proven that $((Y_i)_i, (Z_i)_i)$ converges to (Y, Z) in a suitable sense. This method is quite efficient in small dimension but is limited to dimension 6 or 7 and machine learning methods have been developed in order to solve higher dimensional problems. The usual finite differences methods suffer from the so-called "curse of dimensionality" which prevent us for applying them when the state space is of dimension greater than 4. Indeed too many points are required to discretize the state space which gives a complexity exponentially growing with the dimension d.

1.1.2 Machine learning methods and our proposed schemes

First we need to introduce the neural networks. We define

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

as the set of layer functions with input dimension d_1, output dimension d_2, and activation function $\rho : \mathbb{R}^{d_2} \to \mathbb{R}^{d_2}$. Usually, the activation is applied component-wise via a one-dimensional activation function, i.e., $\rho(x_1, \dots, x_{d_2}) = \big(\hat{\rho}(x_1), \dots, \hat{\rho}(x_{d_2})\big)$ with $\hat{\rho} : \mathbb{R} \mapsto \mathbb{R}$, 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 $\hat{\rho}$ are the sigmoid, the ReLU, the tanh. We then call

$$\mathcal{N}^{\rho}_{d_0,d',\ell,m} = \Big\{\varphi : \mathbb{R}^{d_0} \to \mathbb{R}^{d'} : \exists \phi_0 \in \mathcal{L}^{\rho_0}_{d_0,m_0},\ \exists \phi_i \in \mathcal{L}^{\rho_i}_{m_{i-1},m_i}, i = 1, \dots, \ell - 1,$$
$$\exists \phi_\ell \in \mathcal{L}_{m_{l-1},d'}, \varphi \ = \ \phi_\ell \circ \phi_{\ell-1} \circ \cdots \circ \phi_0\Big\},$$

the set of feedforward neural networks with input layer dimension d_0, output layer dimension d', and ℓ hidden layers with m_i neurons per layer ($i = 0, \cdots, \ell - 1$). These numbers d_0, d', ℓ, the sequence $m = (m_i)_{i=0,\dots,\ell-1}$, and sequence of activation functions $\rho = (\rho_i)_{i=0,\dots,\ell-1}$, form the architecture of the network. It is constructed through the alternative composition of affine maps and a componentwise non-linear activation function.

We shall mostly work with the case $d_0 = d$ (dimension of the state variable x). 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 $\varphi = \varphi^\theta$. Neural networks are usually trained thanks to empirical risk minimization. In the framework of supervised learning, one is given random variables (X, Y) and tries to minimize the risk:

$$\inf_\theta \mathbb{E}[L(\varphi^\theta(X), Y)]$$

with a loss function L quantifying the error between the output $\varphi^\theta(X)$ of the neural network and the label Y. A typical example for L is the quadratic loss function $L : (a, b) \mapsto (a - b)^2$. In practice, the joint law of (X, Y) is unknown and one relies on a finite number N_s of i.i.d. samples $(X_i, Y_i)_{i=1,\cdots,N_s}$ which are used to estimate the expectation by an empirical one:

$$\inf_\theta \frac{1}{N} \sum_{i=1}^{N_s} L(\varphi^\theta(X_i), Y_i),$$

in a Monte-Carlo fashion, as in (1.1.6). Numerically the optimization problem is solved thanks to stochastic gradient descent [RM51] or its variants among which Adam [KB14], Adagrad [DHS11], Adadelta [Zei12]. These optimization methods are implemented in the Tensorflow [Aba+16] and Pytorch [Pas+19] libraries.

Some papers solve PDEs with machine learning but without relying on BSDEs. The Deep Galerkin method [SS18] is a framework able to solve PDEs in any form by looking to a solution in the form of a neural network, sampling point in a domain and trying to enforce the PDE on every sampled point. The same idea is used by physics-informed neural networks [RPK19] which also incorporates data and reconstruct the PDEs solutions by interpolating the data through the PDE dynamics. In the time homogeneous case, the Deep Ritz method of [EY18] solves the variational formulation of elliptic equations thanks to a Deep Galerkin type scheme. Methods using machine learning but not neural networks are also considered in the literature. These schemes have also been designed in order to mitigate the curse of dimensionality, such as sparse grids [Cha+21], nesting Monte-Carlo [War18b], branching [Bou+17; HL+19] and multilevel Picard schemes [E+19; HK20] which are proven to overcome the curse of dimensionality in some cases.

Semilinear case

Let's focus on machine learning methods for semilinear PDEs which rely on BSDEs. The first methods to appear are global methods such as the Deep BSDE method [EHJ17] and then the

Merged Deep BSDE method [CWNMW19]. In that setting one obtains the initial value $u(0, x_0)$ to the PDE solution and a representation of the gradient of the solution by neural networks but not for the solution itself. "Global" refers to the fact that one solves a single large optimization problem taking into account the whole dynamics on $[0, T]$. The Y process is approximated by a forward Euler scheme:

$$Y_{i+1} = Y_i - \tilde{F}(t_i, X_i, Y_{i+1}, Z_i)\Delta t + \mathcal{Z}_i^{\theta_i}(X_i).\Delta W_i,\ i = 0, \cdots, N-1,$$

with $Y_0 = y_0$, a variable y_0 and neural networks $\mathcal{Z}_i^{\theta_i}$. The scheme minimizes the loss

$$\inf_{y_0, \theta_0, \cdots, \theta_N} \mathbb{E}|Y_N - g(X_N)|^2,$$

which is a target problem for the terminal value of Y_T. In that way the method is not part of the supervised learning framework since we don't have targets for the neural networks approximating the Z process. The algorithm is also not an unsupervised algorithm since the coefficients in the dynamics of X and Y are known, which is not the case in the Reinforcement Learning paradigm. The method can be interpreted as a semi-supervised scheme where the labels are implicitly learned through the training.

In [HPW20], the basis function from (1.1.6) is replaced by a neural network and both optimization problems for the solution and its gradient are solved jointly at each time steps. This local method allows us to obtain a functional approximation of the PDE solution at each time step and not only at initial time $t = 0$. Contrarily to the Deep BSDE scheme [EHJ17] several small optimization problems are solved. These optimization problems are close to each other and therefore we can initialize the neural networks parameters to the previously computed ones, which gives a very good starting point because the PDE solution is expected to be continuous in time. Close ideas are used by [Rai18] and in the splitting scheme of [Bec+21]. These schemes successfully solve PDEs in dimension from 10 to 1000 in some cases. The idea of the Deep Backward Dynamic Programming (DBDP) scheme of [HPW20] is to optimize the parameters θ of neural networks $(\mathcal{U}_i^\theta(\cdot), \mathcal{Z}_i^\theta(\cdot)) \in \mathcal{N}_{d,d+1,\ell,m}^\rho$ thanks to the recursive minimization of the backward problems

$$\inf_\theta \mathbb{E}\left|\mathcal{U}_i^\theta(X_i) - \widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \tilde{F}(t_i, X_i, \mathcal{U}_i^\theta(X_i), \mathcal{Z}_i^\theta(X_i))\Delta t + \mathcal{Z}_i^\theta(X_i).\Delta W_i\right|^2,\ i = N-1, \cdots, 0, \tag{1.1.7}$$

where $\widehat{\mathcal{U}}_N(X_N)$ is taken as $g(X_N)$, θ_i^* is the arginf of the previous problem and $\widehat{\mathcal{U}}_{i+1} = \mathcal{U}_{i+1}^{\theta_{i+1}^*}$ (the notations θ^* and $\widehat{\cdot}$ will have the same meaning below). A variant where $\mathcal{Z}_i^\theta$ is replaced by the gradient of $\mathcal{U}_i^\theta$ is also proposed in this paper but yields a bit less accurate results. The Deep Splitting (DS) scheme of [Bec+21] uses this idea and minimize the loss functions

$$\inf_\theta \mathbb{E}\left|\mathcal{U}_i^\theta(X_i) - \widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \tilde{F}(t_i, X_{i+1}, \widehat{\mathcal{U}}_{i+1}(X_{i+1}), \sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1}))\Delta t_i\right|^2,$$

for $i = N-1, \cdots, 0$ with respect to the parameters θ of a single neural network by time step. Similarly the recent paper [NAO21] studies a Malliavin scheme with neural networks. Extensions to more general settings such as the path-dependent case are performed by [RT17; SVSS20; SZ20], whereas linear quadratic stochastic control with control delay is treated by [LM21]. When the objective is to solve a stochastic control problem and not a PDE resolution, the Deep BSDE method can alternatively be used to solve the coupled Forward-Backward Stochastic Differential Equations (FBSDEs) coming from the Pontryagin principle. Contrarily to the previous case where the forward process can be simulated independently of the computation of the BSDE solution (Y, Z), here the dynamics of of X depends on the costate Y. These methods are described for instance in [Ji+20a; Ji+20b].

In this thesis we propose several new schemes and our driving interrogations are:

- How can we improve the existing schemes?
- How do their approximation error behave when the time step vanishes? Can we improve the theoretical error?
- How does the method depend on the neural network architecture?

Multistep DBDP scheme (MDBDP)(see Algorithm 6)
We propose in Chapter 4 a new machine learning multistep method inspired by [GT14] and [HPW20]. It is called Multistep Deep Backward Dynamic Programming (MDBDP). It relies on the following remark. Instead of discretizing the BSDE between time steps t_i and t_{i+1} we can discretize it between t_i and $t_N = T$ and write

$$Y_i = g(X_N) + \sum_{j=i}^{N-1} \tilde{F}(t_j, X_j, Y_j, Z_j)\Delta t - \sum_{j=i}^{N-1} Z_j.\Delta W_j,$$

which can be rewritten by adaptedness of Y and Z as

$$\begin{cases} Y_i = \mathbb{E}[g(X_N) + \sum_{j=i}^{N-1} \tilde{F}(t_j, X_j, Y_j, Z_j)\Delta t - \sum_{j=i}^{N-1} Z_j.\Delta W_j \mid \mathcal{F}_{t_i}] \\ Z_i = \mathbb{E}[(g(X_N) + \sum_{j=i+1}^{N-1} \tilde{F}(t_j, X_j, Y_j, Z_j)\Delta t - \sum_{j=i+1}^{N-1} Z_j.\Delta W_j)\frac{\Delta W_i}{\Delta t} \mid \mathcal{F}_{t_i}],\ i = 0, \cdots, N-1. \end{cases} \tag{1.1.8}$$

In fact, the representations (1.1.5) and (1.1.8) are equal (it can be seen by the tower property of conditional expectation) but because of the numerical approximation of the conditional expectations required for a practical implementation of the scheme, the propagation of the numerical errors will not be the same, see Theorem 1.1.1 and the comments below. Our multistep scheme takes the form of backward iterations as the DBDP scheme (1.1.7) but with the following modified loss function to minimize

$$\begin{aligned} \inf_\theta\ \mathbb{E}\Big|\mathcal{U}_i^\theta(X_i) - g(X_N) - \tilde{F}(t_i, X_i, \mathcal{U}_i^\theta(X_i), \mathcal{Z}_i^\theta(X_i))\Delta t + \mathcal{Z}_i^\theta(X_i)\Delta W_i \\ - \sum_{j=i+1}^{N-1} \tilde{F}(t_j, X_j, \widehat{\mathcal{U}}_j(X_j), \widehat{\mathcal{Z}}_j(X_j))\Delta t + \sum_{j=i+1}^{N-1} \widehat{\mathcal{Z}}_j(X_j).\Delta W_j\Big|^2, \end{aligned} \tag{1.1.9}$$

which uses all the previously optimized neural networks at times steps $j > i$.

Fully nonlinear case

In the fully nonlinear case (1.1.1), an efficient probabilistic scheme is introduced by [FTW11] and it converges locally uniformly to the viscosity solution of the PDE thanks to the monotone scheme framework of [BS91]. Assuming that the PDE solution is smooth, Itô's lemma gives

$$\begin{aligned} Y_t = \ g(X_T) - \int_t^T \big[\mu(s, X_s).Z_s + \frac{1}{2}\mathrm{tr}(\sigma\sigma^\intercal(s, X_s)\Gamma_s) - F(s, X_s, Y_s, Z_s, \Gamma_s)\big]\mathrm{d}s \\ - \int_t^T \sigma^\intercal(s, X_s)Z_s.\mathrm{d}W_s, \quad 0 \le t \le T. \end{aligned}$$

We define $\bar{F}(t, x, u, z, \gamma) := F(t, x, u, z, \gamma) - \mu(t, x).z - \frac{1}{2}\mathrm{tr}(\sigma\sigma^\intercal(s, x)\gamma)$. Contrarily to the semilinear case now we have to estimate also the process corresponding to the second order derivative $\Gamma_t = D_x^2 u(t, X_t)$. Here the coefficients of the forward process X from (1.1.3) can be chosen arbitrarily, contrarily to the semilinear case where the volatility is given in the equation (1.1.2). Indeed, notice that in (1.1.1), the linear part of (1.1.2) does not appear.

A machine learning scheme in this case has been introduced by [BEJ19], relying on the second order BSDEs [Che+07] representation for the solution of fully nonlinear PDEs

$$\begin{cases} Y_t &= g(\mathcal{X}_T) + \int_t^T \bar{F}(s, \mathcal{X}_s, Y_s, Z_s, \Gamma_s)\mathrm{d}s - \int_t^T Z_s.\sigma \mathrm{d}W_s, \\ Z_t &= D_x g(\mathcal{X}_T) - \int_t^T A_s \mathrm{d}s - \int_t^T \Gamma_s \sigma \mathrm{d}W_s, \quad 0 \leq t \leq T, \end{cases}$$

with $A_t = \mathcal{L}D_x u(t, \mathcal{X}_t)$. In particular, in the case of the existence of a classical solution u of class $\mathcal{C}^{1,2}$, Γ_t verifies $\Gamma_t = D_x^2 u(t, X_t)$ and as previously $Y_t = u(t, X_t)$ whereas $Z_t = D_x u(t, X_t)$. The scheme discretizes this system and uses variables to approximate Y_0, Z_0 and neural networks $(\mathcal{A}^\theta, \mathcal{G}^\theta)$ to approximate A, Γ thanks to the minimization of the distance to the terminal conditions $g(X_T), D_x g(X_T)$ of both equations, following the framework of the Deep BSDE scheme.

We propose four alternatives local schemes, by combining ideas from [HPW20; Bec+21] and from our own multistep scheme.

Second order DBDP scheme (2DBDP)(see Algorithm 7)
The first natural idea described in Chapter 5 is an extension of the DBDP scheme with the loss function given by

$$\inf_\theta \mathbb{E}\Big|\mathcal{U}_i^\theta(X_i) - \widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{F}(t_i, X_i, \mathcal{U}_i^\theta(X_i), \mathcal{Z}_i^\theta(X_i), D\widehat{\mathcal{Z}}_{i+1}(\mathcal{T}(X_{t_{i+1}})))\Delta t + \mathcal{Z}_i^\theta(X_i)\Delta W_i\Big|^2,$$

where the second order derivative is estimated by differentiating the gradient obtained at the next time step. We call this method 2DBDP. $\mathcal{T}$ is a truncature at a given quantile made to avoid the propagation of instabilities on the edges of the explored domain. We show that our scheme is able to solve nonlinear PDEs in moderate dimension and gives a better approximation of the control than the Deep 2BSDE scheme.

Thanks to our investigations concerning multistep methods, we also extended this scheme to a multistep setting. We propose in Chapter 3 three variants alongside a survey of machine learning methods for PDEs and control in finance. All the methods consider the loss functions

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

where the definition of $\hat{\Gamma}_{l_j}$ depends on the method. We assume that the drift $\mu = 0$ and that the volatility matrix is a constant invertible matrix σ.

- **Second order Explicit Multistep DBDP scheme (2EMDBDP)**(see Algorithm 3). We combine the multistep scheme and the 2DBDP scheme. If $i = N-1$, define $\widehat{\Gamma}_i = D^2 g$, otherwise $\widehat{\Gamma}_i = D_x \widehat{\mathcal{Z}}_{i+1}$, $\widehat{\Gamma}_j = D_x \widehat{\mathcal{Z}}_j$, $j \in [\![i+1, N-1]\!]$. We also take $l_j = j$.
- **Second order Multistep DBDP (2MDBDP),** (see Algorithm 4). Another method uses Malliavin weights to evaluate this derivative on a subgrid grid $\hat{\pi} = \{t_{\hat{\kappa}\ell}, \ell = 0, \dots, \hat{N}\} \subset \pi$, of modulus $|\hat{\pi}| = \hat{\kappa}|\pi|$, for some $\hat{\kappa} \in \mathbb{N}^*$, with $N = \hat{\kappa}\hat{N}$. Γ_ℓ is obtained by solving

$$\inf_\theta \mathbb{E}\Big|\Gamma_\ell^\theta(X_{\hat{\kappa}\ell}) - \frac{\widehat{\mathcal{Z}}_{\hat{\kappa}(\ell+1)}(X_{\hat{\kappa}(\ell+1)}) - \widehat{\mathcal{Z}}_{\hat{\kappa}(\ell+1)}(\hat{X}_{\hat{\kappa}(\ell+1)})}{2}\hat{H}_\ell^1\Big|^2,$$

with the Malliavin weights

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

and the antithetic variables

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

We also take $l_j = j \div \hat{\kappa} + 1$ where '$\div$' is the symbol for the Euclidian division.

- **Second order Multistep Malliavin DBDP (2M^{2}DBDP)**, (see Algorithm 5). This technique uses second order differentiation of the multistep representation on a subgrid as before thanks to second order Malliavin weights and antithetic variables. We also take $l_j = j \div \hat{\kappa} + 1$.

1.1.3 Lipschitz GroupSort neural networks and contributions to new theoretical results

We provide in Chapter 4 a detailed convergence analysis of both the splitting scheme and our multistep scheme. We focus on the propagation of the discretization and regression errors through the scheme. However we do not consider the statistical error coming from the Monte-Carlo approximation of the expectation in the loss functions such as (1.1.9) nor the optimization error coming from the gradient descent algorithm.

Thanks to recent results on quantitative universal approximation for Lipschitz GroupSort neural networks we obtain explicitly the error in terms of the neural network architecture, that is its number of neurons and layers. For future use, we introduce the L^2-regularity of Z from [Zha04]:

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

We choose sequences $(\gamma_i)_i, (\eta_i)_i$ and consider general approximation classes $\mathcal{N}_i$, $\mathcal{N}_i'$ and $\mathcal{N}_i^{\gamma,\eta}$ respectively from $\mathbb{R}^d \mapsto \mathbb{R}$, $\mathbb{R}^d \mapsto \mathbb{R}^d$, $\mathbb{R}^d \mapsto \mathbb{R}$ (and with γ_i-Lipschitz continuous functions with η_i-Lipschitz continuous gradient for the last one). We define for $i = 0, \dots, N-1$ the L^2-approximation errors in this classes of the functions $v_i^{(1)}, \widehat{z}_i^{(1)}, v_i^{(2)}, v_i^{(3)}, \widehat{z}_i^{(3)}$ defined respectively in (4.3.2), (4.3.4) and (4.3.6):

$$\varepsilon_i^{1,y} := \inf_{\mathcal{U}\in\mathcal{N}_i} \mathbb{E}\big|v_i^{(1)}(X_i) - \mathcal{U}(X_i)\big|^2, \qquad \varepsilon_i^{1,z} := \inf_{\mathcal{Z}\in\mathcal{N}_i'} \mathbb{E}\big|\widehat{z}_i^{(1)}(X_i) - \mathcal{Z}(X_i)\big|_2^2,$$

$$\varepsilon_i^{\gamma,\eta} = \begin{cases} \inf_{\mathcal{U}\in\mathcal{N}_i^{\gamma,\eta}} \mathbb{E}\big|v_i^{(2)}(X_i) - \mathcal{U}(X_i)\big|^2, & i = 0, \dots, N-1, \\ \inf_{\mathcal{U}\in\mathcal{N}_i^{\gamma,\eta}} \mathbb{E}\big|g(X_N) - \mathcal{U}(X_N)\big|^2, & i = N, \end{cases}$$

$$\varepsilon_i^{3,y} := \inf_{\mathcal{U}\in\mathcal{N}_i} \mathbb{E}\big|v_i^{(3)}(X_i) - \mathcal{U}(X_i)\big|^2, \qquad \varepsilon_i^{3,z} := \inf_{\mathcal{Z}\in\mathcal{N}_i'} \mathbb{E}\big|\widehat{z}_i^{(3)}(X_i) - \mathcal{Z}(X_i)\big|_2^2.$$

Our approximation result is the following:

Theorem 1.1.1 (Approximation error of MDBDP)**.** *Under Assumption 4.3.1, there exists a constant $C > 0$ (depending only on the data μ, σ, f, g, d, T) such that in the limit $|\pi| \to 0$*

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

This approximation error of the multistep scheme is better than for the DBDP scheme of [HPW20] where the regression errors term $\sum_{j=0}^{N-1}(\varepsilon_j^{1,y}+\Delta t_j\varepsilon_j^{1,z})$ is replaced by $\sum_{j=0}^{N-1}(N\varepsilon_j^{1,y}+\varepsilon_j^{1,z})$ instead. In particular we answer the point raised by Côme Huré in the Section 1.3.4 of his PhD thesis [Hur19] where he mentioned that the factor N he obtained in front of the sum of errors was unexpected and could not be removed from his proof. In fact the use of a multistep method allows us to remove this term. However, we obtained a similar factor N for the splitting scheme [Bec+21].

Theorem 1.1.2 (Approximation error of DS)**.** *Let Assumption 4.3.1 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 in the limit* $|\pi|\to 0$

$$\sup_{i\in[\![0,N]\!]}\mathbb{E}\big|Y_{t_i}-\widehat{\mathcal{U}}_i^{(2)}(X_i)\big|^2 \le C\Big(\mathbb{E}\big|g(X_N)-g(\mathcal{X}_T)\big|^2+|\pi|+\varepsilon^Z(\pi) + \max_i\big[\gamma_i^2,\eta_i^2\big]|\pi|+\varepsilon_N^{\gamma,\eta}+N\sum_{i=0}^{N-1}\varepsilon_i^{\gamma,\eta}\Big).$$

The first classical approximation results prove that neural networks are dense in function spaces, such as [HSW89; HSW90; Hor91] but no rate of convergence was given. When more regularity is assumed for the target functions, such as boundedness, Lipschitz continuity, convexity or Sobolev regularity, several works later provided explicit approximation results [Pin99a; BGS15; Yar17; Bac17]. We use a more recent approach with Lipschitz neural networks. Thanks to an activation function that divides its input into groups and sorts each one of them (see Figure 1.1), and enforcing bounded weights parameters, the GroupSort network introduced by [ALG19] is 1-Lipschitz. [TSB21] proves explicit approximation results for these networks which allow us to go further into our error analysis by expressing the regression errors as functions of the architecture. As a consequence we can choose the neural networks parameters so that the overall error is equivalent to the time discretization error.

Let $\kappa\in\mathbb{N}^*,\ \kappa\ge 2$, be a grouping size, dividing the number of neurons $m_i=\kappa n_i$, at each layer $i=0,\cdots\ell-1$. $\sum_{i=0}^{\ell-1}m_i$ will be refered to as the width of the network and $\ell+1$ as its depth. The GroupSort networks correspond to classical deep feedforward neural networks in $\mathcal{N}_{d,1,\ell,m}^{\zeta_\kappa}$ with a specific sequence of activation function $\zeta_\kappa=(\zeta_\kappa^i)_{i=0,\dots,\ell-1}$, and one-dimensional output. Each nonlinear function ζ_κ^i divides its input into groups of size κ and sorts each group in decreasing order, see Figure 1.1.

Moreover, by enforcing the parameters of the GroupSort to satisfy with the Euclidian norm $|\cdot|_2$ and the ℓ_∞ norm $|\cdot|_\infty$:

$$\sup_{|x|_2=1}|\mathcal{W}_0x|_\infty\le 1,\quad \sup_{|x|_\infty=1}|\mathcal{W}_ix|_\infty\le 1,\ |\beta_j|_\infty\le M,\ i=1,\cdots,l,\ j=0,\cdots,l,$$

for some $M>0$, the related GroupSort neural networks from $\mathcal{N}_{d,d',\ell,m}^{\zeta_\kappa}$ are 1-Lipschitz. The space of such 1-Lipschitz GroupSort neural networks is called $\mathcal{S}_{d,\ell,m}^{\zeta_\kappa}$:

$$\mathcal{S}_{d,\ell,m}^{\zeta_\kappa}=\{\varphi^{(\mathcal{W}_0,\beta_0,\dots,\mathcal{W}_\ell,\beta_\ell)}\in\mathcal{N}_{d,1,\ell,m}^{\zeta_\kappa},\ \sup_{|x|_2=1}|\mathcal{W}_0x|_\infty\le 1,\ \sup_{|x|_\infty=1}|\mathcal{W}_ix|_\infty\le 1, |\beta_j|_\infty\le M,\ i=1,\cdots,l,\ j=0,\cdots,l\}.$$

We then introduce the set $\mathcal{G}_{K,d,d',\ell,m}^{\zeta_\kappa}$ as

$$\mathcal{G}_{K,d,d',\ell,m}^{\zeta_\kappa}:=\{\Psi=(\Psi_i)_{i=1,\dots,d'}:\mathbb{R}^d\mapsto\mathbb{R}^{d'},\ \Psi_i:x\in\mathbb{R}^d\mapsto K\beta_i\ \ \phi_i\Big(\frac{x+\alpha_i}{\beta_i}\Big)\in\mathbb{R}, \phi_i\in\mathcal{S}_{d,\ell,m}^{\zeta_\kappa},\ \text{ for some }\alpha_i\in\mathbb{R}^d,\ \beta_i>0\}.$$

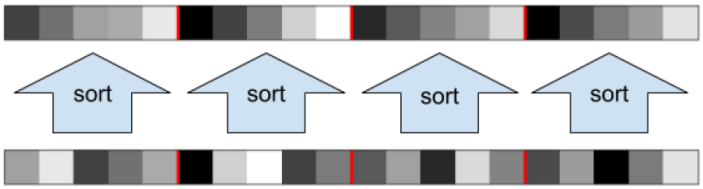


Figure 1.1: GroupSort activation function ζ_κ with grouping size $\kappa = 5$ and $m = 20$ neurons, figure from [ALG19].

Notice that these networks are $\sqrt{d'}K$-Lipschitz and that each of their components is K-Lipschitz. We give the approximation result which is central for our study.

Proposition 1.1.1 (Slight extension of Tanielian, Sangnier, Biau [TSB21] : Approximation theorem for Lipschitz functions by Lipschitz GroupSort neural networks.)**.** *Let $f : [-R, R]^d \mapsto \mathbb{R}^{d'}$ be K-Lipschitz. Then, for all $\varepsilon > 0$, there exists a GroupSort neural network g in $\mathcal{G}^{\zeta_\kappa}_{K,d,d',\ell,m}$ verifying*

$$\sup_{x\in[-R,R]^d} |f(x) - g(x)|_2 \leq \sqrt{d'}2RK\varepsilon,$$

with g of grouping size $\kappa = \lceil \frac{2\sqrt{d}}{\varepsilon} \rceil$, depth $\ell + 1 = O(d^2)$ and width $\sum_{i=0}^{\ell-1} m_i = O((\frac{2\sqrt{d}}{\varepsilon})^{d^2-1})$ in the case $d > 1$. If $d = 1$, the same result holds with g of grouping size $\kappa = \lceil \frac{1}{\varepsilon} \rceil$, depth $\ell + 1 = 3$ and width $\sum_{i=0}^{\ell-1} m_i = O(\frac{1}{\varepsilon})$.

We next study convergence for the approximation error of the MDBDP scheme with GroupSort neural networks and with the additional assumption that the driver $\tilde{F}$ does not depend on z, hence the PDE is linear in z.

Proposition 1.1.2 (Rate of convergence of MDBDP)**.** *Let Assumption 4.3.1 and Assumption 4.3.2 hold, and assume that $\mathcal{X}_0 \in L^{2+\delta}(\mathcal{F}_0, \mathbb{R}^d)$, for some $\delta > 0$, and g is $[g]-$Lipschitz. Then, there exists a bounded sequence K_i (uniformly in i, N) such that for GroupSort neural networks classes $\mathcal{N}_i = \mathcal{G}^{\zeta_\kappa}_{K_i,d,1,\ell,m}$, and $\mathcal{N}'_i = \mathcal{G}^{\zeta_\kappa}_{\sqrt{\frac{d}{\Delta t_i}}K_i,d,d,\ell,m}$, we have*

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

with a grouping size $\kappa = O(2\sqrt{d}N^2)$, depth $\ell+1 = O(d^2)$ and width $\sum_{i=0}^{\ell-1} m_i = O((2\sqrt{d}N^2)^{d^2-1})$ in the case $d > 1$. If $d = 1$, take $\kappa = O(N^2)$, depth $\ell + 1 = 3$ and width $\sum_{i=0}^{\ell-1} m_i = O(N^2)$. Here, the constants in the $O(\cdot)$ term depend only on $\mu, \sigma, \tilde{F}, g, d, T, x_0$.

We are able to perfom the same analysis for the DBDP scheme in the semilinear case where the PDE is nonlinear in z.

Proposition 1.1.3 (Rate of convergence of DBDP)**.** *Let Assumption 4.3.1 hold, and assume that $\mathcal{X}_0 \in L^{2+\delta}(\mathcal{F}_0, \mathbb{R}^d)$, for some $\delta > 0$, and g is $[g]-$Lipschitz. Then, there exists a bounded sequence K_i (uniformly in i, N) such that for $\mathcal{N}_i = \mathcal{G}^{\zeta_\kappa}_{K_i,d,1,\ell,m}$, and $\mathcal{N}'_i = \mathcal{G}^{\zeta_\kappa}_{\sqrt{\frac{d}{\Delta t_i}}K_i,d,d,\ell,m}$, we have*

$$\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}^{(3)}_i(X_i)\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} \big|Z_s - \widehat{\mathcal{Z}}^{(3)}_i(X_i)\big|_2^2 \, \mathrm{d}s\Big] = O(1/N),$$

with a grouping size $\kappa = O(2\sqrt{d}N^3)$, depth $\ell+1 = O(d^2)$ and width $\sum_{i=0}^{\ell-1} m_i = O((2\sqrt{d}N^3)^{d^2-1})$ in the case $d > 1$. If $d = 1$, take $\kappa = O(N^3)$, depth $\ell + 1 = 3$ and width $\sum_{i=0}^{\ell-1} m_i = O(N^3)$. Here, the constants in the $O(\cdot)$ term depend only on $\mu, \sigma, f, g, d, T, \mathcal{X}_0$.

Due to the previously mentioned differences between the error of DBDP and the one of MDBDP we notice that much more neurons are required to obtain a similar error rate. For instance in dimension $d = 1$, to obtain an error of $O(1/N)$ the DBDP scheme requires in theory $O(N^3)$ neurons whereas only $O(N^2)$ are necessary for the MDBDP scheme.

1.2 Mean-field problems and their numerical approximation

1.2.1 Motivation and optimality conditions

Large population games have stimulated a growing interest since the emergence of the mean-field games theory introduced by [LL06a; LL06b] and [HCM06]. This theory considers the limit of an infinite number of similar interacting players and aims at characterizing the resulting equilibria. Two main frameworks are available:

- **Mean-Field Games (MFG).** The MFG theory looks for Nash equilibria, hence consider a competitive interaction between the players.
- **Mean-Field Control (MFC)** (or control of McKean-Vlasov dynamics). Here the framework focuses on collaborative equilibria, with a central planner solving a problem regarding the whole population.

The starting point is a $N-$player stochastic differential game with cost and dynamics for agent i and feedback controls α_t^i

$$\begin{cases} J^i = \mathbb{E}\left[\int_0^T f\left(t, X_t^i, \frac{1}{N}\sum_{k=1}^N \delta_{X_t^k}, \alpha_t^i\right)\,\mathrm{d}t + g\left(X_T^i, \frac{1}{N}\sum_{k=1}^N \delta_{X_T^k}\right)\right] \\ \mathrm{d}X_t^i = b\left(t, X_t^i, \frac{1}{N}\sum_{k=1}^N \delta_{X_t^k}, \alpha_t^i\right)\,\mathrm{d}t + \sigma\left(t, X_t^i, \frac{1}{N}\sum_{k=1}^N \delta_{X_t^k}, \alpha_t^i\right)\,\mathrm{d}W_t^i. \end{cases} \tag{1.2.1}$$

When the number of players N is large, the game becomes difficult to solve and mean-field approximation provide a way to approximately solve the problem by taking the limit $N \to +\infty$. To find an equilibrium, we consider distributed Markovian feedback controls $\alpha_t^i = \alpha_t(X_t^i)$.

To write down the asymptotic MFG problem, we first start by setting a family $(\mu_t)_{t\in[0,T]}$ of probability measures. Then we solve by symmetry only for a representative player:

$$\begin{aligned} \inf_{\alpha_t\in\mathbb{A}} \quad & \mathbb{E}\left[\int_0^T f(t, X_t^\alpha, \mu_t, \alpha_t)\,\mathrm{d}t + g(X_T^\alpha, \mu_T)\right] \\ \text{subject to} \quad & X_t^\alpha = \xi + \int_0^t b(s, X_s^\alpha, \mu_s, \alpha_s)\,\mathrm{d}s + \int_0^t \sigma(s, X_s^\alpha, \mu_s, \alpha_s)\,\mathrm{d}W_t. \end{aligned}$$

Here the initial law of X_0^α, called ν_0, is known, and ξ is sampled from this law. Once we find a solution $X^{\alpha^*,\mu},\mu$, we denote $\nu_t^{\alpha^*,\mu} := \mathcal{L}(X_t^{\alpha^*,\mu},\mu)$ its law. We search for the family of measures μ_t solving the fixed point problem

$$\mu_t = \nu_t^{\alpha^*,\mu},\mu = \mathcal{L}(X_t^{\alpha^*,\mu},\mu).$$

On the other hand the related MFC problem is given by

$$\begin{aligned} \inf_{\alpha_t\in\mathbb{A}} \quad & \mathbb{E}\left[\int_0^T f(t, X_t^\alpha, \mathcal{L}(X_t^\alpha), \alpha_t)\,\mathrm{d}t + g(X_T^\alpha, \mathcal{L}(X_T^\alpha))\right] \\ \text{subject to} \quad & X_t^\alpha = \xi + \int_0^t b(s, X_s^\alpha, \mathcal{L}(X_s^\alpha), \alpha_s)\,\mathrm{d}s + \int_0^t \sigma(s, X_s^\alpha, \mathcal{L}(X_s^\alpha), \alpha_s)\,\mathrm{d}W_t. \end{aligned} \tag{1.2.2}$$

Even if these problems are different, they are nonetheless quite similar. The following enlightening interpretation is given by [CDL13; CD18a]: starting from (1.2.1), if you optimize first then go to the limit $N \to +\infty$ you obtain a MFG, whereas if you first go to the limit $N \to +\infty$ and then optimize you end up with a MFC. In general, the two equilibria are different.

Mean-field theory has several application in applied mathematics such as quantitative finance with optimal execution and price impact [CL18], economics with bitcoin mining [Ber+21] or oil production [CS17], health with the propagation of Epidemics [Lee+21], or even social networks [BTB16]. In the field of energy, the multiplication of small operators, decentralization and smart networks with flexibilities (demand management and storage) have inspired several studies using the mean-field machinery. These works have investigated smart charging [SWA21a; SWA21b], electricity storage and flexibility [ABTM20; GG21a; GG21b], but also price formation and trading in electricity markets [FTT21; FTT20]. These papers often consider linear-quadratic models, that is linear dynamics and quadratic costs so that the exact solution can be computed. But if one wants to use more realistic parameters, numerical methods are necessary in order to obtain an approximation of the solution. As before in the case of stochastic control, neural networks are an interesting tool to solve moderate to high dimensional nonlinear problems. Eventually, for practical use, to respect physical constraints or regulatory frameworks, it is sometimes useful to add state constraints to mean-field control problems. We will consider these two problems. The second part of our thesis aims at answering the following questions:

- How to introduce new machine learning numerical resolution methods?
- How can we impose state constraints in a mean-field control problem? What about probabilistic constraints?

We review the optimality conditions of mean-field control problems which will be useful for their numerical resolution. Similar optimality condition are obtained for the MFG. The dynamic programming approach introduces a value function on $[0,T] \times \mathcal{P}_2(\mathbb{R}^d)$ depending on the starting law:

$$v(t,\mu) \;=\; \inf_{\alpha} \; \mathbb{E}_{t,\mu}\Big[\int_t^T f(X_s^\alpha, \mathcal{L}(X_s^\alpha), \alpha_s) \; \mathrm{d}s + g(X_T^\alpha, \mathcal{L}(X_T^\alpha))\Big],$$

(here $\mathbb{E}_{t,\mu}[\cdot]$ is the conditional expectation given that the law at time t of X solution to (1.2.2) is equal to μ). Similarly as in the standard stochastic control case, this function verifies a partial differential equation, called the Master Bellman equation [BFY13; PW17; PW18; CP19; DPT19] (see [Car+19; CCD15] for MFG):

$$\left\{ \begin{array}{rcll} \partial_t v + \mathcal{H}(t,\mu,v,\partial_\mu v, \partial_x \partial_\mu v, \partial_\mu^2 v) &=& 0, & (t,\mu) \in [0,T) \times \mathcal{P}_2(\mathbb{R}^d), \\ v(T,\mu) &=& G(\mu), & \mu \in \mathcal{P}_2(\mathbb{R}^d), \end{array} \right. \tag{1.2.3}$$

with $G(\mu) = \int g(x,\mu) \; \mu(\mathrm{d}x)$,

$$\mathcal{H}(t,\mu,y,\mathrm{z}(.),\gamma(.),\gamma_0(.,.)) = \int_{\mathbb{R}^d} \big[H(t,x,\mu,y,\mathrm{z}(x),\gamma(x))\big]\mu(\mathrm{d}x),$$

and

$$H(t,x,\mu,y,z,\gamma) = \inf_{a \in A} \big[b(t,x,\mu,a).z + f(x,\mu,a) + \frac{1}{2}\mathrm{tr}\big(\sigma\sigma^\intercal(t,x,\mu,a)\gamma\big)\big].$$

In the equation above, $\partial_\mu v$ is the Lions derivative. It is defined thanks to the Fréchet derivative $[Dv](t,\xi)$ of the lifted function $\tilde{v} : (t,\xi) \in [0,T] \times L^2(\mathbb{R}^d) \mapsto v(t,\mathcal{L}(\xi)) \in \mathbb{R}$ which can be represented by Riesz representation theorem by $\partial_\mu v(t,\xi) \in L^2(\mathbb{R}^d)$ such that $[Dv](t,\xi)(Y) = \mathbb{E}[\partial_\mu v(t,\xi).Y] \in \mathbb{R}$.

The Pontryagin principle [CD15] (see [CD13] for MFG) introduces the Hamiltonian

$$H(t,x,\mu,y,z,\alpha) = b(t,x,\mu,\alpha) \cdot y + \mathrm{Tr}(\sigma(t,x,\mu,\alpha)z) + f(t,x,\mu,\alpha),$$

and gives optimality conditions in the form of McKean-Vlasov Forward-Backward Stochastic Differential Equations (MKVFBSDEs):

$$\begin{cases} \mathrm{d}X_t &= b(t, X_t, \mathcal{L}(X_t), \hat{\alpha}_t)\ \mathrm{d}t + \sigma(t, X_t, \mathcal{L}(X_t), \hat{\alpha}_t)\ \mathrm{d}W_t \\ X_0 &= \xi \\ \mathrm{d}Y_t &= -\partial_x H(t, X_t, \mathcal{L}(X_t), Y_t, Z_t, \hat{\alpha}_t)\ \mathrm{d}t - \mathbb{E}[\partial_\mu H(t, \tilde{X}_t, \mathcal{L}(X_t), \tilde{Y}_t, \tilde{Z}_t, \widetilde{\hat{\alpha}}_t)]\ \mathrm{d}t + Z_t\ \mathrm{d}W_t \\ Y_T &= \partial_x g\left(X_T, \mu_T\right) + \mathbb{E}\Big[\partial_\mu g\left(\widetilde{X_T}, \mu_T\right)\Big], \end{cases} \tag{1.2.4}$$

where $\hat{\alpha}_t = \underset{\alpha}{\operatorname{argmin}}\, H(t, X_t, \mathcal{L}(X_t), Y_t, Z_t, \alpha)$ and $(\tilde{X}, \tilde{Y}, \tilde{Z}, \widetilde{\hat{\alpha}})$ is an independent copy of $(X, Y, Z, \hat{\alpha})$. In the case of MFG, the Lions derivatives ∂_μ disappear in (1.2.4). Concerning mean-field games, the founding paper [LL06b] characterizes the equilibrium thanks to a coupled system of a Hamilton-Jacobi-Bellman and a Fokker-Planck equations which has been used to provide numerical solutions of MFG [ACD10].

A first idea to solve (1.2.2) is presented in [CL22]. A brute force approach consists in finding optimal feedback controls by parameterizing the control α_t by a neural network $\mathcal{A}^\theta : (t_i, x_i) \in [0, T] \times \mathbb{R}^d \mapsto \mathbb{R}^q$ and directly minimize the discretized in time cost:

$$\begin{aligned} \inf_\theta\ & \frac{1}{N_s} \sum_{k=1}^{N_s} \left(\sum_{i=1}^{N} f(t_i, X_i^k, \bar{\mu}_i, \mathcal{A}^\theta(t_i, X_i^k))\ \Delta t + g(X_N^k, \bar{\mu}_N) \right) \\ & X_{i+1}^k = X_i^k + b(t_i, X_i^k, \bar{\mu}_i, \mathcal{A}^\theta(t_i, X_i^k))\ \Delta t + \sigma(t_i, X_i^k, \bar{\mu}_i, \mathcal{A}^\theta(t_i, X_i^k))\ \Delta W_i^k, \\ & \bar{\mu}_i = \frac{1}{N_s} \sum_{k=1}^{N_s} \delta_{X_i^k},\ i = 0, \cdots, N-1, \\ & X_0^k = \xi^k,\ k = 0, \cdots, N_s, \end{aligned} \tag{1.2.5}$$

where the law is approached by the empirical measure of the particles with independent Brownian motions W^k, $k = 1, \cdots, N_s$ and ξ^k, $k = 0, \cdots, N_s$ are independent samples of the initial condition ξ. This is the version for mean-field control of the existing methods [GM05; HE16].

1.2.2 Contributions on the numerical approximation side

Other approaches numerically solve the optimality conditions to construct approximate solutions. We provide in Section 1.2.2 two approaches relying respectively on (1.2.4) and (1.2.3).

A first natural idea for machine learning resolution of mean-field games and mean-field control, introduced by [FZ20; CL22], is to extend the Deep BSDE method to the approximation of McKean-Vlasov FBSDEs, coming from the Pontryagin principle (1.2.4).

$$\begin{cases} X_{i+1}^k &= b(t_i, X_i^k, \bar{\mu}_i, \hat{\alpha}_i(X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k))\ \Delta t + \sigma(t_i, X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k, \hat{\alpha}_i(X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k))\ \Delta W_i^k, \\ X_0^k &= \xi^k,\ k = 0, \cdots, N_s, \\ Y_{i+1}^k &= Y_i^k - \partial_x H(t_i, X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k, \hat{\alpha}_i(X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k))\ \Delta t \\ & \quad -\frac{1}{N} \sum_{j=1}^{N} \partial_\mu H(t_i, X_i^j, \bar{\mu}_i, Y_i^j, Z_i^j, \hat{\alpha}_i(X_i^j, \bar{\mu}_i, Y_i^j, Z_i^j))\ \Delta t + \mathcal{Z}_i^{\theta_i}(X_i^k)\ \Delta W_i^k, \\ \bar{\mu}_i &= \frac{1}{N_s} \sum_{k=1}^{N_s} \delta_{X_i^k},\ i = 0, \cdots, N-1, \end{cases}$$

with $Y_0^k = \mathcal{Y}_0^\eta(X_0^k)$, a neural network $\mathcal{Y}_0^\eta$ and neural networks $\mathcal{Z}_i^{\theta_i}$, $i = 0, \cdots, N-1$. Moreover $\hat{\alpha}_i^k(X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k) = \underset{\alpha}{\operatorname{argmin}}\, H(t_i, X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k, \alpha)$. The scheme minimizes the loss function

$$\inf_{\eta, \theta_0, \cdots, \theta_N}\ \frac{1}{N} \sum_{k=1}^{N} \left| Y_N^k - \partial_x g\left(X_N^k, \mu_N\right) - \frac{1}{N} \sum_{j=1}^{N} \partial_\mu g\left(X_N^j, \mu_N\right) \right|^2,$$

with respect to the neural networks parameters $\eta, \theta_0, \cdots, \theta_N$. In [GMW22], we propose variants and compare all schemes in dimension 10 whereas the previous only tested the schemes in the unidimensional case. Our methods replace the empirical measure by other choices such as an online estimated measure or a neural network. We also consider a local version of the algorithm which solves one optimization problem by time step instead if a simple big optimization step. However it seems that the global method works better than this local one in the context of MKVFBSDEs. A theoretical analysis with a posteriori error estimates is conducted by [RSZ20]. An alternative method in the non-smooth case exploits proximal gradient descent to solve the MFC problem [RSZ21].

Another method arises by trying to solve the Master Bellman equation (1.2.3). Of course, being an infinite dimensional PDE, some discretization has to be performed in order to obtain an implementable scheme. We study this problem in Chapter 6. We consider the equation for a general function $H(t, x, \mu, y, z)$ which does not necessarily come from mean-field control. But the semilinear structure is required in order to use BSDE arguments. We use a particle method and replace this equation by a finite dimensional PDE, in high dimension. This equation is given by

$$\begin{cases} \partial_t v^N + \frac{1}{N}\sum_{i=1}^N H\big(t, x_i, \bar{\mu}(\boldsymbol{x}), v^N, N D_{x_i} v^N\big) + \frac{1}{2}\mathrm{tr}(\Sigma_N(t,\boldsymbol{x}) D^2_{\boldsymbol{x}} v^N) \;=\; 0, \text{ on } [0,T)\times(\mathbb{R}^d)^N \\ v^N(T,\boldsymbol{x}) \;=\; G\big(\bar{\mu}(\boldsymbol{x})\big), \quad \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N, \end{cases} \tag{1.2.6}$$

where $\bar{\mu}(.)$ is the empirical measure function defined by $\bar{\mu}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^N \delta_{x_i}$, for any $\boldsymbol{x} = (x_1, \ldots, x_N)$, $N \in \mathbb{N}^*$, and $\Sigma_N = (\Sigma_N^{ij})_{i,j\in[\![1,N]\!]}$ is the $\mathbb{R}^{Nd\times Nd}$-valued function with block matrices $\Sigma_N^{ij}(t,\boldsymbol{x}) = \sigma(t, x_i, \bar{\mu}(\boldsymbol{x}))\sigma^\intercal(t, x_j, \bar{\mu}(\boldsymbol{x}))\delta_{ij} + \sigma_0(t, x_i, \bar{\mu}(\boldsymbol{x}))\sigma_0^\intercal(t, x_j, \bar{\mu}(\boldsymbol{x})) \in \mathbb{R}^{d\times d}$. This equation is studied in [GMS21] which proves the convergence of its viscosity solution to the viscosity solution of the master equation when N goes to $+\infty$, under some conditions on H and the volatility coefficients σ, σ_0. Our contribution is to give a convergence rate thanks to probabilistic arguments. The finite-dimensional PDE (1.2.6) is linked to a Markovian BSDE through the nonlinear Feynman-Kac formula. The related forward particle system is

$$\mathrm{d}\boldsymbol{X}_t^N = \sigma_N(t, \boldsymbol{X}_t^N)\ \mathrm{d}\boldsymbol{W}_t + \boldsymbol{\sigma}_0(t, \boldsymbol{X}_t^N)\ \mathrm{d}W_t^0,$$

where σ_N is the block diagonal matrix with block diagonals $\sigma_N^{ii}(t,\boldsymbol{x}) = \sigma(t, x_i, \bar{\mu}(\boldsymbol{x}))$, $\boldsymbol{\sigma}_0 = (\sigma_0^i)_{i\in[\![1,N]\!]}$ is the $(\mathbb{R}^{d\times m})^N$-valued function with $\boldsymbol{\sigma}_0^i(t,\boldsymbol{x}) = \sigma_0(t, x_i, \bar{\mu}(\boldsymbol{x}))$, for $\boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]}$, $\boldsymbol{W} = (W^1, \ldots, W^N)$ where W^i, $i = 1, \ldots, N$, are independent n-dimensional Brownian motions, independent of a m-dimensional Brownian motion W^0 on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_{0\le t\le T}, \mathbb{P})$ and where the initial conditions of the particles system, $X_0^{i,N}$, $i = 1, \ldots, N$, are i.i.d. with distribution μ_0. The backward component is defined by the pair process $(Y^N, \boldsymbol{Z}^N = (Z^{i,N})_{i\in[\![1,N]\!]})$ valued in $\mathbb{R}\times(\mathbb{R}^d)^N$, solution to

$$\begin{aligned} Y_t^N =\; & G\big(\bar{\mu}(\boldsymbol{X}_T^N)\big) + \frac{1}{N}\sum_{i=1}^N \int_t^T H_b(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N), Y_s^N, N Z_s^{i,N})\ \mathrm{d}s \\ & - \sum_{i=1}^N \int_t^T (Z_s^{i,N})^\intercal \sigma\big(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N)\big)\ \mathrm{d}W_s^i, \\ & - \sum_{i=1}^N \int_t^T (Z_s^{i,N})^\intercal \sigma_0\big(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N)\big)\ \mathrm{d}W_s^0, \qquad 0 \le t \le T. \end{aligned}$$

The main difficulties in the study of the convergence of this PDE to the Master Bellman equation are:

- The N factor in front of the gradient in the nonlinearity H_b which makes the Lipschitz regularity with respect to the gradient explode.
- The explosion of the dimension of the PDE.

These difficulties can be bypassed thanks to a linearization procedure together with a Girsanov change of measure. We study the pathwise error on v:

$$\mathcal{E}_N^{\mathrm{y}} := \sup_{0\le t\le T} \big|Y_t^N - v(t, \bar{\mu}(\boldsymbol{X}_t^N))\big|,$$

and the L^2-error on its L-derivative

$$\big\|\mathcal{E}_N^{\mathrm{z}}\big\|_2 := \frac{1}{N}\sum_{i=1}^N \Big(\int_0^T \mathbb{E}\big|N Z_t^{i,N} - \partial_\mu v(t, \bar{\mu}(\boldsymbol{X}_t^N))(X_t^{i,N})\big|^2 \mathrm{d}t\Big)^{\frac{1}{2}}.$$

Under assumptions regarding the Lipschitz regularity of the parameters, existence of a smooth enough classical solution with linear growth and bounded second order Lions derivative, we are able to find a convergence rate for the solution. Additional assumptions are required to study the error $\big\|\mathcal{E}_N^{\mathrm{z}}\big\|_2$, such as ellipticity of the common volatility σ_0 and a linear structure regarding the gradient of the PDE solution.

Theorem 1.2.1. *Under Assumptions 6.2.1 and 6.2.2, we have $\mathbb{P}$-almost surely*

$$\mathcal{E}_N^{\mathrm{y}} \le \frac{C_y}{N},$$

where $C_y = \frac{T}{2} e^{[H_b]_1 T} L \|\sigma\|_\infty^2$, with $\|\sigma\|_\infty = \sup_{(t,x,\mu)\in[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)} |\sigma(t,x,\mu)|$.

Theorem 1.2.2. *Under Assumptions 6.2.1, 6.2.2 and 6.2.3, we have*

$$\big\|\mathcal{E}_N^{\mathrm{z}}\big\|_2 \le \frac{C_z}{N^{\frac{1}{2}}},$$

where $C_z = \|\sigma^+\|_\infty \sqrt{2T([H_1]_1 + [H_2]_1 L)\bar{C}_y^2 + \bar{C}_y T \|\sigma\|_\infty^2 L + \frac{2}{c_0}\bar{C}_y^2 T \|H_2\|_\infty^2}$ and $\bar{C}_y = \frac{T}{2} e^{([H_1]_1 + [H_2]_1 L)T} L \|\sigma\|_\infty^2$.

By noticing the symmetry properties of the PDE, we design in Chapter 7 a scheme fitted to these symmetries which allows us to work in high dimension, with 1000 or 10000 particles. More precisely, the solution v^N of (1.2.6) is invariant by permutation of its inputs, namely for $x_1, \cdots, x_N \in (\mathbb{R}^d)^N$ and a permutation σ on $\{1, \cdots, N\}$:

$$v^N(x_1, \cdots, x_N) = v^N(x_{\sigma(1)}, \cdots, x_{\sigma(N)}).$$

This symmetry also implies a structure for its space derivative. Both these behaviors made us design a method fitted to symmetric PDEs. The idea is to consider symmetric neural networks, namely DeepSets [Zah+17] and PointNet [Qi+17] which have been introduced by the machine learning community in particular in order to efficiently classify point clouds.

A *symmetric neural network* function, denoted $\mathcal{U} \in \mathcal{S}_{d,\ell,m,k,d'}^{\mathfrak{s},N,\rho}$, is an $\mathbb{R}^{d'}$-valued exchangeable function to the order N on $\mathbb{R}^d$, in the form:

$$\mathcal{U}(\boldsymbol{x}) = \psi\big(\mathfrak{s}((\varphi(x_i))_{i\in[\![1,N]\!]})\big), \quad \text{for } \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N,$$

where $\varphi \in \mathcal{N}_{d,\ell,m,k}^{\rho}$, $\psi \in \mathcal{N}_{k,\ell,m,d'}^{\rho}$ (here, for simplicity of notations, we assume that the number of hidden layers and neurons of φ and ψ are the same but in practical implementation, they may be different), and $\mathfrak{s}$ is a given $\mathbb{R}^k$-valued exchangeable function to the order N on $\mathbb{R}^k$.

The DeepSet network is given by

$$\mathcal{U}(\boldsymbol{x}) = \psi\Big(\frac{1}{N}\sum_{i=1}^{N}((\varphi(x_i))_{i\in[\![1,N]\!]})\Big), \quad \text{for } \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N,$$

when the PointNet network is given by

$$\mathcal{U}(\boldsymbol{x}) = \psi\big(\max_{i=1\cdots N}((\varphi(x_i))_{i\in[\![1,N]\!]})\big), \quad \text{for } \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N.$$

These networks are symmetric by construction and are expressive enough in the space of symmetric functions. More precisely, by combining Theorem 2.9 of [Wag+19] with Theorem 2 of [Hor91], we obtain the following approximation theorem for DeepSets.

Universal approximation for DeepSets networks. Let $\mathfrak{s}$ be the sum function. The set $\cup_{m=1}^{\infty}\mathcal{S}_{d,\ell,m,N+1,d'}^{\mathfrak{s},N,\rho}$ approximates any N-exchangeable continuous function w arbitrary well on any compact set of $K \subset \mathbb{R}^d$, once ρ is continuous, bounded and non-constant: for all $\varepsilon > 0$, $N \in \mathbb{N}^*$, there exists $\mathcal{U} \in \cup_{m=1}^{\infty}\mathcal{S}_{d,\ell,m,N+1,d'}^{\mathfrak{s},N,\rho}$ such that

$$\big|w(\boldsymbol{x}) - \mathcal{U}(\boldsymbol{x})\big| \leq \varepsilon \quad \forall \boldsymbol{x} \in K^N.$$

Note that a priori the latent space dimension k has to be chosen equal to $N + 1$.

Alternatively, by combining Theorem 1 of [Qi+17] with Theorem 2 of [Hor91], we obtain the following one-dimensional approximation theorem for PointNet.

Universal approximation for PointNet networks. Let $\mathfrak{s}$ be the max function. The set $\cup_{m=1}^{\infty}\cup_{k=1}^{\infty}\mathcal{S}_{1,\ell,m,k,d'}^{\mathfrak{s},N,\rho}$ approximates any N-exchangeable Hausdorff continuous function w (seen as a function on sets) arbitrary well on any compact set of $K \subset \mathbb{R}$, once ρ is continuous, bounded, and non-constant: for all $\varepsilon > 0$, $N \in \mathbb{N}^*$, there exists $\mathcal{U} \in \cup_{m=1}^{\infty}\cup_{k=1}^{\infty}\mathcal{S}_{1,\ell,m,k,d'}^{\mathfrak{s},N,\rho}$ such that

$$\big|w(S) - \mathcal{U}(\boldsymbol{x})\big| \leq \varepsilon, \quad \forall S \subset K,\ S = \{x_1, \cdots, x_N\}.$$

Note here that a priori the latent space dimension k has to be chosen as large as needed.

We adapt the DBDP scheme [HPW20] with these symmetric neural networks. Contrarily to feedforward neural networks, our method does converge even in difficult high dimensional cases. Several variants can be implemented for the approximation of the Z component of the BSDE, corresponding to the gradient of the solution. We can either consider the derivative of the neural networks approximating the Y component or use an architecture satisfying the symmetry properties of the gradient of a symmetric function.

1.2.3 Adding probabilistic state constraints

In many practical situations, it can be useful to impose state constraints on the controlled state. The constraints can be physical (such as non negativity, boundedness...), could be decided by the regulatory framework, or useful to find a particular solution of interest. We refer to [ST02; BEI10; Gel+13; CYZ20; PTZ21; Bal+21] for specific applications of probabilistic constraints, notably in finance, and often written in expectation form. An example that we consider is the one of the optimal control of a battery for renewable electricity storage, subject to randomness in both the production and the market prices. [ABTM20] studies such a problem in a mean-field setting without physical constraints on the battery, in order to obtain an explicitly solvable linear-quadratic model. In Chapter 8 we will numerically solve a related problem with constraints of the size of the storage and the injection or withdrawal capacities.

In the context of mean-field problems, several papers consider mean-field games with the state constrained to stay in a compact set [CC18; CCC18; FH20; GM21; AM21], either at all time or only a terminal time T. Terminal constraints in law are also considered by [BDK20; Dau20] respectively for control of McKean-Vlasov dynamics and stochastic control. For mean-field control, with the point of view of the control of Fokker-Planck equation, the works [Bon19; BF21] are able to enforce terminal or running constraints on a measure, by relying on the Pontryagin principle, in the deterministic case without diffusion. A mean-field control cost for a diffusion with probabilistic state constraints is also studied in [Dau21].

We consider in Chapter 8 the general case of running and discrete time constraints on the probability law of the state, which contains for instance in particular terminal constraint in law and compact set constraints. We rely on the level set approach from [BPZ15; BPZ16; ABZ13]. It amounts to introduce an auxiliary unconstrained problem with an additional state variable. Thanks to a representation result, we can link the solution of this problem to the solution of the original constrained problem. This approach is also useful for numerical purpose since existing methods can be applied to the auxiliary problem. Hence we are also able to develop numerical schemes for law constrained mean-field control problems. We consider the following cost and dynamics:

$$J(X_0,\alpha) = \mathbb{E}\Big[\int_0^T f\big(s, X_s^\alpha, \alpha_s, \mathbb{P}_{(X_s^\alpha,\alpha_s)}\big)\,\mathrm{d}s + g\big(X_T^\alpha, \mathbb{P}_{X_T^\alpha}\big)\Big]$$
$$X_t^\alpha = X_0 + \int_0^t b\big(s, X_s^\alpha, \alpha_s, \mathbb{P}_{(X_s^\alpha,\alpha_s)}\big)\,\mathrm{d}s + \int_0^t \sigma\big(s, X_s^\alpha, \alpha_t, \mathbb{P}_{(X_s^\alpha,\alpha_s)}\big)\,\mathrm{d}W_s,$$

and a probabilistic state constraints in the form

$$\Psi(t, \mathbb{P}_{X_t^\alpha}) \leq 0, \quad 0 \leq t \leq T,$$

where $\Psi = (\Psi^l)_{1\leq l\leq k}$ is a given function from $[0,T]\times\mathcal{P}_2(\mathbb{R}^d)$ into $\mathbb{R}^k$. Here, the multi-dimensional constraint $\Psi(t,\mu)\leq 0$ has to be understood componentwise, i.e., $\Psi^l(t,\mu)\leq 0$, $l=1,\cdots,k$. The problem of interest is therefore

$$V^\Psi := \inf_{\alpha\in\mathcal{A}} \big\{J(X_0,\alpha) : \Psi(t,\mathbb{P}_{X_t^\alpha}) \leq 0,\ \forall\, t\in[0,T]\big\}. \tag{1.2.7}$$

We introduce an additional deterministic state variable

$$Z_t^{z,\alpha} := z - \mathbb{E}\Big[\int_0^t f\big(s, X_s^\alpha, \alpha_s, \mathbb{P}_{(X_s^\alpha,\alpha_s)}\big)\,\mathrm{d}s\Big] = z - \int_0^t \widehat{f}\big(s, \mathbb{P}_{(X_s^\alpha,\alpha_s)}\big)\,\mathrm{d}s, \quad 0\leq t\leq T,$$

and the unconstrained auxiliary problem

$$\mathcal{Y}^\Psi : z\in\mathbb{R} \mapsto \inf_{\alpha\in\mathcal{A}}\Big[\{\widehat{g}(\mathbb{P}_{X_T^\alpha}) - Z_T^{z,\alpha}\}_+ + \sum_{l=1}^k \sup_{s\in[0,T]}\{\Psi^l(s,\mathbb{P}_{X_s^\alpha})\}_+\Big], \tag{1.2.8}$$

with the notation $\{x\}_+ = \max(x,0)$ for the positive part. We see that $\mathcal{Y}^\Psi(z)\geq 0$. Define the infimum of the zero level-set

$$\mathcal{Z}^\Psi := \inf\{z\in\mathbb{R} \mid \mathcal{Y}^\Psi(z) = 0\}. \tag{1.2.9}$$

We prove two representation results of the constrained problem (1.2.7) by the unconstrained problem (1.2.8).

Theorem 1.2.3. *1. If for some $z\in\mathbb{R}$ $\exists\ \alpha\in\mathcal{A}$ s.t. $\widehat{g}(\mathbb{P}_{X_T^\alpha}) \leq Z_T^{z,\alpha}$, $\Psi(s,\mathbb{P}_{X_s^\alpha})\leq 0$, $\forall\ s\in[0,T]$ then $\mathcal{Y}^\Psi(z)=0$.*

2. If V^Ψ is finite then $\mathcal{Y}^\Psi(V^\Psi)=0$. Thus $\mathcal{Z}^\Psi \leq V^\Psi$.

3. Define $1_k = (1, \cdots, 1) \in \mathbb{R}^k$. *We have the upper bound*

$$V^\Psi \leq \inf_{\varepsilon > 0} \mathcal{Z}^{\Psi + \varepsilon 1_k}.$$

Theorem 1.2.4. *Assume that* $V^\Psi < \infty$. *Then we have the representation*

$$\mathcal{Z}^\Psi = V^\Psi.$$

Moreover ε*-optimal controls* α^ε *for the auxiliary problem* $\mathcal{Y}^\Psi(V^\Psi)$ *are* ε*-admissible* ε*-optimal controls for the original problem in the sense that*

$$J(X_0, \alpha^\varepsilon) \leq V^\Psi + \varepsilon, \quad \sup_{0 \leq s \leq T} \Psi(s, \mathbb{P}_{X_s^{\alpha^\varepsilon}}) \leq \varepsilon.$$

A discussion about open loop and closed loop control is given in 8.3. We restrict ourselves for the numerical approximation part to deterministic Markovian feedback (or closed loop) controls but we give assumptions under which restricting the choice of controls to this space is optimal. In general, adapted (open loop) controls yield smaller cost than closed loop controls. These results allow ourselves to numerically solve the unconstrained problem 1.2.8 and the level-set minimization (1.2.9) in order to solve the constrained problem (1.2.7). We adapt the approach (1.2.5) from [CL22] to propose an implementable scheme.

Chapter 2

Introduction (en français)

Cette thèse est divisée en deux parties. La première étudie les méthodes numériques pour résoudre les Équations aux Dérivées Partielles (EDPs) paraboliques non-linéaires. Ces équations apparaissent typiquement dans le contexte de la programmation dynamique pour le contrôle stochastique, ce qui constitue notre principale motivation. Nos schémas utilisent l'apprentissage profond afin d'approcher les solutions des EDPs. Nous décrivons un nouveau schéma multistep pour la résolution d'EDP semilinéaires et réalisons une analyse de convergence de cette méthode et d'autres méthodes proposées dans la littérature. Nous montrons que l'erreur d'approximation théorique et l'erreur numérique empirique sont plus faibles en comparaison avec les méthodes existantes. En particulier, grâce à des résultats récents sur les réseaux de neurones lipschitziens GroupSort, nous sommes en mesure de relier l'erreur du schéma à l'architecture du réseau, c'est-à-dire le nombre de neurones et de couches. Lorsque les EDPs sont entièrement non-linéaires, nous décrivons de nouveaux schémas, one-step ou multistep, capables de traiter ce cas plus difficile. Nous fournissons des tests numériques afin de démontrer la pertinence de nos algorithmes.

La deuxième partie de la thèse est consacrée au contrôle à champ moyen. Nous obtenons d'abord la vitesse de convergence d'une approximation en dimension finie d'une équation sur l'espace de Wasserstein des mesures de probabilité. Nous nous appuyons pour cela sur un argument de linéarisation ainsi que sur un changement de mesure de Girsanov. Ces équations proviennent par exemple de l'équation maîtresse de Bellman du contrôle à champ moyen. Nous utilisons ensuite des réseaux de neurones symétriques DeepSets pour résoudre des EDPs symétriques telles que celles provenant de l'approximation des équations maîtresses de Bellman. Le respect de la symétrie du problème dans le schéma lui-même nous permet de résoudre des problèmes en haute dimension. Finalement, nous considérons le contrôle du champ moyen avec des contraintes d'état. Nous fournissons un résultat de représentation du problème original par un autre problème sans contraintes. Cette représentation nous permet de résoudre numériquement le problème initial.

2.1 Approximation des EDPs non-linéaires

2.1.1 Du contrôle stochastique aux EDPs et à leur résolution numérique

En mathématiques appliquées, notamment en mathématiques financières, le contrôle stochastique est un outil puissant pour concevoir des stratégies efficaces pour un agent dont la dynamique et les objectifs sont soumis au hasard. Des exemples typiques incluent entre autres la couverture quadratique de produits dérivés financiers [Pha00], la couverture du stockage de gaz [War12], ou l'allocation de portefeuille [ZL00]. D'autres applications sont décrites et étudiées dans le livre [Pha09].

La forme générale de tels problèmes est donnée par:

$$\inf_\alpha \mathbb{E}\Big[\int_0^T f(s, X_s^\alpha, \alpha_s)\ \mathrm{d}s + g(X_T^\alpha)\Big]$$
$$X_t^\alpha = X_0 + \int_0^t b(s, X_s^\alpha, \alpha_s)\ \mathrm{d}s + \int_0^t \sigma(s, X_s^\alpha, \alpha_s)\ \mathrm{d}W_s,\ t \geq 0,$$

où α est un processus de contrôle à valeurs dans $\mathbb{R}^q$, W un mouvement Brownien d-dimensionnel, et des fonctions $f : [0,T] \times \mathbb{R}^d \times \mathbb{R}^q \mapsto \mathbb{R}$, $b : [0,T] \times \mathbb{R}^d \times \mathbb{R}^q \mapsto \mathbb{R}^d$, $\sigma : [0,T] \times \mathbb{R}^d \times \mathbb{R}^q \mapsto \mathbb{R}^{d\times d}$. Le processus X est un processus de diffusion, solution d'une Équation Différentielle Stochastique (EDS) contrôlée. En introduisant la fonction valeur

$$v(t,x) = \inf_\alpha \mathbb{E}\Big[\int_t^T f(s, X_s^{t,x,\alpha}, \alpha_s)\ \mathrm{d}s + g(X_T^{t,x,\alpha})\Big]$$
$$X_s^{t,x,\alpha} = x + \int_t^s b(u, X_u^{t,x,\alpha}, \alpha_u)\ \mathrm{d}u + \int_t^s \sigma(u, X_u^{t,x,\alpha}, \alpha_u)\ \mathrm{d}W_u,\ s \geq t,$$

il est connu que celle-ci résout dans le sens des solutions de viscosité l'équation de Hamilton-Jacobi-Bellman (HJB)

$$\begin{cases} \partial_t v(t,x) + \inf_a \{f(t,x,a) + b(t,x,a)\ D_x v(t,x) + \frac{1}{2}\operatorname{Tr}(\sigma\sigma^\top(t,x,a) D_x^2 v(t,x))\} = 0 \\ v(T,x) = g(x). \end{cases}$$

Cette EDP parabolique est un cas particulier d'EDP complétement non-linéaire qui prend la forme

$$\begin{cases} \partial_t u(t,x) + F(t,x,u(t,x), D_x u(t,x), D_x^2 u(t,x)) = 0 \\ u(T,x) = g(x), \end{cases} \tag{2.1.1}$$

pour une fonction $F : [0,T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^{d\times d} \mapsto \mathbb{R}$. Quand la volatilité σ n'est pas contrôlée, c'est-à-dire quand $D_a\sigma = 0$, nous pouvons faire sortir le terme de volatilité de l'infimum et l'équation de HJB devient linéaire en la dérivée seconde (Hessienne) $D_x^2 v$:

$$\begin{cases} \partial_t v(t,x) + \frac{1}{2}\operatorname{Tr}(\sigma\sigma^\top(t,x) D_x^2 v(t,x)) + \inf_a\{f(t,x,a) + b(t,x,a)\ D_x v(t,x)\} = 0 \\ v(T,x) = g(x), \end{cases}$$

qui est un cas de particulier d'EDP semilinéaire

$$\begin{cases} \partial_t \tilde{u}(t,x) + \mu(t,x)\cdot D_x\tilde{u} + \frac{1}{2}\operatorname{Tr}(\sigma\sigma^\top(t,x) D_x^2\tilde{u}(t,x)) + \tilde{F}(t,x,\tilde{u}(t,x), \sigma(t,x) D_x\tilde{u}(t,x)) = 0 \\ \tilde{u}(T,x) = g(x), \end{cases} \tag{2.1.2}$$

pour une fonction $\tilde{F} : [0,T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \mapsto \mathbb{R}$. La résolution numérique de cette équation est plus facile que dans le cas précédent et certains de nos résultats théoriques seront seulement valides dans ce cas.

Dans le cas semilinéaire, les méthodes numériques sur lesquelles nous nous concentrons reposent sur le lien entre les Équations Différentielles Stochastiques Rétrogrades (EDSRs) et les EDPs pour construire des méthodes numériques. Avec le processus de diffusion X ayant pour générateur $\mathcal{L}\phi = \mu(t,x)\cdot D_x\phi(t,x) + \frac{1}{2}\operatorname{Tr}(\sigma\sigma^\top(t,x) D_x^2\phi(t,x))$, c'est-à-dire

$$X_t = \mathcal{X}_0 + \int_0^t \mu(s, X_s)\ \mathrm{d}s + \int_0^t \sigma(s, X_s)\ \mathrm{d}W_s, \tag{2.1.3}$$

la formule de Feynman-Kac non-linéaire [PP90] relie la solution $\tilde{u}$ de (2.1.2) à la solution adaptée (Y,Z) de l'EDSR:

$$Y_t = g(X_T) + \int_t^T \tilde{F}(s, X_s, Y_s, Z_s)\ \mathrm{d}s - \int_t^T Z_s\ \mathrm{d}W_s, \tag{2.1.4}$$

à travers $Y_t = \tilde{u}(t, X_t)$ et quand $\tilde{u}$ est régulière, $Z_t = \sigma(t,x) D_x\tilde{u}(t, X_t)$. Résoudre l'EDSR revient à résoudre l'EDP le long de trajectoires du processus X.

Décrivons l'approche standard de résolution de cette EDSR. On cherche à générer des processus adaptés à temps discret qui convergent vers les solutions de l'EDSR évalués le long d'une grille temporelle. En discrétisant en temps selon la grille régulière $t_k := \frac{kT}{N}$ par un schéma d'Euler pour le processus X (2.1.3) et un schéma d'Euler rétrograde pour la composante rétrograde Y (2.1.4) nous pouvons définir entre deux pas de temps consécutifs les processus à temps discret

$$\begin{aligned} X_{i+1} &= X_i + \mu(t_i, X_i)\ \Delta t + \sigma(t_i, X_i)\ \Delta W_i, \\ Y_{i+1} &= Y_i - \tilde{F}(t_i, X_i, Y_{i+1}, Z_i)\Delta t + Z_i.\Delta W_i,\ i = 0, \cdots, N-1, \end{aligned}$$

avec $X_0 = \mathcal{X}_0$, $\Delta t = \frac{T}{N}$ et $\Delta W_i = W_{t_{i+1}} - W_{t_i}$ ce qui donne par adaptation de Y et de Z

$$\begin{cases} Z_i = \mathbb{E}[Y_{i+1}\frac{\Delta W_i}{\Delta t} \mid \mathcal{F}_{t_i}] \\ Y_i = \mathbb{E}[Y_{i+1} + \tilde{F}(t_i, X_i, Y_{i+1}, Z_i)\Delta t \mid \mathcal{F}_{t_i}],\ i = 0, \cdots, N-1. \end{cases} \tag{2.1.5}$$

Cette représentation est à l'origine de divers schémas numériques [Zha04; BT04; GLW05; LGW06] qui partent de la condition terminale $Y_N = g(X_T)$ et déterminent récursivement Y_i, Z_i en calculant les espérances conditionelles précédentes. Une variante utilise une discrétisation implicite pour Y, qui requiert de résoudre un point fixe par itérations de Picard. Par la structure Markovienne du problème, ces espérances conditionnelles peuvent être écrites comme des fonctions mesurables du processus discret X, c'est-à-dire qu'il existe des fonctions mesurables u_i, z_i telles que

$$\begin{cases} Y_i = u_i(X_i) \\ Z_i = z_i(X_i),\ i = 0, \cdots, N-1. \end{cases}$$

L'idée est d'approcher ces fonctions, ce qui fournit en même temps une solution à l'EDP et l'EDSR. Ceci est fait de manière totalement implémentable dans [GLW05; LGW06] en choisissant une base de fonctions $\Psi_1, \cdots, \Psi_n : \mathbb{R}^d \to \mathbb{R}$ et une autre base de fonctions $\Phi_1, \cdots, \Phi_n : \mathbb{R}^d \to \mathbb{R}^d$ sur lesquelles sont calculées les espérances conditionnelles par régression Monte-Carlo. La méthode génère N_s trajectoires $(X^k)_{k=1,\cdots,N_s}$ à partir de N_s mouvements Brownien indépendants $(W^k)_{k=1,\cdots,N_s}$. Puis on initialise $Y_N^k = g(X_N^k)$ et, récursivement, au i-ème pas de temps, étant donné Y_{i+1}^k on résout le problème de moindres carrés

$$\begin{aligned} &\inf_{\beta_{i,1},\cdots,\beta_{i,n}} \frac{1}{N_s}\sum_{k=1}^{N_s}\Big|\sum_{j=1}^{n}\beta_{i,j}\Phi_j(X_i^k) - Y_{i+1}^k\frac{\Delta W_i^k}{\Delta t}\Big|^2 \\ &\inf_{\alpha_{i,1},\cdots,\alpha_{i,n}} \frac{1}{N_s}\sum_{k=1}^{N_s}\Big|\sum_{j=1}^{n}\alpha_{i,j}\Psi_j(X_i^k) - Y_{i+1}^k - \tilde{F}(t_i, X_i^k, Y_{i+1}^k, Z_i^k)\Delta t\Big|^2, \end{aligned} \tag{2.1.6}$$

où les minimiseurs sont appelés respectivement $\beta_{i,1}^*, \cdots, \beta_{i,n}^*$ et $\alpha_{i,1}^*, \cdots, \alpha_{i,n}^*$. Dans ce cas Z_i^k est défini par $Z_i^k := \sum_{j=1}^n \beta_{i,j}^*\Phi_j(X_i^k)$ et Y_i^k est donné par $Y_i^k := \sum_{j=1}^n \alpha_{i,j}^*\Psi_j(X_i^k)$. Sous des hypothèses techniques, il peut être montré que $((Y_i)_i, (Z_i)_i)$ converge vers (Y, Z) dans un sens approprié. Cette méthode est plutôt efficace en petite dimension mais est limité à la dimension 6 ou 7. Des méthodes d'apprentissage automatique ont été développées pour résoudre des problèmes en dimension plus grande. Les méthodes usuelles de différences finies souffrent de la "malédiction de la dimension" qui nous empêche de les utiliser quand la dimension de l'espace d'état est plus grande que 4. En effet, de trop nombreux points sont nécessaires pour discrétiser l'espace d'état, ce qui donne une complexité exponentielle en la dimension d.

2.1.2 Méthodes d'apprentissage automatique et nos nouveaux algorithmes

D'abord nous devons introduire les réseaux de neurones. On définit

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

comme l'ensemble des fonction de couche avec dimension d'entrée d_1, dimension de sortie d_2, et fonction d'activation $\rho : \mathbb{R}^{d_2} \to \mathbb{R}^{d_2}$. En général, la fonction d'activation est appliqué composante par composante $\rho(x_1, \dots, x_{d_2}) = \big(\hat{\rho}(x_1), \dots, \hat{\rho}(x_{d_2})\big)$ à partir d'une fonction unidimensionnelle $\hat{\rho} : \mathbb{R} \mapsto \mathbb{R}$, à la transformation affine $x \in \mathbb{R}^{d_1} \mapsto \mathcal{W}x + \beta \in \mathbb{R}^{d_2}$, avec une matrice $\mathcal{W}$ appelée poids, et un vecteur β appelé biais. Les examples classiques de fonction d'activation $\hat{\rho}$ sont la sigmoïde, le ReLU, la tangente hyperbolique. Nous appelons ensuite

$$\mathcal{N}^{\rho}_{d_0,d',\ell,m} = \Big\{\varphi : \mathbb{R}^{d_0} \to \mathbb{R}^{d'} : \exists \phi_0 \in \mathcal{L}^{\rho_0}_{d_0,m_0},\ \exists \phi_i \in \mathcal{L}^{\rho_i}_{m_{i-1},m_i}, i = 1, \dots, \ell - 1,$$
$$\exists \phi_\ell \in \mathcal{L}_{m_{l-1},d'}, \varphi \ = \ \phi_\ell \circ \phi_{\ell-1} \circ \cdots \circ \phi_0 \Big\},$$

l'ensemble des réseaux de neurones avec dimension d'entrée d_0, dimension de sortie d', et ℓ couches cachées avec m_i neurones par couche ($i = 0, \cdots, \ell - 1$). Ces nombres d_0, d', ℓ, la suite $m = (m_i)_{i=0,\dots,\ell-1}$, et la suite de fonctions d'activation $\rho = (\rho_i)_{i=0,\dots,\ell-1}$, forment l'architecture du réseau. Il est construit par la composition alternée entre des transformations affines et des non-linéarités.

Nous travaillerons surtout avec le cas $d_0 = d$ (dimension de la variable d'état x). Un réseau donné $\varphi \in \mathcal{N}^{\rho}_{d_0,d',\ell,m}$ est déterminé par les paramètres $\theta = (\mathcal{W}_0, \beta_0, \dots, \mathcal{W}_\ell, \beta_\ell)$ définissant les couches $\phi_0 \dots, \phi_\ell$, et nous écrirons parfois $\varphi = \varphi^\theta$. Les réseaux de neurones sont habituellement entraînés grâce à la minimisation d'un risque empirique. En apprentissage supervisé, à partir de variables aléatoires (X, Y) on cherche à minimiser le risque

$$\inf_\theta \mathbb{E}[L(\varphi^\theta(X), Y)],$$

avec une fonction de perte L quantifiant l'erreur entre la sortie $\varphi^\theta(X)$ du réseau de neurones et la cible Y. Un exemple classique pour L est la fonction quadratique $L : (a, b) \mapsto (a - b)^2$. En pratique, la loi jointe (X, Y) est inconnue et on se fonde sur N_s réalisations de tirages i.i.d. $(X_i, Y_i)_{i=1,\cdots,N_s}$ qui permettent d'approcher l'espérance par une espérance empirique

$$\inf_\theta \frac{1}{N} \sum_{i=1}^{N_s} L(\varphi^\theta(X_i), Y_i),$$

dans le style Monte-Carlo, comme dans (2.1.6). Numériquement le problème d'optimisation est résolu par descente de gradient stochastique [RM51] ou ses variantes parmi lesquelles [KB14], Adagrad [DHS11], Adadelta [Zei12]. Ces méthodes d'optimisation sont implémentées en particulier dans les libraires Tensorflow [Aba+16] et Pytorch [Pas+19].

Certains articles résolvent les EDPs à l'aide de l'apprentissage automatique, mais sans s'appuyer sur les EDSRs. La méthode Deep Galerkin [SS18] est capable de résoudre des EDPs en cherchant une solution sous la forme d'un réseau de neurones. Pour cela il faut tirer au sort un point dans le domaine d'intérêt et essayer de faire respecter l'EDP localement, en chaque point tiré. La même idée est utilisée par les réseaux de neurones physics-informed [RPK19] qui incorporent également des données et reconstruisent les solutions des EDPs en interpolant les données à partir de l'équation. Dans le cas homogène en temps, la méthode Deep Ritz de [EY18] résout la formulation variationnelle des équations elliptiques grâce à un schéma de type Deep Galerkin. Des méthodes utilisant l'apprentissage automatique mais pas les réseaux de neurones sont également envisagées dans la littérature. Ces schémas ont également été conçus afin d'atténuer la malédiction de la dimension, comme les sparse grids [Cha+21], le nesting Monte-Carlo [War18b], le branching [Bou+17; HL+19] et les schémas de Picard multiniveaux [E+19; HK20] qui s'avèrent surmonter la malédiction de la dimension dans certains cas.

Cas semilinéaire

Concentrons-nous sur les méthodes d'apprentissage automatique qui reposent sur les EDSRs pour résoudre les EDP semilinéaires. Les premières méthodes à apparaître sont des méthodes globales telles que la méthode Deep BSDE [EHJ17] puis la méthode Merged Deep BSDE [CWNMW19]. Dans ce cadre, on obtient la valeur initiale $u(0, x_0)$ de la solution de l'EDP et une représentation du gradient de la solution par les réseaux de neurones mais pas de la solution elle-même. Le terme "global" fait référence au fait que l'on résout un seul grand problème d'optimisation en tenant compte de toute la dynamique sur $[0, T]$. Le processus Y est approché par un schéma d'Euler :

$$Y_{i+1} = Y_i - \tilde{F}(t_i, X_i, Y_{i+1}, Z_i)\Delta t + \mathcal{Z}_i^{\theta_i}(X_i).\Delta W_i,\ i = 0, \cdots, N-1,$$

avec $Y_0 = y_0$, une variable y_0 et des réseaux de neurones $\mathcal{Z}_i^{\theta_i}$. Le schéma minimise la fonction de perte

$$\inf_{y_0, \theta_0, \cdots, \theta_N} \mathbb{E}|Y_N - g(X_N)|^2,$$

qui est un problème cible pour la valeur terminale de Y_T. De cette façon, la méthode ne s'inscrit pas dans le cadre de l'apprentissage supervisé puisque nous n'avons pas de cibles pour les réseaux de neurones approximant le processus Z. L'algorithme n'est pas non plus un algorithme non supervisé puisque les coefficients de la dynamique de X et Y sont connus, ce qui n'est pas le cas dans le paradigme de l'apprentissage par renforcement. La méthode peut être vue comme un schéma semi-supervisé où les cibles sont implicitement apprises lors de l'entraînement.

Dans [HPW20], la fonction de base de (2.1.6) est remplacée par un réseau de neurones et les deux problèmes d'optimisation pour la solution et son gradient sont résolus conjointement à chaque pas de temps. Cette méthode locale nous permet d'obtenir une approximation fonctionnelle de la solution de l'EDP à chaque pas de temps et pas seulement au temps initial $t = 0$. Contrairement au schéma Deep BSDE [EHJ17], plusieurs petits problèmes d'optimisation sont résolus. Ces problèmes d'optimisation sont proches les uns des autres et, par conséquent, nous pouvons initialiser les paramètres des réseaux de neurones aux paramètres calculés précédemment, ce qui donne un très bon point de départ, car la solution de l'EDP est censée être continue dans le temps. Des idées proches sont utilisées par [Rai18] et dans le splitting scheme de [Bec+21]. Ces schémas résolvent avec succès des EDP en dimension de 10 à 100 voire 1000 ou plus dans certains cas. L'idée du schéma Deep Backward Dynamic Programming (DBDP) de [HPW20] est d'optimiser les paramètres θ des réseaux de neurones $(\mathcal{U}_i^\theta(\cdot), \mathcal{Z}_i^\theta(\cdot)) \in \mathcal{N}_{d,d+1,\ell,m}^\rho$ grâce à la minimisation récursive des problèmes rétrogrades

$$\inf_\theta\ \mathbb{E}\left|\mathcal{U}_i^\theta(X_i) - \widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \tilde{F}(t_i, X_i, \mathcal{U}_i^\theta(X_i), \mathcal{Z}_i^\theta(X_i))\Delta t + \mathcal{Z}_i^\theta(X_i).\Delta W_i\right|^2,\ i = N-1, \cdots, 0, \tag{2.1.7}$$

où $\widehat{\mathcal{U}}_N(X_N)$ est pris comme $g(X_N)$, θ_i^* est le minimiseur du problème précédent et $\widehat{\mathcal{U}}_{i+1} = \mathcal{U}_{i+1}^{\theta_{i+1}^*}$ (les notations θ^* et $\widehat{\cdot}$ auront le même sens par la suite). Une variante où $\mathcal{Z}_i^\theta$ est remplacé par le gradient de $\mathcal{U}_i^\theta$ est aussi proposée dans ce papier mais donne des résultats un peu moins bons. La méthode de Deep Splitting (DS) de [Bec+21] utilise cette idée et minimise les fonctions de perte

$$\inf_\theta\ \mathbb{E}\left|\mathcal{U}_i^\theta(X_i) - \widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \tilde{F}(t_i, X_{i+1}, \widehat{\mathcal{U}}_{i+1}(X_{i+1}), \sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1}))\Delta t_i\right|^2,$$

pour $i = N-1, \cdots, 0$ par rapport aux paramètres θ d'un seul réseau de neurones par pas de temps. De même, l'article récent [NAO21] étudie un schéma de Malliavin avec des réseaux de neurones. Des extensions à des problèmes plus généraux tels que le cas path-dependent sont réalisées par [RT17; SVSS20; SZ20], tandis que la commande stochastique linéaire quadratique avec retard

est traitée par [LM21]. Lorsque l'objectif est de résoudre un problème de contrôle stochastique et non de résoudre une EDP, la méthode Deep BSDE peut alternativement être utilisée pour résoudre les équations différentielles stochastiques couplées Progressives-Rétrogrades (EDSPRs) provenant du principe de Pontryagin. Contrairement au cas précédent où le processus X peut être simulé indépendamment du calcul de la solution de l'EDSR (Y, Z), ici la dynamique de X dépend de la covariable Y. Ces méthodes sont décrites par exemple dans [Ji+20a; Ji+20b].

Dans cette thèse nous proposons plusieurs nouveaux schémas et nos interrogations principales sont :

- Comment pouvons-nous améliorer les méthodes existantes ?
- Comment se comporte leur erreur d'approximation lorsque le pas de temps décroît vers 0 ? Peut-on améliorer l'erreur théorique ?
- Comment la méthode dépend-elle de l'architecture du réseau de neurones ?

Schéma Multistep DBDP (MDBDP)(voir Algorithme 6)
Nous proposons dans le Chapitre 4 une nouvelle méthode d'apprentissage automatique multistep inspirée de [GT14] et [HPW20]. Elle est appelée Multistep Deep Backward Dynamic Programming (MDBDP). Elle repose sur la remarque suivante. Au lieu de discrétiser la BSDE entre les pas de temps t_i et t_{i+1}, nous pouvons la discrétiser entre t_i et $t_N = T$ et écrire

$$Y_i = g(X_N) + \sum_{j=i}^{N-1} \tilde{F}(t_j, X_j, Y_j, Z_j)\Delta t - \sum_{j=i}^{N-1} Z_j.\Delta W_j,$$

qui peut être réécrit par adaptation de Y et Z comme

$$\begin{cases} Y_i = \mathbb{E}[g(X_N) + \sum_{j=i}^{N-1} \tilde{F}(t_j, X_j, Y_j, Z_j)\Delta t - \sum_{j=i}^{N-1} Z_j.\Delta W_j \mid \mathcal{F}_{t_i}] \\ Z_i = \mathbb{E}[(g(X_N) + \sum_{j=i+1}^{N-1} \tilde{F}(t_j, X_j, Y_j, Z_j)\Delta t - \sum_{j=i+1}^{N-1} Z_j.\Delta W_j)\frac{\Delta W_i}{\Delta t} \mid \mathcal{F}_{t_i}],\ i = 0, \cdots, N-1. \end{cases} \tag{2.1.8}$$

En fait, les représentations (2.1.5) et (2.1.8) sont égales (cela peut se voir par la propriété de tour dans les espérances conditionnelles) mais à cause de l'approximation numérique des espérances conditionnelles nécessaires pour l'implémentation de l'algorithme, la propagation des erreurs numériques ne sera pas la même, voir le Théorème 2.1.1 et les commentaires associés. Notre schéma multistep prend la forme d'itérations rétrogrades comme dans le schéma DBDP (2.1.7) mais avec la fonction de perte modifiée

$$\begin{aligned} \inf_\theta\ \mathbb{E}\Big|&\mathcal{U}_i^\theta(X_i) - g(X_N) - \tilde{F}(t_i, X_i, \mathcal{U}_i^\theta(X_i), \mathcal{Z}_i^\theta(X_i))\Delta t + \mathcal{Z}_i^\theta(X_i)\Delta W_i \\ &- \sum_{j=i+1}^{N-1} \tilde{F}(t_j, X_j, \widehat{\mathcal{U}}_j(X_j), \widehat{\mathcal{Z}}_j(X_j))\Delta t + \sum_{j=i+1}^{N-1} \widehat{\mathcal{Z}}_j(X_j).\Delta W_j\Big|^2, \end{aligned} \tag{2.1.9}$$

qui utilise tous les réseaux de neurones précédemment entraînés aux pas de temps $j > i$.

Cas complétement non-linéaire

Dans le cas complétement non-linéaire (2.1.1), un schéma probabiliste efficace est introduit par [FTW11] et il converge localement uniformément vers la solution de viscosité de l'EDP grâce à la théorie des schémas monotones de [BS91]. En supposant que la solution de l'équation est régulière, la formule d'Itô donne

$$\begin{aligned} Y_t = \ & g(X_T) - \int_t^T \big[\mu(s, X_s).Z_s + \frac{1}{2}\mathrm{tr}(\sigma\sigma^\intercal(s, X_s)\Gamma_s) - F(s, X_s, Y_s, Z_s, \Gamma_s)\big]\mathrm{d}s \\ & - \int_t^T \sigma^\intercal(s, X_s)Z_s.\mathrm{d}W_s, \quad 0 \le t \le T. \end{aligned}$$

On défnit $\bar{F}(t,x,u,z,\gamma) := F(t,x,u,z,\gamma) - \mu(t,x).z - \frac{1}{2}\mathrm{tr}(\sigma\sigma^\intercal(s,x)\gamma)$. Contrairement au cas semilinéaire nous devons désormais estimer également le processus correspondant à la derivée seconde $\Gamma_t = D_x^2 u(t,X_t)$. Ici les coefficients du processus X de (2.1.3) peuvent être choisis arbitrairement, contrairement au cas semilinéaire dans lequel la volatilité est donnée par l'équation (2.1.2). En effet, remarquons que dans (2.1.1), la partie linéaire de (2.1.2) n'apparaît pas.

Un schéma d'apprentissage pour traiter ce cas a été introduit par [BEJ19], à partir des EDSRs du second ordre de [Che+07] pour la solution des EDPs complétement non-linéaires

$$\begin{cases} Y_t &= g(\mathcal{X}_T) + \int_t^T \bar{F}(s,\mathcal{X}_s,Y_s,Z_s,\Gamma_s)\mathrm{d}s - \int_t^T Z_s.\sigma \mathrm{d}W_s, \\ Z_t &= D_x g(\mathcal{X}_T) - \int_t^T A_s \mathrm{d}s - \int_t^T \Gamma_s \sigma \mathrm{d}W_s, \quad 0 \le t \le T, \end{cases}$$

avec $A_t = \mathcal{L}D_x u(t,\mathcal{X}_t)$. En particulier, dans le cas où il existe une solution classique u de classe $\mathcal{C}^{1,2}$, Γ_t vérifie $\Gamma_t = D_x^2 u(t,X_t)$ et comme précédemment $Y_t = u(t,X_t)$ alors que $Z_t = D_x u(t,X_t)$. Le schéma discrétise ce système et utilise des variables pour approcher Y_0, Z_0 et des réseaux de neurones $(\mathcal{A}^\theta, \mathcal{G}^\theta)$ pour approcher A, Γ grâce à la minimisation de la distance à la conditon terminale $(g(X_T), D_x g(X_T))$ des deux équations, en suivant la méthode du schéma Deep BSDE.

Nous proposons quatre schémas locaux alternatifs, en combinant des idées de [HPW20; Bec+21] et de notre propre schéma multistep.

Second order DBDP scheme (2DBDP)(voir Algorithme 7)
La première idée naturelle décrite dans le Chapitre 5 est une extension du schéma DBDP avec la fonction de perte

$$\inf_\theta \mathbb{E}\Big|\mathcal{U}_i^\theta(X_i) - \widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \bar{F}(t_i,X_i,\mathcal{U}_i^\theta(X_i),\mathcal{Z}_i^\theta(X_i),D\widehat{\mathcal{Z}}_{i+1}(\mathcal{T}(X_{t_{i+1}})))\Delta t + \mathcal{Z}_i^\theta(X_i)\Delta W_i\Big|^2,$$

où la dérivée seconde est estimée en différentiant le gradient obtenu au pas de temps suivant. Nous appelons cette méthode 2DBDP. $\mathcal{T}$ est une troncature à un certain quantile qui permet de modérer la propagation d'instabilités sur les bords du domaine exploré. Nous montrons que notre schéma est capable de résoudre des EDPs non-linéaires en dimension modérée et donne une meilleure approximation du contrôle que le schéma Deep 2BSDE.

Grâce à nos investigations concernant les méthodes multistep, nous avons aussi étendu ce schéma dans un cadre multistep. Nous proposons dans le Chapitre 3 trois variantes à côté d'une synthèse de la littérature sur les méthodes d'apprentissage automatique pour les EDPs et le contrôle stochastique en finance. Toutes les méthodes considèrent les fonctions de perte

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

où la définition de $\hat{\Gamma}_{l_j}$ dépend de la méthode. Nous supposons que le drift vérifie $\mu = 0$ et que la matrice de volatilité σ est inversible et constante en temps et espace.

- **Second order Explicit Multistep DBDP scheme (2EMDBDP)**(voir Algorithme 3). On combine le schéma multistep et le schéma 2DBDP. Si $i = N-1$, définissons $\widehat{\Gamma}_i = D^2 g$, sinon $\widehat{\Gamma}_i = D_x\widehat{\mathcal{Z}}_{i+1}$, $\widehat{\Gamma}_j = D_x\widehat{\mathcal{Z}}_j$, $j \in [\![i+1, N-1]\!]$. On prend aussi $l_j = j$.

- **Second order Multistep DBDP (2MDBDP),** (voir Algorithme 4). Une autre méthode utilise des poids de Malliavin pour évaluer cette dérivée sur une sous-grille $\hat{\pi} = \{t_{\hat{\kappa}\ell}, \ell = 0, \dots, \hat{N}\} \subset \pi$, de module $|\hat{\pi}| = \hat{\kappa}|\pi|$, pour $\hat{\kappa} \in \mathbb{N}^*$, avec $N = \hat{\kappa}\hat{N}$. Γ_ℓ obtenu en résolvant

$$\inf_\theta \mathbb{E}\Big|\Gamma_\ell^\theta(X_{\hat{\kappa}\ell}) - \frac{\widehat{\mathcal{Z}}_{\hat{\kappa}(\ell+1)}(X_{\hat{\kappa}(\ell+1)}) - \widehat{\mathcal{Z}}_{\hat{\kappa}(\ell+1)}(\hat{X}_{\hat{\kappa}(\ell+1)})}{2}\hat{H}_\ell^1\Big|^2,$$

avec les poids de Malliavin

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

et les variables antithétiques

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

On prend aussi $l_j = j \div \hat{\kappa} + 1$ où '$\div$' est le symbole pour la division Euclidienne.

- **Second order Multistep Malliavin DBDP (2M^{2}DBDP)**, (voir Algorithme 5). Cette technique utilise une différentiation du second ordre de la représentation multistep sur une sous-grille comme précedemment grâce à des poids de Malliavin du second ordre et des variables antithétiques. On prend également $l_j = j \div \hat{\kappa} + 1$.

2.1.3 Réseaux de neurones lipschitziens GroupSort et contributions à de nouveaux résultats théoriques

Nous fournissons au chapitre 4 une analyse détaillée de la convergence du schéma Deep Splitting et de notre schéma multistep. Nous nous concentrons sur la propagation des erreurs de discrétisation et de régression à travers le schéma. Cependant, nous ne considérons pas l'erreur statistique provenant de l'approximation Monte-Carlo de l'espérance dans les fonctions de perte telles que (2.1.9) ni l'erreur d'optimisation provenant de l'algorithme de descente de gradient.

Grâce à des résultats récents sur l'approximation universelle quantitative pour les réseaux de neurones lipschitziens GroupSort, nous obtenons explicitement l'erreur en termes de l'architecture du réseau de neurones, c'est-à-dire son nombre de neurones et de couches. Pour une utilisation future, nous introduisons la régularité L^2 de Z introduite par [Zha04] :

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

Nous choisissons des suites $(\gamma_i)_i, (\eta_i)_i$ et considérons des classes d'approximation $\mathcal{N}_i$, $\mathcal{N}_i'$ et $\mathcal{N}_i^{\gamma,\eta}$ respectivement de $\mathbb{R}^d \mapsto \mathbb{R}$, $\mathbb{R}^d \mapsto \mathbb{R}^d$, $\mathbb{R}^d \mapsto \mathbb{R}$ (et avec des fonctions γ_i-Lipschitz et gradient η_i-Lipschitz pour le dernier). On définit pour $i = 0, \dots, N-1$ les erreurs d'approximation L^2 dans ces classes de fonction de $v_i^{(1)}, \widehat{z}_i^{(1)}, v_i^{(2)}, v_i^{(3)}, \widehat{z}_i^{(3)}$ definies respectivement dans (4.3.2), (4.3.4) et (4.3.6):

$$\varepsilon_i^{1,y} \;:=\; \inf_{\mathcal{U}\in\mathcal{N}_i}\mathbb{E}\big|v_i^{(1)}(X_i) - \mathcal{U}(X_i)\big|^2, \qquad \varepsilon_i^{1,z} \;:=\; \inf_{\mathcal{Z}\in\mathcal{N}_i'}\mathbb{E}\big|\widehat{z}_i^{(1)}(X_i) - \mathcal{Z}(X_i)\big|_2^2,$$

$$\varepsilon_i^{\gamma,\eta} = \begin{cases} \inf_{\mathcal{U}\in\mathcal{N}_i^{\gamma,\eta}}\mathbb{E}\big|v_i^{(2)}(X_i) - \mathcal{U}(X_i)\big|^2, & i = 0, \dots, N-1, \\ \inf_{\mathcal{U}\in\mathcal{N}_i^{\gamma,\eta}}\mathbb{E}\big|g(X_N) - \mathcal{U}(X_N)\big|^2, & i = N, \end{cases}$$

$$\varepsilon_i^{3,y} \;:=\; \inf_{\mathcal{U}\in\mathcal{N}_i}\mathbb{E}\big|v_i^{(3)}(X_i) - \mathcal{U}(X_i)\big|^2, \qquad \varepsilon_i^{3,z} \;:=\; \inf_{\mathcal{Z}\in\mathcal{N}_i'}\mathbb{E}\big|\widehat{z}_i^{(3)}(X_i) - \mathcal{Z}(X_i)\big|_2^2.$$

Notre résultat d'approxiamtion est le suivant

Theorem 2.1.1 (Erreur d'approximation de MDBDP). *Sous l'Hypothèse 4.3.1, il existe une constante $C > 0$ (dependant seulement des données μ, σ, f, g, d, T) telle qu'à la limite où $|\pi| \to 0$*

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

Cette erreur d'approximation du schéma à plusieurs étapes est meilleure que celle du schéma DBDP de [HPW20] où le terme des erreurs de régression $\sum_{j=0}^{N-1}(\varepsilon_j^{1,y} + \Delta t_j\varepsilon_j^{1,z})$ est remplacé par $\sum_{j=0}^{N-1}(N\varepsilon_j^{1,y} + \varepsilon_j^{1,z})$. En particulier, nous répondons au point soulevé par Côme Huré dans la section 1.3.4 de sa thèse [Hur19] où il mentionne que le facteur N qu'il a obtenu devant la somme des erreurs était inattendu et ne pouvait pas être retiré de sa preuve. En fait, l'utilisation d'une méthode multistep nous permet de supprimer ce terme. Cependant, nous avons obtenu un facteur similaire N pour le schéma Deep Splitting de [Bec+21].

Theorem 2.1.2 (Erreur d'approximation de DS). *Supposons que l'Hypothèse 4.3.1 soit vérifiée, et supposons que $\mathcal{X}_0 \in L^4(\mathcal{F}_0, \mathbb{R}^d)$. Alors, il existe une constante $C > 0$ (dépendant seulement de $\mu, \sigma, f, g, d, T, \mathcal{X}_0$) telle qu'à la limite où $|\pi| \to 0$*

$$\begin{aligned}\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(2)}(X_i)\big|^2 \leq C\Big(&\mathbb{E}\big|g(X_N) - g(\mathcal{X}_T)\big|^2 + |\pi| + \varepsilon^Z(\pi) \\ &+\ \max_i\big[\gamma_i^2, \eta_i^2\big]|\pi| + \varepsilon_N^{\gamma,\eta} + N\sum_{i=0}^{N-1}\varepsilon_i^{\gamma,\eta}\Big).\end{aligned}$$

Les premiers résultats d'approximation classiques prouvent que les réseaux de neurones sont denses dans les espaces de fonctions, tels que [HSW89; HSW90; Hor91] mais aucune vitesse de convergence n'est donnée dans ces travaux. Lorsque l'on suppose une plus grande régularité pour les fonctions cibles, comme la dérivabilité, la continuité de Lipschitz, la convexité ou la régularité de Sobolev, plusieurs articles ont ensuite fourni des résultats d'approximation explicites [Pin99a; BGS15; Yar17; Bac17]. Nous utilisons une approche plus récente avec des réseaux de neurones lipschitziens. Grâce à une fonction d'activation qui divise son entrée en groupes et trie chacun d'entre eux (voir figure 1.1), et en imposant des paramètres bornés, le réseau GroupSort introduit par [ALG19] est 1-Lipschitz. [TSB21] prouve des résultats d'approximation explicites pour ces réseaux qui nous permettent d'aller plus loin dans notre analyse des erreurs en exprimant les erreurs de régression en fonction de l'architecture. En conséquence, nous pouvons choisir les paramètres des réseaux de neurones de sorte que l'erreur globale soit équivalente à l'erreur de discrétisation temporelle.

Soit $\kappa \in \mathbb{N}^*, \kappa \geq 2$, une taille de groupement, divisant le nombre de neurones $m_i = \kappa n_i$, à chaque couche $i = 0, \cdots \ell - 1$. On appellera $\sum_{i=0}^{\ell-1} m_i$ la largeur du réseau et $\ell + 1$ sa profondeur. Les réseaux GroupSort correspondent à des réseaux de neurones classiques dans $\mathcal{N}_{d,1,\ell,m}^{\zeta_\kappa}$ avec une suite spécifique de fonctions d'activation $\zeta_\kappa = (\zeta_\kappa^i)_{i=0,\ldots,\ell-1}$, et une sortie unidimensionnelle. Chaque fonction non-linéaire ζ_κ^i divise son entrée en groupes de taille κ et trie chaque groupe en ordre décroissant, voir la figure 2.1.

De plus, en imposant aux paramètres du GroupSort des contraintes en norme euclidienne $|\cdot|_2$ et de norme ℓ_∞ $|\cdot|_\infty$ sur leurs paramètres:

$$\sup_{|x|_2=1} |\mathcal{W}_0 x|_\infty \leq 1, \quad \sup_{|x|_\infty=1} |\mathcal{W}_i x|_\infty \leq 1, \ |\beta_j|_\infty \leq M, \ i = 1, \cdots, l, \ j = 0, \cdots, l,$$

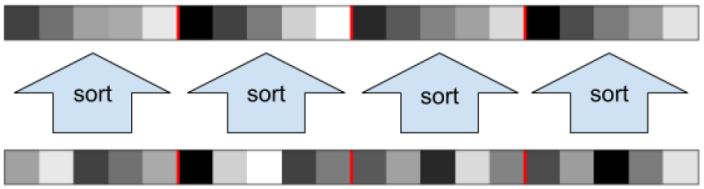


Figure 2.1: Fonction d'activation GroupSort ζ_κ avec taille de groupement $\kappa = 5$ et $m = 20$ neurones, figure provenant de [ALG19].

pour un certain $M > 0$, les réseaux de neurones GroupSort correspondants de $\mathcal{N}^{\zeta_\kappa}_{d,d',\ell,m}$ sont 1-Lipschitz. L'espace de tels réseaux de neurones GroupSort 1-Lipschitz est appelé $\mathcal{S}^{\zeta_\kappa}_{d,\ell,m}$:

$$\mathcal{S}^{\zeta_\kappa}_{d,\ell,m} = \{\varphi^{(\mathcal{W}_0,\beta_0,\dots,\mathcal{W}_\ell,\beta_\ell)} \in \mathcal{N}^{\zeta_\kappa}_{d,1,\ell,m},\ \sup_{|x|_2=1} |\mathcal{W}_0 x|_\infty \le 1,\ \sup_{|x|_\infty=1} |\mathcal{W}_i x|_\infty \le 1,$$
$$|\beta_j|_\infty \le M,\ i = 1,\cdots,l,\ j = 0,\cdots,l\}.$$

On introduit alors l'ensemble $\mathcal{G}^{\zeta_\kappa}_{K,d,d',\ell,m}$

$$\mathcal{G}^{\zeta_\kappa}_{K,d,d',\ell,m} := \{\Psi = (\Psi_i)_{i=1,\dots,d'} : \mathbb{R}^d \mapsto \mathbb{R}^{d'},\ \Psi_i : x \in \mathbb{R}^d \mapsto K\beta_i \quad \phi_i\Big(\frac{x+\alpha_i}{\beta_i}\Big) \in \mathbb{R},$$
$$\phi_i \in \mathcal{S}^{\zeta_\kappa}_{d,\ell,m},\quad \text{pour un certain } \alpha_i \in \mathbb{R}^d,\ \beta_i > 0\}.$$

Remarquons que ces réseaux sont $\sqrt{d'}K$-Lipschitz et que chacune de leur composante est K-Lipschitz. Donnons le résultat d'approximation qui est central pour notre étude:

Proposition 2.1.1 (Légère extension de Tanielian, Sangnier, Biau [TSB21] : Théorème d'approximation pour les fonctions Lipschitz par des réseaux de neurones lipschitziens GroupSort.)**.** *Soit $f : [-R,R]^d \mapsto \mathbb{R}^{d'}$ une fonction K-Lipschitz. Alors, pour tout $\varepsilon > 0$, il existe un réseau GroupSort g dans $\mathcal{G}^{\zeta_\kappa}_{K,d,d',\ell,m}$ vérifiant*

$$\sup_{x\in[-R,R]^d} |f(x) - g(x)|_2 \le \sqrt{d'} 2RK\varepsilon,$$

avec g de taille de groupement $\kappa = \lceil \frac{2\sqrt{d}}{\varepsilon} \rceil$, profondeur $\ell + 1 = O(d^2)$ et largeur $\sum_{i=0}^{\ell-1} m_i = O((\frac{2\sqrt{d}}{\varepsilon})^{d^2-1})$ dans le cas $d > 1$. Si $d = 1$, le même résultat est valide avec g de taille de groupement $\kappa = \lceil \frac{1}{\varepsilon} \rceil$, profondeur $\ell + 1 = 3$ et largeur $\sum_{i=0}^{\ell-1} m_i = O(\frac{1}{\varepsilon})$.

Nous étudions ensuite la convergence pour l'erreur d'approximation du schéma MDBDP avec des réseaux de neurones GroupSort et avec l'hypothèse supplémentaire que le driver $\tilde{F}$ ne dépend pas de z, donc que l'EDP est linéaire en z.

Proposition 2.1.2 (Vitesse de convergence de MDBDP)**.** *Supposons que l'Hypothèse 4.3.1 et l'Hypothèse 4.3.2 soient vérifiées, et supposons que $\mathcal{X}_0 \in L^{2+\delta}(\mathcal{F}_0, \mathbb{R}^d)$, pour un certain $\delta > 0$, et que g soit $[g]-$Lipschitz. Alors, il existe une suite bornée K_i (uniformément en i, N) telle que pour les classes de réseaux GroupSort $\mathcal{N}_i = \mathcal{G}^{\zeta_\kappa}_{K_i,d,1,\ell,m}$, et $\mathcal{N}'_i = \mathcal{G}^{\zeta_\kappa}_{\sqrt{\frac{d}{\Delta t_i}}K_i,d,d,\ell,m}$, on ait*

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

avec une taille de groupement $\kappa = O(2\sqrt{d}N^2)$, une profondeur $\ell + 1 = O(d^2)$ et une largeur $\sum_{i=0}^{\ell-1} m_i = O((2\sqrt{d}N^2)^{d^2-1})$ dans le cas $d > 1$. Si $d = 1$, on peut prendre $\kappa = O(N^2)$, une profondeur $\ell + 1 = 3$ et une largeur $\sum_{i=0}^{\ell-1} m_i = O(N^2)$. Ici, les constantes dans le terme $O(\cdot)$ dépendent seulement de $\mu, \sigma, \tilde{F}, g, d, T, x_0$.

Nous sommes capables d'effectuer la même analyse pour le schéma DBDP dans le cas semi-linéaire où l'EDP est non-linéaire en z.

Proposition 2.1.3 (Rate of convergence of DBDP)**.** *Supoosant que l'Hypothèse 4.3.1 soit vérifiée, et supposons que $\mathcal{X}_0 \in L^{2+\delta}(\mathcal{F}_0, \mathbb{R}^d)$, pour un certain $\delta > 0$, et que g soit $[g]-$Lipschitz. Alors, il existe une suite bornée K_i (uniformément en i, N) telle que pour $\mathcal{N}_i = \mathcal{G}^{\zeta_\kappa}_{K_i,d,1,\ell,m}$, et $\mathcal{N}'_i = \mathcal{G}^{\zeta_\kappa}_{\sqrt{\frac{d}{\Delta t_i}}K_i,d,d,\ell,m}$, on ait*

$$\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}^{(3)}_i(X_i)\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \big|Z_s - \widehat{\mathcal{Z}}^{(3)}_i(X_i)\big|_2^2 \,\mathrm{d}s\Big] = O(1/N),$$

avec une taille de groupement $\kappa = O(2\sqrt{d}N^3)$, une profondeur $\ell + 1 = O(d^2)$ et une largeur $\sum_{i=0}^{\ell-1} m_i = O((2\sqrt{d}N^3)^{d^2-1})$ in the case $d > 1$. Si $d = 1$, on peut prendre $\kappa = O(N^3)$, une profondeur $\ell + 1 = 3$ et une largeur $\sum_{i=0}^{\ell-1} m_i = O(N^3)$. Ici, les constantes dans le terme $O(\cdot)$ dépendent seulement de $\mu, \sigma, f, g, d, T, \mathcal{X}_0$.

En raison des différences mentionnées précédemment entre l'erreur de DBDP et celle de MDBDP, nous remarquons que beaucoup plus de neurones sont nécessaires pour obtenir une vitesse similaire pour l'erreur. Par exemple en dimension $d = 1$, pour obtenir une erreur en $O(1/N)$ le schéma DBDP nécessite en théorie $O(N^3)$ neurones alors que seulement $O(N^2)$ sont nécessaires pour le schéma MDBDP.

2.2 Problèmes à champ moyen et leur approximation numérique

2.2.1 Motivations et conditions d'optimalité

Les jeux à grande population ont suscité un intérêt croissant depuis l'émergence de la théorie des jeux à champ moyen introduite par [LL06a; LL06b] et [HCM06]. Cette théorie considère la limite d'un nombre infini de joueurs similaires en interaction et vise à caractériser les équilibres qui en résultent. Deux cadres principaux sont disponibles :

- **Jeux à champ moyen (MFG)** La théorie MFG s'intéresse aux équilibres de Nash, et considère donc une interaction compétitive entre les joueurs.
- **Contrôle à champ moyen (MFC)** (ou contrôle de dynamique de type McKean-Vlasov). Ici, le cadre se concentre sur les équilibres collaboratifs, avec un planificateur central qui résout un problème concernant l'ensemble de la population.

Le point de départ est un jeu différentiel stochastique à N joueurs, avec des contrôles par rétroaction α^i_t qui influent sur le coût et la dynamique suivantes pour l'agent i :

$$\begin{cases} J^i = \mathbb{E}\left[\int_0^T f\left(t, X^i_t, \frac{1}{N}\sum_{k=1}^N \delta_{X^k_t}, \alpha^i_t\right)\,\mathrm{d}t + g\left(X^i_T, \frac{1}{N}\sum_{k=1}^N \delta_{X^k_T}\right)\right] \\ \mathrm{d}X^i_t = b\left(t, X^i_t, \frac{1}{N}\sum_{k=1}^N \delta_{X^k_t}, \alpha^i_t\right)\,\mathrm{d}t + \sigma\left(t, X^i_t, \frac{1}{N}\sum_{k=1}^N \delta_{X^k_t}, \alpha^i_t\right)\,\mathrm{d}W^i_t. \end{cases} \tag{2.2.1}$$

Lorsque le nombre de joueurs N est grand, le jeu devient difficile à résoudre et l'approximation du champ moyen fournit un moyen de résoudre approximativement le problème en prenant la limite $N \to +\infty$. Pour trouver un équilibre, nous considérons des contrôles de rétroaction Markoviens distribués $\alpha^i_t = \alpha_t(X^i_t)$.

Pour écrire le problème MFG asymptotique, nous commençons par définir une famille $(\mu_t)_{t\in[0,T]}$ de mesures de probabilité. Ensuite, nous résolvons par symétrie uniquement pour un joueur

représentatif :

$$\begin{aligned} \inf_{\alpha_t \in \mathbb{A}} \quad & \mathbb{E}\left[\int_0^T f(t, X_t^\alpha, \mu_t, \alpha_t)\ \mathrm{d}t + g(X_T^\alpha, \mu_T)\right] \\ \text{subject to} \quad & X_t^\alpha = \xi + \int_0^t b(s, X_s^\alpha, \mu_s, \alpha_s)\ \mathrm{d}s + \int_0^t \sigma(s, X_s^\alpha, \mu_s, \alpha_s)\ \mathrm{d}W_t. \end{aligned}$$

Ici, la loi initiale de X_0^α, appelée ν_0, est connue, et ξ est tiré aléatoirement selon cette loi. Une fois que l'on a trouvé une solution $X^{\alpha^*,\mu,\mu}$, on note $\nu_t^{\alpha^*,\mu} := \mathcal{L}(X_t^{\alpha^*,\mu,\mu})$ sa loi. Nous recherchons alors la famille de mesures μ_t résolvant le problème de point fixe

$$\mu_t = \nu_t^{\alpha^*,\mu,\mu} = \mathcal{L}(X_t^{\alpha^*,\mu,\mu}).$$

D'autre part le problème MFC associé est

$$\begin{aligned} \inf_{\alpha_t \in \mathbb{A}} \quad & \mathbb{E}\left[\int_0^T f(t, X_t^\alpha, \mathcal{L}(X_t^\alpha), \alpha_t)\ \mathrm{d}t + g(X_T^\alpha, \mathcal{L}(X_T^\alpha))\right] \\ \text{subject to} \quad & X_t^\alpha = \xi + \int_0^t b(s, X_s^\alpha, \mathcal{L}(X_s^\alpha), \alpha_s)\ \mathrm{d}s + \int_0^t \sigma(s, X_s^\alpha, \mathcal{L}(X_s^\alpha), \alpha_s)\ \mathrm{d}W_t. \end{aligned} \tag{2.2.2}$$

Même si ces problèmes sont différents, ils sont néanmoins assez similaires. L'interprétation éclairante suivante est donnée par [CDL13; CD18a] : à partir de (2.2.1), si l'on optimise d'abord puis que l'on prend la limite $N \to +\infty$ on obtient un MFG, alors que si l'on prend d'abord la limite $N \to +\infty$ puis qu'on optimise on se retrouve avec un MFC. En général, les deux équilibres sont différents.

La théorie du champ moyen a plusieurs applications en mathématiques appliquées comme en finance quantitative avec l'exécution optimale et l'impact des prix [CL18], en économie avec le minage de bitcoins [Ber+21] ou la production de pétrole [CS17], dans la santé avec la propagation des épidémies [Lee+21], ou encore pour les réseaux sociaux [BTB16]. Dans le domaine de l'énergie, la multiplication des petits opérateurs, la décentralisation et les réseaux intelligents avec des flexibilités (gestion de la demande et stockage) ont inspiré plusieurs études utilisant la machinerie du champ moyen. Ces travaux se sont intéressés à la recharge intelligente [SWA21a; SWA21b], au stockage et aux flexibilités pour l'électricité [ABTM20; GG21a; GG21b], mais aussi à la formation des prix et aux échanges sur les marchés de l'électricité [FTT21; FTT20]. Ces articles considèrent souvent des modèles linéaires-quadratiques, c'est-à-dire avec une dynamique linéaire et des coûts quadratiques afin de pouvoir calculer la solution exacte. Mais si l'on veut utiliser des représentations plus réalistes, des méthodes numériques sont nécessaires afin d'obtenir une approximation de la solution. Comme précédemment dans le cas du contrôle stochastique, les réseaux de neurones sont un outil intéressant pour résoudre des problèmes non-linéaires de dimension moyenne à élevée. Enfin, pour une utilisation pratique, pour respecter des contraintes physiques ou des cadres réglementaires, il est parfois utile d'ajouter des contraintes d'état aux problèmes de contrôle à champ moyen. Nous allons considérer ces deux problèmes. La deuxième partie de notre thèse vise à répondre aux questions suivantes :

- Comment introduire de nouvelles méthodes de résolution numérique par apprentissage automatique ?
- Comment imposer des contraintes d'état dans un problème de contrôle à champ moyen ? Qu'en est-il des contraintes probabilistes ?

Nous passons en revue les conditions d'optimalité des problèmes de contrôle du champ moyen qui seront utiles pour leur résolution numérique. Des conditions d'optimalité similaires sont obtenues

pour les MFG. L'approche par programmation dynamique introduit une fonction de valeur sur $[0,T]\times\mathcal{P}_2(\mathbb{R}^d)$ dépendant de la loi de départ :

$$v(t,\mu) \;=\; \inf_{\alpha}\; \mathbb{E}_{t,\mu}\Big[\int_t^T f(X_s^\alpha,\mathcal{L}(X_s^\alpha),\alpha_s)\;\mathrm{d}s + g(X_T^\alpha,\mathcal{L}(X_T^\alpha))\Big],$$

(ici $\mathbb{E}_{t,\mu}[\cdot]$ est l'espérance conditonnelle sachant que la loi à l'instant t de X solution de (2.2.2) est égale à μ). Similairement au cadre du contrôle stochastique, cette fonction vérifie une EDP appelée équation maîtresse de Bellman [BFY13; PW17; PW18; CP19; DPT19] (voir [Car+19; CCD15] pour les MFGs):

$$\begin{cases} \partial_t v + \mathcal{H}(t,\mu,v,\partial_\mu v,\partial_x\partial_\mu v,\partial_\mu^2 v) &= 0, \qquad (t,\mu)\in[0,T)\times\mathcal{P}_2(\mathbb{R}^d),\\ v(T,\mu) &= G(\mu), \qquad \mu\in\mathcal{P}_2(\mathbb{R}^d), \end{cases} \tag{2.2.3}$$

avec $G(\mu)=\int g(x,\mu)\;\mu(\mathrm{d}x)$,

$$\mathcal{H}(t,\mu,y,\mathrm{z}(.),\gamma(.),\gamma_0(.,.)) = \int_{\mathbb{R}^d}\big[H(t,x,\mu,y,\mathrm{z}(x),\gamma(x))\big]\mu(\mathrm{d}x),$$

et

$$H(t,x,\mu,y,z,\gamma) = \inf_{a\in A}\big[b(t,x,\mu,a).z + f(x,\mu,a) + \frac{1}{2}\mathrm{tr}\big(\sigma\sigma^\intercal(t,x,\mu,a)\gamma\big)\big].$$

Dans l'équation précédente, $\partial_\mu v$ est la dérivée de Lions. Elle est définie grâce à la dérivée de Fréchet $[Dv](t,\xi)$ de la fonction liftée $\tilde{v}:(t,\xi)\in[0,T]\times L^2(\mathbb{R}^d)\mapsto v(t,\mathcal{L}(\xi))\in\mathbb{R}$ qui peut être représentée par théorème de representation de Riesz par $\partial_\mu v(t,\xi)\in L^2(\mathbb{R}^d)$ tel que $[Dv](t,\xi)(Y)=\mathbb{E}[\partial_\mu v(t,\xi).Y]\in\mathbb{R}$.

Le principe de Pontryagin [CD15] (voir [CD13] pour les MFGs) introduit l'Hamiltonien

$$H(t,x,\mu,y,z,\alpha) = b(t,x,\mu,\alpha)\cdot y + \mathrm{Tr}(\sigma(t,x,\mu,\alpha)z) + f(t,x,\mu,\alpha),$$

et donne des conditions d'optimalité sous la forme d'équations différentielles stochastiques progressives-rétrogrades de McKean-Vlasov (MKVFBSDEs):

$$\begin{cases} \mathrm{d}X_t &= b(t,X_t,\mathcal{L}(X_t),\hat{\alpha}_t)\;\mathrm{d}t + \sigma(t,X_t,\mathcal{L}(X_t),\hat{\alpha}_t)\;\mathrm{d}W_t\\ X_0 &= \xi\\ \mathrm{d}Y_t &= -\partial_x H(t,X_t,\mathcal{L}(X_t),Y_t,Z_t,\hat{\alpha}_t)\;\mathrm{d}t - \mathbb{E}[\partial_\mu H(t,\tilde{X}_t,\mathcal{L}(X_t),\tilde{Y}_t,\tilde{Z}_t,\widetilde{\hat{\alpha}}_t)]\;\mathrm{d}t + Z_t\;\mathrm{d}W_t\\ Y_T &= \partial_x g\left(X_T,\mu_T\right) + \mathbb{E}\Big[\partial_\mu g\left(\widetilde{X_T},\mu_T\right)\Big], \end{cases} \tag{2.2.4}$$

où $\hat{\alpha}_t = \underset{\alpha}{\mathrm{argmin}}\, H(t,X_t,\mathcal{L}(X_t),Y_t,Z_t,\alpha)$ et $(\tilde{X},\tilde{Y},\tilde{Z},\widetilde{\hat{\alpha}})$ est une copie indépendante de $(X,Y,Z,\hat{\alpha})$. Dans le cas des MFGs, les dérivées de Lions ∂_μ disparaissent dans (2.2.4). En ce qui concerne les MFGs, le papier fondateur [LL06b] caractérise l'équilibre grâce à un système couplé d'équations de Hamilton-Jacobi-Bellman et de Fokker-Planck. Ce système est utilisé pour construire des solutions approchées de MFGs par [ACD10].

Une première idée pour résoudre (2.2.2) est présentée dans [CL22]. Une approche par force brute consiste à chercher des contrôles par rétroaction optimaux en représentant le contrôle α_t par un réseau de neurones $\mathcal{A}^\theta:(t_i,x_i)\in[0,T]\times\mathbb{R}^d\mapsto\mathbb{R}^q$ et de minimiser directement le coût

discrétisé en temps :

$$\begin{aligned}
&\inf_{\theta} \frac{1}{N_s}\sum_{k=1}^{N_s}\left(\sum_{i=1}^{N} f(t_i, X_i^k, \bar{\mu}_i, \mathcal{A}^\theta(t_i, X_i^k))\ \Delta t + g(X_N^k, \bar{\mu}_N)\right) \qquad (2.2.5)\\
&X_{i+1}^k = X_i^k + b(t_i, X_i^k, \bar{\mu}_i, \mathcal{A}^\theta(t_i, X_i^k))\ \Delta t + \sigma(t_i, X_i^k, \bar{\mu}_i, \mathcal{A}^\theta(t_i, X_i^k))\ \Delta W_i^k,\\
&\bar{\mu}_i = \frac{1}{N_s}\sum_{k=1}^{N_s}\delta_{X_i^k},\ i = 0, \cdots, N-1,\\
&X_0^k = \xi^k,\ k = 0, \cdots, N_s,
\end{aligned}$$

où la loi est approchée par la mesure empirique des particules avec des mouvements Bronwiens indépendants $W^k,\ k = 1, \cdots, N_s$ et $\xi^k,\ k = 0, \cdots, N_s$ sont des tirages indépendants de la condition initiale ξ. Il s'agit de la version pour le contrôle à champ moyen des méthodes existantes [GM05; HE16].

2.2.2 Contributions pour l'approximation numérique

D'autres approches résolvent numériquement les conditions d'optimalité pour construire des solutions approchées. Nous décrivons dans la Section 2.2.2 deux approches fondées respectivement sur (2.2.4) et (2.2.3).

Une première idée naturelle pour la résolution par apprentissage du contrôle à champ moyen et des jeux à champs moyen, introduite par [FZ20; CL22], est d'étendre la méthode Deep BSDE pour l'aproximation des FBSDEs de McKean-Vlasov, venant du principe de Pontryagin (2.2.4)

$$\begin{cases}
X_{i+1}^k &= b(t_i, X_i^k, \bar{\mu}_i, \hat{\alpha}_i(X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k))\ \Delta t + \sigma(t_i, X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k, \hat{\alpha}_i(X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k))\ \Delta W_i^k,\\
X_0^k &= \xi^k,\ k = 0, \cdots, N_s,\\
Y_{i+1}^k &= Y_i^k - \partial_x H(t_i, X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k, \hat{\alpha}_i(X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k))\ \Delta t\\
&\quad -\frac{1}{N}\sum_{j=1}^{N}\partial_\mu H(t_i, X_i^j, \bar{\mu}_i, Y_i^j, Z_i^j, \hat{\alpha}_i(X_i^j, \bar{\mu}_i, Y_i^j, Z_i^j))\ \Delta t + \mathcal{Z}_i^{\theta_i}(X_i^k)\ \Delta W_i^k,\\
\bar{\mu}_i &= \frac{1}{N_s}\sum_{k=1}^{N_s}\delta_{X_i^k},\ i = 0, \cdots, N-1,
\end{cases}$$

avec $Y_0^k = \mathcal{Y}_0^\eta(X_0^k)$, un réseau de neurones $\mathcal{Y}_0^\eta$ et des réseaux de neurones $\mathcal{Z}_i^{\theta_i},\ i = 0, \cdots, N-1$. De plus $\hat{\alpha}_i^k(X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k) = \underset{\alpha}{\operatorname{argmin}}\, H(t_i, X_i^k, \bar{\mu}_i, Y_i^k, Z_i^k, \alpha)$. La méthode minimise la fonction de perte

$$\inf_{\eta, \theta_0, \cdots, \theta_N}\ \frac{1}{N}\sum_{k=1}^{N}\left|Y_N^k - \partial_x g\left(X_N^k, \mu_N\right) - \frac{1}{N}\sum_{j=1}^{N}\partial_\mu g\left(X_N^j, \mu_N\right)\right|^2,$$

par rapport aux paramètres des réseaux de neurones $\eta, \theta_0, \cdots, \theta_N$. Dans [GMW22], nous proposons des variantes et comparons tous les schémas en dimension 10 alors que les précédents articles testaient uniquement les schémas dans le cas unidimensionnel. Nos méthodes remplacent la mesure empirique par d'autres choix tels qu'une mesure estimée en ligne ou un réseau de neurones. Nous considérons également une version locale de l'algorithme qui résout un problème d'optimisation par pas de temps au lieu d'un unique problème plus complexe. Cependant, il semble que la méthode globale fonctionne mieux que cette version locale dans le contexte des MKVFBSDEs. Une analyse théorique avec des estimations d'erreur a posteriori est menée par [RSZ20]. Une méthode alternative dans le cas non régulier exploite la descente de gradient proximale pour résoudre le problème MFC [RSZ21].

Une autre méthode apparaît lorsque l'on résout l'équation maîtresse de Bellman (2.2.3). Bien sûr, comme il s'agit d'une équation en dimension infinie, une discrétisation doit être effectuée afin d'obtenir un schéma implémentable. Nous étudions ce problème dans le Chapitre 6. Nous

considérons ce problème pour une fonction générale $H(t,x,\mu,y,z)$ qui ne provient pas nécessairement d'un problème de contrôle à champ moyen. Mais la structure semilinéaire est nécessaire de manière à pouvoir utiliser des arguments de type EDSR. Nous utilisons une méthode particulaire et nous remplaçons cette équation par une EDP en dimension finie, en grande dimension. Cette équation est donnée par

$$\begin{cases} \partial_t v^N + \frac{1}{N}\sum_{i=1}^N H\big(t,x_i,\bar\mu(\boldsymbol{x}),v^N,ND_{x_i}v^N\big) + \dfrac{1}{2}\mathrm{tr}(\Sigma_N(t,\boldsymbol{x})D^2_{\boldsymbol{x}}v^N) \;=\; 0, \text{ on } [0,T)\times(\mathbb{R}^d)^N \\ v^N(T,\boldsymbol{x}) \;=\; G\big(\bar\mu(\boldsymbol{x})\big), \quad \boldsymbol{x}=(x_i)_{i\in[\![1,N]\!]}\in(\mathbb{R}^d)^N, \end{cases} \tag{2.2.6}$$

où $\bar\mu(.)$ est la mesure empirique définie par $\bar\mu(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^N \delta_{x_i}$, pour tout $\boldsymbol{x}=(x_1,\dots,x_N)$, $N \in \mathbb{N}^*$, et $\Sigma_N = (\Sigma_N^{ij})_{i,j\in[\![1,N]\!]}$ est la fonction à valeurs dans $\mathbb{R}^{Nd\times Nd}$ avec des matrices par bloc $\Sigma_N^{ij}(t,\boldsymbol{x}) = \sigma(t,x_i,\bar\mu(\boldsymbol{x}))\sigma^\intercal(t,x_j,\bar\mu(\boldsymbol{x}))\delta_{ij} + \sigma_0(t,x_i,\bar\mu(\boldsymbol{x}))\sigma_0^\intercal(t,x_j,\bar\mu(\boldsymbol{x})) \in \mathbb{R}^{d\times d}$. Cette équation est étudiée dans [GMS21] qui démontre la convergence de ses solutions de viscosité vers la solution de viscosité de l'équation maîtresse quand N tend vers $+\infty$, sous certaines conditions sur H et les coefficients de volatilité σ,σ_0. Notre contribution est de donner une vitesse de convergence grâce à des arguments probabilistes. L'EDP de dimension finie (2.2.6) est liée à une EDSR Markovienne à travers la formule de Feynman-Kac non-linéaire. Le système de particule associé est donné par

$$\mathrm{d}\boldsymbol{X}_t^N = \sigma_N(t,\boldsymbol{X}_t^N)\,\mathrm{d}\boldsymbol{W}_t + \boldsymbol{\sigma}_0(t,\boldsymbol{X}_t^N)\,\mathrm{d}W_t^0,$$

où σ_N est la matrice diagonale par blocs avec blocs diagonaux $\sigma_N^{ii}(t,\boldsymbol{x}) = \sigma(t,x_i,\bar\mu(\boldsymbol{x}))$, $\boldsymbol{\sigma}_0 = (\sigma_0^i)_{i\in[\![1,N]\!]}$ est la fonction à valeurs dans $(\mathbb{R}^{d\times m})^N$ avec $\boldsymbol{\sigma}_0^i(t,\boldsymbol{x}) = \sigma_0(t,x_i,\bar\mu(\boldsymbol{x}))$, pour $\boldsymbol{x}=(x_i)_{i\in[\![1,N]\!]}$, $\boldsymbol{W}=(W^1,\dots,W^N)$ où W^i, $i=1,\dots,N$, est un mouvement Brownien n-dimensionnel, indépendant d'un mouvement Brownien m-dimensionnel W^0 sur un espace probabilisé filtré $(\Omega,\mathcal{F},\mathbb{F}=(\mathcal{F}_t)_{0\le t\le T},\mathbb{P})$ et où les conditiones initiales du système de particules, $X_0^{i,N}$, $i=1,\dots,N$, sont i.i.d. avec distribution μ_0. La composante rétrograde est définie par la paire de processus $(Y^N,\boldsymbol{Z}^N=(Z^{i,N})_{i\in[\![1,N]\!]})$ à valeurs dans $\mathbb{R}\times(\mathbb{R}^d)^N$, solution de

$$\begin{aligned} Y_t^N =\; & G\big(\bar\mu(\boldsymbol{X}_T^N)\big) + \frac{1}{N}\sum_{i=1}^N\int_t^T H_b(s,X_s^{i,N},\bar\mu(\boldsymbol{X}_s^N),Y_s^N,NZ_s^{i,N})\,\mathrm{d}s \\ & - \sum_{i=1}^N\int_t^T (Z_s^{i,N})^\intercal\sigma\big(s,X_s^{i,N},\bar\mu(\boldsymbol{X}_s^N)\big)\,\mathrm{d}W_s^i, \\ & - \sum_{i=1}^N\int_t^T (Z_s^{i,N})^\intercal\sigma_0\big(s,X_s^{i,N},\bar\mu(\boldsymbol{X}_s^N)\big)\,\mathrm{d}W_s^0, \qquad 0\le t\le T. \end{aligned}$$

Les principales difficultés dans l'étude de la convergence de cette EDP vers l'équation de Bellman maître sont :

- Le facteur N devant le gradient de la non-linéarité H_b qui fait exploser la régularité de Lipschitz par rapport au gradient.
- L'explosion de la dimension de l'EDP.

Ces difficultés peuvent être contournées grâce à une procédure de linéarisation associée à un changement de mesure de Girsanov. Nous étudions l'erreur trajectorielle sur v :

$$\mathcal{E}_N^{\mathrm{y}} := \sup_{0\le t\le T}\big|Y_t^N - v(t,\bar\mu(\boldsymbol{X}_t^N))\big|,$$

et l'erreur L^2 sur sa L-derivative

$$\big\|\mathcal{E}_N^{\mathrm{z}}\big\|_2 := \frac{1}{N}\sum_{i=1}^N\Big(\int_0^T \mathbb{E}\big|NZ_t^{i,N} - \partial_\mu v(t,\bar\mu(\boldsymbol{X}_t^N))(X_t^{i,N})\big|^2\mathrm{d}t\Big)^{\frac12}.$$

Sous des hypothèses concernant la régularité de Lipschitz des paramètres, l'existence d'une solution classique suffisamment régulière avec une croissance linéaire et une dérivée de Lions du second ordre bornée, nous sommes capables de trouver une vitesse de convergence pour la solution. Des hypothèses supplémentaires sont nécessaires pour étudier l'erreur $\left\|\mathcal{E}_N^{\mathrm{z}}\right\|_2$, comme l'ellipticité de la volatilité commune σ_0 et une structure linéaire concernant le gradient de la solution de l'EDP.

Theorem 2.2.1. *Sous les Hypothèses 6.2.1 et 6.2.2, nous avons $\mathbb{P}$-presque surement*

$$\mathcal{E}_N^{\mathrm{y}} \leq \frac{C_y}{N},$$

où $C_y = \frac{T}{2} e^{[H_b]_1 T} L \|\sigma\|_\infty^2$, avec $\|\sigma\|_\infty = \sup_{(t,x,\mu)\in[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)} |\sigma(t,x,\mu)|$.

Theorem 2.2.2. *Sous les Hypothèses 6.2.1, 6.2.2 et 6.2.3, nous avons*

$$\left\|\mathcal{E}_N^{\mathrm{z}}\right\|_2 \leq \frac{C_z}{N^{\frac{1}{2}}},$$

où $C_z = \|\sigma^+\|_\infty \sqrt{2T([H_1]_1 + [H_2]_1 L)\bar{C}_y^2 + \bar{C}_y T \|\sigma\|_\infty^2 L + \frac{2}{c_0}\bar{C}_y^2 T \|H_2\|_\infty^2}$ et $\bar{C}_y = \frac{T}{2} e^{([H_1]_1 + [H_2]_1 L)T} L \|\sigma\|_\infty^2$.

En remarquant les propriétés de symétrie de l'EDP, nous concevons dans le Chapitre 7 un schéma adapté à ces symétries qui nous permet de travailler en haute dimension, avec 1000 ou 10000 particules. Plus précisément, la solution v^N of (2.2.6) est invariante par permutation de ces arguments, c'est-à-dire que pour $x_1, \cdots, x_N \in (\mathbb{R}^d)^N$ et une permutation σ sur $\{1, \cdots, N\}$:

$$v^N(x_1, \cdots, x_N) = v^N(x_{\sigma(1)}, \cdots, x_{\sigma(N)}).$$

Cette symétrie implique également une structure pour sa dérivée spatiale. Ces deux comportements nous ont incités à concevoir une méthode adaptée aux EDP symétriques. L'idée est de considérer des réseaux de neurones symétriques, à savoir DeepSets [Zah+17] et PointNet [Qi+17] qui ont été introduits par la communauté de l'apprentissage automatique notamment afin de classifier efficacement les nuages de points.

Une fonction de *réseau de neurones symétrique*, notée $\mathcal{U}$ dans $\mathcal{S}_{d,\ell,m,k,d'}^{\mathfrak{s},N,\rho}$, est une fonction échangeable à valeurs dans $\mathbb{R}^{d'}$ à l'ordre N sur $\mathbb{R}^d$, de la forme :

$$\mathcal{U}(\boldsymbol{x}) = \psi\big(\mathfrak{s}((\varphi(x_i))_{i\in[\![1,N]\!]})\big), \quad \text{for } \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N,$$

où $\varphi \in \mathcal{N}_{d,\ell,m,k}^{\rho}$, $\psi \in \mathcal{N}_{k,\ell,m,d'}^{\rho}$ (ici, pour simplifier les notations, nous supposons que le nombre de couches cachées et de neurones de φ et ψ est le même, mais en pratique, ils peuvent être différents), et $\mathfrak{s}$ est une fonction échangeable à valeur dans $\mathbb{R}^k$ à l'ordre N sur $\mathbb{R}^k$.

Le réseau DeepSet est donné par

$$\mathcal{U}(\boldsymbol{x}) = \psi\Big(\frac{1}{N}\sum_{i=1}^{N}((\varphi(x_i))_{i\in[\![1,N]\!]})\Big), \quad \text{for } \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N,$$

alors que le réseau PointNet network vérifie

$$\mathcal{U}(\boldsymbol{x}) = \psi\big(\max_{i=1\cdots N}((\varphi(x_i))_{i\in[\![1,N]\!]})\big), \quad \text{for } \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N.$$

Ces réseaux sont symétriques par construction et sont suffisamment expressifs dans l'espace des fonctions symétriques. Plus précisément, en combinant le théorème 2.9 de [Wag+19] avec le théorème 2 de [Hor91], nous obtenons le théorème d'approximation suivant pour les DeepSets.

Approximation universelle pour les réseaux DeepSets. Soit $\mathfrak{s}$ la fonction somme. L'ensemble $\cup_{m=1}^{\infty}\mathcal{S}_{d,\ell,m,N+1,d'}^{\mathfrak{s},N,\rho}$ approche n'importe quelle fonction N-échangeable w arbitrairement bien sur

n'importe quel compact de $K \subset \mathbb{R}^d$, dès que ρ est continu, borné et non-constant: pour tout $\varepsilon > 0$, $N \in \mathbb{N}^*$, il existe $\mathcal{U} \in \cup_{m=1}^{\infty} \mathcal{S}_{d,\ell,m,N+1,d'}^{\mathfrak{s},N,\rho}$ tel que

$$\big|w(\boldsymbol{x}) - \mathcal{U}(\boldsymbol{x})\big| \leq \varepsilon \quad \forall \boldsymbol{x} \in K^N.$$

Notez qu'a priori la dimension k de l'espace latent doit être prise égale à $N+1$.

Alternativement, en combinant le Théorème 1 de [Qi+17] avec le Théorème 2 de [Hor91], nous obtenons un théorème d'approximation universelle pour PointNet en dimension un.

Approximation universelle pour les réseaux PointNet. Soit $\mathfrak{s}$ la fonction max. L'ensemble $\cup_{m=1}^{\infty} \cup_{k=1}^{\infty} \mathcal{S}_{1,\ell,m,k,d'}^{\mathfrak{s},N,\rho}$ approche n'importe quelle function N-échangeable Hausdorff continue w (vue comme une fonction sur les ensembles) arbitrairement bien sur tout compact de $K \subset \mathbb{R}$, dès que ρ est continue, bornée et non constante: pour tout $\varepsilon > 0$, $N \in \mathbb{N}^*$, il existe $\mathcal{U} \in \cup_{m=1}^{\infty} \cup_{k=1}^{\infty} \mathcal{S}_{1,\ell,m,k,d'}^{\mathfrak{s},N,\rho}$ tel que

$$\big|w(S) - \mathcal{U}(\boldsymbol{x})\big| \leq \varepsilon, \quad \forall S \subset K,\ S = \{x_1, \cdots, x_N\}.$$

Notez ici qu'a priori la dimension k de l'espace latent doit être choisie aussi grande que nécessaire.

Nous adaptons le schéma DBDP [HPW20] à ces réseaux de neurones symétriques. Contrairement aux réseaux de neurones classiques, notre méthode converge même dans les cas difficiles de haute dimension. Plusieurs variantes peuvent être mises en œuvre pour l'approximation de la composante Z de l'EDSR, correspondant au gradient de la solution. Nous pouvons soit considérer la dérivée des réseaux de neurones approximant la composante Y, soit utiliser une architecture satisfaisant les propriétés de symétrie du gradient d'une fonction symétrique.

2.2.3 Ajouter des contraintes d'état probabilistes

Dans de nombreuses situations pratiques, il peut être utile d'imposer des contraintes d'état à l'état contrôlé. Les contraintes peuvent être physiques (telles que la non négativité, le caractère borné...), peuvent être imposées par le cadre réglementaire, ou être utiles pour trouver une solution particulière d'intérêt. Nous nous référons à [ST02; BEI10; Gel+13; CYZ20; PTZ21; Bal+21] pour des applications spécifiques de contraintes probabilistes, notamment en finance, et souvent écrites sous forme d'espérance. Un exemple que nous considérons est celui du contrôle optimal d'une batterie pour le stockage d'électricité renouvelable, soumis au hasard à la fois au travers de la production intermittente mais aussi des prix de marché. [ABTM20] étudie un tel problème dans un cadre de champ moyen sans contraintes physiques sur la batterie, afin d'obtenir un modèle linéaire-quadratique explicitement résoluble. Dans le Chapitre 8, nous résoudrons numériquement un problème proche avec des contraintes sur la taille du stockage et les capacités d'injection et de soutirage.

Dans le contexte des problèmes de champ moyen, plusieurs articles considèrent les jeux à champ moyen où l'état est contraint de rester dans un ensemble compact [CC18; CCC18; FH20; GM21; AM21], soit à tout moment, soit seulement à un temps terminal T. Les contraintes terminales en loi sont également considérées par [BDK20; Dau20] respectivement pour la commande de dynamique de McKean-Vlasov et la commande stochastique. Pour le contrôle à champ moyen, du point de vue du contrôle de l'équation de Fokker-Planck, les travaux [Bon19; BF21] sont en mesure d'imposer des contraintes terminales ou continues sur la distribution de l'état, en s'appuyant sur le principe de Pontryagin, dans le cas déterministe sans diffusion. Un coût du type du contrôle à champ moyen pour une diffusion classique mais avec des contraintes d'état probabilistes est également étudié dans [Dau21].

Nous considérons dans le Chapitre 8 le cas général des contraintes en temps continu et en temps discret sur la loi de probabilité de l'état, qui contient par exemple en particulier la contrainte terminale en loi et les contraintes en ensemble compact. Nous nous appuyons sur l'approche level-set de [BPZ15; BPZ16; ABZ13]. Cela revient à introduire un problème auxiliaire non contraint avec une variable d'état supplémentaire. Grâce à un résultat de représentation, on peut lier la solution de ce problème à la solution du problème contraint original. Cette approche est également utile à des fins numériques puisque les méthodes existantes peuvent être appliquées au problème auxiliaire. Par conséquent, nous sommes également en mesure de développer des schémas numériques pour les problèmes de contrôle à champ moyen sous contrainte de loi. Nous considérons le coût et la dynamique suivants :

$$J(X_0,\alpha)=\mathbb{E}\Big[\int_0^T f\big(s,X_s^\alpha,\alpha_s,\mathbb{P}_{(X_s^\alpha,\alpha_s)}\big)\,\mathrm{d}s+g\big(X_T^\alpha,\mathbb{P}_{X_T^\alpha}\big)\Big]$$
$$X_t^\alpha=X_0+\int_0^t b\big(s,X_s^\alpha,\alpha_s,\mathbb{P}_{(X_s^\alpha,\alpha_s)}\big)\,\mathrm{d}s+\int_0^t \sigma(s,X_s^\alpha,\alpha_t,\mathbb{P}_{(X_s^\alpha,\alpha_s)})\,\mathrm{d}W_s,$$

et des contraintes probabilistes sous la forme

$$\Psi(t,\mathbb{P}_{X_t^\alpha})\le\ 0,\quad 0\le t\le T,$$

où $\Psi=(\Psi^l)_{1\le l\le k}$ est une fonction de $[0,T]\times\mathcal{P}_2(\mathbb{R}^d)$ dans $\mathbb{R}^k$. Ici, la contrainte multi-dimensionnelle $\Psi(t,\mu)\le 0$ doit être comprise composante par composante, i.e., $\Psi^l(t,\mu)\le 0$, $l=1,\cdots,k$. Le problème d'intérêt est donc

$$V^\Psi:=\inf_{\alpha\in\mathcal{A}}\big\{J(X_0,\alpha):\Psi(t,\mathbb{P}_{X_t^\alpha})\le 0,\ \forall\ t\in[0,T]\big\}. \tag{2.2.7}$$

Nous introduisons une variable d'état déterministe

$$Z_t^{z,\alpha}:=z-\mathbb{E}\Big[\int_0^t f\big(s,X_s^\alpha,\alpha_s,\mathbb{P}_{(X_s^\alpha,\alpha_s)}\big)\,\mathrm{d}s\Big]=z-\int_0^t \widehat{f}\big(s,\mathbb{P}_{(X_s^\alpha,\alpha_s)}\big)\,\mathrm{d}s,\quad 0\le t\le T,$$

et le problème auxiliaire sans contrainte

$$\mathcal{Y}^\Psi:z\in\mathbb{R}\mapsto\inf_{\alpha\in\mathcal{A}}\Big[\{\widehat{g}(\mathbb{P}_{X_T^\alpha})-Z_T^{z,\alpha}\}_+ +\sum_{l=1}^k\sup_{s\in[0,T]}\{\Psi^l(s,\mathbb{P}_{X_s^\alpha})\}_+\Big], \tag{2.2.8}$$

avec la notation $\{x\}_+=\max(x,0)$ pour la partie positive. Nous observons que $\mathcal{Y}^\Psi(z)\ge 0$. Definissons l'infimum du zéro level-set

$$\mathcal{Z}^\Psi:=\inf\{z\in\mathbb{R}\ |\ \mathcal{Y}^\Psi(z)=0\}. \tag{2.2.9}$$

Nous prouvons deux résultats de représentation du problème contraint (2.2.7) par le problème sans contrainte (2.2.8).

Theorem 2.2.3. *1. Si pour un certain $z\in\mathbb{R}$ $\exists\ \alpha\in\mathcal{A}$ s.t. $\widehat{g}(\mathbb{P}_{X_T^\alpha})\le Z_T^{z,\alpha}$, $\Psi(s,\mathbb{P}_{X_s^\alpha})\le 0$, $\forall\ s\in[0,T]$ alors $\mathcal{Y}^\Psi(z)=0$.*

2. Si V^Ψ est fini alors $\mathcal{Y}^\Psi(V^\Psi)=0$. Donc $\mathcal{Z}^\Psi\le V^\Psi$.

3. Définissons $1_k=(1,\cdots,1)\in\mathbb{R}^k$. Nous avons la borne supérieure

$$V^\Psi\le\inf_{\varepsilon>0}\mathcal{Z}^{\Psi+\varepsilon 1_k}.$$

Theorem 2.2.4. *Supposons que $V^\Psi<\infty$. Alors nous avons la représentation*

$$\mathcal{Z}^\Psi=V^\Psi.$$

De plus les contrôles ε-optimaux α^ε pour le problème auxiliaire $\mathcal{Y}^\Psi(V^\Psi)$ sont ε-admissibles ε-optimaux pour le problème originel dans le sens que

$$J(X_0,\alpha^\varepsilon)\le V^\Psi+\varepsilon,\ \sup_{0\le s\le T}\Psi(s,\mathbb{P}_{X_s^{\alpha^\varepsilon}})\le\varepsilon.$$

Une discussion sur les contrôles en boucle ouverte et en boucle fermée est donnée dans 8.3. Nous nous limitons, pour la partie approximation numérique, aux commandes déterministes markoviennes à rétroaction (ou en boucle fermée), mais nous donnons des hypothèses sous lesquelles restreindre le choix des commandes à cet espace se fait sans perte de généralité. En général, les contrôles adaptés (en boucle ouverte) atteignent en effet un coût plus faible que les contrôles en boucle fermée. Ces résultats nous permettent de résoudre numériquement le problème sans contrainte 2.2.8 et la minimisation par level-set (2.2.9) afin de résoudre le problème contraint (2.2.7). Nous adaptons l'approche (2.2.5) de [CL22] pour proposer un schéma implémentable.

Part I

Numerical resolution of non-linear partial differential equations

Chapter 3

Neural networks-based algorithms for stochastic control and PDEs in finance

This chapter is based on the paper [GPW22a]
M. Germain, H. Pham, and X. Warin. "Neural networks based algorithms for stochastic control and PDEs in finance". In: to appear in Machine Learning And Data Sciences For Financial Markets: A Guide To Contemporary Practices. Ed. by A. Capponi. and C.A. Lehalle Cambridge University Press, 2022. Chap. New Frontiers for Stochastic Control in Finance.

We provide in this chapter a survey of machine learning methods for the resolution of stochastic control problems. We also describe new schemes for fully nonlinear PDEs, by combining ideas from the multistep scheme from Chapter 4, from the 2DBDP method of Chapter 5 and the Malliavin weights used by [FTW11]. Our main focus are problems coming from financial mathematics, especially derivatives pricing and portfolio selection and we demonstrate on numerical examples that our second order mutlistep schemes improve the one step scheme 2DBDP of Chapter 5. We also compare the Deep BSDE method [HJE18] with the DBDP scheme [HPW20] on a derivatives pricing example in high dimension.

Abstract

This chapter presents machine learning techniques and deep reinforcement learning-based algorithms for the efficient resolution of nonlinear partial differential equations and dynamic optimization problems arising in investment decisions and derivative pricing in financial engineering. We survey recent results in the literature, present new developments, and compare the different schemes illustrated by numerical tests on various financial applications. We conclude by highlighting some future research directions.

3.1 Breakthrough in the resolution of high dimensional non-linear problems

The numerical resolution of control problems and nonlinear PDEs– arising in several financial applications such as portfolio selection, hedging, or derivatives pricing–is subject to the so-called "curse of dimensionality", making impractical the discretization of the state space in dimension greater than 3 by using classical PDE resolution methods such as finite differences schemes. Probabilistic regression Monte-Carlo methods based on a Backward Stochastic Differential Equation (BSDE) representation of semilinear PDEs have been developed in [Zha04], [BT04], [GLW05] to overcome this obstacle. These mesh-free techniques are successfully applied upon dimension 6 or 7, nevertheless, their use of regression methods requires a number of basis functions growing fastly with the dimension. What can be done to further increase the dimension of numerically solvable problems?

A breakthrough with deep learning based-algorithms has been made in the last five years towards this computational challenge, and we mention the recent survey by [Bec+20]. The main interest in the use of machine learning techniques for control and PDEs

is the ability of deep neural networks to efficiently represent high dimensional functions without using spatial grids, and with no curse of dimensionality [Gro+18], [Hut+20].

Although the use of neural networks for solving PDEs is not new, see e.g. [DPT94], the approach has been successfully revived with new ideas and directions. Neural networks have known a increasing popularity since the works on Reinforcement Learning for solving the game of Go by Google DeepMind teams.

These empirical successes and the introduced methods allow to solve control problems in moderate or large dimension. Moreover, recently developed open source libraries like Tensorflow and Pytorch also offer an accessible framework to implement these algorithms.

A first natural use of neural networks for stochastic control concerns the discrete time setting, with the study of Markov Decision Processes, either in a brute force fashion or by using dynamic programming approaches.

In the continuous time setting, and in the context of PDE resolution, we present various methods. A first kind of schemes is rather generic and can be applied to a variety of PDEs coming from a large range of applications. Other schemes rely on BSDE representations, strongly linked to stochastic control problems. In both cases, numerical evidence seems to indicate that the methods can be used in large dimension, greater than 10 and up to 1000 in certain studies. Some theoretical results also illustrates the convergence of specific algorithms. These advances pave the way for new methods dedicated to the study of large population games, studied in the context of mean field games and mean field control problems.

The outline of this article is the following. We first focus on some schemes for discrete time control in Section 3.2 before presenting generic machine learning schemes for PDEs in Subsection 3.3.1. Then we review BSDE-based machine learning methods for semilinear equations in Subsection 3.3.2. Existing algorithms for fully non-linear PDEs are detailed in Subsection 3.3.2 before presenting new BSDE schemes designed to treat this more difficult case. Numerical tests on CVA pricing and portfolio selection are conducted in Section 3.4 to compare the different approaches.

Finally, we highlight in Section 3.5 further directions and perspectives including recent advances for the resolution of mean field games and mean field control problems with or without model.

3.2 Deep learning approach for stochastic control

We present in this section some recent breakthrough in the numerical resolution of stochastic control in high dimension by means of machine learning techniques. We consider a model-based setting in discrete-time, i.e., a Markov decision process, that could possibly be obtained from the time discretization of a continuous-time stochastic control problem.

Let us fix a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ equipped with a filtration $\mathbb{F} = (\mathcal{F}_t)_t$ representing the available information at any time $t \in \mathbb{N}$ ($\mathcal{F}_0$ is the trivial σ-algebra). The evolution of the system is described by a model dynamics for the state process $(X_t)_{t\in\mathbb{N}}$ valued in $\mathcal{X} \subset \mathbb{R}^d$:

$$X_{t+1} = F(X_t, \alpha_t, \varepsilon_{t+1}), \quad t \in \mathbb{N}, \tag{3.2.1}$$

where $(\varepsilon_t)_t$ is a sequence of i.i.d. random variables valued in E, with ε_{t+1} $\mathcal{F}_{t+1}$-measurable containing all the noisy information arriving between t and $t+1$, and $\alpha = (\alpha_t)_t$ is the control process valued in $A \subset \mathbb{R}^q$. The dynamics function F is a measurable function from $\mathbb{R}^d \times \mathbb{R}^q \times E$ into $\mathbb{R}^d$, and assumed to be known. Given a running cost function f, a finite horizon $T \in \mathbb{N}^*$, and a terminal cost function, the problem is to minimize over control process α a functional cost

$$J(\alpha) = \mathbb{E}\Big[\sum_{t=0}^{T-1} f(X_t, \alpha_t) + g(X_T)\Big]. \tag{3.2.2}$$

In some relevant applications, we may require constraints on the state and control in the form: $(X_t, \alpha_t) \in \mathcal{S}$, $t \in \mathbb{N}$.

for some subset $\mathcal{S}$ of $\mathbb{R}^d \times \mathbb{R}^q$. This can be handled by relaxing the state/constraint and introducing into the costs a penalty function $L(x,a)$: $f(x,a) \leftarrow f(x,a) + L(x,a)$, and $g(x) \leftarrow g(x) + L(x,a)$. For example, if the constraint set is in the form: $\mathcal{S} = \{(x,a) \in \mathbb{R}^d \times \mathbb{R}^q : h_k(x,a) = 0, k = 1, \ldots, p, h_k(x,a) \geq 0, k = p+1, \ldots, m\}$, then one can take as penalty functions:

$$L(x,a) = \sum_{k=1}^{p} \mu_k |h_k(x,a)|^2 + \sum_{k=p+1}^{m} \mu_k \max(0, -h_k(x,a)),$$

where μ_k are penalization parameters (large in practice) see e.g. [HE16].

3.2.1 Global approach

The method consists simply in approximating at any time t, the feedback control, i.e. a function of the state process, by a neural network (NN):

$$\alpha_t \simeq \pi^{\theta_t}(X_t), \quad t = 0, \ldots, T-1,$$

where π^θ is a feedforward neural network on $\mathbb{R}^d$ with parameters θ, and then to minimize over the global set of parameters $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_{T-1})$ the quantity (playing the role of loss function)

$$\tilde{J}(\boldsymbol{\theta}) = \mathbb{E}\Big[\sum_{t=0}^{T-1} f(X_t^{\boldsymbol{\theta}}, \pi^{\theta_t}(X_t^{\boldsymbol{\theta}})) + g(X_T^{\boldsymbol{\theta}})\Big],$$

where $X^{\boldsymbol{\theta}}$ is the state process associated with the NN feedback controls:

$$X_{t+1}^{\boldsymbol{\theta}} = F(X_t^{\boldsymbol{\theta}}, \pi^{\theta_t}(X_t^{\boldsymbol{\theta}}), \varepsilon_{t+1}), \quad t = 0, \ldots, T-1.$$

This basic idea of approximating control by parametric function of the state was proposed in [GM05], and updated with the use of (deep) neural networks by [HE16]. This method met success due to its simplicity and the easy accessibility of common libraries like TensorFlow for optimizing the parameters of the neural networks. Some recent extensions of this approach dealt with stochastic control problems with delay, see [HH21]. However, such global optimization over a huge set of parameters $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_{T-1})$ may suffer from being stuck in suboptimal traps and thus does not converge, especially for large horizon T. An alternative is to consider controls $\alpha_t \simeq \pi^\theta(t, X_t)$, $t = 0, \ldots, T-1$, with a single neural network π^θ giving more stabilized results as studied by [FMW21]. We focus here on feedback controls, which is not a restriction as we are in a Markov setting. For path-dependent control problems, we may consider recurrent neural networks to take into consideration the past of state trajectory as input of the policy.

3.2.2 Backward dynamic programming approach

In [Bac+21], the authors propose methods that combine ideas from numerical probability and deep reinforcement learning. Their algorithms are based on the classical dynamic programming (DP), (deep) neural networks for the approximation/learning of the optimal policy and value function, and Monte-Carlo regressions with performance and value iterations.

The first algorithm, called NNContPI, is a combination of dynamic programming and the approach in [HE16]. It learns sequentially the control by NN $\pi^\theta(.)$ and performance iterations, and is designed as follows:

Algorithm 1: NNContPI

Input: the training distributions $(\mu_t)_{t=0}^{T-1}$;
Output: estimates of the optimal strategy $(\hat{\pi}_t)_{t=0}^{T-1}$;
for $t = T-1, \ldots, 0$ **do**

Compute $\hat{\pi}_t := \pi^{\hat{\theta}_t}$ with

$$\hat{\theta}_t \in \arg\min_\theta \mathbb{E}\Big[f\big(X_t, \pi^\theta(X_t)\big) + \sum_{s=t+1}^{T-1} f\big(X_s^\theta, \hat{\pi}_s(X_s^\theta)\big) + g\big(X_T^\theta\big)\Big]$$

where $X_t \sim \mu_t$ and where $\big(X_s^\theta\big)_{s=t+1}^T$ is defined by induction as:

$$\begin{cases} X_{t+1}^\theta &= F\big(X_t, \pi^\theta(X_t), \varepsilon_{t+1}\big), \\ X_{s+1}^\theta &= F\big(X_s^\theta, \hat{\pi}_s(X_s^\theta), \varepsilon_{s+1}\big), \quad \text{for } s = t+1, \ldots, T-1. \end{cases}$$

The second algorithm, refereed to as Hybrid-Now, combines optimal policy estimation by neural networks and dynamic programming principle, and relies on an hybrid procedure between value and performance iteration to approximate the value function by neural network $\Phi^\eta(.)$ on $\mathbb{R}^d$ with parameters η.

The convergence analysis of Algorithms NNContPI and Hybrid-Now are studied in [Hur+21], and various applications in finance are implemented in [Bac+21]. These algorithms are well-designed for control problems with continuous control space $A = \mathbb{R}^q$ or a ball in $\mathbb{R}^q$. In the case where the control space A is finite, it is relevant to randomize controls, and then use classification methods by approximating the distribution of controls with neural networks and Softmax activation function.

Algorithm 2: Hybrid-Now

Input: the training distributions $(\mu_t)_{t=0}^{T-1}$;
Output:
– estimate of the optimal strategy $(\hat{\pi}_t)_{t=0}^{T-1}$;
– estimate of the value function $(\hat{V}_t)_{t=0}^{T-1}$;
Set $\hat{V}_T = g$;
for $t = T-1, \dots, 0$ **do**
Compute:

$$\hat{\theta}_t \in \arg\min_{\theta} \mathbb{E}\Big[f\big(X_t, \pi^{\theta}(X_t)\big) + \hat{V}_{t+1}(X_{t+1}^{\theta})\Big]$$

where $X_t \sim \mu_t$, and $X_{t+1}^{\theta} = F(X_t, \pi^{\theta}(X_t), \varepsilon_{t+1})$;
Set $\hat{\pi}_t := \pi^{\hat{\theta}_t}$; ▷ $\hat{\pi}_t$ is the estimate of the optimal policy at time t
Compute

$$\hat{\eta}_t \in \arg\min_{\eta} \mathbb{E}\Big|f(X_t, \hat{\pi}_t(X_t)) + \hat{V}_{t+1}(X_{t+1}^{\hat{\theta}_t}) - \Phi^{\eta}(X_t)\Big|^2.$$

Set $\hat{V}_t = \Phi^{\hat{\eta}_t}$; ▷ $\hat{V}_t$ is the estimate of the value function at time t

3.3 Machine learning algorithms for nonlinear PDEs

By change of time scale, Markov decision process (3.2.1)-(3.2.2) can be obtained from the time discretization of a continuous-time stochastic control problem with controlled diffusion dynamics on $\mathbb{R}^d$

$$dX_t = b(X_t, \alpha_t)dt + \sigma(X_t, \alpha_t)dW_t,$$

and cost functional to be minimized over control process α valued in A

$$J(\alpha) = \mathbb{E}\Big[\int_0^T f(X_t, \alpha_t)dt + g(X_T)\Big].$$

In this case, it is well-known, see e.g. [Pha09], that the dynamic programming Bellman equation leads to a PDE in the form

$$\begin{cases} \partial_t u + H(x, D_x u, D_x^2 u) = 0, \text{ on } [0,T) \times \mathbb{R}^d \\ u(T,.) = g \quad \text{on } \mathbb{R}^d, \end{cases}$$

where $H(x,z,\gamma) = \inf_{a \in A}\big[b(x,a).z + \frac{1}{2}\mathrm{tr}(\sigma\sigma^{\intercal}(x,a)\gamma) + f(x,a)\big]$ is the so-called Hamiltonian function. The numerical resolution of such class of second-order parabolic PDEs will be addressed in this section.

3.3.1 Deterministic approach by neural networks

In the schemes below, differential operators are evaluated by automatic differentiation of the network function approximating the solution of the PDE. Machine learning libraries such as Tensorflow or Pytorch allow to efficiently compute these derivatives. The studied PDE problem is

$$\begin{cases} \partial_t u + \mathcal{F} u = 0 & \text{on } [0,T) \times \Lambda \\ u(T,\cdot) = g & \text{on } \Lambda \\ u(t,x) = h(t,x) & \text{on } [0,T) \times \partial\Lambda, \end{cases}$$

with $\mathcal{F}$ a space differential operator, Λ a subset of $\mathbb{R}^d$.

• Deep Galerkin Method [SS18].

The Deep Galerkin Method is a meshfree machine learning algorithm to solve PDEs on a domain, eventually with boundary conditions. The principle is to sample time and space points according to a training measure, e.g. uniform on a bounded domain, and minimize a performance measure quantifying how well a neural network satisfies the differential operator and boundary conditions. The method consists in minimizing over neural network $\mathcal{U} : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$, the L^2 loss

$$\mathbb{E}|\partial_t \mathcal{U}(\tau,\kappa) + \mathcal{F}\mathcal{U}(\tau,\kappa)|^2 + \mathbb{E}|\mathcal{U}(T,\xi) - g(\xi)|^2 + \mathbb{E}|\mathcal{U}(\tau,\kappa) - h(\tau,\kappa)|^2$$

with κ, τ, ξ independent random variables in $\Lambda \times [0,T) \times \partial\Lambda$. [SS18] also prove a convergence result (without rate) for the Deep Galerkin method. This method is tested on financial problems by [AA+18]. A major advantage to this method is its adaptability to a large range of PDEs with or without boundary conditions. Indeed the loss function is straightforwardly modified according to changes in the constraints one wishes to enforce on the PDE solution. A related approach is the deep parametric PDE method, see [KJY20], and [GW20] applied to option pricing.

• Other approximation methods

(i) *Physics informed neural networks* [RPK19]. Physics informed neural networks use both data (obtained for a limited amount of samples from a PDE solution), and theoretical dynamics to reconstruct solutions from PDEs. The convergence of this method in the Second Order linear parabolic (or elliptic) case is proven in [SDEK20], see also [GAS20].

(ii) *Deep Ritz method* [EY18]. The Deep Ritz method focuses on the resolution of the variational formulation from elliptic problems where the integral is evaluated by randomly sampling time and space points, like in the Deep Galerkin method [SS18] and the minimization is performed over the parameters of a neural network. This scheme is tested on Poisson equation with different types of boundary conditions. [MZ19] studies the convergence of the Deep Ritz algorithm.

3.3.2 Probabilistic approach by neural networks

Semi-linear case

In this paragraph, we consider semilinear PDEs of the form

$$\begin{cases} \partial_t u + \mu \cdot D_x u + \frac{1}{2}\,\mathrm{Tr}(\sigma\sigma^\intercal D_x^2 u) \;=\; f(\cdot,\cdot,u,\sigma^\intercal D_x u) & \text{on } [0,T)\times\mathbb{R}^d \\ u(T,\cdot) \;=\; g & \text{on } \mathbb{R}^d\,. \end{cases} \tag{3.3.1}$$

for which we have the forward backward SDE representation

$$\begin{cases} Y_t &=& g(\mathcal{X}_T) - \int_t^T f(s,\mathcal{X}_s,Y_s,Z_s)\mathrm{d}s - \int_t^T Z_s\cdot \mathrm{d}W_s, \quad 0\le t\le T, \\ \mathcal{X}_t &=& \mathcal{X}_0 + \int_0^t \mu(s,\mathcal{X}_s)\mathrm{d}s + \int_0^t \sigma(s,\mathcal{X}_s)\mathrm{d}W_s, \end{cases} \tag{3.3.2}$$

via the (non-linear) Feynman-Kac formula: $Y_t = v(t,X_t)$, $Z_t = \sigma^\intercal(t,\mathcal{X}_t) D_x v(t,X_t)$, $0 \le t \le T$, see [PP90].

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 $\Delta W_j := W_{t_{j+1}} - W_{t_j}$, $j = 0, \ldots, N$. Sample paths of $(X_i)_i$ act as training data in the machine learning setting. Thus our training set can be chosen as large as desired, which is relevant for training purposes as it does not lead to overfitting.

The time discretization of the BSDE (3.3.2) can be written in backward induction as

$$Y_i^\pi = Y_{i+1}^\pi - f(t_i, X_i, Y_i^\pi, Z_i^\pi)\Delta t_i - Z_i^\pi.\Delta W_i, \quad i = 0, \ldots, N-1, \tag{3.3.3}$$

which can be described as conditional expectation formulae

$$\left\{ \begin{array}{rcl} Y_i^\pi & = & \mathbb{E}_i\Big[Y_{i+1}^\pi - f(t_i, X_i, Y_i^\pi, Z_i^\pi)\Delta t_i\Big] \\ Z_i^\pi & = & \mathbb{E}_i\Big[\frac{\Delta W_i}{\Delta t_i} Y_{i+1}^\pi\Big], \qquad i = 0, \ldots, N-1, \end{array} \right. \tag{3.3.4}$$

where $\mathbb{E}_i$ is a notation for the conditional expectation w.r.t. $\mathcal{F}_{t_i}$.

• Deep BSDE scheme [EHJ17], [HJE18].

The essence of this method is to write down the backward equation (3.3.3) as a forward equation. One approximates the initial condition Y_0 and the Z component at each time by networks functions taking the forward process X as input. The objective function to optimize is the error between the reconstructed dynamics and the true 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 $\boldsymbol{\mathcal{Z}} = (\mathcal{Z}_i)_i$, $\mathcal{Z}_i : \mathbb{R}^d \to \mathbb{R}^d$, $i = 0, \ldots, N-1$, 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

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

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, supplies 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 (3.3.1) at times t_i evaluated at $\mathcal{X}_{t_i}$, i.e., of $Y_{t_i} = u(t_i, \mathcal{X}_{t_i})$, $i = 0, \ldots, N$. The convergence of this algorithm through a posteriori error is studied by [HL20], see also [JL21]. A variant is proposed by [CWNMW19] which introduces a single neural network $\mathcal{Z}(t, x) : [0, T] \times \mathbb{R}^d \mapsto \mathbb{R}^d$ instead of N independent neural networks. This simplifies the optimization problem and leads to more stable solutions. A close method introduced by [Rai18] uses also a single neural network $\mathcal{U}(t, x) : [0, T] \times \mathbb{R}^d \mapsto \mathbb{R}$ and estimates Z as the automatic derivative in space of $\mathcal{U}$. We also refer to [JO19] for a variation of this deep BSDE scheme to curve-dependent PDEs arising in the pricing under rough volatility model, to [NR20] for approximations methods for Hamilton-Jacobi-Bellman PDEs, and to [KSS20] for extension of deep BSDE scheme to elliptic PDEs with applications in insurance.

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

The method builds upon the backward dynamic programming relation (3.3.3) stemming from the time discretization of the BSDE, and approximates simultaneously at each time step t_i the processes (Y_{t_i}, Z_{t_i}) with neural networks trained with the forward diffusion process X_i as input. The scheme can be implemented in two similar versions:

1. *DBDP1.* Starting from $\widehat{\mathcal{U}}_N^{(1)} = g$, proceed by backward induction for $i = N-1, \ldots, 0$, 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 quadratic loss function

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

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 quadratic loss function

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

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 supplies 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 (3.3.1) on the time grid t_i, $i = 0, \dots, N-1$. The study of the approximation error due to the time discretization and the choice of the loss function is accomplished in [HPW20].

Variants and extensions of DBDP schemes

(i) A regression-based machine learning scheme inspired by regression Monte-Carlo methods for numerically computing condition expectations in the time discretization (3.3.4) of the BSDE, is given by: 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 simultaneously estimates the pair component (Y, Z) through 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), $i = N-1, \dots, 0$. Interestingly, the convergence of the DBDP scheme can be confirmed by computing at each time step the infimum of loss function, which should vanish for the exact solution (up to the time discretization). In contrast, the infimum of the loss functions in usual regression-based schemes is unknown for the true solution as it is supposed to match the residual of L^2-projection. Therefore the scheme accuracy cannot be directly verified.

(ii) The DBDP scheme is based on local resolution, and was first used to solve linear PDEs, see [VSS18]. It is also suitable to solve variational inequalities and can be used to valuate American options as shown in [HPW20]. Alternative methods consists in using the Deep Optimal Stopping scheme [BCJ19] or the method from [Bec+19]. Some tests on Bermudan options are also performed by [LXL19] and [FTT19] with some refinements of the Deep BSDE scheme.

(iii) The **Deep Splitting (DS) scheme in [Bec+21]** combines ideas from the DBDP2 and regression-based schemes. Indeed the current regression-approximation on Z is estimated by the automatic differentiation of the neural network computed at the previous

optimization step. The current approximation of Y is then computed by a regression-type optimization problem. It can be seen as a local version of the global algorithm from [Rai18] or as a step by step Feynman-Kac approach. As the scheme is a local one, it can be used to valuate American options. The convergence of this method is studied by [GPW22a].

(iv) Local resolution permits to add other constraints such as constraints on a replication portfolio using facelifting techniques as in [KLW21].

(v) The **Deep Backward Multistep (MDBDP) scheme [GPW22a]** 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

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

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 (3.3.1) at times t_i, $i = 0, \dots, N-1$.

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 as it can be shown comparing the error estimated in [GPW22a] and [HPW20] both at theoretical and numerical level.

Case of fully non-linear PDEs

In this paragraph, we consider fully non-linear PDEs in the form

$$\begin{cases} \partial_t u + \mu \cdot D_x u + \frac{1}{2}\,\mathrm{Tr}(\sigma\sigma^\intercal D_x^2 u) \;=\; F(\cdot, \cdot, u, D_x u, D_x^2 u) & \text{on } [0,T) \times \mathbb{R}^d \\ u(T, \cdot) \;=\; g & \text{on } \mathbb{R}^d, \end{cases} \tag{3.3.5}$$

For this purpose, we introduce a forward diffusion process $\mathcal{X}$ in $\mathbb{R}^d$ as in (3.3.2), and associated to the linear part $\mathcal{L}$ of the differential operator in the l.h.s. of the PDE (3.3.5).

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 (3.3.5) 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. We refer to Section 3.1 in [PWG21] for a discussion on the choice of the parameters μ, σ.

In the sequel, we assume for simplicity that $\mu = 0$, and σ is a constant invertible matrix.

Let us derive formally a BSDE representation for the nonlinear PDE (3.3.5) on which we shall rely for designing our machine learning algorithm. Assuming that the solution u to this PDE 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

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

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

$$Y_t = \; g(\mathcal{X}_T) - \int_t^T F(s, \mathcal{X}_s, Y_s, Z_s, \Gamma_s)\mathrm{d}s - \int_t^T Z_s^\intercal \sigma \mathrm{d}W_s, \; 0 \le t \le T. \tag{3.3.6}$$

Compared to the case of semi-linear PDE of the form (3.3.1), the key point is the approximation/learning of the Hessian matrix $D_x^2 u$, hence of the Γ-component of the BSDE (3.3.6). We present below different approaches for the approximation of the Γ-component. To the best of our knowledge, no theoretical convergence result is available for machine learning schemes in the fully nonlinear case but several methods show good empirical performances.

• Deep 2BSDE scheme [BEJ19].

This scheme relies on the 2BSDE representation of [Che+07]

$$\left\{ \begin{array}{rcl} Y_t &=& g(\mathcal{X}_T) - \int_t^T F(s, \mathcal{X}_s, Y_s, Z_s, \Gamma_s)\mathrm{d}s - \int_t^T Z_s^\intercal \sigma \mathrm{d}W_s, \\ Z_t &=& D_x g(\mathcal{X}_T) - \int_t^T A_s \mathrm{d}s - \int_t^T \Gamma_s \sigma \mathrm{d}W_s, \quad 0 \le t \le T, \end{array} \right. \tag{3.3.7}$$

with $A_t = \mathcal{L} D_x u(t, \mathcal{X}_t)$. The idea is to adapt the Deep BSDE algorithm to the fully non-linear case. Again, we treat the backward system (3.3.7) as a forward equation by approximating the initial conditions Y_0, Z_0 and the A, Γ components of the 2BSDE at each time by networks functions taking the forward process $\mathcal{X}$ as input, and aiming to match the terminal condition.

• Second order DBDP (2DBDP) [PWG21]

The basic idea is to adapt the DBDP scheme by 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 $\mathcal{Z}$ (or double automatic differentiation $D_x^2 \mathcal{U}$ of the network function $\mathcal{U}$), via a learning approach relying on the time discretization of the BSDE (3.3.6). It turns out that such method approximates poorly Γ inducing instability of the scheme: indeed, while the unique pair solution (Y, Z) to classical BSDEs (3.3.2) completely characterizes the solution to the related semilinear PDE and its gradient, the relation (3.3.6) does not allow to characterize directly the triple (Y, Z, Γ). This approach was proposed and tested in [PWG21] where the automatic differentiation is performed on the previous value of $\mathcal{Z}$ with a truncation $\mathcal{T}$ which allows to reduce instabilities.

• Second Order Multistep schemes.

To overcome the instability in the approximation of the Γ-component in the Second order DBDP scheme, we propose a finer approach based on a suitable probabilistic representation of the Γ-component for learning accurately the Hessian function $D_x^2 u$ by using also Malliavin weights. We start from the training simulations of the forward process $(X_i)_i$ on the uniform grid $\pi = \{t_i = i|\pi|, i = 0, \ldots, N\}$, $|\pi| = T/N$, and notice that $X_i = \mathcal{X}_{t_i}$, $i = 0, \ldots, N$ as μ and σ are constants. The approximation of the value function u and its gradient $D_x u$ is learnt simultaneously on the grid π 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.

For the second and the third one, we need to introduce a subgrid $\hat{\pi} = \{t_{\hat{\kappa}\ell}, \ell = 0, \ldots, \hat{N}\} \subset \pi$, of modulus $|\hat{\pi}| = \hat{\kappa}|\pi|$, for some $\hat{\kappa} \in \mathbb{N}^*$, with $N = \hat{\kappa}\hat{N}$.

- *Version 1:* Extending the methodology introduced in [PWG21], 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:

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

- *Version 2:* The time discretization of (3.3.6) on the time grid $\hat{\pi}$, where $(Y_\ell^{\hat{\pi}}, Z_\ell^{\hat{\pi}}, \Gamma_\ell^{\hat{\pi}})$ denotes an approximation of the triple

$$\big(u(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}), D_x u(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell}), D_x^2 u(t_{\hat{\kappa}\ell}, X_{\hat{\kappa}\ell})\big), \quad \ell = 0, \ldots, \hat{N},$$

leads to the standard representation formula for the Z component:

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

(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 $\hat\pi$ for $\ell = 0,\dots,\hat N-1$ with

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

- *Version 3:* Alternatively, the time discretization of (3.3.6) on $\hat\pi$ yields the iterated conditional expectation relation:

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

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

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

for $\ell = 0,\dots,\hat N$, where

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

By introducing again the antithetic variables

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

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

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

for $\ell = 0, \dots, N-1$, 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

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

for all $m = \ell + 1, \dots, \hat{N} - 1$, 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.

Remark 3.3.1. 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 $D^2 g(X_{\hat{\kappa}\hat{N}})$. □

Remark 3.3.2. We point out that in our machine learning setting for the versions 2 and 3 of the scheme, we only solve two optimization problems by time step instead of three as in [FTW11]. One optimization is dedicated to the computation of the Γ component but the $\mathcal{U}$ and $\mathcal{Z}$ components are simultaneously learned by the algorithm. □

We can now describe the three versions of second order MDBDP schemes for the numerical resolution of the fully non-linear PDE (3.3.5). We emphasize that these schemes do not require *a priori* that the solution to the PDE is smooth.

Algorithm 3: Second order Explicit Multistep DBDP (2EMDBDP)

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

If $i = N-1$, update $\widehat{\Gamma}_i = D^2 g$, 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:

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

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

end

The proposed algorithms 3, 4, 5 are in backward iteration, and involve one optimization at each step. Moreover, as the computation of Γ requires a further derivation for Algorithms 4 and 5, we may expect that the additional propagation error varies according to $\frac{|\pi|}{|\bar{\pi}|} = \frac{1}{\hat{\kappa}}$, and thus the convergence of the scheme when $\hat{\kappa}$ is large. In the numerical implementation, the expectation in the loss functions are replaced by empirical average and the minimization over network functions is performed by stochastic gradient descent.

Algorithm 4: Second order Multistep DBDP (2MDBDP)

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

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

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, $i = (\ell-1)\hat{\kappa} + k$:

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

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

end

end

Algorithm 5: Second order Multistep 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

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

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, $i = (\ell - 1)\hat{\kappa} + k$:

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

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

end

end

Dimension d	DBDP [HPW20]	DBSDE [HJE18]
1	0.05950 (0.000257)	0.05949 (0.000264)
3	0.17797 (0.000421)	0.17807 (0.000288)
5	0.25956 (0.000467)	0.25984 (0.000331)
10	0.40930 (0.000623)	0.40886 (0.000196)
15	0.52353 (0.000591)	0.52389 (0.000551)
30	0.78239 (0.000832)	0.78231 (0.001266)

Table 3.1: CVA value with $X_0 = 1, T = 1, \beta = 0.03, \sigma = 0.2$ and 50 time steps.

3.4 Numerical applications

We test our different algorithms 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 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.

3.4.1 Numerical tests on credit valuation adjustment pricing

We consider an example of model from [HL17] for the pricing of CVA in a d-dimensional Black-Scholes model

$$\mathrm{d}X_t = \sigma X_t \, \mathrm{d}W_t, \; X_0 = 1_d$$

with $\sigma > 0$, given by the nonlinear PDE

$$\begin{cases} \partial_t u + \frac{\sigma^2}{2} \mathrm{Tr}(x^\top D_x^2 u \; x) + \beta(u_+ - u) = 0 & \text{on } [0,T] \times \mathbb{R}^d \\ u(T,x) = |\sum_{i=1}^d x_i - d| - 0.1 & \text{on } \mathbb{R}^d \end{cases}$$

with a straddle type payoff. We compare our results with the DBDP scheme [HPW20] with the ones from the Deep BSDE solver [HJE18]. The results in Table 3.1 are averaged over 10 runs and the standard deviation is written in parentheses. We use ReLu activation functions.

We observe in Table 3.1 that both algorithms give very close results and are able to solve the nonlinear pricing problem in high dimension d. The variance of the results is quite small and similar from one to another but increases with the dimension. The same conclusions arise when solving the PDE for the larger maturity $T = 2$.

3.4.2 Portfolio allocation in stochastic volatility models

We consider several examples from [PWG21] that we solve with Algorithms 3 (2EMDBDP), 4 (2MDBDP), and 5 (2M^{2}DBDP) designed in this paper. Notice that some comparison tests with the 2DBSDE scheme [BEJ19] have been already done in [PWG21]. 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. We use tanh as activation function.

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 $[0, T) \times \mathbb{R} \times \mathbb{R}^n$:

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

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

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

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

We shall test this example when the utility function U is of exponential form: $U(\mathrm{x}) = -\exp(-\eta \mathrm{x})$, with $\eta > 0$, and under different cases for which explicit solutions are available. We refer to [PWG21] where these solutions are described.

(1) *Merton problem.* This corresponds to a degenerate case where the factor V, hence the volatility σ and the risk premium λ are constant ($v_i = \theta_i,\ \nu_i = 0$). We train our algorithms with the forward process

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

(2) *One risky asset:* $n = 1$. We train our algorithms with the forward process

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

We test our algorithm with $\lambda(v) = \lambda v$, $\lambda > 0$, for which we have an explicit solution.

(3) *No leverage effect, i.e.,* $\rho_i = 0$, $i = 1, \dots, n$. We train with the forward process

$$\begin{aligned} \mathcal{X}_{k+1} &= \; \mathcal{X}_k + \sum_{i=1}^n \lambda_i(\theta_i)\Delta t_k + \Delta W_k, \quad k = 0, \dots, N-1,\ \mathcal{X}_0 \;=\; \mathrm{x}_0 \\ V_{k+1}^i &= \; V_k^i + \nu_i \Delta B_k^i, \qquad k = 0, \dots, N-1,\ V_0^i \;=\; \theta_i. \end{aligned}$$

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.

	Average	Standard deviation	Relative error (%)
[PWG21]	-0.50561	0.00029	0.20
2EMDBDP	**-0.50673**	**0.00019**	**0.022**
2MDBDP	-0.50647	0.00033	0.030
2M^{2}DBDP	-0.50644	0.00022	0.035

Table 3.2: Estimate of $u(0,1.)$ in the Merton problem with $N = 120$, $\hat{N} = 30$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is -0.50662.

	Average	Standard deviation	Relative error (%)
[PWG21]	-0.53431	0.00070	0.34
2EMDBDP	**-0.53613**	**0.00045**	**0.007**
2MDBDP	-0.53772	0.00046	0.304
2M^{2}DBDP	-0.53205	0.00050	0.755

Table 3.3: 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.

Merton Problem. We take $\eta = 0.5$, $\lambda = 0.6$, $N = 120$, $\hat{N} = 30$, $T = 1$, $x_0 = 1$. We plot in Figure 3.1 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 Table 3.2, the multistep algorithms improve significantly the results obtained in [PWG21], where the estimation of the Hessian is not really accurate (see blue curve in Figure 3.1).

One asset $n = 1$ in Scott volatility model. We take $\eta = 0.5$, $\lambda = 1.5$, $\theta = 0.4$, $\nu = 0.4$, $\kappa = 1$, $\rho = -0.7$, $T = 1$, $x_0 = 1$. For all tests we choose $N = 120$, $\hat{N} = 30$ and $\sigma(v) = e^v$. We report in Table 3.3 the relative error between the neural networks approximation of $u, D_{\mathrm{x}} u, D_{\mathrm{x}}^2 u$ computed from our different algorithms and their analytic values. It turns out that the multistep extension of [PWG21], namely 2EMDBDP scheme, yields a very accurate approximation result, much better than the other algorithms, with also a reduction of the standard deviation.

No Leverage in Scott model. In the case with one asset we take $\eta = 0.5$, $\lambda = 1.5$, $\theta = 0.4$, $\nu = 0.2$, $\kappa = 1$, $T = 1$, $x_0 = 1$. For all tests we choose $N = 120$, $\hat{N} = 30$ and $\sigma(v) = e^v$. We report in Table 3.4 the relative error between the neural networks approximation of $u, D_{\mathrm{x}} u, D_{\mathrm{x}}^2 u$ computed from our different algorithms and their analytic values. All the algorithms yield quite accurate results, but compared to the case with correlation in Table 3.3, it appears here that the best performance in terms of precision is achieved by Algorithm 2M^{2}DBDP.

In the case with four assets ($n = 4$, $d = 5$), we take $\eta = 0.5$,
$\lambda = \begin{pmatrix}1.5 & 1.1 & 2. & 0.8\end{pmatrix}, \theta = \begin{pmatrix}0.1 & 0.2 & 0.3 & 0.4\end{pmatrix}, \nu = \begin{pmatrix}0.2 & 0.15 & 0.25 & 0.31\end{pmatrix}, \kappa = \begin{pmatrix}1. & 0.8 & 1.1 & 1.3\end{pmatrix}$.
The results are reported in Table 3.5. We observe that the algorithm in [PWG21] provides a not so accurate outcome, while its multistep version (2EMDBDP scheme) divides by 10 the relative

	Average	Standard deviation	Relative error (%)
[PWG21]	-0.49980	0.00073	0.35
2EMDBDP	-0.50400	0.00229	0.485
2MDBDP	-0.50149	**0.00024**	0.015
2M^{2}DBDP	**-0.50157**	0.00036	**0.001**

Table 3.4: Estimate of $u(0,1,\theta)$, with 120 time steps on the No Leverage problem 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.

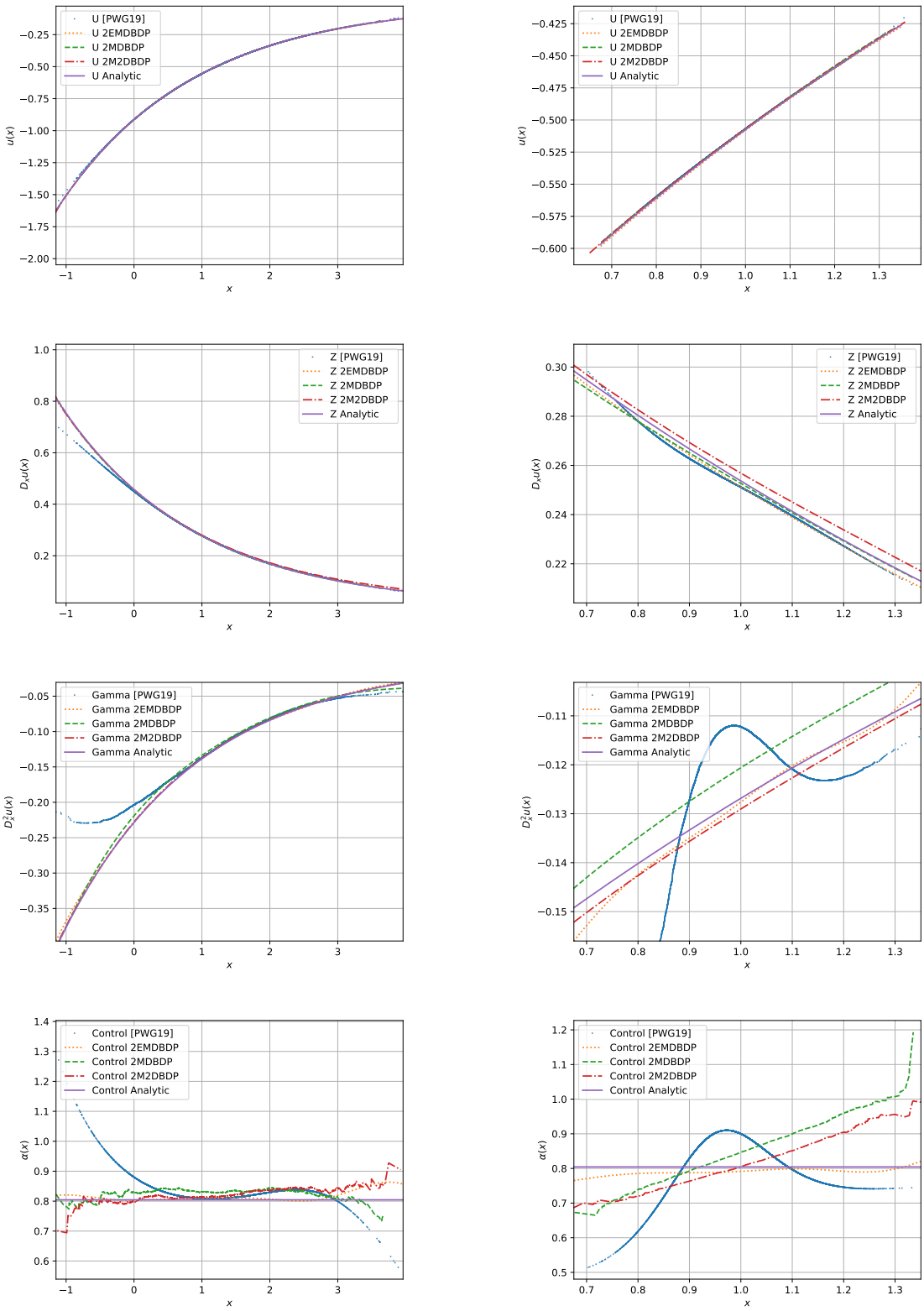


Figure 3.1: Estimates of u, $D_x u$, $D_x^2 u$ and of the optimal control α on the Merton problem with $N = 120$, $\hat{N} = 30$. We take $x_0 = 1.$, at the left $t = 0.5042$, and at the right $t = 0.0084$.

	Average	Standard deviation	Relative error (%)
[PWG21]	-0.43768	0.00137	0.92
2EMDBDP	**-0.4401**	**0.00051**	**0.239**
2MDBDP	-0.43796	0.00098	0.861
2M^{2}DBDP	-0.44831	0.00566	1.481

Table 3.5: Estimate of $u(0,1,\theta)$, with 120 time steps on the No Leverage problem 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.

	$\hat{N}$	Average	S.d.	Relative error (%)
[PWG21]		**-0.27920**	0.05734	**1.49**
2EMDBDP		-0.26631	**0.00283**	3.19
2MDBDP	30	-0.28979	0.00559	5.34
2MDBDP	60	-0.28549	0.00948	3.78
2MDBDP	120	**-0.28300**	0.01129	**2.87**
2M^{2}DBDP	30	NC	NC	NC

Table 3.6: 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 (S.d.) observed over 10 independent runs are reported. The theoretical solution is -0.27509173.

error and the standard deviation.

In the case with nine assets ($n=9$, $d=10$), 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 Table 3.6. The approximation is less accurate than in lower dimension, but we observe again that compared to one-step scheme in [PWG21] , the multistep versions improve significantly the standard deviation of the result. However the best performance in precision is obtained here by the [PWG21] scheme.

3.5 Extensions and perspectives

• Solving mean-field control and mean-field games through McKean-Vlasov FBSDEs.

These methods solve the optimality conditions for mean-field problems through the stochastic Pontryagin principle from [CD18c]. The law of the solution influences the coupled FBSDEs dynamics so they are of McKean-Vlasov type. Variations around the Deep BSDE method [HJE18] are used to solve such a system by [CL22], [FZ20]. [GMW22] uses the Merged method from [CWNMW19] and solves several numerical examples in dimension 10 by introducing an efficient law estimation technique. [CL22] also proposes another method dedicated to mean field control to directly tackle the optimization problem with a neural network as the control in the stochastic dynamics. The $N-$player games, before going to the mean-field limit of an infinite number of players, are solved by [Hu19], [HHL20].

• Solving mean-field control through master Bellman equation and symmetric neural networks.

[Ger+22] solves the master Bellman equation arising from dynamic programming principle applied to mean-field control problems (see [PW17]). The paper approximates the value function evaluated on the empirical measure stemming from particles simulation of a training forward process. The symmetry between iid particles is enforced by optimizing over exchangeable high-

dimensional neural networks, invariant by permutation of their inputs. The companion paper [GPW22c] provides a rate for the particle method convergence.

• Reinforcement Learning for mean-field control and mean-field games [CLT19; AKS19; AFL20; Gu+20; Guo+20].

Some works focus on similar problems but with unknown dynamics. Thus they rely on trajectories sampled from a simulator and reinforcement learnings– especially Q-learning– to estimate the state action value function and optimal control without a model. The idea is to optimize a neural network by relying on a memory of past state action transitions used to train the network in order for it to verify the Bellman equation on samples from memory replay.

• Machine learning framework for solving high-dimensional mean field game and mean field control problems [Rut+20]

This paper focuses on potential mean field games, in which the cost functions depending on the law can be written as the linear functional derivative of a function with respect to a measure. A Lagrangian method with Deep Galerkin type penalization is used. In this case the potential is approached by a neural network and solving mean-field games amounts to solve an unconstrained optimization problem.

• Deep quantum neural networks [Sak20]

We briefly mention this work studying the use of deep quantum neural networks which exploit the quantum superposition properties by replacing bits by “qubits". Promising results are obtained when using these networks for regression in financial contexts such as implied volatility estimation. Future works may study the application of such neural networks to control problems and PDEs.

• Path signature for path-dependent PDE [SVSS20]

This work extends previously developed methods for solving state-dependent PDEs to the linear path-dependent setting coming for instance from the pricing and hedging of path-dependent options. A path-dependent Feynman-Kac representation is numerically computed through a global minimization over neural networks. The authors show that using LSTM networks taking the forward process' path signatures (coming from the rough paths literature) as input yields better results than taking the discretized path as input of a feedforward network.

Chapter 4

Approximation Error Analysis of Some Deep Backward Schemes for Nonlinear PDEs

This chapter is based on the paper [GPW22a]
M. Germain, H. Pham, and X. Warin. "Approximation Error Analysis of Some Deep Backward Schemes for Nonlinear PDEs". In: *SIAM Journal on Scientific Computing 44.1* (5 January 2022), A28–A56.

In this chapter we propose a new multistep machine learning scheme for the resolution of semi-linear PDEs. We combine ideas from [HPW20] and [GT14] and demonstrate both the theoretical and numerical performances of our method. A theoretical analysis is also performed on the Deep Splitting scheme [Bec+21] and the Deep Backward Dynamic Programming scheme [HPW20]. We are able to relate the approximation error of the algorithm to the number of neurons and layers of the approximating neural networks, in the case of the GroupSort architecture [ALG19]. These neural networks enjoy Lipschitz continuity properties and quantitative approximation results are given by [TSB21]. However these results still exhibit the curse of dimensionality, even though in practice a small number of neurons and layers is enough on the tested examples.

Abstract

Recently proposed numerical algorithms for solving high-dimensional nonlinear partial differential equations (PDEs) based on neural networks have shown their remarkable performance. We review some of them and study their convergence properties. The methods rely on probabilistic representation of PDEs by backward stochastic differential equations (BSDEs) and their iterated time discretization. Our proposed algorithm, called deep backward multistep scheme (MDBDP), is a machine learning version of the LSMDP scheme of Gobet, Turkedjiev (Math. Comp. 85, 2016). It 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. Our main theoretical contribution is to provide an approximation error analysis of the MDBDP scheme as well as the deep splitting (DS) scheme for semilinear PDEs designed in Beck, Becker, Cheridito, Jentzen, Neufeld (2019). We also supplement the error analysis of the DBDP scheme of Huré, Pham, Warin (Math. Comp. 89, 2020). This yields notably convergence rate in terms of the number of neurons for a class of deep Lipschitz continuous GroupSort neural networks when the PDE is linear in the gradient of the solution for the MDBDP scheme, and in the semilinear case for the DBDP scheme. We illustrate our results with some numerical tests that are compared with some other machine learning algorithms in the literature.

4.1 Introduction

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

$$\begin{cases} \partial_t u + \mu \cdot D_x u + \frac{1}{2}\,\mathrm{Tr}(\sigma\sigma^\intercal D_x^2 u) \;=\; f(\cdot,\cdot,u,\sigma^\intercal D_x u) & \text{on } [0,T)\times\mathbb{R}^d \\ u(T,\cdot) \;=\; g & \text{on } \mathbb{R}^d, \end{cases} \tag{4.1.1}$$

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] 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 (ii) on multilevel Picard methods, developed in [E+19] with algorithms based on Picard iterations, multi-level techniques and automatic differentiation. These methods permit to handle some 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.

Over the last few years, machine learning methods have emerged since the pioneering papers by [HJE18] and [SS18], and have shown their efficiency for solving high-dimensional nonlinear PDEs by means of neural networks approximation. The work [HJE18] introduces a global machine learning resolution technique via a 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 [HL20].

The Deep Galerkin method of [SS18] proposes another global meshfree method with a random sampling of time and space points inside a bounded domain.

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 (DBDP) of [HPW20] verifies better stability and accuracy properties than the global method in [HJE18], as illustrated on several test cases. The recent paper [Bec+21] also introduces machine learning schemes based on local loss functions, called Deep Splitting (DS) 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.

In this paper, we propose machine learning schemes that use multistep methods introduced in [BD07] and [GT14]. The idea is to rely on the whole previously computed values of the discretized processes in the backward computations of the approximation as it is expected to yield a better propagation of regression errors. We shall develop this approach to the DBDP scheme 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, through standard stability-type arguments for BSDEs (see e.g. Section 4.4 in [Zha17] for the continuous time case). The arguments can be adapted to obtain the convergence of the DS scheme introduced in [Bec+21]. Furthermore, by relying on recent approximation results for deep neural networks in [TSB21], we obtain a rate of convergence of our scheme in terms of the number of neurons, and supplement the convergence analysis of the DBDP scheme [HPW20].

We provide some numerical tests of our proposed algorithms, which show the benefit of multistep schemes, and compare our results with the cited machine learning schemes. Notice that the GroupSort network is used for theoretical analysis but in the numerical implementation, we applied standard networks with tanh as activation function. The theoretical analysis of the convergence of methods relying on standard neural networks is left to future research. More numerical examples and tests are presented in the extended first arXiv version [GPW22a] of this paper.

The plan of the paper is the following. In Section 4.2, we give a brief reminder on neural networks and notably on a specific class of deep network functions considered in [ALG19; TSB21] that yields an approximation result with rate of convergence for Lipschitz functions. We also review machine learning schemes for the numerical resolution of semilinear PDEs. We then describe in detail the MDBDP scheme.

We state in Section 4.3 the convergence of the MDBDP, DS, and DBDP schemes, while Section 4.4 is devoted to the proof of these results. Section 4.5 gives some numerical tests for illustration.

4.2 BSDE Machine Learning Schemes for Semilinear PDEs

In this section, we review recent numerical schemes, and present our new scheme for the resolution of the semi-linear PDE (4.1.1) by approximations in the class of neural networks and relying on probabilistic representation of the solution to the PDE.

4.2.1 Neural Networks

We denote by

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

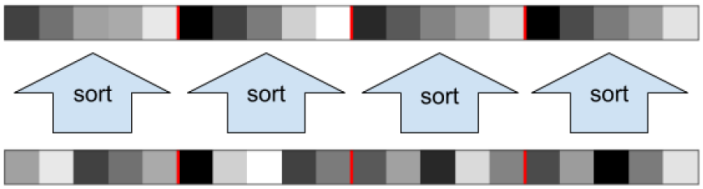


Figure 4.1: GroupSort activation function ζ_κ with grouping size $\kappa = 5$ and $m = 20$ neurons, figure from [ALG19].

the set of layer functions with input dimension d_1, output dimension d_2, and activation function $\rho : \mathbb{R}^{d_2} \to \mathbb{R}^{d_2}$. Usually, the activation is applied component-wise via a one-dimensional activation function, i.e., $\rho(x_1, \ldots, x_{d_2}) = \big(\hat\rho(x_1), \ldots, \hat\rho(x_{d_2})\big)$ with $\hat\rho : \mathbb{R} \mapsto \mathbb{R}$, 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 $\hat\rho$ are the sigmoid, the ReLU, the tanh. When ρ is the identity function, we simply write $\mathcal{L}_{d_1,d_2}$.

We then define

$$\mathcal{N}^\rho_{d_0,d',\ell,m} = \Big\{\varphi : \mathbb{R}^{d_0} \to \mathbb{R}^{d'} : \exists \phi_0 \in \mathcal{L}^{\rho_0}_{d_0,m_0},\ \exists \phi_i \in \mathcal{L}^{\rho_i}_{m_{i-1},m_i}, i = 1, \ldots, \ell-1,$$
$$\exists \phi_\ell \in \mathcal{L}_{m_{l-1},d'}, \varphi \ = \ \phi_\ell \circ \phi_{\ell-1} \circ \cdots \circ \phi_0\Big\},$$

as the set of feedforward neural networks with input layer dimension d_0, output layer dimension d', and ℓ hidden layers with m_i neurons per layer ($i = 0, \cdots, \ell-1$). These numbers d_0, d', ℓ, the sequence $m = (m_i)_{i=0,\ldots,\ell-1}$, and sequence of activation functions $\rho = (\rho_i)_{i=0,\ldots,\ell-1}$, form the architecture of the network.

In the sequel, we shall mostly work with the case $d_0 = d$ (dimension of the state variable x).

A given network function $\varphi \in \mathcal{N}^\rho_{d_0,d',\ell,m}$ is determined by the weight/bias parameters $\theta = (\mathcal{W}_0, \beta_0, \ldots, \mathcal{W}_\ell, \beta_\ell)$ defining the layer functions $\phi_0 \ldots, \phi_\ell$, and we shall sometimes write $\varphi = \varphi_\theta$.

We recall the fundamental result of [HSW89] that justifies the use of neural networks as function approximators, in the usual case of activation functions applied componentwise at each hidden layer.

Universal approximation theorem. The space $\bigcup_{i=0}^{\ell-1} \bigcup_{m_i=0}^{\infty} \mathcal{N}^\rho_{d_0,d',\ell,m}$ is dense in $L^2(\nu)$, the set of measurable functions $h : \mathbb{R}^{d_0} \to \mathbb{R}^{d'}$ s.t. $\int |h(x)|_2^2 \nu(\mathrm{d}x) < \infty$, 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. Some results give rates for the approximation of functions in Sobolev spaces [Pin99a], for bounded convex subdifferentiable Lipschitz functions [BGS15] or bounded Lipschitz functions [Yar17], but here, we need a result related to (possibly unbounded) Lipschitz functions. The paper [Bac17] provides a possible answer in this direction, but we instead rely on a simpler approach in [TSB21], building on the GroupSort deep neural networks introduced by [ALG19]. Let $\kappa \in \mathbb{N}^*$, $\kappa \geq 2$, be a grouping size, dividing the number of neurons $m_i = \kappa n_i$, at each layer $i = 0, \cdots \ell - 1$. $\sum_{i=0}^{\ell-1} m_i$ will be refered to as the width of the network and $\ell+1$ as its depth. The GroupSort networks correspond to classical deep feedforward neural networks in $\mathcal{N}^{\zeta_\kappa}_{d,1,\ell,m}$ with a specific sequence of activation function $\zeta_\kappa = (\zeta^i_\kappa)_{i=0,\ldots,\ell-1}$, and one-dimensional output. Each nonlinear function ζ^i_κ divides its input into groups of size κ and sorts each group in decreasing order, see Figure 4.1. Moreover, by enforcing the parameters of the GroupSort to satisfy with the Euclidian norm $|\cdot|_2$ and the ℓ_∞ norm $|\cdot|_\infty$:

$$\sup_{|x|_2=1} |\mathcal{W}_0 x|_\infty \leq 1, \quad \sup_{|x|_\infty=1} |\mathcal{W}_i x|_\infty \leq 1, \ |\beta_j|_\infty \leq M, \ i = 1, \cdots, l, \ j = 0, \cdots, l$$

for some $M > 0$, the related GroupSort neural networks from $\mathcal{N}^{\zeta_\kappa}_{d,d',\ell,m}$ are 1-Lipschitz.

The space of such 1-Lipschitz GroupSort neural networks is called $\mathcal{S}^{\zeta_\kappa}_{d,\ell,m}$:

$$\mathcal{S}^{\zeta_\kappa}_{d,\ell,m} = \{\varphi_{(\mathcal{W}_0,\beta_0,\dots,\mathcal{W}_\ell,\beta_\ell)} \in \mathcal{N}^{\zeta_\kappa}_{d,1,\ell,m},\ \sup_{|x|_2=1} |\mathcal{W}_0 x|_\infty \leq 1,\ \sup_{|x|_\infty=1} |\mathcal{W}_i x|_\infty \leq 1,$$
$$|\beta_j|_\infty \leq M,\ i = 1,\cdots,l,\ j = 0,\cdots,l\}.$$

We then introduce the set $\mathcal{G}^{\zeta_\kappa}_{K,d,d',\ell,m}$ as

$$\mathcal{G}^{\zeta_\kappa}_{K,d,d',\ell,m} := \{\Psi = (\Psi_i)_{i=1,\dots,d'} : \mathbb{R}^d \mapsto \mathbb{R}^{d'},\ \Psi_i : x \in \mathbb{R}^d \mapsto K\beta_i \quad \phi_i\Big(\frac{x+\alpha_i}{\beta_i}\Big) \in \mathbb{R},$$
$$\phi_i \in \mathcal{S}^{\zeta_\kappa}_{d,\ell,m},\ \text{ for some } \alpha_i \in \mathbb{R}^d,\ \beta_i > 0\}.$$

Notice that these networks are $\sqrt{d'}K$-Lipschitz and that each of their components is K-Lipschitz. We rely on the the following quantitative approximation result which directly follows from [TSB21].

Proposition 4.2.1 (Slight extension of Tanielian, Sangnier, Biau [TSB21] : Approximation theorem for Lipschitz functions by Lipschitz GroupSort neural networks.)**.** *Let $f : [-R,R]^d \mapsto \mathbb{R}^{d'}$ be K-Lipschitz. Then, for all $\varepsilon > 0$, there exists a GroupSort neural network g in $\mathcal{G}^{\zeta_\kappa}_{K,d,d',\ell,m}$ verifying*

$$\sup_{x\in[-R,R]^d} |f(x) - g(x)|_2 \leq \sqrt{d'}2RK\varepsilon$$

with g of grouping size $\kappa = \lceil \frac{2\sqrt{d}}{\varepsilon} \rceil$, depth $\ell + 1 = O(d^2)$ and width $\sum_{i=0}^{\ell-1} m_i = O((\frac{2\sqrt{d}}{\varepsilon})^{d^2-1})$ in the case $d > 1$. If $d = 1$, the same result holds with g of grouping size $\kappa = \lceil \frac{1}{\varepsilon} \rceil$, depth $\ell + 1 = 3$ and width $\sum_{i=0}^{\ell-1} m_i = O(\frac{1}{\varepsilon})$.

Proof. With f_i the i-th component of f, define

$$\widetilde{f_i} : z \in [0,1]^d \mapsto \frac{f_i(2R(z - 1/2))}{2RK}. \tag{4.2.1}$$

Then $\widetilde{f_i}$ is 1-Lipschitz and by Theorem 3 from [TSB21] if $d > 1$ (or Proposition 5 from [TSB21] if $d = 1$), there exists a 1-Lipschitz GroupSort neural network $g_i \in \mathcal{S}^{\zeta_\kappa}_{d,\ell,m}$ verifying

$$\sup_{z\in[0,1]^d} |\widetilde{f_i}(z) - g_i(z)| \leq \varepsilon$$

with g_i of grouping size $\kappa = O(\frac{2\sqrt{d}}{\varepsilon})$, depth $\ell+1 = O(d^2)$ and width $\sum_{i=0}^{\ell-1} m_i = O((\frac{2\sqrt{d}}{\varepsilon})^{d^2-1})$(respectively grouping size $\kappa = O(\frac{1}{\varepsilon})$, depth $\ell + 1 = 3$ and width $\sum_{i=0}^{\ell-1} m_i = O(\frac{1}{\varepsilon})$ if $d = 1$). Inverting (4.2.1) we have $f_i(x) = 2KR\widetilde{f_i}(\frac{x+R}{2R})$ hence

$$\sup_{x\in[-R,R]^d} \Big| f_i(x) - 2KRg_i\Big(\frac{x+R}{2R}\Big)\Big| \leq 2KR\varepsilon.$$

The result is proven by concatenating the d' K-Lipschitz GroupSort networks $x \mapsto 2KRg_i(\frac{x+R}{2R})$, $i = 1,\cdots,d'$. □

Remark 4.2.1. *As mentioned in [TSB21], GroupSort neural networks generalize the ReLU networks and, thanks to their Lipschitz continuity, offer better stability regarding noisy inputs and adversarial attacks. It also appears that GroupSort networks are more expressive than ReLU ones.*

4.2.2 Existing Schemes

We review recent machine learning schemes that will serve as benchmarks for our new scheme described in the next section. All these schemes rely on 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 (4.1.1), namely:

$$\mathcal{X}_t = \mathcal{X}_0 + \int_0^t \mu(s, \mathcal{X}_s)\mathrm{d}s + \int_0^t \sigma(s, \mathcal{X}_s)\mathrm{d}W_s, \quad 0 \le t \le T, \tag{4.2.2}$$

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 (4.1.1) admits a probabilistic representation in terms of the BSDE:

$$Y_t = g(\mathcal{X}_T) - \int_t^T f(s, \mathcal{X}_s, Y_s, Z_s)\mathrm{d}s - \int_t^T Z_s.\mathrm{d}W_s, \quad 0 \le t \le T, \tag{4.2.3}$$

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 scheme

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

where $\Delta W_j := W_{t_{j+1}} - W_{t_j}$, $j = 0, \dots, N$. When the diffusion $\mathcal{X}$ cannot be 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 (4.2.3) is written in backward induction as

$$Y_i^\pi = Y_{i+1}^\pi - f(t_i, X_i, Y_i^\pi, Z_i^\pi)\Delta t_i - Z_i^\pi.\Delta W_i, \quad i = 0, \dots, N-1, \tag{4.2.4}$$

which also reads as conditional expectation formulae

$$\begin{cases} Y_i^\pi &= \mathbb{E}_i\Big[Y_{i+1}^\pi - f(t_i, X_i, Y_i^\pi, Z_i^\pi)\Delta t_i\Big] \\ Z_i^\pi &= \mathbb{E}_i\Big[\frac{\Delta W_i}{\Delta t_i} Y_{i+1}^\pi\Big], \qquad i = 0, \dots, N-1, \end{cases} \tag{4.2.5}$$

where $\mathbb{E}_i$ denotes the conditional expectation w.r.t. $\mathcal{F}_{t_i}$. Alternatively, by iterating relations (4.2.4) together with the terminal relation $Y_N^\pi = g(X_N)$, we have

$$Y_i^\pi = g(X_N) - \sum_{j=i}^{N-1} \big[f(t_j, X_j, Y_j^\pi, Z_j^\pi)\Delta t_j + Z_j^\pi.\Delta W_j\big], \quad i = 0, \dots, N-1. \tag{4.2.6}$$

• Deep BSDE scheme [HJE18].

The idea of the method is to treat the backward equation (4.2.4) 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 (4.1.1) 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 (4.2.4) 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

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

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 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 $\sigma^\intercal(t_i,.)D_xu(t_i,.)$ to the PDE (4.1.1) at times t_i, $i = 0, \dots, N-1$. The approximation error has been analyzed in [HPW20].

Remark 4.2.2. A machine learning scheme in the spirit of regression-based Monte-Carlo methods ([BT04], [GLW05]) for approximating condition expectations in the time discretization (4.2.5) of the BSDE, can be formulated as follows: starting from $\widehat{\mathcal{U}}_N = g$, 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. Moreover, a variant where the automatic differentiation $D_x\mathcal{U}_i(X_i)$ is performed to estimate Z_{t_i} instead of using a second neural network $\widehat{\mathcal{Z}}_i$ (similarly as in the previous DBDP2 scheme) can also be considered. In this case, one only needs to solve for each time step the (b) optimization problem and not the (a) problem anymore. □

• **Deep Splitting (DS) scheme [Bec+21].**

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

$$\begin{aligned} &J_i^S(\mathcal{U}_i) \\ &= \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}(X_{i+1}) - \mathcal{U}_i(X_i) \\ &\qquad - f(t_i, X_{i+1}, \widehat{\mathcal{U}}_{i+1}(X_{i+1}), \sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1}))\Delta t_i\Big|^2, \end{aligned} \tag{4.2.7}$$

  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 4.3, we shall analyze the approximation error of the DS scheme. Please note that in (4.2.7) we consider a slight modification of the original DS scheme from [Bec+21]. In their loss function, the term $f(t_i, X_{i+1}, \widehat{\mathcal{U}}_{i+1}(X_{i+1}), \sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1}))$ is replaced by $f(t_{i+1}, X_{i+1}, \widehat{\mathcal{U}}_{i+1}(X_{i+1}), \sigma(t_{i+1}, X_{i+1})^\intercal D_x\widehat{\mathcal{U}}_{i+1}(X_{i+1}))$.

4.2.3 Deep Backward Multi-step Scheme (MDBDP)

The starting point of the MDBDP scheme is the iterated representation (4.2.6) 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

$$\begin{aligned} J_i^{MB}(\mathcal{U}_i, \mathcal{Z}_i) = \mathbb{E}\Big|&g(X_N) - \sum_{j=i+1}^{N-1} f(t_j, X_j, \widehat{\mathcal{U}}_j(X_j), \widehat{\mathcal{Z}}_j(X_j))\Delta t_j - \sum_{j=i+1}^{N-1} \widehat{\mathcal{Z}}_j(X_j).\Delta W_j \\ &- \mathcal{U}_i(X_i) - f(t_i, X_i, \mathcal{U}_i(X_i), \mathcal{Z}_i(X_i))\Delta t_i - \mathcal{Z}_i(X_i).\Delta W_i\Big|^2 \end{aligned} \tag{4.2.8}$$

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 (4.1.1) at times t_i, $i = 0, \ldots, N-1$. This approximation error will be analyzed in Section 4.3.

MDBDP is a machine learning version of the Multi-step 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.

Remark 4.2.3. *We could have also considered, as in the DBDP2 scheme, the automatic differentiation of $\widehat{\mathcal{U}}_i$ for the approximation of the gradient Z_{t_i}. However, as shown in the numerical tests of [HPW20], this approach leads to less accurate results than the DBDP1 algorithm which uses an additional neural network. Moreover, at least for theoretical analysis, it requires to optimize over C^1 neural networks, which is a restrictive assumption. Hence we focus on a DBDP1-type method.*

In the numerical implementation, the expectation defining the loss function J_i^{MB} in (4.2.8) 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+21], [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 $i = N-1, \ldots, 0$, at each iteration of the SGD, we pick a sample $(X_j^k, \Delta W_j^k)_{j=i,\ldots,N}$ of the Euler process and increment of Brownian motion $(X_j, \Delta W_j)_j$, $k = 1, \ldots, K$, of mini-batch size K, and consider the empirical loss function:

$$\begin{aligned}
&\mathbb{J}_i^K(\theta) \\
&= \frac{1}{K}\sum_{k=1}^{K} \Big| g(X_N^k) - \sum_{j=i+1}^{N-1} f(t_j, X_j^k, \widehat{\mathcal{U}}_j(X_j^k), \widehat{\mathcal{Z}}_j(X_j^k))\Delta t_j - \sum_{j=i+1}^{N-1} \widehat{\mathcal{Z}}_j(X_j^k).\Delta W_j^k \\
&\qquad - \mathcal{U}^\theta(X_i^k) - f(t_i, X_i^k, \mathcal{U}^\theta(X_i^k), \mathcal{Z}^\theta(X_i^k))\Delta t_i - \mathcal{Z}^\theta(X_i^k).\Delta W_i^k \Big|^2, \quad (4.2.9)
\end{aligned}$$

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 $j \in [\![i+1, N-1]\!]$. In practice, the number of iterations for SGD at the initial induction time $N-1$ should be large enough so as to learn accurately the value function $u(t_{N-1}, .)$ and its gradient $D_x u(t_{N-1}, .)$ via $\widehat{\mathcal{U}}^{\hat\theta_{N-1}}$ and $\widehat{\mathcal{Z}}^{\hat\theta_{N-1}}$. However, it is then expected 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 SGD 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], [BJK19], and the references therein.

4.3 Convergence Analysis

This section is devoted to the approximation error and rate of convergence of the MDBDP, DS, and DBDP schemes

described in Section 4.2.

We make the following standard assumptions on the coefficients of the forward-backward equation associated to semilinear PDE (4.1.1).

Algorithm 6: 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 (4.2.9)
    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

Assumption 4.3.1. *(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 $(t',x',y',z') \in [0,T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d$,

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

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

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

Assumption 4.3.1 guarantees the existence and uniqueness of an adapted solution $(\mathcal{X}, Y, Z)$ to the forward-backward equation (4.2.2)-(4.2.3), satisfying

$$\mathbb{E}\Big[\sup_{0\leq t\leq T} |\mathcal{X}_t|_2^2 + \sup_{0\leq t\leq T} |Y_t|^2 + \int_0^T |Z_t|_2^2 \mathrm{d}t\Big] <\ \infty,$$

(see for instance Theorem 3.3.1, Theorem 4.2.1, Theorem 4.3.1 from [Zha17]). Given the time grid $\pi = \{t_i : i = 0, \dots, N\}$, let us introduce the L^2-regularity of Z:

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

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 g is also Lipschitz, we have

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

Here, the standard notation $O(|\pi|)$ means that $\limsup_{|\pi|\to 0}\ |\pi|^{-1} O(|\pi|) < \infty$.

Lemma 4.3.1. *Under Assumption 4.3.2 (ii), the following standard estimate for the Euler-Maruyama scheme holds when $\Delta t_i \to 0$*

$$\mathbb{E}|X_{i+1}^x - X_{i+1}^{x'}|_2^2 \leq (1 + C\Delta t_i)|x - x'|_2^2,$$

where $X_{i+1}^x := x + \mu(t_i, x)\Delta t_i + \sigma(t_i, x)\Delta W_i$.

Proof. By expanding the square, simply notice that the dominant terms when $\Delta t_i \to 0$ are of older Δt_i because the term of order $\sqrt{\Delta t_i}$, namely $(x - x') \cdot (\sigma(t_i, x) - \sigma(t_i, x'))\Delta W_i$ has a null expectation and all other terms are dominated by Δt_i. □

4.3.1 Convergence of the MDBDP Scheme

We fix classes of functions $\mathcal{N}_i$ and $\mathcal{N}_i'$ for the approximations respectively of the solution and its gradient, and define $(\widehat{\mathcal{U}}_i^{(1)}, \widehat{\mathcal{Z}}_i^{(1)})$ as the output of the MDBDP scheme at times t_i, $i = 0, \dots, N$.

Let us define (implicitly) the process

$$\begin{cases} V_i^{(1)} &= \mathbb{E}_i\Big[g(X_N) - f\big(t_i, X_i, V_i^{(1)}, \overline{\widehat{Z}}_i^{(1)}\big)\Delta t_i - \sum_{j=i+1}^{N-1} f\big(t_j, X_j, \widehat{\mathcal{U}}_j^{(1)}(X_j), \widehat{\mathcal{Z}}_j^{(1)}(X_j)\big)\Delta t_j\Big], \\ \overline{\widehat{Z}}_i^{(1)} &= \mathbb{E}_i\Big[\frac{g(X_N)\Delta W_i}{\Delta t_i} - \sum_{j=i+1}^{N-1} f\big(t_j, X_j, \widehat{\mathcal{U}}_j^{(1)}(X_j), \widehat{\mathcal{Z}}_j^{(1)}(X_j)\big)\frac{\Delta W_i \Delta t_j}{\Delta t_i}\Big], \ i = 0, \dots, N, \end{cases} \tag{4.3.1}$$

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

$$V_i^{(1)} \ = \ v_i^{(1)}(X_i), \qquad \overline{\widehat{Z}}_i^{(1)} \ = \ \widehat{z}_i^{(1)}(X_i), \quad i = 0, \dots, N, \tag{4.3.2}$$

for some deterministic functions $v_i^{(1)}, \widehat{z}_i^{(1)}$. Let us then introduce

$$\varepsilon_i^{1,y} \ := \ \inf_{\mathcal{U} \in \mathcal{N}_i} \mathbb{E}\big|v_i^{(1)}(X_i) - \mathcal{U}(X_i)\big|^2, \qquad \varepsilon_i^{1,z} \ := \ \inf_{\mathcal{Z} \in \mathcal{N}_i'} \mathbb{E}\big|\widehat{z}_i^{(1)}(X_i) - \mathcal{Z}(X_i)\big|_2^2,$$

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

Theorem 4.3.1 (Approximation error of MDBDP)**.** *Under Assumption 4.3.1, there exists a constant $C > 0$ (depending only on the data μ, σ, f, g, d, T) such that in the limit $|\pi| \to 0$*

$$\begin{aligned} &\sup_{i \in [\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_i)\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} \big|Z_s - \widehat{\mathcal{Z}}_i^{(1)}(X_i)\big|_2^2 \, \mathrm{d}s\Big] \\ &\leq \ C\Big(\mathbb{E}\big|g(\mathcal{X}_T) - g(X_N)\big|^2 + |\pi| + \varepsilon^Z(\pi) + \sum_{j=0}^{N-1} (\varepsilon_j^{1,y} + \Delta t_j \varepsilon_j^{1,z})\Big). \end{aligned} \tag{4.3.3}$$

Remark 4.3.1. The upper bound in (4.3.3)

consists of four terms. The first three terms correspond to the time discretization of BSDE, similarly as in [Zha04], [BT04], namely (i) the strong approximation of the terminal condition (depending on the forward scheme and 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 $\sum_{j=0}^{N-1}(\varepsilon_j^{1,y} + \Delta t_j \varepsilon_j^{1,z})$ in (4.3.3) is better than the one for the DBDP scheme derived in [HPW20], with an order $\sum_{j=0}^{N-1}(N\varepsilon_j^{1,y} + \varepsilon_j^{1,z})$. In the work [GT14] which introduced the multistep scheme with linear regression, the authors noticed the same improvement in the error propagation in comparison with the one-step classical scheme [GLW05]. □

We next study convergence for the approximation error of the MDBDP scheme, for a specific choice of functions classes $\mathcal{N}_i$ and $\mathcal{N}_i'$ and with the additional assumption that f does not depend on z.

Assumption 4.3.2. *The generator function f is independent of z. Namely, for all $(t, x, y, z, z') \in [0, T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^d$,*

$$f(t, x, y, z) = f(t, x, y, z').$$

Actually, if f is linear in z: $f(t,x,y,z) = \bar{f}(t,x,y) + \lambda(t,x).z$, one can boil down to Assumption 4.3.2 for $\bar{f}$ by incorporating the linearity in the drift function μ, namely with the modified drift: $\bar{\mu}(t,x) = \mu(t,x) - \sigma\lambda(t,x)$.

Proposition 4.3.1 (Rate of convergence of MDBDP)**.** *Let Assumption 4.3.1 and Assumption 4.3.2 hold, and assume that $\mathcal{X}_0 \in L^{2+\delta}(\mathcal{F}_0, \mathbb{R}^d)$, for some $\delta > 0$, and g is $[g]-$Lipschitz. Then, there exists a bounded sequence K_i (uniformly in i, N) such that for GroupSort neural networks classes $\mathcal{N}_i = \mathcal{G}^{\zeta_\kappa}_{K_i,d,1,\ell,m}$, and $\mathcal{N}'_i = \mathcal{G}^{\zeta_\kappa}_{\sqrt{\frac{d}{\Delta t_i}}K_i,d,d,\ell,m}$, we have*

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

with a grouping size $\kappa = O(2\sqrt{d}N^2)$, depth $\ell+1 = O(d^2)$ and width $\sum_{i=0}^{\ell-1} m_i = O((2\sqrt{d}N^2)^{d^2-1})$ in the case $d > 1$. If $d = 1$, take $\kappa = O(N^2)$, depth $\ell + 1 = 3$ and width $\sum_{i=0}^{\ell-1} m_i = O(N^2)$. Here, the constants in the $O(\cdot)$ term depend only on $\mu, \sigma, f, g, d, T, \mathcal{X}_0$.

4.3.2 Convergence of the DS Scheme

We consider classes $\mathcal{N}_i^{\gamma,\eta}$ of differentiable γ_i-Lipschitz functions with η_i-Lipschitz derivative for sequences $\gamma = (\gamma_i)_i$, $\eta = (\eta_i)_i$ and define $\widehat{\mathcal{U}}_i^{(2)}$ as the output of the DS scheme at times t_i, $i = 0, \dots, N$. .

Let us define the process

$$V_i^{(2)} = \mathbb{E}_i\Big[\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - f\big(t_i, X_i, \mathbb{E}_i[\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1})], \mathbb{E}_i[\sigma(t_i,X_i)^\intercal D_x[\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1})]\big)\Delta t_i\Big], \tag{4.3.4}$$

for $i \in [\![0, N-1]\!]$, and $V_N^{(2)} = \widehat{\mathcal{U}}_N^{(2)}(X_N)$. By the Markov property of $(X_i)_i$, we have $V_i^{(2)} = v_i^{(2)}(X_i)$, for some functions $v_i^{(2)} : \mathbb{R}^d \to \mathbb{R}$, $i \in [\![0, N-1]\!]$, and we introduce

$$\varepsilon_i^{\gamma,\eta} = \begin{cases} \inf_{\mathcal{U}\in\mathcal{N}_i^{\gamma,\eta}} \mathbb{E}\big|v_i^{(2)}(X_i) - \mathcal{U}(X_i)\big|^2, & i = 0, \dots, N-1, \\ \inf_{\mathcal{U}\in\mathcal{N}_i^{\gamma,\eta}} \mathbb{E}\big|g(X_N) - \mathcal{U}(X_N)\big|^2, & i = N. \end{cases}$$

the L^2-approximation error in the class $\mathcal{N}_i^{\gamma,\eta}$ of the functions $v_i^{(2)}$, $i = 0, \dots, N-1$, and g.

Theorem 4.3.2 (Approximation error of DS)**.** *Let Assumption 4.3.1 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 in the limit $|\pi| \to 0$*

$$\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(2)}(X_i)\big|^2 \le C\Big(\mathbb{E}\big|g(X_N) - g(\mathcal{X}_T)\big|^2 + |\pi| + \varepsilon^Z(\pi) + \max_i\big[\gamma_i^2, \eta_i^2\big]|\pi| + \varepsilon_N^{\gamma,\eta} + N\sum_{i=0}^{N-1}\varepsilon_i^{\gamma,\eta}\Big). \tag{4.3.5}$$

Remark 4.3.2. We retrieve a similar error as in the analysis of the DBDP2 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 $\varepsilon_N^{\gamma,\eta}$ is removed in (4.3.5). □

The GroupSort neural networks being only continuous but not differentiable, we are not able to express a convergence rate for the Deep Splitting scheme in terms of the architecture and number of neurons to choose, like in Propositions 4.3.1, 4.3.2. It would require a quantitative approximation result for C^1 neural networks with bounded Lipschitz gradient, and this is left to future research.

4.3.3 Convergence of the DBDP Scheme

We consider classes of functions $\mathcal{N}_i$ and $\mathcal{N}_i'$ for the approximations of the solution and its gradient, and define $(\widehat{\mathcal{U}}_i^{(3)}, \widehat{\mathcal{Z}}_i^{(3)})$ as the output of the DBDP scheme at times t_i, $i = 0, \dots, N$. Let us define (implicitly) the process

$$\begin{cases} V_i^{(3)} &= \mathbb{E}_i\Big[\widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}) - f\big(t_i, X_i, V_i^{(3)}, \overline{\widehat{Z}_i}^{(3)}\big)\Delta t_i\Big] \\ \overline{\widehat{Z}_i}^{(3)} &= \mathbb{E}_i\Big[\widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1})\frac{\Delta W_i}{\Delta t_i}\Big], \quad i = k, \dots, N-1. \end{cases}$$

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

$$V_i^{(3)} \;=\; v_i^{(3)}(X_i), \qquad \overline{\widehat{Z}_i} \;=\; \widehat{z}_i^{(3)}(X_i), \quad i = 0, \dots, N, \tag{4.3.6}$$

for some deterministic functions $v_i^{(3)}, \widehat{z}_i^{(3)}$. Let us then introduce

$$\varepsilon_i^{3,y} \;:=\; \inf_{\mathcal{U}\in\mathcal{N}_i} \mathbb{E}\big|v_i^{(3)}(X_i) - \mathcal{U}(X_i)\big|^2, \qquad \varepsilon_i^{3,z} \;:=\; \inf_{\mathcal{Z}\in\mathcal{N}_i'} \mathbb{E}\big|\widehat{z}_i^{(3)}(X_i) - \mathcal{Z}(X_i)\big|_2^2,$$

for $i = 0, \dots, N-1$, which represent the L^2-approximation errors of the functions $v_i^{(3)}, \widehat{z}_i^{(3)}$ in the classes $\mathcal{N}_i$ and $\mathcal{N}_i'$.

Theorem 4.3.3 (Huré, Pham, Warin [HPW20] : Approximation error of DBDP)**.** *Under Assumption 4.3.1, there exists a constant $C > 0$ (depending only on the data μ, σ, f, g, d, T) such that in the limit $|\pi| \to 0$*

$$\begin{aligned} &\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(3)}(X_i)\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \big|Z_s - \widehat{\mathcal{Z}}_i^{(3)}(X_i)\big|_2^2 \,\mathrm{d}s\Big] \\ &\le\; C\Big(\mathbb{E}\big|g(\mathcal{X}_T) - g(X_N)\big|^2 + |\pi| + \varepsilon^Z(\pi) + N\sum_{j=0}^{N-1}(\varepsilon_j^{3,y} + \Delta t_j \varepsilon_j^{3,z})\Big). \end{aligned} \tag{4.3.7}$$

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

Proposition 4.3.2 (Rate of convergence of DBDP)**.** *Let Assumption 4.3.1 hold, and assume that $\mathcal{X}_0 \in L^{2+\delta}(\mathcal{F}_0, \mathbb{R}^d)$, for some $\delta > 0$, and g is $[g]-$Lipschitz. Then, there exists a bounded sequence K_i (uniformly in i, N) such that for $\mathcal{N}_i = \mathcal{G}^{\zeta_\kappa}_{K_i,d,1,\ell,m}$, and $\mathcal{N}_i' = \mathcal{G}^{\zeta_\kappa}_{\sqrt{\frac{d}{\Delta t_i}}K_i,d,d,\ell,m}$, we have*

$$\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(3)}(X_i)\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \big|Z_s - \widehat{\mathcal{Z}}_i^{(3)}(X_i)\big|_2^2 \,\mathrm{d}s\Big] \;=\; O(1/N),$$

with a grouping size $\kappa = O(2\sqrt{d}N^3)$, depth $\ell+1 = O(d^2)$ and width $\sum_{i=0}^{\ell-1} m_i = O((2\sqrt{d}N^3)^{d^2-1})$ in the case $d > 1$. If $d = 1$, take $\kappa = O(N^3)$, depth $\ell + 1 = 3$ and width $\sum_{i=0}^{\ell-1} m_i = O(N^3)$. Here, the constants in the $O(\cdot)$ term depend only on $\mu, \sigma, f, g,\ d, T, \mathcal{X}_0$.

4.4 Proof of the Main Theoretical Results

4.4.1 Proof of Theorem 4.3.1

Let us introduce the processes $(\bar{V}_i, \bar{Z}_i)_i$ arising from the time discretization of the BSDE (4.2.3), and defined by the *implicit* backward Euler scheme:

$$\begin{cases} \bar{V}_i^{(1)} &= \mathbb{E}_i\Big[\bar{V}_{i+1}^{(1)} - f\big(t_i, X_i, \bar{V}_i^{(1)}, \bar{Z}_i^{(1)}\big)\Delta t_i\Big] \\ \bar{Z}_i^{(1)} &= \mathbb{E}_i\Big[\bar{V}_{i+1}^{(1)}\frac{\Delta W_i}{\Delta t_i}\Big], \quad i = 0, \dots, N-1, \end{cases} \tag{4.4.1}$$

starting from $\bar{V}_N^{(1)} = g(X_N)$. We recall from [Zha04] the time discretization error:

$$\begin{aligned}&\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|Y_{t_i} - \bar{V}_i^{(1)}\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \big|Z_s - \bar{Z}_i^{(1)}\big|_2^2\,\mathrm{d}s\Big]\\ &\le C\Big(\mathbb{E}\big|g(\mathcal{X}_T) - g(X_N)\big|^2 + |\pi| + \varepsilon^Z(\pi)\Big),\end{aligned} \tag{4.4.2}$$

for some constant C depending only on the coefficients satisfying Assumption 4.3.1.

Let us introduce the auxiliary process

$$\hat{V}_i^{(1)} = \mathbb{E}_i\Big[g(X_N) - \sum_{j=i}^{N-1} f\big(t_j, X_j, \widehat{\mathcal{U}}_j^{(1)}(X_j), \widehat{\mathcal{Z}}_j^{(1)}(X_j)\big)\Delta t_j\Big], \quad i = 0,\dots,N, \tag{4.4.3}$$

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

$$\hat{V}_i^{(1)} = \mathbb{E}_i\Big[\hat{V}_{i+1}^{(1)} - f\big(t_i, X_i, \widehat{\mathcal{U}}_i^{(1)}(X_i), \widehat{\mathcal{Z}}_i^{(1)}(X_i)\big)\Delta t_i\Big], \quad i = 0,\dots,N-1. \tag{4.4.4}$$

Observe also that $\overline{\widehat{Z}}_i^{(1)}$ defined in (4.3.1) satisfies

$$\overline{\widehat{Z}}_i^{(1)} = \mathbb{E}_i\Big[\hat{V}_{i+1}^{(1)}\frac{\Delta W_i}{\Delta t_i}\Big], \quad i = 0,\dots,N-1. \tag{4.4.5}$$

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

$$\begin{aligned}&\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_i)\big|^2\\ &\le 4\Big(\mathbb{E}\big|Y_{t_i} - \bar{V}_i^{(1)}\big|^2 + \mathbb{E}\big|\bar{V}_i^{(1)} - \hat{V}_i^{(1)}\big|^2 + \mathbb{E}\big|\hat{V}_i^{(1)} - V_i^{(1)}\big|^2 + \mathbb{E}\big|V_i^{(1)} - \hat{\mathcal{U}}_i^{(1)}(X_i)\big|^2\Big)\\ &=: 4(I_i^1 + I_i^2 + I_i^3 + I_i^4),\end{aligned} \tag{4.4.6}$$

and analyze each of these contribution 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 4.3.1. Notice that the first contribution term is the time discretization error for BSDE given by (4.4.2), and we shall study the three other terms in the following steps.

Step 1. Fix $i \in [\![0, N-1]\!]$. From the definition (4.3.1) of $V_i^{(1)}$ and by the martingale representation theorem, there exists a square integrable process $\{\widehat{Z}_s^{(1)}, t_i \le s \le T\}$ s.t.

$$\begin{aligned}&g(X_N) - f\big(t_i, X_i, V_i^{(1)}, \overline{\widehat{Z}_i^{(1)}}\big)\Delta t_i - \sum_{j=i+1}^{N-1} f\big(t_j, X_j, \widehat{\mathcal{U}}_j^{(1)}(X_j), \widehat{\mathcal{Z}}_j^{(1)}(X_j)\big)\Delta t_j\\ &= V_i + \int_{t_i}^{t_N} \widehat{Z}_s^{(1)}.\mathrm{d}W_s.\end{aligned} \tag{4.4.7}$$

From the definition (4.3.1) of $\overline{\widehat{Z}}_i^{(1)}$, and by Itô isometry, we then have

$$\overline{\widehat{Z}}_i^{(1)} = \frac{\mathbb{E}_i\big[\int_{t_i}^{t_{i+1}} \widehat{Z}_s^{(1)}\mathrm{d}s\big]}{\Delta t_i}, \text{ i.e. } \mathbb{E}_i\Big[\int_{t_i}^{t_{i+1}} \big(\widehat{Z}_s^{(1)} - \overline{\widehat{Z}}_i^{(1)}\big)\mathrm{d}s\Big] = 0. \tag{4.4.8}$$

Plugging (4.4.7) into (4.2.8), we see that the loss function of the MDBDP scheme can be rewritten

as

$$\begin{aligned}
&\mathcal{J}_i^{MB}(\mathcal{U}_i, \mathcal{Z}_i)\\
&= \mathbb{E}\Big|V_i^{(1)} - \mathcal{U}_i(X_i) + \Delta t_i\big[f\big(t_i, X_i, V_i^{(1)}, \overline{\widehat{Z}}_i^{(1)}\big) - f\big(t_i, X_i, \mathcal{U}_i(X_i), \mathcal{Z}_i(X_i)\big)\big]\\
&\quad + \sum_{j=i+1}^{N-1}\int_{t_j}^{t_{j+1}}\big[\widehat{Z}_s^{(1)} - \widehat{\mathcal{Z}}_j(X_j)\big].\mathrm{d}W_s + \int_{t_i}^{t_{i+1}}\big[\widehat{Z}_s^{(1)} - \mathcal{Z}_i(X_i)\big].\mathrm{d}W_s\Big|^2\\
&= \widetilde{\mathcal{J}}_i^{MB}(\mathcal{U}_i, \mathcal{Z}_i) + \mathbb{E}\Big[\sum_{j=i}^{N-1}\int_{t_j}^{t_{j+1}}\big|\widehat{Z}_s^{(1)} - \overline{\widehat{Z}}_j^{(1)}\big|_2^2\mathrm{d}s\Big]\\
&\quad + \sum_{j=i+1}^{N-1}\Delta t_j\mathbb{E}\big|\overline{\widehat{Z}}_j^{(1)} - \widehat{\mathcal{Z}}_j(X_j)\big|_2^2, \qquad (4.4.9)
\end{aligned}$$

where we use (4.4.8), and

$$\begin{aligned}
&\widetilde{\mathcal{J}}_i^{MB}(\mathcal{U}_i, \mathcal{Z}_i)\\
&:= \mathbb{E}\Big|V_i^{(1)} - \mathcal{U}_i(X_i) + \Delta t_i\big[f\big(t_i, X_i, V_i^{(1)}, \overline{\widehat{Z}}_i^{(1)}\big) - f\big(t_i, X_i, \mathcal{U}_i(X_i), \mathcal{Z}_i(X_i)\big)\big]\Big|^2\\
&\quad + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}}_i^{(1)} - \mathcal{Z}_i(X_i)\big|_2^2.
\end{aligned}$$

It is clear by Lipschitz continuity of f in Assumption 4.3.1 that

$$\widetilde{\mathcal{J}}_i^{MB}(\mathcal{U}_i, \mathcal{Z}_i) \le C\Big(\mathbb{E}\big|V_i^{(1)} - \mathcal{U}_i(X_i)\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}}_i^{(1)} - \mathcal{Z}_i(X_i)\big|_2^2\Big). \qquad (4.4.10)$$

On the other hand, by the Young inequality: $(1-\beta)a^2 + \big(1-\frac{1}{\beta}\big)b^2 \le (a+b)^2 \le (1+\beta)a^2 + \big(1+\frac{1}{\beta}\big)b^2$, for all $(a,b)\in\mathbb{R}^2$, and $\beta > 0$, we have

$$\begin{aligned}
&\widetilde{\mathcal{J}}_i^{MB}(\mathcal{U}_i, \mathcal{Z}_i)\\
&\ge (1-\beta)\mathbb{E}\big|V_i^{(1)} - \mathcal{U}_i(X_i)\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}}_i^{(1)} - \mathcal{Z}_i(X_i)\big|_2^2\\
&+ \quad \Big(1 - \frac{1}{\beta}\Big)|\Delta t_i|^2\mathbb{E}\big|f\big(t_i, X_i, \mathcal{U}_i(X_i), \mathcal{Z}_i(X_i)\big) - f\big(t_i, X_i, V_i^{(1)}, \overline{\widehat{Z}}_i^{(1)}\big)\big|^2\\
&\ge (1-\beta)\mathbb{E}\big|V_i^{(1)} - \mathcal{U}_i(X_i)\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}}_i^{(1)} - \mathcal{Z}_i(X_i)\big|_2^2\\
&\quad - \frac{2[f]_L^2}{\beta}|\Delta t_i|^2\Big(\mathbb{E}\big|\mathcal{U}_i(X_i) - V_i^{(1)}\big|^2 + \mathbb{E}\big|\mathcal{Z}_i(X_i) - \overline{\widehat{Z}}_i^{(1)}\big|_2^2\Big)\\
&\ge \Big(1 - \big(4[f]_L^2 + \frac{1}{2}\big)\Delta t_i\Big)\mathbb{E}\big|V_i^{(1)} - \mathcal{U}_i(X_i)\big|^2 + \frac{1}{2}\Delta t_i\mathbb{E}\big|\overline{\widehat{Z}}_i^{(1)} - \mathcal{Z}_i(X_i)\big|_2^2, \qquad (4.4.11)
\end{aligned}$$

where we use the Lipschitz continuity of f in the second inequality, and choose explicitly $\beta = 4[f]_L^2\Delta t_i$ (< 1 for Δt_i small enough) in the last one. By applying inequality (4.4.11) to $(\mathcal{U}_i, \mathcal{Z}_i) = (\widehat{\mathcal{U}}_i^{(1)}, \widehat{\mathcal{Z}}_i^{(1)})$, which is a minimizer of $\tilde{\mathcal{J}}_i^{MB}$ by (4.4.9), and combining with (4.4.10), this yields for Δt_i small enough and for all functions $\mathcal{U}_i$, $\mathcal{Z}_i$:

$$\begin{aligned}
&\mathbb{E}\big|V_i^{(1)} - \widehat{\mathcal{U}}_i^{(1)}(X_i)\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}}_i^{(1)} - \widehat{\mathcal{Z}}_i^{(1)}(X_i)\big|_2^2\\
&\le C\Big(\mathbb{E}\big|V_i - \mathcal{U}_i(X_i)\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}}_i - \mathcal{Z}_i(X_i)\big|_2^2\Big).
\end{aligned}$$

By minimizing over $\mathcal{U}_i, \mathcal{Z}_i$ in the right hand side, we get the approximation error in the classes $\mathcal{N}_i, \mathcal{N}_i'$ of the regressed functions $V_i^{(1)}$, $\overline{\widehat{Z}}_i^{(1)}$:

$$\mathbb{E}\big|V_i^{(1)} - \widehat{\mathcal{U}}_i^{(1)}(X_i)\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}}_i^{(1)} - \widehat{\mathcal{Z}}_i^{(1)}(X_i)\big|_2^2 \le C(\varepsilon_i^{1,y} + \Delta t_i\varepsilon_i^{1,z}). \qquad (4.4.12)$$

Step 2. From the expressions of $V_i^{(1)}$ and $\hat{V}_i^{(1)}$ in (4.3.1), (4.4.3), and by Lipschitz continuity of f, we have by (4.4.12):

$$\begin{aligned}\mathbb{E}\big|\hat{V}_i^{(1)} - V_i^{(1)}\big|^2 =&\ \Delta t_i^2 \mathbb{E}\Big|\mathbb{E}_i\big[f\big(t_i, X_i, V_i^{(1)}, \overline{\widehat{Z}}_i^{(1)}\big) - f\big(t_i, X_i, \widehat{\mathcal{U}}_i^{(1)}(X_i), \widehat{\mathcal{Z}}_i^{(1)}(X_i)\big)\big]\Big|^2 \\ \le&\ 2[f]_L^2 |\Delta t_i|^2 \Big(\mathbb{E}\big|V_i^{(1)} - \widehat{\mathcal{U}}_i^{(1)}(X_i)\big|^2 + \mathbb{E}\big|\overline{\widehat{Z}}_i^{(1)} - \widehat{\mathcal{Z}}_i^{(1)}(X_i)\big|_2^2\Big) \\ \le&\ C\Delta t_i(\varepsilon_i^{1,y} + \Delta t_i \varepsilon_i^{1,z}), \qquad i = 0, \dots, N. \end{aligned} \tag{4.4.13}$$

Step 3. From the recursive expressions of $\bar{V}_i^{(1)}$, $\hat{V}_i^{(1)}$ in (4.4.1), (4.4.4), and applying the Young, the Cauchy-Schwarz inequalities, together with the Lipschitz condition of f, we get for $\beta > 0$:

$$\begin{aligned}&\mathbb{E}\big|\bar{V}_i^{(1)} - \hat{V}_i^{(1)}\big|^2 \\ &\le\ (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\bar{V}_{i+1}^{(1)} - \hat{V}_{i+1}^{(1)}\big]\Big|^2 + 2[f]_L^2\Big(1 + \frac{1}{\beta}\Big)|\Delta t_i|^2\Big(\mathbb{E}\big|\bar{V}_i^{(1)} - \hat{\mathcal{U}}_i^{(1)}(X_i)\big|^2 + \mathbb{E}\big|\bar{Z}_i^{(1)} - \widehat{\mathcal{Z}}_i^{(1)}(X_i)\big|_2^2\Big) \\ &\le\ (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\bar{V}_{i+1}^{(1)} - \hat{V}_{i+1}^{(1)}\big]\Big|^2 + 2[f]_L^2\Big(1 + \frac{1}{\beta}\Big)|\Delta t_i|^2\big(3\mathbb{E}|\bar{V}_i^{(1)} - \hat{V}_i^{(1)}|^2 + 2\mathbb{E}\big|\bar{Z}_i^{(1)} - \overline{\widehat{Z}}_i^{(1)}\big|_2^2\big) \\ &+\ 2[f]_L^2\Big(1 + \frac{1}{\beta}\Big)|\Delta t_i|^2\Big(3\mathbb{E}|\hat{V}_i^{(1)} - V_i^{(1)}|^2 + 3\mathbb{E}|V_i^{(1)} - \widehat{\mathcal{U}}_i^{(1)}(X_i)|^2 + 2\mathbb{E}\big|\overline{\widehat{Z}}_i^{(1)} - \widehat{\mathcal{Z}}_i(X_i)\big|_2^2\Big) \\ &\le\ (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\bar{V}_{i+1}^{(1)} - \hat{V}_{i+1}^{(1)}\big]\Big|^2 + (1+\beta)\frac{2[f]_L^2|\Delta t_i|^2}{\beta}\big(3\mathbb{E}|\bar{V}_i^{(1)} - \hat{V}_i^{(1)}|^2 + 2\mathbb{E}\big|\bar{Z}_i^{(1)} - \overline{\widehat{Z}}_i^{(1)}\big|_2^2\big) \\ &+\ C[f]_L^2\Big(1 + \frac{1}{\beta}\Big)\Delta t_i(\varepsilon_i^{1,y} + \Delta t_i\varepsilon_i^{1,z}), \end{aligned} \tag{4.4.14}$$

where we use (4.4.12), (4.4.13) in the last inequality. Moreover, by (4.4.1), (4.4.5), we have

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

and thus by the Cauchy-Schwarz inequality

$$\Delta t_i \mathbb{E}\big|\bar{Z}_i^{(1)} - \overline{\widehat{Z}}_i^{(1)}\big|_2^2 \le\ d\Big(\mathbb{E}\big|\bar{V}_{i+1}^{(1)} - \hat{V}_{i+1}^{(1)}\big|^2 - \mathbb{E}\Big|\mathbb{E}_i\big[\bar{V}_{i+1}^{(1)} - \hat{V}_{i+1}^{(1)}\big]\Big|^2\Big). \tag{4.4.15}$$

Plugging into (4.4.14), and choosing $\beta = 4d[f]_L^2\Delta t_i$, gives

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

By discrete Gronwall lemma, and recalling that $\bar{V}_N^{(1)} = \hat{V}_N^{(1)}$ $(= g(X_N))$, we then obtain

$$\sup_{i \in [\![0,N]\!]} \mathbb{E}\big|\bar{V}_i^{(1)} - \hat{V}_i^{(1)}\big|^2 \le\ C\sum_{i=0}^{N-1}\big(\varepsilon_i^{1,y} + \Delta t_i\varepsilon_i^{1,z}\big). \tag{4.4.16}$$

The required bound for the approximation error on Y follows by plugging (4.4.2), (4.4.12), (4.4.13), and (4.4.16) into (4.4.6).

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

$$\begin{aligned}&\mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}}\big|Z_s^{(1)} - \widehat{\mathcal{Z}}_i^{(1)}(X_i)\big|_2^2\,\mathrm{d}s\Big] \\ &\le\ 3\sum_{i=0}^{N-1}\Big(\mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Z_s^{(1)} - \bar{Z}_i^{(1)}\big|_2^2\,\mathrm{d}s\Big] + \Delta t_i\mathbb{E}\big|\bar{Z}_i^{(1)} - \overline{\widehat{Z}}_i^{(1)}\big|_2^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}}_i^{(1)} - \widehat{\mathcal{Z}}_i^{(1)}(X_i)\big|_2^2\Big) \end{aligned} \tag{4.4.17}$$

By summing the inequality (4.4.15) (recalling that $\bar{V}_N^{(1)} = \hat{V}_N^{(1)}$), and using (4.4.14), we have for $\beta \in (0,1)$:

$$
\begin{aligned}
&\sum_{i=0}^{N-1} \Delta t_i \mathbb{E}|\bar{Z}_i^{(1)} - \overline{\widehat{Z}}_i^{(1)}|^2 \\
&\leq\ d \sum_{i=0}^{N-1} \Big(\mathbb{E}\big|\bar{V}_i^{(1)} - \hat{V}_i^{(1)}\big|^2 - \mathbb{E}\big|\mathbb{E}_i\big[\bar{V}_{i+1}^{(1)} - \hat{V}_{i+1}^{(1)}\big]\big|^2 \Big) \\
&\leq\ d \sum_{i=0}^{N-1} \Big(\beta \mathbb{E}\Big|\mathbb{E}_i\big[\bar{V}_{i+1}^{(1)} - \hat{V}_{i+1}^{(1)}\big]\Big|^2 + \big(1 + \frac{1}{\beta}\big)\big(2[f]_L^2 |\Delta t_i|^2\big)\big(3\mathbb{E}|\bar{V}_i^{(1)} - \hat{V}_i^{(1)}|^2 + 2\mathbb{E}\big|\bar{Z}_i^{(1)} - \overline{\widehat{Z}}_i^{(1)}\big|_2^2\big) \\
&\qquad +\ C[f]_L^2 \big(1 + \frac{1}{\beta}\big)\Delta t_i(\varepsilon_i^{1,y} + \Delta t_i \varepsilon_i^{1,z})\Big) \\
&\leq\ d \sum_{i=0}^{N-1} \Big(\frac{8d[f]_L^2 \Delta t_i}{1 - 8d[f]_L^2 \Delta t_i} \mathbb{E}\big|\mathbb{E}_i\big[\bar{V}_{i+1}^{(1)} - \hat{V}_{i+1}^{(1)}\big]\big|^2 + \frac{3}{4d}\Delta t_i \mathbb{E}|\bar{V}_i^{(1)} - \hat{V}_i^{(1)}|^2 \ +\ \frac{C}{8d}(\varepsilon_i^{1,y} + \Delta t_i \varepsilon_i^{1,z})\Big) \\
&\qquad +\ \frac{1}{2}\sum_{i=0}^{N-1} \Delta t_i \mathbb{E}\big|\bar{Z}_i^{(1)} - \overline{\widehat{Z}}_i^{(1)}\big|_2^2, \qquad (4.4.18)
\end{aligned}
$$

by choosing explicitly $\beta = \frac{8d[f]_L^2 \Delta t_i}{1 - 8d[f]_L^2 \Delta t_i} = O(\Delta t_i)$ for Δt_i small enough. Plugging (4.4.2), (4.4.12), (4.4.16), and (4.4.18) (using the Jensen inequality) into (4.4.17), this proves the required bound for the approximation error on Z, and completes the proof. □

4.4.2 Proof of Proposition 4.3.1

Let us introduce the flow of the Euler scheme (X_i) by:

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

starting from $X_k^{k,x} = x$ at time step $j = k \in \mathbb{N}^*$. Under Assumption 4.3.2, f does not depend on z so by slight abuse of notation we write $f(t,x,y) = f(t,x,y,z)$. Define

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

and observe that we have the recursive relations:

$$
\begin{aligned}
\hat{V}_{i,1}^{k,x} &= \mathbb{E}_i\Big[\hat{V}_{i+1,1}^{k,x} - f\big(t_i, X_i^{k,x}, \widehat{\mathcal{U}}_i^{(1)}(X_i^{k,x})\big)\Delta t_i\Big],\ i = k, \dots, N, \\
V_{i,1}^{k,x} &= \mathbb{E}_i\Big[\hat{V}_{i+1,1}^{k,x} - f\big(t_i, X_i^{k,x}, V_{i,1}^{k,x}\big)\Delta t_i\Big],\ i = k, \dots, N.
\end{aligned}
$$

Notice by the Markov property of the discretized forward process $(X_i^{k,x})_i$ that

$$V_{j,1}^{k,x} \ =\ v_j^{(1)}(X_j^{k,x}),\ \hat{V}_{j,1}^{k,x} \ =\ \hat{v}_j^{(1)}(X_j^{k,x}),\ \overline{\widehat{Z}}_{j,1}^{k,x} \ =\ \hat{z}_j^{(1)}(X_j^{k,x}),\ j = k, \dots, N$$

for some deterministic function $v_j^{(1)}, \hat{v}_j^{(1)}, \hat{z}_j^{(1)}$ which do not depend on k. Notably $v_j^{(1)}, \hat{z}_j^{(1)}$ are the same functions as in (4.3.2).

Step 1. We first estimate the evolution of the Lipschitz constant of $\hat{v}_i^{(1)}$ when i varies. Let $x' \in \mathbb{R}^d$. By the Cauchy-Schwarz inequality

$$\Delta t_k \mathbb{E}\Big|\overline{\widehat{Z}}_{k,1}^{k,x} - \overline{\widehat{Z}}_{k,1}^{k,x'}\Big|^2 \leq \frac{1}{\Delta t_k}\mathbb{E}\Big|\mathbb{E}_k\Big[(\hat{V}_{k+1,1}^{k,x} - \hat{V}_{k+1,1}^{k,x'})\Delta W_k\Big]\Big|^2 \leq d\ \mathbb{E}\Big|\hat{V}_{k+1,1}^{k,x} - \hat{V}_{k+1,1}^{k,x'}\Big|^2 \quad (4.4.19)$$

Moreover, assuming that $\widehat{\mathcal{U}}_k^{(1)}$ is $[\widehat{\mathcal{U}}_k^{(1)}]-$Lipschitz yields

$$\begin{aligned}\mathbb{E}\Big|\hat{V}_{k,1}^{k,x} - \hat{V}_{k,1}^{k,x'}\Big| &\leq \mathbb{E}\Big|\hat{V}_{k+1,1}^{k,x} - \hat{V}_{k+1,1}^{k,x'}\Big| + \Delta t_i \mathbb{E}\Big|\{f\big(t_k, x', \widehat{\mathcal{U}}_k^{(1)}(x')\big) - f\big(t_k, x, \widehat{\mathcal{U}}_k^{(1)}(x)\big)\}\Big| \\ &\leq \mathbb{E}\Big|\hat{V}_{k+1,1}^{k,x} - \hat{V}_{k+1,1}^{k,x'}\Big| + [f]\Delta t_i(1 + [\widehat{\mathcal{U}}_k^{(1)}])|x - x'|_2.\end{aligned}$$

Step 2. Then for the $v_k^{(1)}$ function, the Young inequality gives

$$\begin{aligned}\mathbb{E}\Big|V_{k,1}^{k,x} - V_{k,1}^{k,x'}\Big|^2 &\leq (1 + \gamma\Delta t_k)\mathbb{E}\Big|\mathbb{E}_k\Big[\hat{V}_{k+1,1}^{k,x} - \hat{V}_{k+1,1}^{k,x'}\Big]\Big|^2 \\ &\quad + (1 + \frac{1}{\gamma\Delta t_k})\Delta t_k^2 \mathbb{E}\Big|\{f\big(t_k, x', V_{k,1}^{k,x'}\big) - f\big(t_k, x, V_{k,1}^{k,x}\big)\}\Big|^2 \\ &\leq (1 + \gamma\Delta t_k)\mathbb{E}\Big|\mathbb{E}_k\Big[\hat{V}_{k+1,1}^{k,x} - \hat{V}_{k+1,1}^{k,x'}\Big]\Big|^2 \\ &\quad + 2[f]^2(1 + \frac{1}{\gamma\Delta t_k})\Delta t_k^2 \mathbb{E}\big[|x - x'|_2^2 + |V_{k,1}^{k,x} - V_{k,1}^{k,x'}|^2\big].\end{aligned}$$

Therefore by choosing $\gamma = 2[f]^2$ for Δt_k small enough

$$\mathbb{E}\Big|V_{k,1}^{k,x} - V_{k,1}^{k,x'}\Big|^2 \leq (1 + (\gamma + 3)\Delta t_k)\mathbb{E}\Big|\hat{V}_{k+1,1}^{k,x} - \hat{V}_{k+1,1}^{k,x'}\Big|^2 + (1 + (\gamma + 3)\Delta t_i)\Delta t_k|x - x'|_2^2.$$

Hence assuming $\hat{v}_{k+1}^{(1)}$ is $[\hat{v}_{k+1}^{(1)}]-$Lipchitz we obtain with Lemma 4.3.1

$$\begin{aligned}|v_k^{(1)}(x) - v_k^{(1)}(x')|^2 &= \mathbb{E}\Big|V_{k,1}^{k,x} - V_{k,1}^{k,x'}\Big|^2 \quad (4.4.20) \\ &\leq (1 + (\gamma + 3)\Delta t_k)((1 + C\Delta t_k)[\hat{v}_{k+1}^{(1)}]^2 + \Delta t_k)|x - x'|_2^2 \\ &\leq (1 + \tilde{C}\Delta t_k)([\hat{v}_{k+1}^{(1)}]^2 + \Delta t_k)|x - x'|_2^2 := [v_k^{(1)}]^2|x - x'|_2^2, \quad (4.4.21)\end{aligned}$$

for Δt_k small enough and another constant $\tilde{C}$.

Step 3. Let $\epsilon > 0$, $\kappa \in \mathbb{N}$, $\ell \in \mathbb{N}$, $m \in \mathbb{R}^\ell$ to be chosen after. Recursively, we choose $\mathcal{N}_k = \mathcal{G}^{\zeta_\kappa}_{[v_k^{(1)}],d,1,\ell,m}$ (with $[v_{N-1}^{(1)}]^2 = (1 + \tilde{C}\Delta t_{N-1})([g]^2 + \Delta t_{N-1})$ by (4.4.21)) to approximate $v_k^{(1)}$ by $[v_k^{(1)}]-$Lipschitz GroupSort neural networks with uniform error $2[v_k]R\epsilon$ on $[-R, R]^d$, see Proposition 4.2.1. Therefore, by Lemma 4.3.1, estimations (4.4.20) and the definition of $[v_k^{(1)}]$ in (4.4.21), for Δt_k small enough

$$\begin{aligned}|\hat{v}_k^{(1)}(x) - \hat{v}_k^{(1)}(x')| &\leq \mathbb{E}\Big|\hat{V}_{k+1,1}^{k,x} - \hat{V}_{k+1,1}^{k,x'}\Big| + [f]\Delta t_k(1 + [\widehat{\mathcal{U}}_k^{(1)}])|x - x'|_2 \\ &\leq (1 + (C + 2[f])\Delta t_k)[\hat{v}_{k+1}^{(1)}]|x - x'|_2 + [f](1 + C\Delta t_k)\Delta t_k|x - x'|_2.\end{aligned}$$

Thus $\hat{v}_k^{(1)}$ is $[\hat{v}_k^{(1)}]$ Lipschitz with

$$[\hat{v}_k^{(1)}] \leq (1 + \hat{C}\Delta t_k)[\hat{v}_{k+1}^{(1)}] + [f](1 + C\Delta t_k)\Delta t_i$$

for a constant $\hat{C}$. By discrete Gronwall lemma over $k = N - 1, \ldots, 0$,

$$[\hat{v}_i^{(1)}]^2 \leq K,\ [v_i^{(1)}]^2 \leq K,$$

uniformly in i, N for some constant K. By (4.4.19) and Proposition 4.2.1, we choose $\mathcal{N}'_k = \mathcal{G}^{\zeta_\kappa}_{\sqrt{\frac{d}{\Delta t_i}}[v_k^{(1)}],d,d,\ell,m}$ to approximate $\widehat{z}_k^{(1)}$ by GroupSort neural networks with uniform error $2\frac{d}{\sqrt{\Delta t_k}}[v_k]R\epsilon$ on $[-R,R]^d$. Thus $\sqrt{\Delta t_k}\widehat{z}_k^{(1)}, \sqrt{\Delta t_k}\mathcal{Z}_k^{(1)}$ are dK Lipschitz, uniformly.

Step 4. The regression errors $\varepsilon_i^{1,y}$ verify from, localization of X_i on $B_2(R)$, the Hölder inequality, and the Markov inequality, the approximation error of $v_i^{(1)}$, $i \in [\![0, N-1]\!]$, by the class of GroupSort neural networks (Proposition 4.2.1)

$$\begin{aligned}
\sqrt{\varepsilon_i^{1,y}} &= \inf_{\mathcal{U}\in\mathcal{G}_{[v_i],d,1}} \big\|v_i^{(1)}(X_i) - \mathcal{U}(X_i)\big\|_2 \\
&\le \inf_{\mathcal{U}\in\mathcal{G}_{[v_i],d,1}} \Big\|\big(v_i^{(1)}(X_i) - \mathcal{U}(X_i)\big)\mathbf{1}_{X_i\in B_2(R)}\Big\|_2 + \Big\|\big(v_i^{(1)}(X_i) - \widehat{\mathcal{U}}_i^{(1)}(X_i)\big)\mathbf{1}_{|X_i|_2\ge R}\Big\|_2 \\
&\le 2KR\epsilon + \mathbb{E}\Big|\big(v_i^{(1)}(X_i) - \widehat{\mathcal{U}}_i^{(1)}(X_i)\big)^{2q}\Big|^{1/2q}\mathbb{E}\Big|\mathbf{1}_{|X_i|_2\ge R}^{\frac{2q}{2q-1}}\Big|^{\frac{2q-1}{2q}} \\
&= 2KR\epsilon + \mathbb{E}\Big|\big(v_i^{(1)}(X_i) - \widehat{\mathcal{U}}_i^{(1)}(X_i)\big)^{2q}\Big|^{1/2q}\mathbb{E}[\mathbf{1}_{|X_i|_2\ge R}]^{\frac{2q-1}{2q}} \\
&\le 2KR\epsilon + \frac{\big(\big\|v_i^{(1)}(X_i) - v_i^{(1)}(0)\big\|_{2q} + \big\|\widehat{\mathcal{U}}_i^{(1)}(X_i) - v_i^{(1)}(0)\big\|_{2q}\big)\big\|X_i\big\|_{\frac{2q}{2q-1}}}{R}, \qquad (4.4.22)
\end{aligned}$$

for $q > 1$ and $2q = 2 + \delta$ with δ as in the statement of the Proposition and by noticing that $(v_i^{(1)}(X_i) - \widehat{\mathcal{U}}_i^{(1)}(X_i)) = (v_i^{(1)}(X_i) - v_i^{(1)}(0) - (\widehat{\mathcal{U}}_i^{(1)}(X_i) - v_i^{(1)}(0)))$. Now, by Lipschitz continuity of $v_i^{(1)}$,$\widehat{\mathcal{U}}^{(1)}$ and because $0 \in B_2(R)$ we have

$$\begin{aligned}
&\big\|\widehat{\mathcal{U}}_i^{(1)}(X_i) - v_i^{(1)}(0)\big\|_{2q} + \big\|v_i^{(1)}(X_i) - v_i^{(1)}(0)\big\|_{2q} \\
&\le \big\|\widehat{\mathcal{U}}_i^{(1)}(0) - v_i^{(1)}(0)\big\|_{2q} + \big\|\widehat{\mathcal{U}}_i^{(1)}(X_i) - \widehat{\mathcal{U}}_i^{(1)}(0)\big\|_{2q} + \big\|v_i^{(1)}(X_i) - v_i^{(1)}(0)\big\|_{2q} \\
&\le 2KR\epsilon + 2K\big\|X_i\big\|_{2q}. \qquad (4.4.23)
\end{aligned}$$

Recalling the standard estimate $\|X_i\|_{2q} \le C(1 + \|\mathcal{X}_0\|_{2q})$, $i = 0, \dots, N$, we then have

$$\varepsilon_i^{1,y} \le C\Big\{R^2\epsilon^2 + \frac{1+R^2\epsilon^2}{R^2}\Big\},$$

for some constant $C(d, \mathcal{X}_0)$ independent of N, R, ϵ. Similarly, repeating (4.4.22) and (4.4.23) by replacing respectively $\widehat{\mathcal{U}}_i^{(1)}$ by $\widehat{\mathcal{Z}}_i^{(1)}$ and $v_i^{(1)}$ by $\hat{z}_i^{(1)}$ and recalling that $\sqrt{\Delta t_k}\widehat{z}_k^{(1)}, \sqrt{\Delta t_k}\mathcal{Z}_k^{(1)}$ are dK Lipschitz uniformly w.r.t N, we obtain

$$\Delta t_i \varepsilon_i^{1,z} \le C\{R^2\epsilon^2 + \frac{1+R^2\epsilon^2}{R^2}\},$$

Then to obtain a convergence rate of $O(1/N)$ in (4.3.3), it suffices to choose R, ϵ such that

$$NR^2\epsilon^2 = O(1/N),\ N\frac{1+R^2\epsilon^2}{R^2} = O(1/N),$$

which is verified with if $d > 1$ with $R = O(N)$, $\epsilon = O(\frac{1}{N^2})$. Then by Proposition 4.2.1, we can choose the previously GroupSort neural networks with grouping size $\kappa = O(2\sqrt{d}N^2)$, depth $\ell+1 = O(d^2)$ and width $\sum_{i=0}^{\ell-1} m_i = O((2\sqrt{d}N^2)^{d^2-1})$ if $d > 1$. If $d = 1$, we can take $\kappa = O(N^2)$, depth $\ell + 1 = 3$ and width $\sum_{i=0}^{\ell-1} m_i = O(N^2)$.

4.4.3 Proof of Theorem 4.3.2

Let us introduce the *explicit* backward Euler scheme of the BSDE (4.2.3):

$$\begin{cases} \bar{V}_i^{(2)} &= \mathbb{E}_i\Big[\bar{V}_{i+1}^{(2)} - f\big(t_i, X_i, \bar{V}_{i+1}^{(2)}, \bar{Z}_i^{(2)}\big)\Delta t_i\Big] \\ \bar{Z}_i^{(2)} &= \mathbb{E}_i\Big[\bar{V}_{i+1}^{(2)}\frac{\Delta W_i}{\Delta t_i}\Big], \quad i = 0, \dots, N-1, \end{cases} \qquad (4.4.24)$$

starting from $\bar{V}_N^{(2)} = g(X_N)$, and which is also known to converge with the same time discretization error (4.4.2) than the implicit backward scheme.

We decompose the approximation error into three terms:

$$\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(2)}(X_i)\big|^2 \le 3\Big(\mathbb{E}\big|Y_{t_i} - \bar{V}_i^{(2)}\big|^2 + \mathbb{E}\big|\bar{V}_i^{(2)} - V_i^{(2)}\big|^2 + \mathbb{E}\big|V_i^{(2)} - \widehat{\mathcal{U}}_i^{(2)}(X_i)\big|^2\Big). \quad (4.4.25)$$

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 $V_i^{(2)}$ in (4.3.4) and the martingale representation theorem, there exists a square integrable process $\{\widehat{Z}_s^{(2)}, t_i \le s \le t_{i+1}\}$ such that

$$\begin{aligned}&\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - f\big(t_i, X_i, \mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1})\big], \mathbb{E}_i\big[\sigma(t_i, X_i)^\intercal D_x \widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1})\big]\big)\Delta t_i \\ &= V_i + \int_{t_i}^{t_{i+1}} \widehat{Z}_s^{(2)}.\mathrm{d}W_s.\end{aligned}$$

It follows that the quadratic loss function of the DS scheme in (4.2.7) is written as

$$\begin{aligned}&J_i^S(\mathcal{U}_i) \\ &:= \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \mathcal{U}_i(X_i) - f\big(t_i, X_{i+1}, \widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}), \sigma(t_i, X_i)^\intercal D_x \widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1})\big)\Delta t_i\Big|^2 \\ &= \tilde{J}_i^S(\mathcal{U}_i) + \mathbb{E}\Big[\int_{t_i}^{t_{i+1}} |\widehat{Z}_s^{(2)}|_2^2 \mathrm{d}s\Big], \end{aligned} \quad (4.4.26)$$

where

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

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

$$\tilde{J}_i^S(\mathcal{U}_i) + |\Delta t_i|^2 \mathbb{E}\big|\Delta f_i\big|^2 \ge \frac{1}{2}\mathbb{E}\big|V_i^{(2)} - \mathcal{U}_i(X_i)\big|^2. \quad (4.4.27)$$

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

$$\begin{aligned}&\tilde{J}_i^S(\mathcal{U}_i) + |\Delta t_i|^2 \mathbb{E}\big|\Delta f_i\big|^2 \\ &\le 2\mathbb{E}\big|V_i^{(2)} - \mathcal{U}_i(X_i)\big|^2 + 3|\Delta t_i|^2 \mathbb{E}\big|\Delta f_i\big|^2 \\ &\le 2\mathbb{E}\big|V_i^{(2)} - \mathcal{U}_i(X_i)\big|^2 + 9|\Delta t_i|^2 [f]_L^2 \mathbb{E}|X_{i+1} - X_i|_2^2 \\ &\quad + 9|\Delta t_i|^2 [f]_L^2 \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \mathbb{E}_i[\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1})]\Big|^2 \\ &\quad + 9|\Delta t_i|^2 [f]_L^2 \mathbb{E}\Big|\sigma(t_i, X_i)^\intercal D_x \widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \mathbb{E}_i\big[\sigma(t_i, X_i)^\intercal D_x \widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1})\big]\Big|_2^2 \\ &\le 2\,\mathbb{E}\big|V_i^{(2)} - \mathcal{U}_i^{(2)}(X_i)\big|^2 + 9|\Delta t_i|^2 [f]_L^2 \mathbb{E}|X_{i+1} - X_i|_2^2 \\ &\quad + 9|\Delta t_i|^2 [f]_L^2 \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \widehat{\mathcal{U}}_{i+1}^{(2)}(X_i)\Big|^2 \\ &\quad + 9|\Delta t_i|^2 [f]_L^2 \mathbb{E}\Big[|\sigma(t_i, X_i)|_2^2 \mathbb{E}_i\big|D_x \widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - D_x \widehat{\mathcal{U}}_{i+1}^{(2)}(X_i)\big|_2^2\Big], \end{aligned} \quad (4.4.28)$$

where we use the definition of conditional expectation $\mathbb{E}_i[.]$, and the tower property of conditional expectation in the last inequality. Recall that $\widehat{\mathcal{U}}_{i+1} \in \mathcal{N}_i^{\gamma,\eta}$ is Lipschitz on $\mathbb{R}^d$. Actually, we have

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

By the Cauchy-Schwarz inequality, we then have

$$\begin{aligned}\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \widehat{\mathcal{U}}_{i+1}^{(2)}(X_i)\big|^2 &\leq C\gamma_i^2 \big\|X_{i+1} - X_i\big\|_4^2 \\ &\leq C\gamma_i^2 \Delta t_i\end{aligned}$$

for Δt_i small enough, $R \geq 1$, and we used again the standard estimate: $\|X_i\|_{2p} \leq C(1 + \|\mathcal{X}_0\|_{2p})$, $\|X_{i+1} - X_i\|_{2p} \leq C(1 + \|\mathcal{X}_0\|_{2p})\sqrt{\Delta t_i}$, for $p \geq 1$. By using also the Lipschitz condition on $D_x\widehat{\mathcal{U}}_{i+1}$, and plugging into (4.4.28), we get

$$\tilde{J}_i^S(\mathcal{U}_i) + |\Delta t_i|^2 \mathbb{E}\big|\Delta f_i\big|^2 \leq 2\mathbb{E}\big|V_i^{(2)} - \mathcal{U}_i(X_i)\big|^2 + C(d)\max\big[\gamma_i^2, \eta_i^2\big]\Big(1 + \big\|\mathcal{X}_0\big\|_4^2\Big)^2 |\Delta t_i|^3 \quad (4.4.29)$$

By applying inequality (4.4.27) to $\mathcal{U}_i = \widehat{\mathcal{U}}_i^{(2)}$, which is a minimizer of $\tilde{\mathcal{J}}_i^S$ by (4.4.26), and combining with (4.4.29), this yields for all functions $\mathcal{U}_i$ in $\mathcal{N}_i^{\gamma,\eta}$:

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

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

$$\mathbb{E}\big|V_i^{(2)} - \widehat{\mathcal{U}}_i^{(2)}(X_i)\big|^2 \leq C\Big(\varepsilon_i^{\gamma,\eta} + (1 + \|\mathcal{X}_0\|_4^2)^2 |\Delta t_i|^3 \max\big[\gamma_i^2, \eta_i^2\big]\Big). \quad (4.4.30)$$

Step 2. From the expressions of $V_i^{(2)}$, and $\bar{V}_i^{(2)}$ in (4.3.4) and (4.4.24), and by applying the Young, the Cauchy-Schwarz inequalities, we get with $\beta \in (0,1)$

$$\begin{aligned}&\mathbb{E}\big|\bar{V}_i^{(2)} - V_i^{(2)}\big|^2 \\ &\leq (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \bar{V}_{i+1}^{(2)}\big]\Big|^2 \\ &+ \big(1 + \frac{1}{\beta}\big)|\Delta t_i|^2 \mathbb{E}\Big|f\big(t_i, X_i, \bar{V}_{i+1}^{(2)}, \bar{Z}_i^{(2)}\big) \\ &\qquad - f\big(t_i, X_i, \mathbb{E}_i[\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1})], \mathbb{E}_i[\sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1})]\big)\Big|^2 \\ &\leq (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \bar{V}_{i+1}^{(2)}\big]\Big|^2 \\ &+ 2[f]_L^2\big(1 + \frac{1}{\beta}\big)|\Delta t_i|^2\Big(\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \bar{V}_{i+1}^{(2)}\big|^2 + \mathbb{E}\Big|\mathbb{E}_i\big[\sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1})\big] - \bar{Z}_i^{(2)}\Big|_2^2\Big) \quad (4.4.31)\end{aligned}$$

Now, recalling the expression of $\bar{Z}_i$ in (4.4.24), and by a standard integration by parts argument (see e.g. Lemma 2.1 in [FTW11]), we have

$$\begin{aligned}&\mathbb{E}_i\big[\sigma(t_i, X_i)^\intercal D_x\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1})\big] - \bar{Z}_i^{(2)} \\ &= \mathbb{E}_i\Big[\big(\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \bar{V}_{i+1}^{(2)}\big)\frac{\Delta W_i}{\Delta t_i}\Big] \\ &= \mathbb{E}_i\Big[\Big(\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \bar{V}_{i+1}^{(2)} - \mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \bar{V}_{i+1}^{(2)}\big]\Big)\frac{\Delta W_i}{\Delta t_i}\Big].\end{aligned}$$

By plugging into (4.4.31), we then obtain by the Cauchy-Schwarz inequality

$$\begin{aligned}&\mathbb{E}\big|\bar{V}_i^{(2)} - V_i^{(2)}\big|^2 \\ &\leq (1+\beta)\mathbb{E}\Big|\mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \bar{V}_{i+1}^{(2)}\big]\Big|^2 + 2[f]_L^2(1+\beta)\frac{|\Delta t_i|^2}{\beta}\Big\{\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \bar{V}_{i+1}^{(2)}\big|^2 \\ &+ \frac{d}{\Delta t_i}\Big[\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \bar{V}_{i+1}^{(2)}\big|^2 - \mathbb{E}\Big|\mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \bar{V}_{i+1}^{(2)}\big]\Big|^2\Big]\Big\} \\ &\leq (1 + C\Delta t_i)\mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{i+1}) - \bar{V}_{i+1}^{(2)}\big|^2, \quad (4.4.32)\end{aligned}$$

by choosing explicitly $\beta = 2d[f]_L^2 \Delta t_i$ for Δt_i small enough. By using again the Young inequality on the r.h.s. of (4.4.32), and since $\Delta t_i = O(1/N)$, we then get

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

By discrete Gronwall lemma, and recalling that $\bar{V}_N^{(2)} = g(X_N)$, $V_N^{(2)} = \widehat{\mathcal{U}}_N(X_N)$, we deduce with (4.4.30) that

$$\sup_{i\in[\![0,N]\!]} \mathbb{E}\big|\bar{V}_i^{(2)} - V_i^{(2)}\big|^2 \leq C\varepsilon_N^{\gamma,\eta} + CN\sum_{i=1}^{N-1}\Big(\varepsilon_i^{\gamma,\eta} + (1 + \|\mathcal{X}_0\|_4^2)^2|\Delta t_i|^3 \max\big[\gamma_i^2, \eta_i^2\big]\Big) \quad (4.4.33)$$

The required bound (4.3.5) for the approximation error on Y follows by plugging (4.4.2), (4.4.30) and (4.4.33) into (4.4.25). □

4.4.4 Proof of Proposition 4.3.2

For $x \in \mathbb{R}^d$, we define the processes $X_{j+1}^{j,x}$, $j = 0, \ldots, N$,

$$X_{j+1}^{j,x} := x + \mu(t_j, x)\Delta t_j + \sigma(t_j, x)\Delta W_j, \quad j = 0, \ldots, N-1.$$

Define also

$$\begin{cases} V_{i,3}^x &= \mathbb{E}_i\Big[\widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x}) - f\big(t_i, x, V_{i,3}^x, \overline{\widehat{Z}}_{i,3}^x\big)\Delta t_i\Big] = v_i^{(3)}(x) \\ \overline{\widehat{Z}}_{i,3}^x &= \mathbb{E}_i\Big[\widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x})\frac{\Delta W_i}{\Delta t_i}\Big] = \widehat{z}_i^{(3)}(x) \end{cases}$$

with $v_i^{(3)}, \widehat{z}_i^{(3)}$ as in (4.3.6) by Markov property.
Step 1. Let $x' \in \mathbb{R}^d$. By the Cauchy-Schwarz inequality, we have the standard estimate

$$\begin{aligned} &\Delta t_i \mathbb{E}\Big|\overline{\widehat{Z}}_{i,3}^x - \overline{\widehat{Z}}_{i,3}^{x'}\Big|_2^2 \\ &= \frac{1}{\Delta t_i}\mathbb{E}\Big|\mathbb{E}_i\Big[\{\widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x}) - \widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x'}) - \mathbb{E}_i\Big[\widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x}) - \widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x'})\Big]\}\Delta W_i\Big]\Big|^2 \\ &\leq d\,\Big(\mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x}) - \widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x'})\Big|^2 - \mathbb{E}\Big|\mathbb{E}_i\Big[\widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x}) - \widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x'})\Big]\Big|^2\Big). \end{aligned} \quad (4.4.34)$$

We then apply the Young inequality to see that

$$\begin{aligned} &\mathbb{E}\Big|V_{i,3}^x - V_{i,3}^{x'}\Big|^2 \\ &\leq (1 + \gamma\Delta t_i)\mathbb{E}\Big|\mathbb{E}_i\Big[\widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x}) - \widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x'})\Big]\Big|^2 \\ &+ (1 + \frac{1}{\gamma\Delta t_i})\Delta t_i^2\mathbb{E}\Big|\{f\big(t_i, x', V_{i,3}^{x'}, \overline{\widehat{Z}}_i^{3,x'}\big) - f\big(t_i, x, V_{i,3}^x, \overline{\widehat{Z}}_{i,3}^x\big)\}\Big|^2 \\ &\leq (1 + \gamma\Delta t_i)\mathbb{E}\Big|\mathbb{E}_i\Big[\widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x}) - \widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x'})\Big]\Big|^2 \\ &+ 3[f]^2(1 + \frac{1}{\gamma\Delta t_i})\Delta t_i^2\mathbb{E}\{|x - x'|_2^2 + |V_{i,3}^x - V_{i,3}^{x'}|^2 + |\overline{\widehat{Z}}_{i,3}^x - \overline{\widehat{Z}}_{i,3}^{x'}|_2^2\}. \end{aligned}$$

Hence for $\gamma = 3[f]^2 d$ and Δt_i small enough, using (4.4.34) we obtain

$$\begin{aligned} \mathbb{E}\Big|V_{i,3}^x - V_{i,3}^{x'}\Big|^2 \leq (1 + (\gamma + 3d)\Delta t_i)\mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x}) - \widehat{\mathcal{U}}_{i+1}^{(3)}(X_{i+1}^{i,x'})\Big|^2 \\ + (1 + (\gamma + 3d)\Delta t_i)\Delta t_i\mathbb{E}|x - x'|_2^2. \end{aligned}$$

Therefore, with Lemma 4.3.1

$$\begin{aligned}|v_{N-1}^{(3)}(x)-v_{N-1}^{(3)}(x')|^2 &= \mathbb{E}\Big|V_{N-1,3}^{x}-V_{N-1,3}^{x'}\Big|^2\\ &\le (1+(\gamma+3d)\Delta t_{N-1})((1+C\Delta t_{N-1})[g]^2+\Delta t_i)|x-x'|_2^2\\ &\le (1+\hat{C}\Delta t_{N-1})([g]^2+\Delta t_i)|x-x'|_2^2,\end{aligned}$$

for some constant $\hat{C}$. Similarly, assuming $\widehat{\mathcal{U}}_{i+1}^{(3)}$ is $[\widehat{\mathcal{U}}_{i+1}^{(3)}]-$Lipschitz, $v_i^{(3)}$ is Lipschitz with constant $[v_i^{(3)}]$ verifying

$$[v_i^{(3)}]^2 \le (1+\hat{C}\Delta t_i)([\widehat{\mathcal{U}}_{i+1}^{(3)}]^2+\Delta t_i).$$

Step 2. Let $\epsilon>0,\ \kappa\in\mathbb{N},\ \ell\in\mathbb{N},\ m\in\mathbb{R}^\ell$ to be chosen after. Recursively, we approximate $v_i^{(3)}$ by a $[v_i^{(3)}]$-Lipschitz GroupSort neural network $\mathcal{U}_i^{(3)}$ in $\mathcal{N}_i=\mathcal{G}^{\zeta_\kappa}_{[v_i^{(3)}],d,1,\ell,m}$ with uniform error $2[v_i]R\epsilon$ on $[-R,R]^d$ (Proposition 4.2.1). Then by discrete Gronwall inequality

$$[\mathcal{U}_i^{(3)}]^2\le K,\ [v_i^{(3)}]^2\le K,$$

uniformly in i,N for some constant K. Thus $v_i^{(3)},\mathcal{U}_i^{(3)}$ are K Lipschitz, uniformly. Then we approximate by (4.4.34) $\widehat{z}_i^{(3)}$ by a $\sqrt{\frac{d}{\Delta t_i}}[v_i^{(3)}]$-Lipschitz GroupSort neural network $\mathcal{Z}_i$ in $\mathcal{N}_i'=\mathcal{G}^{\zeta_\kappa}_{\sqrt{\frac{d}{\Delta t_i}}[v_i^{(3)}],d,d,\ell,m}$ with uniform error $2\frac{d}{\sqrt{\Delta t_i}}[v_i^{(3)}]R\varepsilon$ on $[-R,R]^d$ thanks to Proposition 4.2.1. Thus $\sqrt{\Delta t_i}\widehat{z}_i^{(3)},\sqrt{\Delta t_i}\mathcal{Z}_i^{(3)}$ are dK Lipschitz, uniformly.

Step 3. The regression errors $\varepsilon_i^{3,y}$ verify from, localization of X_i on $B_2(R)$, the Hölder inequality, and the Markov inequality, the approximation error of $v_i^{(3)}$, $i\in[\![0,N-1]\!]$, by the class of GroupSort neural networks (Proposition 4.2.1)

$$\begin{aligned}\sqrt{\varepsilon_i^{3,y}} &= \inf_{\mathcal{U}\in\mathcal{G}_{[v_k],d,1}}\big\|v_i^{(3)}(X_i)-\mathcal{U}(X_i)\big\|_2\\ &\le \inf_{\mathcal{U}\in\mathcal{G}_{[v_k],d,1}}\Big\|\big(v_i^{(3)}(X_i)-\mathcal{U}(X_i)\big)\mathbf{1}_{X_i\in B_2(R)}\Big\|_2+\Big\|\big(v_i^{(3)}(X_i)-\widehat{\mathcal{U}}_i^{(3)}(X_i)\big)\mathbf{1}_{|X_i|_2\ge R}\Big\|_2\\ &\le 2KR\epsilon+\mathbb{E}\Big|\big(v_i^{(3)}(X_i)-\widehat{\mathcal{U}}_i^{(3)}(X_i)\big)^{2q}\Big|^{1/2q}\mathbb{E}\Big|\mathbf{1}_{|X_i|_2\ge R}^{\frac{2q}{2q-1}}\Big|^{\frac{2q-1}{2q}}\\ &= 2KR\epsilon+\mathbb{E}\Big|\big(v_i^{(3)}(X_i)-\widehat{\mathcal{U}}_i^{(3)}(X_i)\big)^{2q}\Big|^{1/2q}\mathbb{E}[\mathbf{1}_{|X_i|_2\ge R}]^{\frac{2q-1}{2q}}\\ &\le 2KR\epsilon+\frac{\big(\big\|v_i^{(3)}(X_i)-v_i^{(3)}(0)\big\|_{2q}+\big\|\widehat{\mathcal{U}}_i^{(3)}(X_i)-v_i^{(3)}(0)\big\|_{2q}\big)\big\|X_i\big\|_{\frac{2q}{2q-1}}^{\frac{2q}{2q-1}}}{R},\end{aligned}\tag{4.4.35}$$

by noticing that $\big(v_i^{(3)}(X_i)-\widehat{\mathcal{U}}_i^{(3)}(X_i)\big)=\big(v_i^{(3)}(X_i)-v_i^{(3)}(0)-(\widehat{\mathcal{U}}_i^{(3)}(X_i)-v_i^{(3)}(0))\big)$ for $q>0$ and $2q=2+\delta$ with δ as in the statement of the Proposition. Now, by Lipschitz continuity of $v_i^{(3)},\widehat{\mathcal{U}}^{(3)}$ and because $0\in B_2(R)$ we have

$$\begin{aligned}&\big\|\widehat{\mathcal{U}}_i^{(3)}(X_i)-v_i^{(3)}(0)\big\|_{2q}+\big\|v_i^{(3)}(X_i)-v_i^{(3)}(0)\big\|_{2q}\\ &\le\big\|\widehat{\mathcal{U}}_i^{(3)}(0)-v_i^{(3)}(0)\big\|_{2q}+\big\|\widehat{\mathcal{U}}_i^{(3)}(X_i)-\widehat{\mathcal{U}}_i^{(3)}(0)\big\|_{2q}+\big\|v_i^{(3)}(X_i)-v_i^{(3)}(0)\big\|_{2q}\\ &\le 2KR\epsilon+2K\big\|X_i\big\|_{2q}.\end{aligned}\tag{4.4.36}$$

Recalling the standard estimate $\|X_i\|_{2q}\le C(1+\|\mathcal{X}_0\|_{2q})$, $i=0,\dots,N$, we then have

$$\varepsilon_i^{3,y}\le\ C\Big\{R^2\epsilon^2+\frac{1+R^2\epsilon^2}{R^2}\Big\},$$

for some constant $C(d, \mathcal{X}_0)$ independent of N, R, ϵ. Similarly repeating (4.4.35) and (4.4.36) by replacing respectively $\widehat{\mathcal{U}}_i^{(3)}$ by $\widehat{\mathcal{Z}}_i^{(3)}$ and $v_i^{(3)}$ by $\hat{z}_i^{(3)}$ and recalling that $\sqrt{\Delta t_i}\widehat{z}_i^{(3)}, \sqrt{\Delta t_i}\mathcal{Z}_i^{(3)}$ are dK Lipschitz uniformly w.r.t. N, we obtain

$$\Delta t_i \varepsilon_i^{3,z} \leq C\Big\{R^2\epsilon^2 + \frac{1+R^2\epsilon^2}{R^2}\Big\}.$$

Then to obtain a convergence rate of $O(1/N)$ in (4.3.7), it suffices to choose R, ε such that

$$N^2R^2\epsilon^2 = O(1/N), \quad N^2\frac{1+R^2\epsilon^2}{R^2} = O(1/N),$$

which is verified with $R = O(N^{3/2})$, $\epsilon = O(\frac{1}{N^3})$. Then by Proposition 4.2.1, if $d > 1$ we can choose the previously GroupSort neural networks with grouping size $\kappa = O(\lceil 2\sqrt{d}N^3\rceil)$, depth $\ell + 1 = O(d^2)$ and width $\sum_{i=0}^{\ell-1} m_i = O((2\sqrt{d}N^3)^{d^2-1})$. If $d = 1$, we can take $\kappa = O(N^3)$, depth $\ell + 1 = 3$ and width $\sum_{i=0}^{\ell-1} m_i = O(N^3)$.

4.5 Numerical Tests

We test our different algorithms and the cited ones in this paper on some examples and by varying the state space dimension. 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. Each numerical experiment is conducted using a node composed of 2 Intel® Xeon® Gold 5122 Processors, 192 Gb of RAM, and 2 GPU nVidia® Tesla® V100 16Gb. We use a batch size of 1000. We do not implement the GroupSort network because even if it is useful for theoretical analysis, it would be costly to use in practice: on the one hand, it will induce a cost of order $O(n \ln n)$ where n is the batch size, compared to a linear cost $O(n)$ for standard activation function; on the other hand, it requires to track the Lipschitz constant of the functions and adapt the networks architecture accordingly. Whereas theoretical results suggest to take deep neural networks with depth increasing with the dimension, we observe that two hidden layers are enough to obtain a good accuracy. According to our experience tanh activation function provides the best results. ReLU or Elu being not bounded, some explosion tends to appear when the learning rates are not small enough.

We consider examples from [HPW20] to compare its DBDP scheme with the DS and MDBDP schemes.

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 $N = 120$. More numerical examples and tests are presented in the extended version [GPW22a] of this paper, and the codes at: `https://github.com/MaxGermain/MultistepBSDE`.

4.5.1 PDE with Bounded Solution and Simple Structure

We take the parameters: $\mu = \frac{0.2}{d}$, $\sigma = \frac{I_d}{\sqrt{d}}$, terminal condition $g(x) = \cos(\overline{x})$, with $\overline{x} = \sum_{i=1}^d x_i$, and generator

$$\begin{aligned} & f(x,y,z) \\ = \; & -\Big(\cos(\overline{x}) + 0.2\sin(\overline{x})\Big)e^{\frac{T-t}{2}} + \frac{1}{2}(\sin(\overline{x})\cos(\overline{x})e^{T-t})^2 - \frac{1}{2d}(y(1_d \cdot z))^2. \end{aligned}$$

so that the PDE solution is given by $u(t,x) = \cos(\overline{x})\exp\big(\frac{T-t}{2}\big)$.

We fix $T = 1$, and increase the dimension d. The results are reported in Table 4.1 for $d = 10$, in Table 4.2 for $d = 20$, and in Table 4.3 for $d = 50$. It is observed that all the schemes

	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**

Table 4.1: Estimate of $u(0, x_0)$ in the case (4.5.1), 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**

Table 4.2: Estimate of $u(0, x_0)$ in the case (4.5.1), 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.

DBDP, DBSDE and MDBDP provide quite accurate results with smallest standard deviation for MDBDP, and largely outperforms the DS scheme.

4.5.2 PDE with Unbounded Solution and more Complex Structure

We consider a toy example with solution given by

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

Therefore we take the parameters

$$\mu = 0,\ \sigma = \frac{I_d}{\sqrt{d}},\ \ T = 1, \quad f(t,x,y,z) = k(t,x) - \frac{y}{\sqrt{d}}(1_d \cdot z) - \frac{y^2}{2} \tag{4.5.2}$$

with $k(t,x) = \partial_t u + \frac{1}{2d}\operatorname{Tr}(D_x^2 u) + \frac{u}{\sqrt{d}}\sum_i D_{x_i} u + \frac{u^2}{2}$.

We start with tests in dimension $d = 1$. The results are reported in Table 4.4.

We next increase the dimension to $d = 8$, and report the results in the following figure. 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 (above dimension $d = 10$, all the schemes do not give good approximation results).

	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**

Table 4.3: Estimate of $u(0, x_0)$ in the case (4.5.1), 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.

	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**

Table 4.4: Estimate of $u(0, x_0)$ in the case (4.5.2), where $d = 1, x_0 = 0.5$, $T = 1$ with 120 time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 1.3776.

	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**

Table 4.5: Estimate of $u(0, x_0)$ in the case (4.5.2), where $d = 8, x_0 = 0.5\ \mathbb{1}_8$, $T = 1$ with 120 time steps. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 1.1603.

Chapter 5

Neural networks-based backward scheme for fully nonlinear PDEs

This chapter is based on the paper [PWG21]
H. Pham, X. Warin, and M. Germain. "Neural networks-based backward scheme for fully nonlinear PDEs." In: *SN Partial Differential Equations and Applications* 2, 16 (27 January 2021).

The objective of this Chapter is to design a new deep learning scheme for solving fully nonlinear PDEs. The existing scheme in the literature is the Deep 2BSDE method of [BEJ19], which is a modified version of the Deep BSDE scheme [HJE18] able to treat full nonlinearity. Similarly, thanks to ideas from the Deep Splitting scheme [Bec+21] we provide an adaptation of the DBDP scheme of [HPW20] to the more challenging case of non-linearity in the second order derivative of the solution. Our study is only focused on numerical aspects and the theoretical study of the algorithm is left to future research.

Abstract

We propose a numerical method for solving high dimensional fully nonlinear partial differential equations (PDEs). Our algorithm estimates simultaneously by backward time induction the solution and its gradient by multi-layer neural networks, while the Hessian is approximated by automatic differentiation of the gradient at previous optimization step.

This methodology extends to the fully nonlinear case the approach recently proposed in [HPW20] for semi-linear PDEs. Numerical tests illustrate the performance and accuracy of our method on several examples in high dimension with non-linearity on the Hessian term including a linear quadratic control problem with control on the diffusion coefficient, Monge-Ampère equation and Hamilton-Jacobi-Bellman equation in portfolio optimization.

5.1 Introduction

This paper is devoted to the resolution in high dimension of fully nonlinear parabolic partial differential equations (PDEs) of the form

$$\begin{cases} \partial_t u + f(.,.,u, D_x u, D_x^2 u) = 0, & \text{on } [0,T) \times \mathbb{R}^d, \\ u(T,.) = g, & \text{on } \mathbb{R}^d, \end{cases} \tag{5.1.1}$$

with a non-linearity in the solution, its gradient $D_x u$ and its hessian $D_x^2 u$ via the function $f(t,x,y,z,\gamma)$ defined on $[0,T]\times\mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \times \mathbb{S}^d$ (where $\mathbb{S}^d$ is the set of symmetric $d\times d$ matrices), and a terminal condition g.

The numerical resolution of this class of PDEs is far more difficult than the one of classical semi-linear PDEs where the nonlinear function f does not depend on γ. In fact, rather few methods are available to solve fully nonlinear equations even in moderate dimension.

- First based on the work of [Che+07], an effective scheme developed in [FTW11] using some regression techniques has been shown to be convergent under some ellipticity conditions later removed by [Tan13]. 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 [War18a]. It seems to be effective in very high dimension for maturities T not too long and linearities not too important.

- A numerical algorithm to solve fully nonlinear equations 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. This extends the idea introduced in the pioneering papers [EHJ17; HJE18], which were the first serious works for using machine learning methods to solve high dimensional PDEs.

- The Deep Galerkin method proposed in [SS18] based on some machine learning techniques and using some automatic differentiation of the solution seems to be effective on some cases. It has been tested in [AA+18] for example on the Merton problem.

In this article, we introduce a numerical method based on machine learning techniques and backward in time iterations, which extends the proposed schemes in [VSS18] for linear problems, and in the recent work [HPW20] for semi-linear PDEs. The approach in these works consists in estimating simultaneously the solution and its gradient by multi-layer neural networks by minimizing a sequence of loss functions defined in backward induction. A basic idea to extend this method to the fully nonlinear case would rely on the representation proposed in [Che+07]: at each time step t_n of an Euler scheme, the Hessian $D_x^2 u$ at t_n is approximated by a neural

network minimizing some local L_2 criterion associated to a BSDE involving $D_x u$ at date t_{n+1} and $D_x^2 u$. Then, the pair $(u, D_x u)$ at date t_n is approximated/learned with a second minimization similarly as in the method described by [HPW20]. The first minimization can be implemented with different variations but numerical results show that the global scheme does not scale well with the dimension. Instability on the $D_x^2 u$ calculation rapidly propagates during the backward resolution. Besides, the methodology appears to be costly when using two optimizations at each time step. An alternative approach that we develop here, is to combine the ideas of [HPW20] and the splitting method in [Bec+21] in order to derive a new deep learning scheme that requires only one local optimization during the backward resolution for learning the pair $(u, D_x u)$ and approximating $D_x^2 u$ by automatic differentiation of the gradient computed at the previous step.

The outline of the paper is organized as follows. In Section 5.2, we briefly recall the mathematical description of the classical feedforward approximation, and then derive the proposed neural networks-based backward scheme. We test our method in Section 5.3 on various examples. First we illustrate our results with a PDE involving a non-linearity of type $uD_x^2 u$. Then, we consider a stochastic linear quadratic problem with controlled volatility where an analytic solution is available, and we test the performance and accuracy of our algorithm up to dimension 20. Next, we apply our algorithm to a Monge-Ampère equation, and finally, we provide numerical tests for the solution to fully nonlinear Hamilton-Jacobi-Bellman equation, with non-linearities of the form $|D_x u|^2 / D_x^2 u$, arising in portfolio selection problem with stochastic volatilities.

5.2 The proposed deep backward scheme

Our aim is to numerically approximate the function $u : [0, T] \times \mathbb{R}^d \mapsto \mathbb{R}$, assumed to be the unique smooth solution to the fully nonlinear PDE (5.1.1) under suitable conditions. This will be achieved by means of neural networks approximations for u and its gradient $D_x u$, relying on a backward scheme and training simulated data of some forward diffusion process. Approximations of PDE in high dimension by neural networks have now become quite popular, and are supported theoretically by recent results in [Hut+18] and [DLM20] showing their efficiency to overcome the curse of dimensionality.

5.2.1 Feedforward neural network to approximate functions

We denote by d_0 the dimension of the input variables, and d_1 the dimension of the output variable. A (deep) neural network is characterized by a number of layers $L + 1 \in \mathbb{N} \setminus \{1, 2\}$ with m_ℓ, $\ell = 0, \ldots, L$, the number of neurons (units or nodes) on each layer: the first layer is the input layer with $m_0 = d$, the last layer is the output layer with $m_L = d_1$, and the $L - 1$ layers between are called hidden layers, where we choose for simplicity the same dimension $m_\ell = m$, $\ell = 1, \ldots, L - 1$.

A feedforward neural network is a function from $\mathbb{R}^{d_0}$ to $\mathbb{R}^{d_1}$ defined as the composition

$$x \in \mathbb{R}^d \longmapsto A_L \circ \varrho \circ A_{L-1} \circ \ldots \circ \varrho \circ A_1(x) \in \mathbb{R}. \tag{5.2.1}$$

Here A_ℓ, $\ell = 1, \ldots, L$ are affine transformations: A_1 maps from $\mathbb{R}^{d_0}$ to $\mathbb{R}^m$, $A_2, \ldots, A_{L-1}$ map from $\mathbb{R}^m$ to $\mathbb{R}^m$, and A_L maps from $\mathbb{R}^m$ to $\mathbb{R}^{d_1}$, represented by

$$A_\ell(x) = \mathcal{W}_\ell x + \beta_\ell,$$

for a matrix $\mathcal{W}_\ell$ called weight, and a vector β_ℓ called bias term, $\varrho : \mathbb{R} \to \mathbb{R}$ is a nonlinear function, called activation function, and applied component-wise on the outputs of A_ℓ, i.e., $\varrho(x_1, \ldots, x_m) = (\varrho(x_1), \ldots, \varrho(x_m))$. Standard examples of activation functions are the sigmoid, the ReLu, the Elu, tanh.

All these matrices $\mathcal{W}_\ell$ and vectors β_ℓ, $\ell = 1, \ldots, L$, are the parameters of the neural network, and can be identified with an element $\theta \in \mathbb{R}^{N_m}$, where $N_m = \sum_{\ell=0}^{L-1} m_\ell(1 + m_{\ell+1}) = d_0(1 + m) + m(1 + m)(L - 2) + m(1 + d_1)$ is the number of parameters. We denote by $\mathcal{N}_{d_0,d_1,L,m}$ the set of all functions generated by (5.2.1) for $\theta \in \mathbb{R}^{N_m}$.

5.2.2 Forward-backward representation

Let us introduce a forward diffusion process

$$X_t = X_0 + \int_0^t \mu(s, X_s)ds + \int_0^t \sigma(s, X_s)dW_s, \quad 0 \le t \le T, \tag{5.2.2}$$

where μ is a function defined on $[0,T] \times \mathbb{R}^d$ with values in $\mathbb{R}^d$, σ is a function defined on $[0,T] \times \mathbb{R}^d$ with values in $\mathbb{M}^d$ the set of $d \times d$ matrices, and W a d-dimensional Brownian motion on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$ equipped with a filtration $\mathbb{F} = (\mathcal{F}_t)_{0 \le t \le T}$ satisfying the usual conditions. The process X will be used for the simulation of training data in our deep learning algorithm, and we shall discuss later the choice of the drift and diffusion coefficients μ and σ, see Remark 3.

Let us next denote 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

$$Y_t \;=\; u(t, X_t), \quad Z_t \;=\; D_x u(t, X_t), \quad \Gamma_t \;=\; D_x^2 u(t, X_t), \quad 0 \le t \le T. \tag{5.2.3}$$

By Itô's formula applied to $u(t, X_t)$, and since u is solution to (5.1.1), we see that (Y, Z, Γ) satisfies the backward equation:

$$\begin{aligned} Y_t = \; g(X_T) &- \int_t^T \big[\mu(s, X_s).Z_s + \frac{1}{2}\mathrm{tr}(\sigma\sigma^\intercal(s, X_s)\Gamma_s) - f(s, X_s, Y_s, Z_s, \Gamma_s)\big]ds \\ &- \int_t^T \sigma^\intercal(s, X_s)Z_s.dW_s, \qquad 0 \le t \le T. \end{aligned} \tag{5.2.4}$$

Remark 1. This BSDE does not uniquely characterize a triple (Y, Z, Γ) contrarily to the semilinear case (without a non-linearity with respect to Γ) in which proper assumptions on the equation coefficients provide existence and uniqueness for a solution couple (Y, Z). In the present case at least two options can be used to estimate the Γ component:

- Rely on the 2BSDE representation from [Che+07] which extends the probabilistic representation of [PP90] for semilinear equations to the fully nonlinear case. It is the approach used by [BEJ19] with a global large minimization problem, as in [HJE18].
- Compute the second order derivative by automatic differentiation. This is the point of view we adopt in this paper together with a local approach solving several small optimization problems. In this way, we provide an extension of [HPW20] to cover a broader range of nonlinear PDEs.

□

5.2.3 Algorithm

We now provide a numerical approximation of the forward backward system (5.2.2)-(5.2.4), and consequently of the solution u (as well as its gradient $D_x u$) to the PDE (5.1.1).

We start from a time grid $\pi = \{t_i, i = 0, \dots, N\}$ of $[0,T]$, with $t_0 = 0 < t_1 < \dots < t_N = T$, and time steps $\Delta t_i := t_{i+1} - t_i$, $i = 0, \dots, N-1$. The time discretization of the forward process X on π is then equal (typically when μ and σ are constants) or approximated by an Euler scheme:

$$X_{t_{i+1}} = \; X_{t_i} + \mu(t_i, X_{t_i})\Delta t_i + \sigma(t_i, X_{t_i})\Delta W_{t_i}, \qquad i = 0, \dots, N-1,$$

where we set $\Delta W_{t_i} := W_{t_{i+1}} - W_{t_i}$ (by misuse of notation, we keep the same notation X for the continuous time diffusion process and its Euler scheme). The backward SDE (5.2.4) is approximated by the time discretized scheme

$$Y_{t_i} \simeq \; Y_{t_{i+1}} - \big[\mu(t_i, X_{t_i}).Z_{t_i} + \frac{1}{2}\mathrm{tr}\big(\sigma\sigma^\intercal(t_i, X_{t_i}\Gamma_{t_i}\big) - f(t_i, X_{t_i}, Y_{t_i}, Z_{t_i}, \Gamma_{t_i})\big]\Delta t_i - \sigma^\intercal(t_i, X_{t_i})Z_{t_i}.\Delta W_{t_i},$$

that is written in forward form as

$$Y_{t_{i+1}} \simeq F(t_i, X_{t_i}, Y_{t_i}, Z_{t_i}, \Gamma_{t_i}, \Delta t_i, \Delta W_{t_i}), \quad i = 0, \dots, N-1, \tag{5.2.5}$$

with

$$\begin{aligned} F(t,x,y,z,\gamma,h,\Delta) &:= y - \tilde{f}(t,x,y,z,\gamma)h + z^\intercal \sigma(t,x)\Delta, \\ \tilde{f}(t,x,y,z,\gamma) &:= f(t,x,y,z,\gamma) - \mu(t,x).z - \frac{1}{2}\mathrm{tr}\big(\sigma\sigma^\intercal(t,x)\gamma\big). \end{aligned} \tag{5.2.6}$$

The idea of the proposed scheme is the following. Similarly as in [HPW20], we approximate at each time t_i, $u(t_i,.)$ and its gradient $D_x u(t_i,.)$, by neural networks $x \in \mathbb{R}^d \mapsto (\mathcal{U}_i(x;\theta), \mathcal{Z}_i(x;\theta))$ with parameter θ that are learned optimally by backward induction: suppose that $\hat{\mathcal{U}}_{i+1} := \mathcal{U}_{i+1}(.;\theta^*_{i+1})$, $\hat{\mathcal{Z}}_{i+1} := \mathcal{Z}_{i+1}(.;\theta^*_{i+1})$ is an approximation of $u(t_{i+1},.)$ and $D_x u(t_{i+1},.)$ at time t_{i+1}, then θ^*_i is computed from the minimization of the quadratic loss function:

$$\hat{L}_i(\theta) = \mathbb{E}\Big|\hat{\mathcal{U}}_{i+1} - F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta), \mathcal{Z}_i(X_{t_i};\theta), D\hat{\mathcal{Z}}_{i+1}(\mathcal{T}(X_{t_{i+1}})), \Delta t_i, \Delta W_{t_i})\Big|^2$$

where $\mathcal{T}$ is a truncation operator such that $\mathcal{T}(X)$ is bounded for example by a quantile of the diffusion process and $D\hat{\mathcal{Z}}_{i+1}$ stands for the automatic differentiation of $\hat{\mathcal{Z}}_{i+1}$. The idea behind the truncation is the following. During one step resolution, the estimation of the gradient is less accurate at the edge of the explored domain where samples are rarely generated. Differentiating the gradient gives a very oscillating Hessian at the edge of the domain. At the following time step resolution, these oscillations propagate to the gradient and the solution even if the domain where the oscillations occur is rarely attained. In order to avoid these oscillations, a truncation is achieved, permits to avoid that the oscillations of the neural network fit in zone where the simulations propagate scarcely to areas of importance. This truncation may be necessary to get convergence on some rather difficult cases. Of course this truncation is only valid if the real Hessian does not varies too much.

The intuition for the relevance of this scheme to the approximation of the PDE (5.1.1) is the following. From (5.2.3) and (5.2.5), the solution u to (5.1.1) should approximately satisfy

$$u(t_{i+1}, X_{t_{i+1}}) \simeq F(t_i, X_{t_i}, u(t_i, X_{t_i}), D_x u(t_i, X_{t_i}), D_x^2 u(t_i, X_{t_i}), \Delta t_i, \Delta W_{t_i}).$$

Suppose that at time t_{i+1}, $\hat{\mathcal{U}}_{i+1}$ is an estimation of $u(t_{i+1},.)$. Recalling the expression of F in (5.2.6), the quadratic loss function at time t_i is then approximately equal to

$$\begin{aligned} \hat{L}_i(\theta) \simeq \mathbb{E}\Big|& u(t_i, X_{t_i}) - \mathcal{U}_i(X_{t_i};\theta) + \big(D_x u(t_i, X_{t_i}) - \mathcal{Z}_i(X_{t_i};\theta)\big)^\intercal \sigma(t_i, X_{t_i})\Delta W_{t_i} \\ &- \Delta t_i \big[\tilde{f}(t_i, X_{t_i}, u(t_i, X_{t_i}), D_x u(t_i, X_{t_i}), D_x^2 u(t_i, X_{t_i})) - \tilde{f}(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta), \mathcal{Z}_i(X_{t_i};\theta), D\hat{\mathcal{Z}}_{i+1}(\mathcal{T}(X_{t_{i+1}}))\big)\Big| \end{aligned}$$

By assuming that $\tilde{f}$ has small non-linearities in its arguments (y,z,γ), say Lipschitz, possibly with a suitable choice of μ, σ, the loss function is thus approximately equal to

$$\hat{L}_i(\theta) \simeq (1 + O(\Delta t_i))\mathbb{E}\big|u(t_i, X_{t_i}) - \mathcal{U}_i(X_{t_i};\theta)\big|^2 + O(\Delta t_i)\mathbb{E}\big|D_x u(t_i, X_{t_i}) - \mathcal{Z}_i(X_{t_i};\theta)\big|^2 + O(|\Delta t_i|^2).$$

Therefore, by minimizing over θ this quadratic loss function, via stochastic gradient descent (SGD) based on simulations of $(X_{t_i}, X_{t_{i+1}}, \Delta W_{t_i})$ (called training data in the machine learning language), one expects the neural networks $\mathcal{U}_i$ and $\mathcal{Z}_i$ to learn/approximate better and better the functions $u(t_i,.)$ and $D_x u(t_i,)$ in view of the universal approximation theorem for neural networks. The rigorous convergence of this algorithm is postponed to a future work.

To sum up, the global algorithm is given in Algo 7 in the case where g is Lipschitz and the derivative can be analytically calculated almost everywhere. If the derivative of g is not available, it can be calculated by automatic differentiation of the neural network approximation of g.

Algorithm 7: Second order DBDP (2DBDP) from [PWG21]

Use a single deep neural network $(\mathcal{U}_N(.;\theta), \mathcal{Z}_N(.;\theta)) \in \mathcal{N}_{d,1+d,L,m}$ and minimize (by SGD)

$$\begin{cases} \hat{L}_N(\theta) := \mathbb{E}\Big|\mathcal{U}_N(X_{t_N};\theta) - g(X_{t_N})\Big|^2 + \dfrac{\Delta t_{N-1}}{d}\mathbb{E}\Big|\mathcal{Z}_N(X_{t_N};\theta) - Dg(X_{t_N})\Big|^2 \\ \quad \theta_N^* \in \arg\min_{\theta\in\mathbb{R}^{N_m}} \hat{L}_N(\theta). \end{cases}$$

$\widehat{\mathcal{U}}_N = \mathcal{U}_N(.;\theta_N^*)$, and set $\widehat{\mathcal{Z}}_N = \mathcal{Z}_N(.;\theta_N^*)$

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

Use a single deep neural network $(\mathcal{U}_i(.;\theta), \mathcal{Z}_i(.;\theta)) \in \mathcal{N}_{d,1+d,L,m}$ for the approximation of $(u(t_i,.), D_x u(t_i,.))$, and compute (by SGD) the minimizer of the expected quadratic loss function

$$\begin{cases} \hat{L}_i(\theta) \\ \quad := \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) - F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta), \mathcal{Z}_i(X_{t_i};\theta), D\hat{\mathcal{Z}}_{i+1}(\mathcal{T}(X_{t_{i+1}})), \Delta t_i, \Delta W_{t_i})\Big|^2 \\ \theta_i^* \in \arg\min_{\theta\in\mathbb{R}^{N_m}} \hat{L}_i(\theta). \end{cases} \tag{5.2.7}$$

Update: $\widehat{\mathcal{U}}_i = \mathcal{U}_i(.;\theta_i^*)$, and set $\widehat{\mathcal{Z}}_i = \mathcal{Z}_i(.;\theta_i^*)$.

end

Remark 2. Several alternatives can be implemented for the computation of the second order derivative. A natural candidate would consist in choosing to approximate the solution u at time t_i by a neural network $\mathcal{U}_i$ and estimate Γ_i as the iterated automatic differentiation $D_x^2\mathcal{U}_i$. However, it is shown in [HPW20] that choosing only a single neural network for u and using its automatic derivative to estimate the Z component degrades the error in comparison to the choice of two neural networks $\mathcal{U}, \mathcal{Z}$. A similar behavior has been observed during our tests for this second order case and the most efficient choice was to compute the derivative of the $\mathcal{Z}$ network. This derivative can also be estimated at the current time step t_i instead of t_{i+1}. However this method leads to an additional cost for the neural networks training by complicating the computation of the automatic gradients performed by Tensorflow during the backpropagation. It also leads numerically to worse results on the control estimation, as empirically observed in Table 5.5 and described in the related paragraph "Comparison with an implicit version of the scheme". For this reason, we decided to apply a splitting method and evaluate the Hessian at time t_{i+1}. For this reason, we decided to apply a splitting method and evaluate the Hessian at time t_{i+1}. □

Remark 3. The diffusion process X is used for the training simulations in the stochastic gradient descent method for finding the minimizer of the quadratic loss function in (5.2.7), where the expectation is replaced by empirical average for numerical implementation. The choice of the drift and diffusion parameters are explained in Section 5.3.1. □

5.3 Numerical results

We first construct an example with different non-linearities in the Hessian term and the solution. We graphically show that the solution is very well calculated in dimension $d = 1$ and then move to higher dimensions. We then use an example derived from a stochastic optimization problem with an analytic solution and show that we are able to accurately calculate the solution. Next, we consider the numerical resolution of the Monge-Ampère equation, and finally, give some tests for a fully nonlinear Hamilton-Jacobi-Bellman equation arising from portfolio optimization with

stochastic volatilities.

5.3.1 Choice of the algorithm hyperparameters

We describe in this paragraph how we choose the various hyperparameters of the algorithm and explain the learning strategy.

• PARAMETERS OF THE TRAINING SIMULATIONS: the choice of the drift coefficient is typically related to the underlying probabilistic problem associated to the PDE (for example a stochastic control problem), and should drive the training process to regions of interest, e.g.., that are visited with large probability by the optimal state process in stochastic control. In practice, we can take a drift function $\mu(.)$ equal to the drift associated to some a priori control. This choice of control could be an optimal control for a related problem for which we know the solution, or could be the control obtained by the first iteration of the algorithm. The choice of the diffusion coefficient σ is also important: large σ induces a better exploration of the state space, but as we will see in most of examples below, it gives a scheme slowly converging to the solution with respect to the time discretization and it generates a higher variance on the results. Moreover, for the applications in stochastic control, we might explore some region that are visited with very small probabilities by the optimal state process, hence representing few interest. On the other hand, small σ means a weak exploration, and we might lack information and precision on some region of the state space: the solution calculated at each time step is far more sensitive to very local errors induced by the neural network approximation and tends to generate a bias. Therefore a trade off has to be found between rather high variance with slow convergence in time and fast convergence in time with a potential bias. We also refer to [NR20] for a discussion on the role of the diffusion coefficient.

In practice and for the numerical examples in the next section, we test the scheme for different σ and by varying the number of time steps, and if it converges to the same solution, one can consider that we have obtained the correct solution. We also show the impact of the choice of the diffusion coefficient σ.

• PARAMETERS OF TRUNCATION: Given the training simulations X, we choose a truncation operator $\mathcal{T}_p$ indexed by a parameter p close to 1, so that $\mathcal{T}_p(X_t)$ corresponds to a truncation of X_t at a given quantile ϕ_p. In the numerical tests, we shall vary p between 0.95 and 0.999.

• PARAMETERS OF THE OPTIMIZATION ALGORITHM OVER NEURAL NETWORKS: In the whole numerical part, we use a classical Feedforward network using layers with m neurons each and a tanh activation function, the output layer uses an identity activation function. At each time step the resolution of equation (5.2.7) is achieved using a mini-batch with 1000 training trajectories. The training and learning rate adaptation procedure is the following:

- Every 40 inner gradient descent iterations, the loss is checked on 10000 validation trajectories.
- This optimization sequence is repeated with 200 outer iterations for the first optimization step at date $t_N = T$ and only 100 outer iterations at the dates t_i with $i < N$.
- An average of the loss calculated on 10 successive outer iterations is performed. If the decrease of the average loss every 10 outer iterations is less than 5% then the learning rate is divided by 2.

The optimization is performed using the Adam gradient descent algorithm, see [KB14]. Notice that the adaptation of the learning rate is not common with the Adam method but in our case it appears to be crucial to have a steady diminution of the loss of the objective function. The procedure is also described in [CWNMW19] and the chosen parameters are similar to this article. At the initial optimization step at time $t_N = T$, the learning rate is taken equal to $1E-2$ and at the following optimization steps, we start with a learning rate equal to $1E-3$.

During time resolution, it is far more effective to initialize the solution of equations (5.2.7) with the solution $(\mathcal{U}, \mathcal{Z})$ at the next time step. Indeed the previously computed values at time step t_{i+1} are good approximations of the processes at time step t_i if the PDE solution and its gradient are continuous.

All experiments are achieved using Tensorflow [Aba+16]. In the sequel, the PDE solutions on curves are calculated as the average of 10 runs. We provide the standard deviation associated to these results. We also show the influence of the number of neurons on the accuracy of the results.

5.3.2 A non-linearity in uD_x^2u

We consider a generator in the form

$$f(t,x,y,z,\gamma) = \ y\mathrm{tr}(\gamma) + \frac{y}{2} + 2y^2 - 2y^4 e^{-(T-t)},$$

and $g(x) = \tanh\big(\frac{\sum_{i=1}^d x_i}{\sqrt{d}}\big)$, so that an analytical solution is available:

$$u(t,x) = \ \tanh\Big(\frac{\sum_{i=1}^d x_i}{\sqrt{d}}\Big) e^{-\frac{T-t}{2}}.$$

We fix the horizon $T = 1$, and choose to evaluate the solution at $t = 0$ and $x = 0.5\frac{\mathbf{I}_d}{\sqrt{d}}$ (here $\mathbb{I}_d$ denotes the vector in $\mathbb{R}^d$ with all components equal to 1), for which $u(t,x) = 0.761902$ while its derivative is equal to 1.2966.

This initial value x is chosen such that independently of the dimension the solution is varying around this point and not in a region where the tanh function is close to -1 or 1.

The coefficients of the forward process used to solve the equation are (here $\mathbf{I}_d$ is the identity $d \times d$-matrix)

$$\sigma = \frac{\hat{\sigma}}{\sqrt{d}}\mathbf{I}_d, \quad \mu = 0,$$

and here the truncation operator is chosen equal to

$$\mathcal{T}_p(X_t^{0,x}) = \min\big\{\max[x - \sigma\sqrt{t}\phi_p, X_t^{0,x}], x + \sigma\sqrt{t}\phi_p\big\},$$

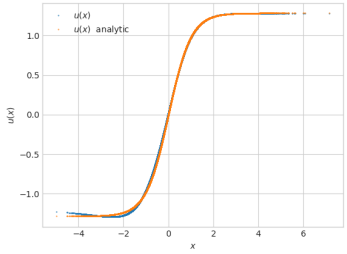
where $\phi_p = \mathcal{N}^{-1}(p)$, with $\mathcal{N}$ is the CDF of a unit centered Gaussian random variable.

In the numerical results, we take $p = 0.999$ and $m = 20$ neurons. We first begin in dimension $d = 1$, and show in Figure 5.1 how u, $D_x u$ and $D_x^2 u$ are well approximated by the resolution method.

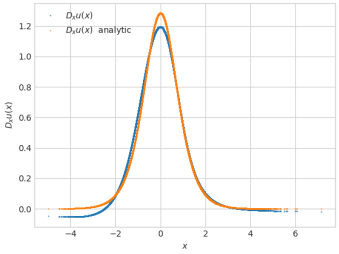
On Figure 5.2, we check the convergence, for different values of $\hat{\sigma}$ of both the solution u and its derivative at point x and date 0. Standard deviation of the function value is very low and the standard deviation of the derivative still being low.

As the dimension increases, we have to increase the value of $\hat{\sigma}$ of the forward process. In dimension 3, the value $\hat{\sigma} = 0.5$ gives high standard deviation in the result obtained as shown on Figure 5.3, while in dimension 10, see Figure 5.4, we see that the value $\hat{\sigma} = 1$ is too low to give good results. We also clearly notice that in 10D, a smaller time step should be used but in our test cases we decided to consider a maximum number of time steps equal to 160.

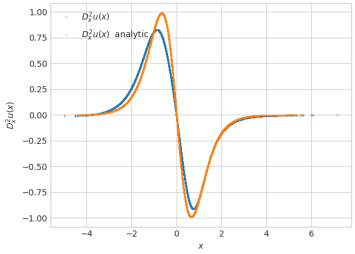
On this simple test case, the dimension is not a problem and very good results are obtained in dimension 20 or above with only 20 neurons and 2 layers.



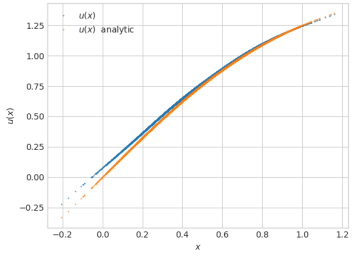
Y at date $t = 0.5$.



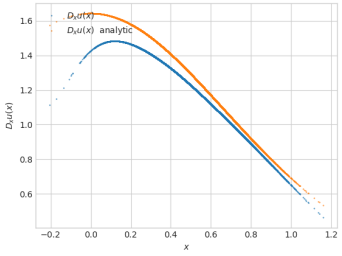
Z at date $t = 0.5$



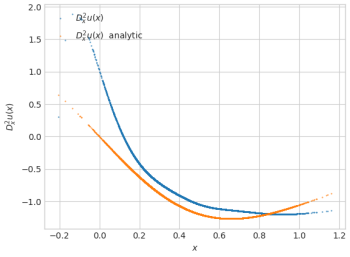
Γ at date $t = 0.5$



Y at date $t = 0.006125$.

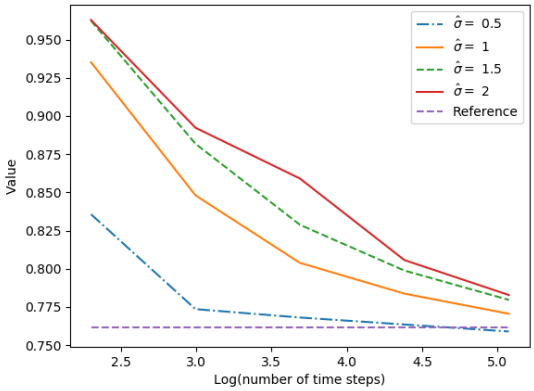


Z at date $t = 0.006125$

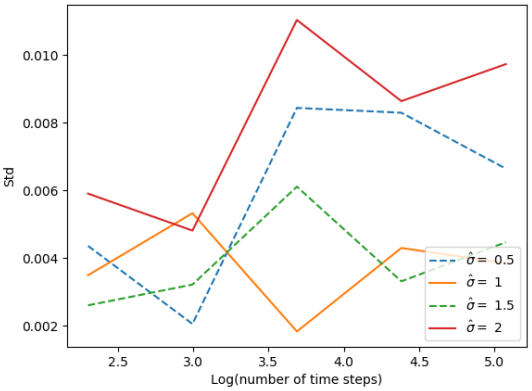


Γ at date $t = 0.006125$

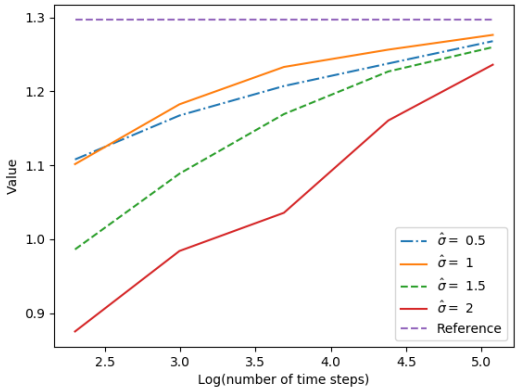
Figure 5.1: A single valuation run for test case one 1D using 160 time steps, $\hat{\sigma} = 2.$, $p = 0.999$, 20 neurons, 2 layers.



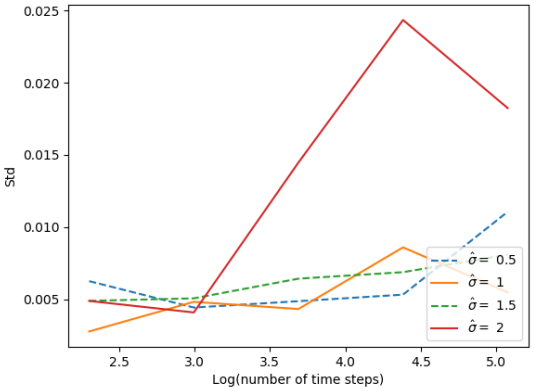
Convergence of u depending on $\hat{\sigma}$



Standard deviation of u

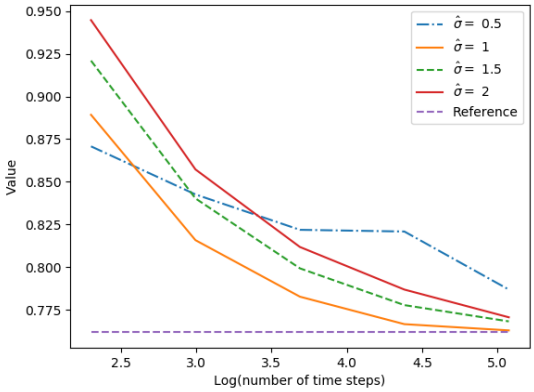


Convergence of $D_x u$ depending on $\hat{\sigma}$

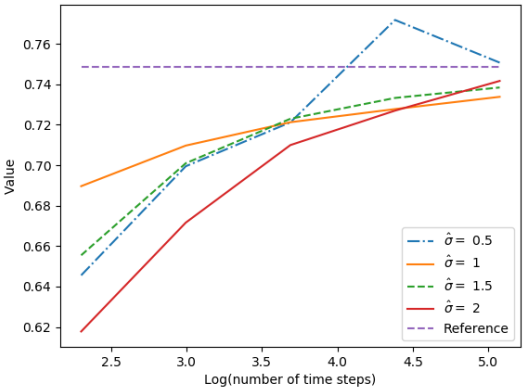


Standard deviation of $D_x u$

Figure 5.2: Convergence in 1D of the case one, number of neurons par layer equal to 20, 2 layers, $p = 0.999$.

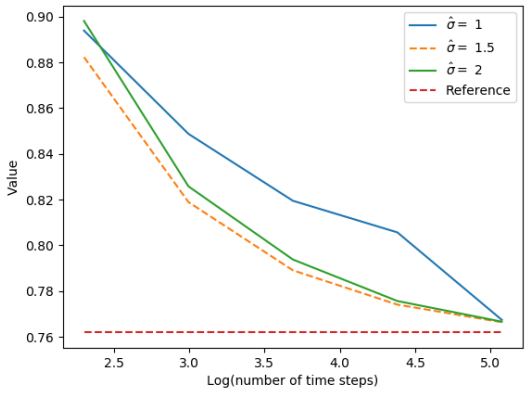


Convergence of u depending on $\hat{\sigma}$

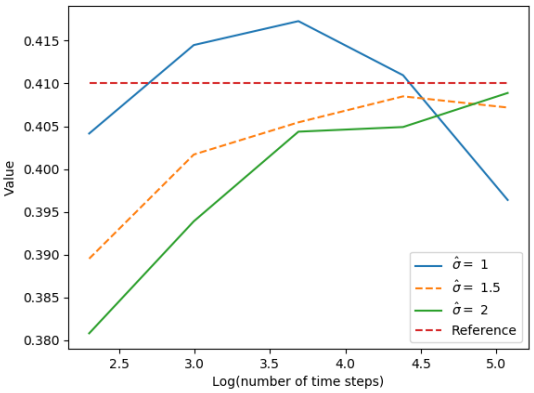


Convergence of $D_x u$ (first component) depending on $\hat{\sigma}$

Figure 5.3: Convergence in 3D of the case one, number of neurons par layer equal to 20, 2 layers, $p = 0.999$.



Convergence of u depending on $\hat{\sigma}$



Convergence of $D_x u$ depending on $\hat{\sigma}$ (first component)

Figure 5.4: Convergence in 10D of the case one, number of neurons par layer equal to 20, 2 layers, $p = 0.999$.

5.3.3 A linear quadratic stochastic test case.

In this example, we consider a controlled process $\mathcal{X} = \mathcal{X}^\alpha$ with dynamics in $\mathbb{R}^d$ according to

$$d\mathcal{X}_t = (A\mathcal{X}_t + B\alpha_t)dt + D\alpha_t dW_t, \quad 0 \le t \le T, \;\; \mathcal{X}_0 = x,$$

where W is a real Brownian motion, the control process α is valued in $\mathbb{R}$, and the constant coefficients $A \in \mathbb{M}^d$, $B \in \mathbb{R}^d$, $D \in \mathbb{R}^d$. The quadratic cost functional to be minimized is

$$J(\alpha) = \mathbb{E}\Big[\int_0^T \big(\mathcal{X}_t^\intercal Q \mathcal{X}_t + \alpha_t^2 N\big)dt + \mathcal{X}_T^\intercal P \mathcal{X}_T\Big],$$

where P, Q are non negative $d \times d$ symmetric matrices and $N \in \mathbb{R}$ is strictly positive.

The Bellman equation associated to this stochastic control problem is:

$$\frac{\partial u}{\partial t} + \inf_{a\in\mathbb{R}} \big[(Ax + Ba).D_x u + \frac{a^2}{2}\mathrm{tr}(DD^\intercal D_x^2 u) + x^\intercal Q x + Na^2\big] = 0, \quad (t,x) \in [0,T) \times \mathbb{R}^d,$$
$$u(T,x) = x^\intercal P x, \quad x \in \mathbb{R}^d,$$

which can be rewritten as a fully nonlinear equation in the form (5.1.1) with

$$f(t,x,y,z,\gamma) = x^\intercal Q x + Ax.z - \frac{1}{2}\frac{|B^\intercal z|^2}{\mathrm{tr}(DD^\intercal\gamma) + 2N}.$$

An explicit solution to this PDE is given by

$$u(t,x) = x^\intercal K(t) x,$$

where $K(t)$ is non negative $d \times d$ symmetric matrix function solution to the Riccati equation:

$$\dot{K} + A^\top K + KA + Q - \frac{KBB^\top K}{N + D^\top K D} = 0, \quad K(T) = P.$$

We take $T = 1$. The coefficients of the forward process used to solve the equation are

$$\sigma = \frac{\hat{\sigma}}{\sqrt{d}}\mathbf{I}_d, \quad \mu(t,x) = Ax.$$

In our numerical example we take the following parameters for the optimization problem:

$$A = \mathbf{I}_d, \; B = D = \mathbb{1}_d, \quad Q = P = \frac{1}{d}\mathbf{I}_d, \quad N = d$$

and we want to estimate the solution at $x = \mathbb{1}_d$.

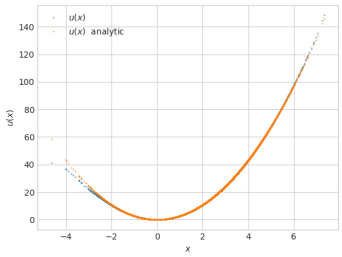
In this example, the truncation operator (indexed by p between 0 and 1 and close to 1) is as follows:

$$\mathcal{T}_p(X_t^x) = \min\big\{\max\big[xe^{\hat{A}t} - \sigma\sqrt{\frac{e^{2\hat{A}t} - \hat{1}}{2\hat{A}}}\phi_p, X_t^x\big], xe^{\hat{A}t} + \sigma\sqrt{\frac{e^{2\hat{A}t} - \hat{1}}{2\hat{A}}}\phi_p\big\},$$

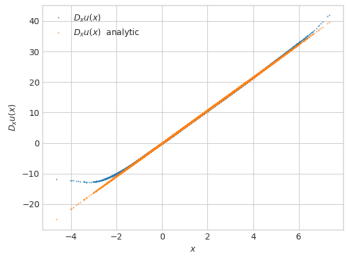
where $\phi_p = \mathcal{N}^{-1}(p)$, $\hat{A}$ is a vector so that $\hat{A}_i = A_{ii}$, $i = 1, ..., d$, $\hat{1}$ is a unit vector, and the square root is taken componentwise.

On Figure 5.5 we give the solution of the PDE with $d = 1$ using $\hat{\sigma} = 1.5$ obtained for two dates: at $t = 0.5$ and at t close to zero. We observe that we have a very good estimation of the function value and a correct one of the Γ value at date $t = 0.5$. The precision remains good for Γ close to $t = 0$ and very good for u and $D_x u$.

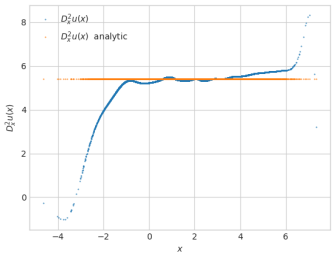
On Figure 5.6, we give the results obtained in dimension $d = 1$ by varying $\hat{\sigma}$. For a value of $\hat{\sigma} = 2$, the standard deviation of the result becomes far higher than with $\hat{\sigma} = 0.5$ or 1.



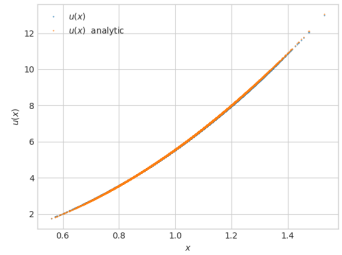
Y at date $t = 0.5$.



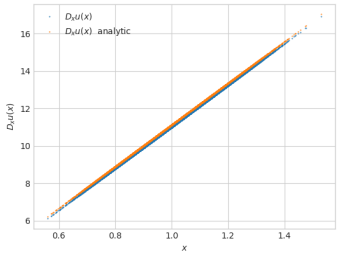
Z at date $t = 0.5$



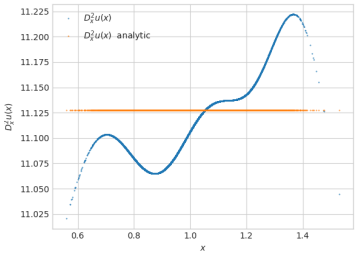
Γ at date $t = 0.5$



Y at date $t = 0.006125$.

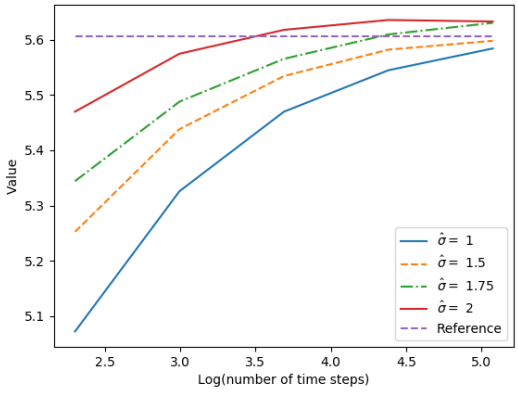


Z at date $t = 0.006125$

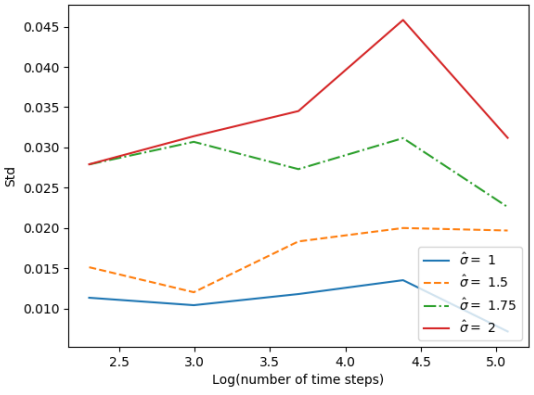


Γ at date $t = 0.006125$

Figure 5.5: Test case linear quadratic 1D using 160 time steps, $\hat{\sigma} = 1.5$, $p = 0.999$, 100 neurons.

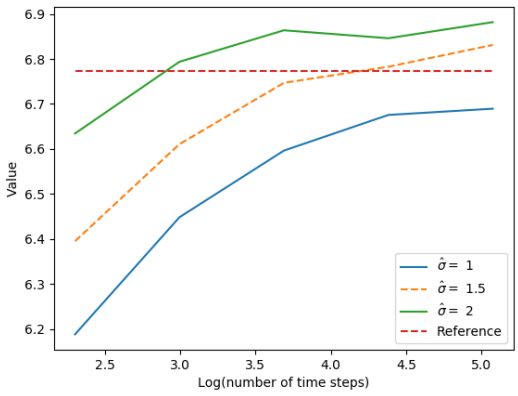


Convergence of u depending on $\hat{\sigma}$.

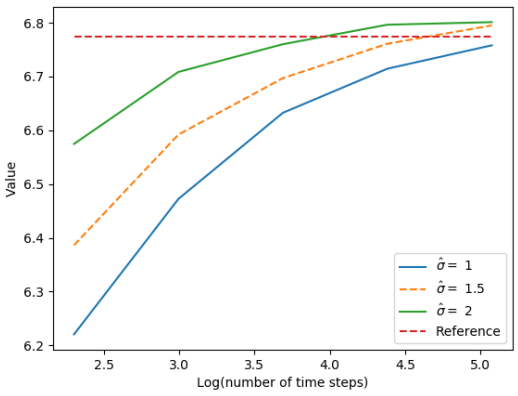


Standard deviation of u

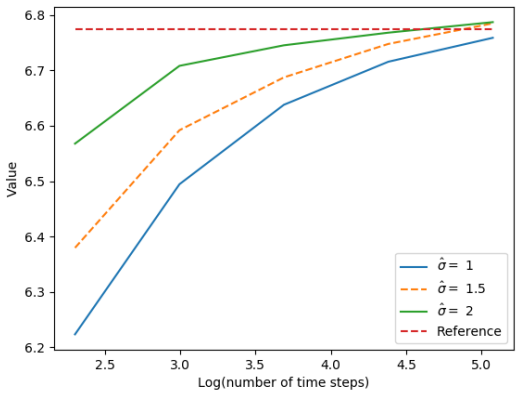
Figure 5.6: Convergence in 1D of the linear quadratic case, number of neurons par layer equal to 50, 2 layers, $p = 0.999$.



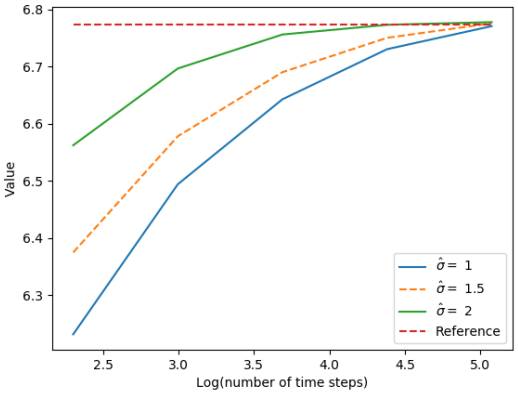
10 neurons



20 neurons

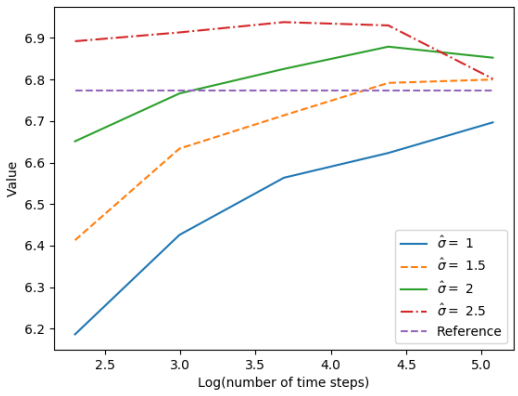


30 neurons

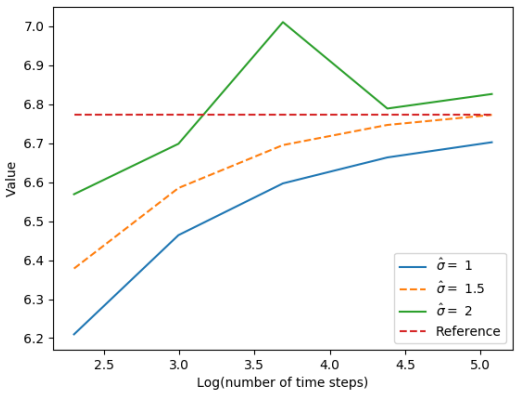


50 neurons

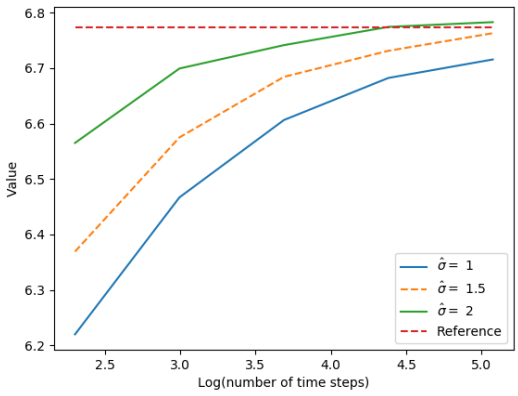
Figure 5.7: Convergence in 3D of the linear quadratic case, 2 layers, testing the influence of the number of neurons, truncation $p = 0.95$.



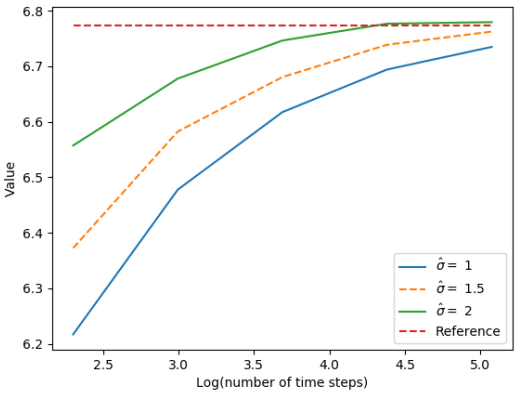
10 neurons



20 neurons

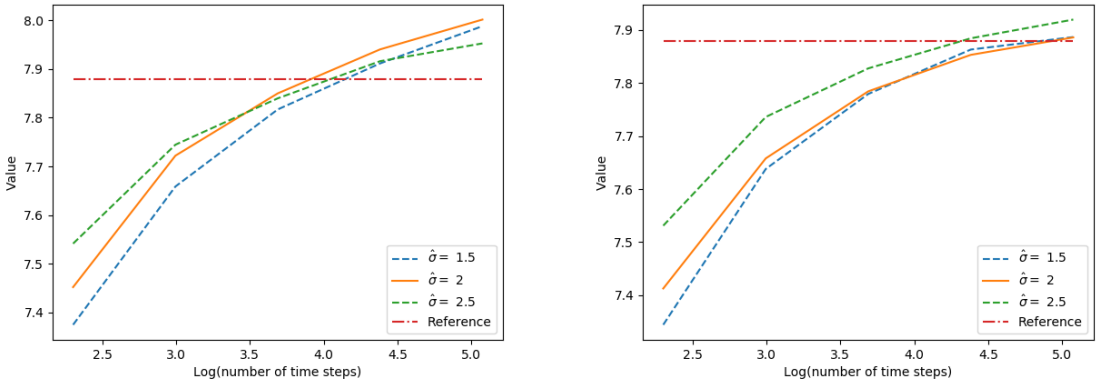


50 neurons



100 neurons

Figure 5.8: Convergence in 3D of the linear quadratic case, 2 layers, testing the influence of the number of neurons, truncation $p = 0.99$.



Convergence with 50 neurons Convergence with 100 neurons

Figure 5.9: Convergence in 7D of the linear quadratic case, 2 layers, $p = 0.999$.

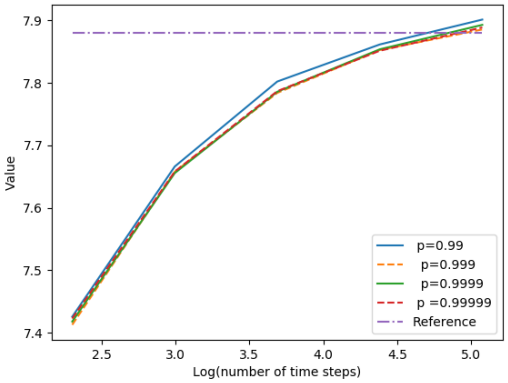


Figure 5.10: Function value convergence in 7D of the linear quadratic case with 2 layers, 100 neurons, testing p, using $\hat{\sigma} = 2$

.

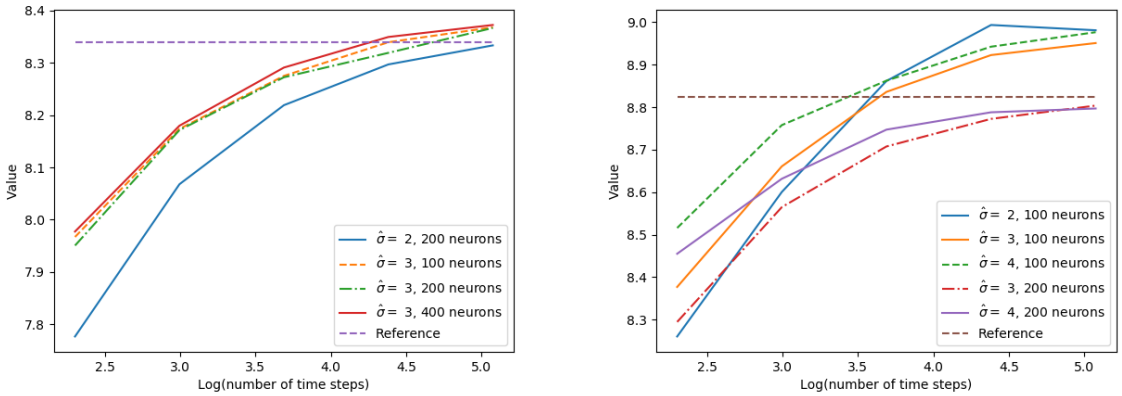
On Figure 5.7, for $d = 3$, we take a quite low truncation factor $p = 0.95$ and observe that the number of neurons to take has to be rather high. We have also checked that taking a number of hidden layers equal to 3 does not improve the results.

On Figure 5.8, for $d = 3$, we give the same graphs for a higher truncation factor. As we take a higher truncation factor, the results are improved by taking a higher number of neurons (100 in the figure below).

On Figure 5.9, we observe in dimension 7 the influence of the number of neurons on the result for a high truncation factor $p = 0.999$. We clearly have a bias for a number of neurons equal to 50. This bias disappears when the number of neurons increases to 100.

On Figure 5.10, for $d = 7$, we check that influence of the truncation factor appears to be slow for higher dimensions.

Finally, we give results in dimension 10, 15 and 20 for $p = 0.999$ on Figures 5.11, 5.12. We observe that the number a neurons with 2 hidden layers has to increase with the dimension but also that the increase is rather slow in contrast with the case of one hidden layer as theoretically shown in [Pin99b]. For $\hat{\sigma} = 5$ we had to take 300 neurons to get very accurate results.



10D 15D

Figure 5.11: Function value convergence in 10D and 15D of the linear quadratic case with 2 layers, $p = 0.999$.

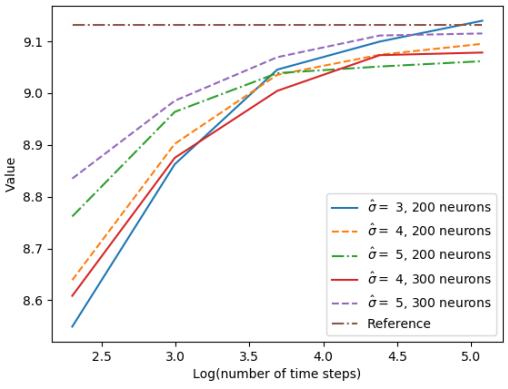


Figure 5.12: Function value convergence in 20D of the linear quadratic case with 2 layers, $p = 0.999$.

Dimension d	Averaged value	Standard deviation	Relative error (%)	Theoretical solution
5	0.37901	0.00312	0.97	0.382727
15	0.25276	0.00235	1.17	0.255754

Table 5.1: Estimate of $u(0, x_0 = 1_5)$ on the Monge Ampere problem (5.3.1) with $N = 120$. Average and standard deviation observed over 10 independent runs are reported.

5.3.4 Monge-Ampère equation

Let us consider the parabolic Monge-Ampère equation

$$\begin{cases} \partial_t u + \det(D_x^2 u) \;=\; h(x), \quad (t,x) \in [0,T] \times \mathbb{R}^d, \\ u(T,x) \;=\; g(x), \end{cases} \tag{5.3.1}$$

where $\det(D_x^2 u)$ is the determinant of the Hessian matrix $D_x^2 u$. It is in the form (5.1.1) with

$$f(t,x,\gamma) = \det(\gamma) - h(x).$$

We test our algorithm by choosing a C^2 function g, then compute $G = \det(D_x^2 g)$, and set $h := G - 1$. Then, by construction, the function

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

is solution to the Monge-Ampère equation (5.3.1). We choose $g(x) = \cos(\sum_{i=1}^d x_i/\sqrt{d})$, and we shall train with the forward process $X = x_0 + W$, where W is a d-dimensional Brownian motion. On this example, we use neural networks with 3 hidden layers, $d + 10$ neurons per layer, and we do not need to apply any truncation to the forward process X. Actually, we observe that adding a truncation worsens the results. For choosing the truncation level, we first test the method with no truncation before decreasing the quantile parameter p. In the Monge-Ampère case the best results are obtained without any truncation. It may be caused by the oscillation of the Hessian.

The following table gives the results in dimension $d = 5$, 15, and for $T = 1$.

5.3.5 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

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

where $W = (W^1, \dots, W^n)$ is a n-dimensional Brownian motion, $b = (b^1, \dots, b^n)$ is the rate of return of the assets, $\lambda = (\lambda^1, \dots, \lambda^n)$ is the risk premium 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

$$\mathrm{d}V_t^i = \kappa_i[\theta_i - V_t^i]\mathrm{d}t + \nu_i \mathrm{d}B_t^i, \quad i = 1, \dots, n, \tag{5.3.2}$$

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 $i=1,\ldots,n$. This Bellman equation is in the form (5.1.1) with

$$f(t,x,y,z,\gamma)=\;\sum_{i=1}^n\big[\kappa_i(\theta_i-v_i)z_i+\frac{1}{2}\nu_i^2\gamma_{ii}\big]-\frac{1}{2}R(v)\frac{z_0^2}{\gamma_{00}}-\sum_{i=1}^n\big[\rho_i\lambda_i(v_i)\nu_i\frac{z_0\gamma_{0i}}{\gamma_{00}}+\frac{1}{2}\rho_i^2\nu_i^2\frac{(\gamma_{0i})^2}{\gamma_{00}}\big],$$

for $x=(\mathrm{x},v)\in\mathbb{R}^{n+1}$, $z=(z_0,\ldots,z_n)\in\mathbb{R}^{n+1}$, $\gamma=(\gamma_{ij})_{0\le i,j\le n}\in\mathbb{S}^{n+1}$, and displays a high non-linearity in the Hessian argument γ.

The truncation operator indexed by a parameter p is chosen equal to

$$\mathcal{T}_p(X_t^{0,x})=\min\big\{\max[x+\mu t-\sigma\sqrt{t}\phi_p,X_t^{0,x}],x+\mu t+\sigma\sqrt{t}\phi_p\big\},$$

where $\phi_p=\mathcal{N}^{-1}(p)$, $\mathcal{N}$ is the CDF of a unit centered Gaussian random variable. We use neural networks with 2 hidden layers and $d+10$ neurons per layer. 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 closed-form 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, so that the generator of Bellman equation reduces to

$$f(t,x,y,z,\gamma)\;=\;-\frac{1}{2}|\lambda|^2\frac{z^2}{\gamma},\qquad (t,x,y,z)\in[0,T]\times\mathbb{R}\times\mathbb{R}\times\mathbb{R},\tag{5.3.3}$$

with explicit solution given by:

$$u(t,x)=\;e^{-(T-t)\frac{|\lambda|^2}{2}}U(x),\quad \hat{a}_i\;=\;\frac{\lambda_i}{\eta\sigma}.$$

We train with the forward process

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

(2) *One risky asset:* $n=1$. A quasi-explicit solution is provided in [Zar01]:

$$u(t,\mathrm{x},v)=\;U(\mathrm{x})w(t,v),\qquad \text{with } w(t,v)\;=\;\Big\|\exp\Big(-\frac{1}{2}\int_t^T R(\hat{V}_s^{t,v})ds\Big)\Big\|_{L^{1-\rho^2}}$$

where $\hat{V}_s^{t,v}$ is the solution to the modified O.U. model

$$\mathrm{d}\hat{V}_s=\;\big[\kappa(\theta-\hat{V}_s)-\rho\nu\lambda(\hat{V}_s)\big]\mathrm{d}s+\nu\mathrm{d}B_s,\quad s\ge t,\;\hat{V}_t=v.$$

We test our algorithm with $\lambda(v) = \lambda v$, $\lambda > 0$, for which we have an explicit solution:

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

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])

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

with $\hat\kappa = \sqrt{\kappa^2 + 2\rho\nu\lambda\kappa + \gamma^2\lambda^2}$. We train with the forward process

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

(3) *No leverage effect, i.e., $\rho_i = 0$, $i = 1,\dots,n$.* In this case, there is a quasi-explicit solution given by

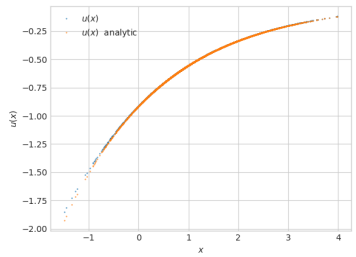
$$u(t,\mathrm{x},v) = U(\mathrm{x})w(t,v), \quad \text{with } w(t,v) = \mathbb{E}\Big[\exp\Big(-\frac{1}{2}\int_t^T R(V_s^{t,v})ds\Big)\Big], \ \ (t,v)\in[0,T]\times\mathbb{R}^n, \tag{5.3.4}$$

where $V^{t,v}$ is the solution to (5.3.2), starting from v at time t. 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 given by

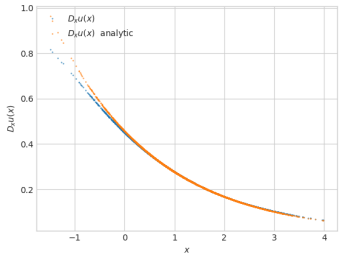
$$\begin{aligned}
w(t,v) &= \exp\Big(-\sum_{i=1}^n\big[\phi_i(t)\frac{v_i^2}{2} + \psi_i(t)v_i + \chi_i(t)\big]\Big), \quad (t,v)\in[0,T]\times\mathbb{R}^n,\\
\phi_i(t) &= \lambda_i^2\frac{\sinh(\hat\kappa_i(T-t))}{\kappa_i\sinh(\hat\kappa_i(T-t)) + \hat\kappa_i\cosh(\hat\kappa_i(T-t))}\\
\psi_i(t) &=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

	Averaged value	Standard deviation	Theoretical value	Relative error (%)
$u(0, x_0 = 1)$	-0.50561	0.00029	-0.50662	0.20
$D_x u(0, x_0 = 1)$	0.25081	0.00088	0.25331	0.99
$\alpha(0, x_0 = 1)$	0.83552	0.02371	0.80438	3.87

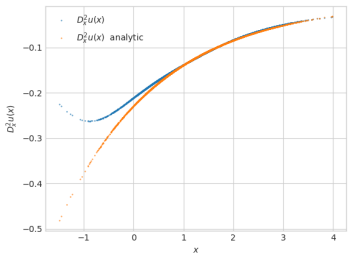
Table 5.2: Estimate of the solution, its derivative and the optimal control at the initial time $t = 0$ in the Merton problem (5.3.3). Average and standard deviation observed over 10 independent runs are reported.



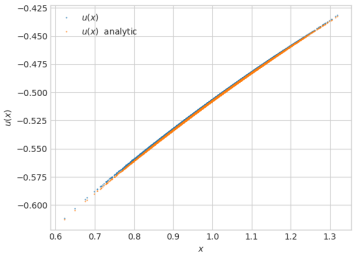
Y at date $t = 0.5042$.



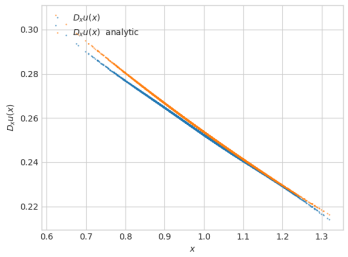
Z at date $t = 0.5042$



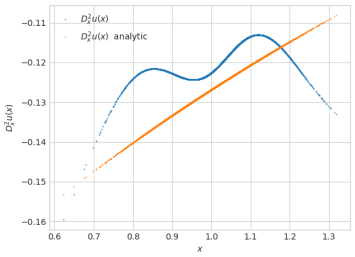
Γ at date $t = 0.5042$



Y at date $t = 0.0084$.



Z at date $t = 0.0084$



Γ at date $t = 0.0084$

Figure 5.13: Estimates of the solution and its derivatives on the Merton problem (5.3.3) using 120 time steps.

with $\hat{\kappa}_i = \sqrt{\kappa_i^2 + \nu_i^2 \lambda_i^2}$. We train with the forward process

$$\begin{aligned} \mathcal{X}_{k+1} &= \mathcal{X}_k + \Delta W_k, \quad k = 0, \dots, N-1, \quad \mathcal{X}_0 = \mathrm{x}_0, \\ V_{k+1}^i &= V_k^i + \nu_i \Delta B_k^i, \qquad k = 0, \dots, N-1, \quad V_0^i = \theta_i, \end{aligned}$$

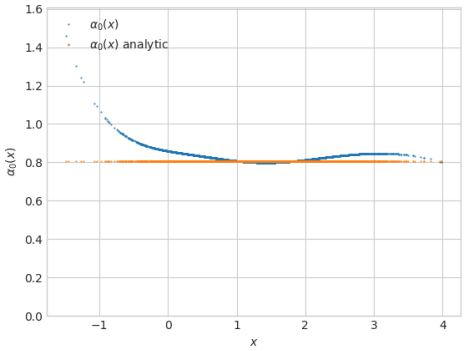
with $< W, B^i >_t = 0$.

Merton Problem. We take $\eta = 0.5$, $\lambda = 0.6$, $T = 1$, $N = 120$, and $\sigma(v) = e^v$. We plot the neural networks approximation of $u, D_x u, D_x^2 u, \alpha$ (in blue) together with their analytic values (in orange). For comparison with Figures 5.6 and 5.7, we report the error on the gradient and the initial control. In practice, after empirical tests, we choose $p = 0.98$ for the truncation.

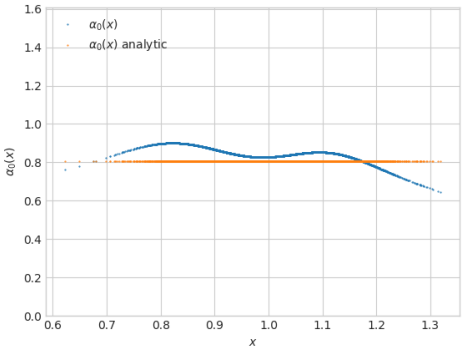
One asset ($n = 1$) in Scott volatility model. We take $\eta = 0.5$, $\lambda = 1.5$, $\theta = 0.4$, $\nu = 0.4$, $\kappa = 1$, $\rho = -0.7$. For all tests we choose $T = 1$, $N = 120$, and $\sigma(v) = e^v$. In practice, after empirical tests, we choose $p = 0.98$ for the truncation.

No Leverage in Scott model. In the case with one asset ($n = 1$), we take $\eta = 0.5$, $\lambda = 1.5$, $\theta = 0.4$, $\nu = 0.2$, $\kappa = 1$. For all tests we choose $T = 1$, $N = 120$, and $\sigma(v) = e^v$. In practice, after empirical tests, we choose $p = 0.95$ for the truncation.

In the case with four assets ($n = 4$, $d = 5$), we take $\eta = 0.5$, $\lambda = \begin{pmatrix} 1.5 & 1.1 & 2. & 0.8 \end{pmatrix}$, $\theta = \begin{pmatrix} 0.1 & 0.2 & 0.3 & 0.4 \end{pmatrix}$, $\nu = \begin{pmatrix} 0.2 & 0.15 & 0.25 & 0.31 \end{pmatrix}$, $\kappa = \begin{pmatrix} 1. & 0.8 & 1.1 & 1.3 \end{pmatrix}$.



α at date $t = 0.5042$.



α at date $t = 0.0084$.

Figure 5.14: Estimates of the optimal control α on the Merton problem (5.3.3).

Averaged value	Standard deviation	Relative error (%)
-0.53431	0.00070	0.34

Table 5.3: Estimate of $u(0, \mathrm{x}_0 = 1, \theta)$ on the One Asset problem with stochastic volatility ($d = 2$). Average and standard deviation observed over 10 independent runs are reported. The exact solution is -0.53609477.

Dimension d	Averaged value	Standard deviation	Relative error (%)	Theoretical solution
2	-0.49980	0.00073	0.35	-0.501566
5	-0.43768	0.00137	0.92	-0.441765
8	-0.38720	0.00363	1.96	-0.394938
10	-0.27920	0.05734	1.49	-0.275092

Table 5.4: Estimate of $u(0, \mathrm{x}_0 = 1, \theta)$ on the No Leverage problem (5.3.4). Average and standard deviation observed over 10 independent runs are reported.

	Average	Std	True value	Relative error (%)
$u(0, x_0 = 1)$	-0.50572	0.00034	-0.50662	0.18
$D_x u(0, x_0 = 1)$	0.25091	0.00067	0.25331	0.95
$\alpha(0, x_0 = 1)$	0.85254	0.01956	0.80438	5.99

Table 5.5: Estimate of the solution, its derivative and the optimal control at the initial time $t = 0$ in the Merton problem (5.3.3) with implicit estimation of the Hessian. Average and standard deviation (Std) observed over 10 independent runs are reported

In the case with seven assets ($n = 7$, $d = 8$) we take $\eta = 0.5$, $\lambda = \big(1.5 \quad 1.1 \quad 2. \quad 0.8 \quad 0.5 \quad 1.7 \quad 0.9\big)$, $\theta = \big(0.1 \quad 0.2 \quad 0.3 \quad 0.4 \quad 0.25 \quad 0.15 \quad 0.18\big)$, $\nu = \big(0.2 \quad 0.15 \quad 0.25 \quad 0.31 \quad 0.4 \quad 0.35 \quad 0.22\big)$, $\kappa = \big(1. \quad 0.8 \quad 1.1 \quad 1.3 \quad 0.95 \quad 0.99 \quad 1.02\big)$.

In the case with nine assets ($n = 9$, $d = 10$), we take $\eta = 0.5$, $\lambda = \big(1.5 \quad 1.1 \quad 2. \quad 0.8 \quad 0.5 \quad 1.7 \quad 0.9 \quad 1. \quad 0.9\big)$, $\theta = \big(0.1 \quad 0.2 \quad 0.3 \quad 0.4 \quad 0.25 \quad 0.15 \quad 0.18 \quad 0.08 \quad 0.91\big)$, $\nu = \big(0.2 \quad 0.15 \quad 0.25 \quad 0.31 \quad 0.4 \quad 0.35 \quad 0.22 \quad 0.4 \quad 0.1$ $\kappa = \big(1. \quad 0.8 \quad 1.1 \quad 1.3 \quad 0.95 \quad 0.99 \quad 1.02 \quad 1.06 \quad 1.6\big)$.

Hamilton-Jacobi-Bellman equation from portfolio optimization is a typical example of full-nonlinearity in the second order derivative, and the above results show that our algorithm performs quite well up to dimension $d = 8$, but gives a high variance in dimension $d = 10$.

Comparison with an implicit version of the scheme.
As explained in Remark 2, an alternative option for the estimation of the Hessian is to approximate it by the automatic differentiation of the current neural network for the Z component. It corresponds to the replacement of $D\hat{\mathcal{Z}}_{i+1}(\mathcal{T}(X_{t_{i+1}}))$ by $D\mathcal{Z}_i(\mathcal{T}(X_{t_i}); \theta)$ in (5.2.7). An additional change has to be made to the method for it to work. At the last optimization step (for time step $t_0 = 0$), we notice empirically that the variable Γ_0 is not able to properly learn the initial Hessian value at all. Therefore for this last step we use variables Y_0, Z_0 and an explicit estimation of the second order derivative given by $D\hat{\mathcal{Z}}_1(\mathcal{T}(X_{t_1}))$. We see in Table 5.5 that the results for the Merton problem are very similar to the ones from Table 5.2 for the splitting scheme but with a worse estimation of the Hessian and optimal control (the error is multiplied by around 1.5). When we tested this implicit scheme on the Monge Ampere problem we also faced computational problems during the optimization step of Tensorflow. The numerical computation of the gradient of the objective function for the backpropagation step, more precisely for the determinant part, often gives rise to matrix invertibility errors which stops the algorithm execution. For these two reasons, we focused our study on the explicit scheme.

Comparison with the 2BSDE scheme of [BEJ19]. We conclude this paper with a comparison of our algorithm with the global scheme of [BEJ19], called Deep 2BDSE. The tests below concern the Merton problem (5.3.3) but similar behavior happens on the other examples with stochastic volatilities. This scheme was implemented in the original paper only for small number of time steps (e.g. $N = 30$). Thus we tested this algorithm on two discretizations, respectively with $N = 20$ and $N = 120$ time steps, as shown in Figure 5.15, for $T = 1$ where we plotted the learning curve of the Deep BSDE method. These curves correspond to the values taken by the loss function during the gradient descent iterations. For this algorithm the loss function to minimize in the training of neural networks is defined as the mean L^2 error between the generated Y_N value and the true terminal condition $g(X_N)$. We observe that for this choice of maturity $T = 1$ the loss function oscillates during the training process and does not vanish. As a consequence the Deep 2BSDE does not converge in this case. Even when decreasing the learning rate, we noticed that we cannot obtain the convergence of the scheme.

However, the Deep 2BSDE method does converge for small maturities T, as illustrated in Table 5.6 with $T = 0.1$ and different values for the number of time steps N. Nevertheless, even if the value function is well approximated, the estimation of the gradient and control did not

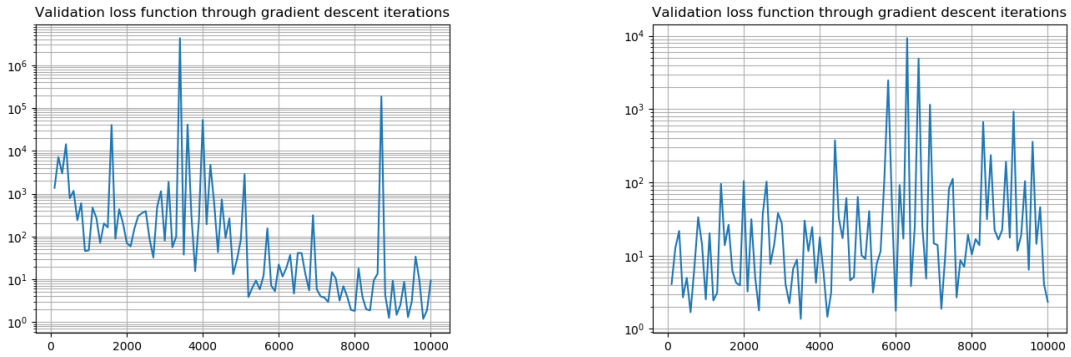


Figure 5.15: Learning curve in logarithmic scale for the scheme [BEJ19] on the Merton problem (5.3.3) with $N = 20$ times steps on the left and $N = 120$ time steps on the right. The maturity is $T = 1$
. 10000 gradient descent iterations were conducted.

	N	Averaged value	Standard deviation	Theoretical value	Relative error (%)
$u(0, x_0 = 1)$	5	-0.60667	**0.01588**	-0.59571	1.84
$u(0, x_0 = 1)$	10	-0.59841	0.02892	-0.59571	0.45
$u(0, x_0 = 1)$	20	**-0.59316**	0.04251	-0.59571	**0.43**
$D_x u(0, x_0 = 1)$	5	**0.09668**	**0.25630**	0.29786	**67.54**
$D_x u(0, x_0 = 1)$	10	0.03810	0.44570	0.29786	93.36
$D_x u(0, x_0 = 1)$	20	0.07557	0.55030	0.29786	74.63
$\alpha(0, x_0 = 1)$	5	-0.15243	0.61096	0.80438	118.95
$\alpha(0, x_0 = 1)$	10	**0.59971**	1.97906	0.80438	**25.44**
$\alpha(0, x_0 = 1)$	20	0.28385	**0.43775**	0.80438	64.71

Table 5.6: Estimate of the solution, its derivative and the optimal control at the initial time $t = 0$ in the Merton problem (5.3.3) with maturity $T = 0.1$ for the [BEJ19] scheme. Average and standard deviation observed over 10 independent runs are reported.

converge (the corresponding variance is very large), in comparison with our scheme whereas the gradient is very well approximated and the control is quite precise. We also have a much smaller variance in the results. Table 5.7 shows the results obtained by our method with $T = 0.1$ in order to compare it with the performance of [BEJ19]. It illustrates the limitations of the global approach and justifies our introduction of a local method.

	N	Averaged value	Standard deviation	Theoretical value	Relative error (%)
$u(0, x_0 = 1)$	5	**-0.59564**	0.01136	-0.59571	**0.01**
$u(0, x_0 = 1)$	10	-0.59550	**0.00037**	-0.59571	0.04
$u(0, x_0 = 1)$	20	-0.59544	0.00054	-0.59571	0.04
$D_x u(0, x_0 = 1)$	5	0.29848	**0.00044**	0.29786	0.21
$D_x u(0, x_0 = 1)$	10	0.29842	0.00084	0.29786	0.19
$D_x u(0, x_0 = 1)$	20	**0.29785**	0.00054	0.29786	**0.001**
$\alpha(0, x_0 = 1)$	5	**0.82322**	**0.01014**	0.80438	**2.34**
$\alpha(0, x_0 = 1)$	10	0.85284	0.07565	0.80438	6.02
$\alpha(0, x_0 = 1)$	20	0.84201	0.09892	0.80438	4.68

Table 5.7: Estimate of the solution, its derivative and the optimal control at the initial time $t = 0$ in the Merton problem (5.3.3) with maturity $T = 0.1$ for our scheme. Average and standard deviation observed over 10 independent runs are reported.

Part II

McKean-Vlasov equations and mean-field control

Chapter 6

Rate of convergence for particles approximation of PDEs in Wasserstein space

This chapter is based on the paper [GPW22c]
M. Germain, H. Pham, X. Warin. "Rate of convergence for particles approximation of PDEs in Wasserstein space", to appear in *Journal of Applied Probability* 59.4 (December 2022)

In this part of the thesis we move to the problem of mean-field control. In this generalization of stochastic control, the law of the state appears in the dynamics but also in the cost to minimize. As a consequence, it complicates the resolution and we need to design new numerical methods. In this Chapter we study the discretization in space of the semilinear PDEs on the Wasserstein space, like the Master Bellman equation characterizing the value function of mean-field control. We show the convergence speed of an approximating PDE in finite dimension by probabilistic arguments. Our work complements [GMS21] which studies a very similar problem but obtain a convergence without rate. Nevertheless, our proofs require existence and uniqueness of smooth classical solutions for the limit PDE, which is quite restrictive. [GMS21] works instead with the weaker notion of viscosity solutions and hence require less regularity.

Abstract

We prove a rate of convergence for the N-particle approximation of a second-order partial differential equation in the space of probability measures, like the Master equation or Bellman equation of mean-field control problem under common noise. The rate is of order $1/N$ for the pathwise error on the solution v and of order $1/\sqrt{N}$ for the L^2 error on its L-derivative $\partial_\mu v$. The proof relies on backward stochastic differential equations techniques.

6.1 Introduction

Let us consider the second-order parabolic partial differential equation (PDE) on the Wasserstein space $\mathcal{P}_2(\mathbb{R}^d)$ of square-integrable probability measures on $\mathbb{R}^d$, in the form:

$$\left\{ \begin{array}{rcll} \partial_t v + \mathcal{H}(t,\mu,v,\partial_\mu v,\partial_x\partial_\mu v,\partial_\mu^2 v) &=& 0, & (t,\mu)\in[0,T)\times\mathcal{P}_2(\mathbb{R}^d), \\ v(T,\mu) &=& G(\mu), & \mu\in\mathcal{P}_2(\mathbb{R}^d). \end{array} \right. \tag{6.1.1}$$

Here, $\partial_\mu v(t,\mu)$ is the L-derivative on $\mathcal{P}_2(\mathbb{R}^d)$ (see [CD18a]) of $\mu\mapsto v(t,\mu)$, and it is a function from $\mathbb{R}^d$ into $\mathbb{R}^d$, $\partial_x\partial_\mu v(t,\mu)$ is the usual derivative on $\mathbb{R}^d$ of $x\in\mathbb{R}^d\mapsto\partial_\mu v(t,\mu)(x)\in\mathbb{R}^d$, hence valued in $\mathbb{R}^{d\times d}$ the set of $d\times d$-matrices with real coefficients, and $\partial_\mu^2 v(t,\mu)$ is the L-derivative of $\mu\mapsto\partial_\mu v(t,\mu)(.)$, hence a function from $\mathbb{R}^d\times\mathbb{R}^d$ into $\mathbb{R}^{d\times d}$. The terminal condition is given by a real-valued function G on $\mathcal{P}_2(\mathbb{R}^d)$, and the Hamiltonian $\mathcal{H}$ of this PDE is assumed to be in semi-linear (non linear w.r.t. v, $\partial_\mu v$, and linear w.r.t. $\partial_x\partial_\mu$, $\partial_\mu^2 v$) expectation form:

$$\begin{aligned} \mathcal{H}(t,\mu,y,\mathrm{z}(.),\gamma(.),\gamma_0(.,.)) = &\int_{\mathbb{R}^d}\big[H(t,x,\mu,y,\mathrm{z}(x)) + \frac{1}{2}\mathrm{tr}\big((\sigma\sigma^\intercal+\sigma_0\sigma_0^\intercal)(t,x,\mu)\gamma(x)\big)\big]\mu(\mathrm{d}x) \\ &+ \frac{1}{2}\int_{\mathbb{R}^d\times\mathbb{R}^d}\mathrm{tr}\big(\sigma_0(t,x,\mu)\sigma_0^\intercal(t,x',\mu)\gamma_0(x,x')\big)\mu(\mathrm{d}x)\mu(\mathrm{d}x'), \end{aligned} \tag{6.1.2}$$

for some real-valued measurable function H defined on $[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)\times\mathbb{R}\times\mathbb{R}^d$, and where σ, σ_0 are measurable functions on $[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)$, valued respectively in $\mathbb{R}^{d\times n}$, and $\mathbb{R}^{d\times m}$. Here $\mathrm{tr}(M)$ denotes the trace of a square matrix M, while $M^\intercal$ is its transpose, and . is the scalar product.

PDEs in Wasserstein space have been largely studied in the literature over the last years, notably with the emergence of the mean-field game theory, and we mention among others the papers [BFY15], [GS15], [PW18], [Car+19], [SZ19], [Bur+20], and other references in the two-volume monographs [CD18a]-[CD18b].

An important application concerns mean-field type control problems with common noise. The controlled stochastic McKean-Vlasov dynamics is given by

$$\begin{aligned} \mathrm{d}X_s^\alpha = \ &\beta(s,X_s^\alpha,\mathbb{P}^0_{X_s^\alpha},\alpha_s)\mathrm{d}s + \sigma(s,X_s^\alpha,\mathbb{P}^0_{X_s^\alpha})\mathrm{d}W_s \\ &+ \ \sigma_0(s,X_s^\alpha,\mathbb{P}^0_{X_s^\alpha})\mathrm{d}W_s^0, \quad t\le s\le T,\ X_t^\alpha=\xi, \end{aligned} \tag{6.1.3}$$

where W is a n-dimensional Brownian motion, independent of a m-dimensional Brownian motion W^0 (representing the common noise) on a filtered probability space $(\Omega,\mathcal{F},\mathbb{F}=(\mathcal{F}_t)_{0\le t\le T},\mathbb{P})$, the control process α is $\mathbb{F}$-adapted valued in some Polish space A, and here $\mathbb{P}^0$ denotes the conditional law given W^0. The value function defined on $[0,T]\times\mathcal{P}_2(\mathbb{R}^d)$ by

$$v(t,\mu) \ = \ \inf_\alpha \mathbb{E}_{t,\mu}\Big[\int_t^T e^{-r(s-t)}f(X_s^\alpha,\mathbb{P}^0_{X_s^\alpha},\alpha_s)\mathrm{d}s + e^{-r(T-t)}g(X_T^\alpha,\mathbb{P}^0_{X_T^\alpha})\Big],$$

(here $\mathbb{E}_{t,\mu}[\cdot]$ is the conditional expectation given that the law at time t of X solution to (6.1.3) is equal to μ) is shown to satisfy the Bellman equation (6.1.1)-(6.1.2) (see [BFY13], [CP19], [DPT19]) with $G(\mu)=\int g(x,\mu)\mu(dx)$, σ, σ_0 as in (6.1.3) and

$$H(t,x,\mu,y,z) = \ -ry + \inf_{a\in A}\big[\beta(t,x,\mu,a).z + f(x,\mu,a)\big]. \tag{6.1.4}$$

We now consider a finite-dimensional approximation of the PDE (6.1.1)-(6.1.2) in the Wasserstein space. This can be derived formally by looking at the PDE for μ to averages of Dirac masses, and it turns out that the corresponding PDE takes the form

$$\begin{cases} \partial_t v^N + \frac{1}{N}\sum_{i=1}^N H\big(t, x_i, \bar{\mu}(\boldsymbol{x}), v^N, N D_{x_i} v^N\big) + \frac{1}{2}\mathrm{tr}(\Sigma_N(t,\boldsymbol{x}) D^2_{\boldsymbol{x}} v^N) \;=\; 0, \text{ on } [0,T)\times(\mathbb{R}^d)^N \\ v^N(T,\boldsymbol{x}) \;=\; G\big(\bar{\mu}(\boldsymbol{x})\big), \quad \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N, \end{cases} \tag{6.1.5}$$

where $\bar{\mu}(.)$ is the empirical measure function defined by $\bar{\mu}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^N \delta_{x_i}$, for any $\boldsymbol{x} = (x_1,\dots,x_N)$, $N \in \mathbb{N}^*$, and $\Sigma_N = (\Sigma_N^{ij})_{i,j\in[\![1,N]\!]}$ is the $\mathbb{R}^{Nd\times Nd}$-valued function with block matrices $\Sigma_N^{ij}(t,\boldsymbol{x}) = \sigma(t,x_i,\bar{\mu}(\boldsymbol{x}))\sigma^\intercal(t,x_j,\bar{\mu}(\boldsymbol{x}))\delta_{ij} + \sigma_0(t,x_i,\bar{\mu}(\boldsymbol{x}))\sigma_0^\intercal(t,x_j,\bar{\mu}(\boldsymbol{x})) \in \mathbb{R}^{d\times d}$. In the special case where H has the form (6.1.4), we notice that (6.1.5) is the Bellman equation for the N-cooperative problem, whose convergence to the mean-field control problem has been studied in [Lac17], [CD18b], [LT19; LT20], when $\sigma_0 \equiv 0$ (no common noise), and recently by [Dje20] in the common noise case. We point out that these works do not consider the same master equation. In particular their master equation is stated on $[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)$ and is linear in $\partial_\mu u$ whereas we allow a non-linear dependence in this derivative. Moreover our master equation is in expectation form. In [LT20] the master equation is approached by a system of N coupled PDEs on $[0,T]\times(\mathbb{R}^d)^N$ whereas we consider a single approximating PDE on $[0,T]\times(\mathbb{R}^d)^N$. For more general Hamiltonian functions H, it has been recently proved in [GMS21] that the sequence of viscosity solutions $(v^N)_N$ to (6.1.5) converge locally uniformly to the viscosity solution v to (6.1.1) when $\sigma = 0$ and σ_0 does not depend on space and measure arguments. For a detailed comparison between this work and ours, we refer to Remark 6.2.4.

In this paper, we adopt a probabilistic approach by considering a backward stochastic differential equation (BSDE) representation for the finite-dimensional PDE (6.1.5) according to the classical work [PP90]. The solution $(Y^N, \boldsymbol{Z}^N = (Z^{i,N})_{1\le i\le N})$ to this BSDE is written with an underlying forward particle system $\boldsymbol{X}^N = (X^{i,N})_{1\le i\le N}$ of a McKean-Vlasov SDE, and connected to the PDE (6.1.5) via the Feynman-Kac formula: $Y_t^N = v^N(t,\boldsymbol{X}_t^N)$, $Z_t^{i,N} = D_{x_i}v^N(t,\boldsymbol{X}_t^{i,N})$, $0\le t\le T$. By using BSDE techniques, our main contribution is to show a rate of convergence of order $1/N$ of $|Y_t^N - v(t,\bar{\mu}(\boldsymbol{X}_t^N))|$, and also of $|NZ_t^{i,N} - \partial_\mu v(t,\bar{\mu}(\boldsymbol{X}_t^N))(X_t^{i,N})|^2$, $i=1,\dots,N$, for suitable norms, and under some regularity conditions on v (see Theorem 6.2.1 and Theorem 6.2.2). This rate of convergence on the particles approximation of v and its L-derivative is new to the best of our knowledge. We point out that classical BSDE arguments for proving the rate of convergence do not apply directly due to the presence of the factor N in front of $D_{x_i}v^N$ in the generator H, and we rather use linearization arguments and change of probability measures to overcome these issues. Another issue is due to the fact that the BSDE dimension $d\times N$ is exploding with the number of particles therefore we have to track down the influence of the dimension in the estimations, whereas classical BSDE works usually consider a fixed dimension d which is incorporated into constants.

The outline of the paper is organized as follows. In Section 6.2, we formulate the particle approximation of the PDE and its BSDE representation, and state the rate of convergence for v and its L-derivative. Section 6.3 is devoted to the proof of these results.

6.2 Particles approximation of Wasserstein PDEs

The formal derivation of the finite-dimensional approximation PDE is obtained as follows. We look at the PDE (6.1.1)-(6.1.2) for $\mu = \bar{\mu}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^N \delta_{x_i} \in \mathcal{P}_2(\mathbb{R}^d)$, when $\boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]}$ runs over $(\mathbb{R}^d)^N$. By setting $\tilde{v}^N(t,\boldsymbol{x}) = v(t,\bar{\mu}(\boldsymbol{x}))$, and assuming that v is smooth, we have for all $(i,j)\in[\![1,N]\!]$ (see Proposition 5.35 and Proposition 5.91 in [CD18a]):

$$\begin{cases} D_{x_i}\tilde{v}^N(t,\boldsymbol{x}) \;=\; \frac{1}{N}\partial_\mu v(t,\bar{\mu}(\boldsymbol{x}))(x_i), \\ D^2_{x_ix_j}\tilde{v}^N(t,\boldsymbol{x}) \;=\; \frac{1}{N}\partial_x\partial_\mu v(t,\bar{\mu}(\boldsymbol{x}))(x_i)\delta_{ij} + \frac{1}{N^2}\partial^2_\mu v(t,\bar{\mu}(\boldsymbol{x}))(x_i,x_j). \end{cases} \tag{6.2.1}$$

By substituting into the PDE (6.1.1)-(6.1.2) for $\mu = \bar{\mu}(\boldsymbol{x})$, and using (6.2.1), we then see that $\tilde{v}^N$ satisfies the relation:

$$\partial_t \tilde{v}^N + \frac{1}{N}\sum_{i=1}^N H\big(t, x_i, \bar{\mu}(\boldsymbol{x}), \tilde{v}^N, N D_{x_i}\tilde{v}^N\big) \tag{6.2.2}$$
$$+ \frac{1}{2}\sum_{i=1}^N \mathrm{tr}\big[\big(\sigma\sigma^\intercal + \sigma_0\sigma_0^\intercal\big)(t, x_i, \bar{\mu}(\boldsymbol{x}))\big(D^2_{x_i}\tilde{v}^N - \frac{1}{N^2}\partial^2_\mu v(t, \bar{\mu}(\boldsymbol{x}))(x_i, x_i)\big)\big]$$
$$+ \frac{1}{2}\sum_{i\neq j\in[\![1,N]\!]} \mathrm{tr}\big(\sigma_0(t, x_i, \bar{\mu}(\boldsymbol{x}))\sigma_0^\intercal(t, x_j, \bar{\mu}(\boldsymbol{x}))D^2_{x_i x_j}\tilde{v}^N\big)$$
$$+ \frac{1}{2N^2}\sum_{i=1}^N \mathrm{tr}\big(\sigma_0\sigma_0^\intercal(t, x_i, \bar{\mu}(\boldsymbol{x}))\partial^2_\mu v(t, \bar{\mu}(\boldsymbol{x}))(x_i, x_i)\big) = 0$$

for $(t, \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]}) \in [0,T)\times(\mathbb{R}^d)^N$, together with the terminal condition $\tilde{v}^N(t, \boldsymbol{x}) = G(\bar{\mu}(\boldsymbol{x}))$. By neglecting the terms $\partial^2_\mu v/N^2$ in the above relation, we obtain the PDE (6.1.5) for $v^N \simeq \tilde{v}^N$. The purpose of this section is to rigorously justify this approximation and state a rate of convergence for v^N towards v, as well as a convergence for their gradients.

6.2.1 Particles BSDE approximation

Let us introduce an arbitrary measurable $\mathbb{R}^d$-valued function b on $[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)$, and set B_N the $(\mathbb{R}^d)^N$-valued function defined on $[0,T]\times(\mathbb{R}^d)^N$ by $B_N(t,\boldsymbol{x}) = (b(t, x_i, \bar{\mu}(\boldsymbol{x}))_{i\in[\![1,N]\!]}$ for $(t, \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]}) \in [0,T)\times(\mathbb{R}^d)^N$. The finite-dimensional PDE (6.1.5) may then be written as

$$\begin{cases} \partial_t v^N + B_N(t,\boldsymbol{x}).D_{\boldsymbol{x}} v^N + \frac{1}{2}\mathrm{tr}\big(\Sigma_N(t,\boldsymbol{x})D^2_{\boldsymbol{x}} v^N\big) \\ \qquad + \frac{1}{N}\sum_{i=1}^N H_b\big(t, x_i, \bar{\mu}(\boldsymbol{x}), v^N, N D_{x_i} v^N\big) = 0, \quad \text{on } [0,T)\times(\mathbb{R}^d)^N, \\ v^N(T,\boldsymbol{x}) = G\big(\bar{\mu}(\boldsymbol{x})\big), \quad \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N, \end{cases} \tag{6.2.3}$$

where $H_b(t,x,\mu,y,z) := H(t,x,\mu,y,z) - b(t,x,\mu).z$. For error analysis purpose, the function b can be simply taken to be zero. The introduction of the function b is actually motivated by numerical purpose. It corresponds indeed to the drift of training simulations for approximating the function v^N, notably by machine learning methods, and should be chosen for suitable exploration of the state space, see a detailed discussion in our companion paper [Ger+22]. In this paper, we fix an arbitrary function b (satisfying Lipschitz condition to be precised later).

Following [PP90], it is well-known that the semi-linear PDE (6.2.3) admits a probabilistic representation in terms of forward backward SDE. The forward component is defined by the process $\boldsymbol{X}^N = (X^{i,N})_{i\in[\![1,N]\!]}$ valued in $(\mathbb{R}^d)^N$, solution to the SDE:

$$\mathrm{d}\boldsymbol{X}^N_t = B_N(t, \boldsymbol{X}^N_t)\mathrm{d}t + \sigma_N(t, \boldsymbol{X}^N_t)\mathrm{d}\boldsymbol{W}_t + \boldsymbol{\sigma}_0(t, \boldsymbol{X}^N_t)\mathrm{d}W^0_t \tag{6.2.4}$$

where σ_N is the block diagonal matrix with block diagonals $\sigma^{ii}_N(t,\boldsymbol{x}) = \sigma(t, x_i, \bar{\mu}(\boldsymbol{x}))$, $\boldsymbol{\sigma}_0 = (\sigma^i_0)_{i\in[\![1,N]\!]}$ is the $(\mathbb{R}^{d\times m})^N$-valued function with $\boldsymbol{\sigma}^i_0(t,\boldsymbol{x}) = \sigma_0(t, x_i, \bar{\mu}(\boldsymbol{x}))$, for $\boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]}$, $\boldsymbol{W} = (W^1, \ldots, W^N)$ where W^i, $i = 1, \ldots, N$, are independent n-dimensional Brownian motions, independent of a m-dimensional Brownian motion W^0 on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_{0\le t\le T}, \mathbb{P})$. Notice that $\Sigma_N = \sigma_N\sigma_N^\intercal + \boldsymbol{\sigma}_0\boldsymbol{\sigma}_0^\intercal$, and $\boldsymbol{X}^N$ is the particles system of the McKean-Vlasov SDE:

$$\mathrm{d}X_t = b(t, X_t, \mathbb{P}_{X_t})\mathrm{d}t + \sigma(t, X_t, \mathbb{P}^0_{X_t})\mathrm{d}W_t + \sigma_0(t, X_t, \mathbb{P}^0_{X_t})\mathrm{d}W^0_t, \tag{6.2.5}$$

where W is an n-dimensional Brownian motion independent of W^0. The backward component is defined by the pair process $(Y^N, \boldsymbol{Z}^N = (Z^{i,N})_{i\in[\![1,N]\!]})$ valued in $\mathbb{R}\times(\mathbb{R}^d)^N$, solution to

$$\begin{aligned} Y_t^N = \; & G\big(\bar\mu(\boldsymbol{X}_T^N)\big) + \frac{1}{N}\sum_{i=1}^N \int_t^T H_b(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N), Y_s^N, NZ_s^{i,N})\mathrm{d}s \\ & - \sum_{i=1}^N \int_t^T (Z_s^{i,N})^\intercal \sigma\big(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N)\big)\mathrm{d}W_s^i, \\ & - \sum_{i=1}^N \int_t^T (Z_s^{i,N})^\intercal \sigma_0\big(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N)\big)\mathrm{d}W_s^0, \qquad 0\le t\le T. \end{aligned} \tag{6.2.6}$$

We shall assume that the measurable functions $(t,x,\mu)\mapsto b(t,x,\mu)$, $\sigma(t,x,\mu)$ satisfy a Lipschitz condition in $(x,\mu)\in\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)$ uniformly w.r.t. $t\in[0,T]$, which ensures the existence and uniqueness of a strong solution $\boldsymbol{X}^N\in\mathcal{S}_{\mathbb{F}}^2((\mathbb{R}^d)^N)$ to (6.2.4) given an initial condition. Here, $\mathcal{S}_{\mathbb{F}}^2(\mathbb{R}^q)$ is the set of $\mathbb{F}$-adapted process $(V_t)_t$ valued in $\mathbb{R}^q$ s.t. $\mathbb{E}\big[\sup_{0\le t\le T}|V_t|^2\big]<\infty$, ($|.|$ is the Euclidian norm on $\mathbb{R}^q$, and for a matrix M, we choose the Frobenius norm $|M|=\sqrt{\mathrm{tr}(MM^\intercal)}$) and the Wasserstein space $\mathcal{P}_2(\mathbb{R}^d)$ is endowed with the Wasserstein distance

$$\mathcal{W}_2(\mu,\mu') = \Big(\inf\big\{\mathbb{E}|\xi-\xi'|^2 : \xi\sim\mu, \xi'\sim\mu'\big\}\Big)^{\frac12},$$

and we set $\|\mu\|_2 := \big(\int_{\mathbb{R}^d}|x|^2\,\mu(\mathrm{d}x)\big)^{\frac12}$ for $\mu\in\mathcal{P}_2(\mathbb{R}^d)$. Assuming also that the measurable function $(t,x,\mu,y,z)\mapsto H_b(t,x,\mu,y,z)$ is Lipschitz in $(y,z)\in\mathbb{R}\times\mathbb{R}^d$ uniformly with respect to $(t,x,\mu)\in[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)$, and the measurable function G satisfies a quadratic growth condition on $\mathcal{P}_2(\mathbb{R}^d)$, we have the existence and uniqueness of a solution $(Y^N,\boldsymbol{Z}^N=(Z^{i,N})_{i\in[\![1,N]\!]})\in\mathcal{S}_{\mathbb{F}}^2(\mathbb{R})\times\mathbb{H}_{\mathbb{F}}^2((\mathbb{R}^d)^N)$ to (6.2.6), and the connection with the PDE (6.2.3) (satisfied in general in the viscosity sense) via the (non linear) Feynman-Kac formula:

$$Y_t^N = v^N(t,\boldsymbol{X}_t^N), \quad \text{and} \quad Z_t^{i,N} = D_{x_i}v^N(t,\boldsymbol{X}_t^N),\;\; i=1,\dots,N,\;\; 0\le t\le T, \tag{6.2.7}$$

(when v^N is smooth for the last relation). Here, $\mathbb{H}_{\mathbb{F}}^2(\mathbb{R}^q)$ is the set of $\mathbb{F}$-adapted process $(V_t)_t$ valued in $\mathbb{R}^q$ s.t. $\mathbb{E}\big[\int_0^T|V_t|^2\mathrm{d}t\big]<\infty$.

6.2.2 Main results

We aim to analyze the particles approximation error on the solution v to the PDE (6.1.1), and its L-derivative $\partial_\mu v$ by considering the pathwise error on v:

$$\mathcal{E}_N^{\mathrm{y}} := \sup_{0\le t\le T}\big|Y_t^N - v(t,\bar\mu(\boldsymbol{X}_t^N))\big|,$$

and the L^2-error on its L-derivative

$$\big\|\mathcal{E}_N^{\mathrm{z}}\big\|_2 := \frac{1}{N}\sum_{i=1}^N\Big(\int_0^T \mathbb{E}\big|NZ_t^{i,N} - \partial_\mu v(t,\bar\mu(\boldsymbol{X}_t^N))(X_t^{i,N})\big|^2\mathrm{d}t\Big)^{\frac12},$$

where the initial conditions of the particles system, $X_0^{i,N}$, $i=1,\dots,N$, are i.i.d. with distribution μ_0.

Here, it is assumed that we have the existence and uniqueness of a classical solution v to the PDE (6.1.1)-(6.1.2). More precisely, we make the following assumption:

Assumption 6.2.1 (Smooth solution to the Master Bellman PDE)**.** *There exists a unique solution v to to* (6.1.1)*, which lies in $C_b^{1,2}([0,T]\times\mathcal{P}_2(\mathbb{R}^d))$ that is:*

- *$v(.,\mu) \in C^1([0,T])$, and continuous on $[0,T]$, for any $\mu \in \mathcal{P}_2(\mathbb{R}^d)$,*
- *$v(t,.)$ is fully C^2 on $\mathcal{P}_2(\mathbb{R}^d)$, for any $t \in [0,T]$ in the sense that: $(x,\mu) \in \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d) \mapsto \partial_\mu v(t,\mu)(x) \in \mathbb{R}^d$, $(x,\mu) \in \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d) \mapsto \partial_x \partial_\mu v(t,\mu)(x) \in \mathbb{M}^d$, and $(x,x',\mu) \in \mathbb{R}^d \times \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d) \mapsto \partial^2_\mu v(t,\mu)(x,x') \in \mathbb{M}^d$, are well-defined and jointly continuous,*
- *there exists some constant $L > 0$ s.t. for all $(t,x,\mu) \in [0,T] \times \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$*

$$\big|\partial_\mu v(t,\mu)(x)\big| \le L\big(1 + |x| + \|\mu\|_2\big), \quad \big|\partial^2_\mu v(t,\mu)(x,x)\big| \le L.$$

The existence of classical solutions to mean-field PDE in Wasserstein space is a challenging problem, and beyond the scope of this paper. We refer to [Buc+17], [CCD15], [SZ19], [WZ20] for conditions ensuring regularity results of some Master PDEs. Notice also that linear-quadratic mean-field control problems have explicit smooth solutions as in Assumption 6.2.1, see e.g. [PW18].

We also make some rather standard assumptions on the coefficients of the forward backward SDE:

Assumption 6.2.2 (Lipschitz condition on the coefficients of the forward backward SDE)**.**

(i) The drift and volatility coefficients b, σ, σ_0 are Lipschitz: there exist positive constants $[b]$, $[\sigma]$, and $[\sigma_0]$ s.t. for all $t \in [0,T]$, $x, x' \in \mathbb{R}^d$, $\mu, \mu' \in \mathcal{P}_2(\mathbb{R}^d)$,

$$\begin{aligned}
|b(t,x,\mu) - b(t,x',\mu')| &\le [b]\big(|x-x'| + \mathcal{W}_2(\mu,\mu')\big)\\
|\sigma(t,x,\mu) - \sigma(t,x',\mu')| &\le [\sigma]\big(|x-x'| + \mathcal{W}_2(\mu,\mu')\big)\\
|\sigma_0(t,x,\mu) - \sigma_0(t,x',\mu')| &\le [\sigma_0]\big(|x-x'| + \mathcal{W}_2(\mu,\mu')\big).
\end{aligned}$$

(ii) For all $(t,x,\mu) \in [0,T] \times \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$, $\Sigma(t,x,\mu) := \sigma\sigma^\intercal(t,x,\mu)$ is invertible, and the function σ, and its pseudo-inverse $\sigma^+ := \sigma^\intercal \Sigma^{-1}$ are bounded.

(iii) $\mu_0 \in \mathcal{P}_{4q}(\mathbb{R}^d)$ for some $q > 1$, i.e., $\|\mu_0\|_{4q} := \big(\int |x|^{4q} \mu_0(\mathrm{d}x)\big)^{\frac{1}{4q}} < \infty$, and

$$\int_0^T |b(t,0,\delta_0)|^{4q} + |\sigma(t,0,\delta_0)|^{4q} + |\sigma_0(t,0,\delta_0)|^{4q} \,\mathrm{d}t < \infty.$$

(iv) The driver H_b satisfies the Lipschitz condition: there exist positive constants $[H_b]_1$ and $[H_b]_2$ s.t. for all $t \in [0,T]$, $x, x' \in \mathbb{R}^d$, $\mu, \mu' \in \mathcal{P}_2(\mathbb{R}^d)$, $y, y' \in \mathbb{R}$, $z, z' \in \mathbb{R}^d$,

$$\begin{aligned}
|H_b(t,x,\mu,y,z) - H_b(t,x,\mu,y',z')| &\le [H_b]_1(|y-y'| + |z-z'|)\\
|H_b(t,x,\mu,y,z) - H_b(t,x',\mu',y,z)| &\le [H_b]_2\big(1 + |x| + |x'| + \|\mu\|_2 + \|\mu'\|_2\big)\\
&\qquad \big(|x-x'| + \mathcal{W}_2(\mu,\mu')\big).
\end{aligned}$$

(v) The terminal condition satisfies the (locally) Lipschitz condition: there exists some positive constant $[G]$ s.t. for all $\mu, \mu' \in \mathcal{P}_2(\mathbb{R}^d)$

$$|G(\mu) - G(\mu')| \le [G]\big(\|\mu\|_2 + \|\mu'\|_2\big)\mathcal{W}_2(\mu,\mu').$$

In order to have a convergence result for the first order Lions derivative we have to make a stronger assumption.

Assumption 6.2.3.

(i) The function H_b is in the form:

$$H_b(t,x,\mu,y,z) = H_1(t,x,\mu,y) + H_2(t,\mu,y).z,$$

where $H_1 : [0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)\times\mathbb{R}\mapsto\mathbb{R}$ verifies for all $t\in[0,T]$, $x,x'\in\mathbb{R}^d$, $\mu,\mu'\in\mathcal{P}_2(\mathbb{R}^d)$, $y,y'\in\mathbb{R}$, $z,z'\in\mathbb{R}^d$,

$$\begin{aligned}|H_1(t,x,\mu,y)-H_1(t,x,\mu,y')| &\le [H_1]_1|y-y'|\\ |H_1(t,x,\mu,y)-H_1(t,x',\mu',y)| &\le [H_1]_2\big(1+|x|+|x'|+\|\mu\|_2+\|\mu'\|_2\big)\\ &\quad\big(|x-x'|+\mathcal{W}_2(\mu,\mu')\big),\end{aligned}$$

and $H_2 : [0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)\times\mathbb{R}\mapsto\mathbb{R}^d$ is bounded and verifies for all $t\in[0,T]$, $x,x'\in\mathbb{R}^d$, $\mu,\mu'\in\mathcal{P}_2(\mathbb{R}^d)$, $y,y'\in\mathbb{R}$, $z,z'\in\mathbb{R}^d$,

$$\begin{aligned}|H_2(t,x,\mu,y)-H_2(t,x,\mu,y')| &\le [H_2]_1|y-y'|\\ |H_2(t,x,\mu,y)-H_2(t,x',\mu',y)| &\le [H_2]_2\big(1+|x|+|x'|+\|\mu\|_2+\|\mu'\|_2\big)\\ &\quad\big(|x-x'|+\mathcal{W}_2(\mu,\mu')\big).\end{aligned}$$

(ii) σ_0 is uniformly elliptic and does not depend on x, namely there exists $c_0>0$ such that for all $t\in[0,T]$, $\mu\in\mathcal{P}_2(\mathbb{R}^d)$, $z\in\mathbb{R}^d$

$$z^\intercal\sigma_0(t,\mu)\sigma_0^\intercal(t,\mu)z \;\ge\; c_0|z|^2.$$

(iii) There exists some constant $L>0$ s.t. for all $(t,x,\mu)\in[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)$

$$\big|\partial_\mu v(t,\mu)(x)\big| \le\; L.$$

Remark 6.2.1. *The Lipschitz condition on b,σ in Assumption 6.2.2(i) implies that the functions $\boldsymbol{x}\in(\mathbb{R}^d)^N\mapsto B_N(t,\boldsymbol{x})$, resp. $\sigma_N(t,\boldsymbol{x})$ and $\boldsymbol{\sigma}_0(t,\boldsymbol{x})$, defined in (6.2.4), are Lipschitz (with Lipschitz constant $2[b]$, resp. $2[\sigma]$ and $2[\sigma_0]$). Indeed, we have*

$$\begin{aligned}|B_N(t,\boldsymbol{x})-B_N(t,\boldsymbol{x}')|^2 &= \sum_{i=1}^N |b(t,x_i,\bar\mu(\boldsymbol{x}))-b(t,x_i,\bar\mu(\boldsymbol{x}'))|^2\\ &\le 2[b]^2\sum_{i=1}^N(|x_i-x_i'|^2+\mathcal{W}_2(\bar\mu(\boldsymbol{x}),\bar\mu(\boldsymbol{x}'))^2)\\ &\le 2[b]^2(|\boldsymbol{x}-\boldsymbol{x}'|^2+\sum_{i=1}^N\frac{1}{N}|\boldsymbol{x}-\boldsymbol{x}'|^2) \;=\; 4[b]^2|\boldsymbol{x}-\boldsymbol{x}'|^2,\end{aligned}$$

for $\boldsymbol{x}=(x_i)_{i\in[\![1,N]\!]}$, and similarly for σ_N and $\boldsymbol{\sigma}_0$.

This yields the existence and uniqueness of a solution $\boldsymbol{X}^N=(X^{i,N})_{i\in[\![1,N]\!]}$ to (6.2.4) given initial conditions. Moreover, under Assumption 6.2.2(iii), we have the standard estimate:

$$\mathbb{E}\big[\sup_{0\le t\le T}|\boldsymbol{X}_t^N|^{4q}\big] \le\; C\big(1+\|\mu_0\|_{4q}^{4q}\big) \;<\; \infty,\quad i=1,\dots,N, \tag{6.2.8}$$

for some constant C (possibly depending on N). The Lipschitz condition on H_b w.r.t. (y,z) in Assumption 6.2.2(iv), and the quadratic growth condition on G from Assumption 6.2.2(v) gives the existence and uniqueness of a solution $(Y^N,\boldsymbol{Z}^N=(Z^{i,N})_{i\in[\![1,N]\!]})\in\mathcal{S}_{\mathbb{F}}^2(\mathbb{R})\times\mathbb{H}_{\mathbb{F}}^2((\mathbb{R}^d)^N)$ to

(6.2.6). *Moreover, by Assumption 6.2.2(iv)(v), we see that*

$$\begin{aligned}
&\big|\frac{1}{N}\sum_{i=1}^N H_b(t,x_i,\bar\mu(\boldsymbol{x}),y,z_i)-\frac{1}{N}\sum_{i=1}^N H_b(t,x_i',\bar\mu(\boldsymbol{x}'),y,z_i)\big|\\
&\le [H_b]_2\frac{1}{N}\sum_{i=1}^N\big(1+|x_i|+|x_i'|+\frac{1}{\sqrt{N}}(|\boldsymbol{x}|+|\boldsymbol{x}'|)\big)\big(|x_i-x_i'|+\frac{1}{\sqrt{N}}|\boldsymbol{x}-\boldsymbol{x}'|\big)\\
&\le 4[H_b]_2\big(1+|\boldsymbol{x}|+|\boldsymbol{x}'|\big)|\boldsymbol{x}-\boldsymbol{x}'|\\
\big|G(\bar\mu(\boldsymbol{x}))-G(\bar\mu(\boldsymbol{x}'))\big|&\le \frac{[G]}{N}(|\boldsymbol{x}|+|\boldsymbol{x}'|)|\boldsymbol{x}-\boldsymbol{x}'|,
\end{aligned}$$

for all $x,x'\in\mathbb{R}^d$, $\boldsymbol{x}=(x_i)_{i\in[\![1,N]\!]}$, $\boldsymbol{x}'=(x_i')_{i\in[\![1,N]\!]}\in(\mathbb{R}^d)^N$, which yields by standard stability results for BSDE (see e.g. Theorems 4.2.1 and 5.2.1 in [Zha17]) that the function v^N in (6.2.7) *inherits the locally Lipschitz condition:*

$$\big|v^N(t,\boldsymbol{x})-v^N(t,\boldsymbol{x}')\big|\le C(1+|\boldsymbol{x}|+|\boldsymbol{x}'|)|\boldsymbol{x}-\boldsymbol{x}'|,\quad\forall\boldsymbol{x},\boldsymbol{x}'\in(\mathbb{R}^d)^N,$$

for some constant C (possibly depending on N). This implies

$$|Z_t^{i,N}|\le C\big(1+|\boldsymbol{X}_t^N|\big),\quad 0\le t\le T,\quad i=1,\dots,N,\tag{6.2.9}$$

(this is clear when v^N is smooth, and otherwise obtained by a mollifying argument as in Theorem 5.2.2 in [Zha17]).

Remark 6.2.2. *Assumption 6.2.1 is verified in the case of linear quadratic control problems for which explicit smooth solutions are found in [PW18; PW17] respectively without and with common noise. These papers prove that the second order Lions derivative ∂_μ^2 is a continuous function of time which does not depend on the μ,x arguments hence is bounded whereas ∂_μ is affine in both the state and the first moment of the measure thus satisfies linear growth. Notice that in general, Assumptions 6.2.2 and 6.2.3 are not satisfied due to the quadratic nature of H_b in the z. However, in the uncontrolled case*

$$\begin{aligned}
v(t,\mu)&=\mathbb{E}_{t,\mu}\Big[\int_t^T\Big(X_t^\top A(t)X_t+\mathbb{E}[X_t]^\top B(t)\mathbb{E}[X_t]+C(t)X_t+D(t)\mathbb{E}[X_t]\Big)dt\\
&\qquad+X_T^\top EX_T+\mathbb{E}[X_T]^\top F\mathbb{E}[X_T]+GX_T+H\mathbb{E}[X_T]\Big]\\
\mathrm{d}X_t&=(b_0(t)+b_1(t)X_t+b_2(t)\mathbb{E}[X_t])\,\mathrm{d}t+\sigma(t)\,\mathrm{d}W_t,
\end{aligned}$$

we see that v is a solution to the linear PDE

$$\begin{cases}
\partial_t v+\int_{\mathbb{R}^d}\big[x^\top A(t)x+\bar\mu^\top B(t)\bar\mu+C(t)x+D(t)\bar\mu\\
+(b_0(t)+b_1(t)x+b_2(t)\bar\mu)\partial_\mu v(t,\mu)(x)\\
+\frac12\mathrm{tr}\big((\sigma\sigma^\intercal)(t)\partial_x\partial_\mu v(t,\mu)(x)\big)\big]\mu(\mathrm{d}x)=0, & (t,\mu)\in[0,T)\times\mathcal{P}_2(\mathbb{R}^d),\\
v(T,\mu)=E\ \mathrm{Var}(\mu)+\bar\mu^\top(E+F)\bar\mu+(G+H)\bar\mu, & \mu\in\mathcal{P}_2(\mathbb{R}^d),
\end{cases}$$

where $\bar\mu=\int_{\mathbb{R}^d}x\mu(\mathrm{d}x)$, $\mathrm{Var}(\mu)=\int_{\mathbb{R}^d}x^2\mu(\mathrm{d}x)-\bar\mu^2$. In that case both assumptions 6.2.1 and 6.2.2 are satisfied.

Theorem 6.2.1. *Under Assumptions 6.2.1 and 6.2.2, we have $\mathbb{P}$-almost surely*

$$\mathcal{E}_N^{\mathrm{y}}\le\frac{C_y}{N},$$

where $C_y=\frac{T}{2}e^{[H_b]_1T}L\|\sigma\|_\infty^2$, with $\|\sigma\|_\infty=\sup_{(t,x,\mu)\in[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)}|\sigma(t,x,\mu)|$.

Theorem 6.2.2. *Under Assumptions 6.2.1, 6.2.2 and 6.2.3, we have*

$$\big\|\mathcal{E}_N^{\mathrm{z}}\big\|_2 \le \frac{C_z}{N^{\frac{1}{2}}},$$

where $C_z = \|\sigma^+\|_\infty \sqrt{2T([H_1]_1 + [H_2]_1 L)\bar{C}_y^2 + \bar{C}_y T\|\sigma\|_\infty^2 L + \frac{2}{c_0}\bar{C}_y^2 T\|H_2\|_\infty^2}$ *and* $\bar{C}_y = \frac{T}{2} e^{([H_1]_1+[H_2]_1 L)T} L\|\sigma\|_\infty^2$.

Remark 6.2.3. *Let us consider the global weak errors on v and its L-derivative $\partial_\mu v$ along the limiting McKean-Vlasov SDE, and defined by*

$$E_N^{\mathrm{y}} := \sup_{0\le t\le T} \big|\mathbb{E}[Y_t^N] - \mathbb{E}[v(t, \mathbb{P}^0_{X_t})]\big|$$
$$E_N^{\mathrm{z}} := \frac{1}{N}\sum_{i=1}^N \Big(\int_0^T \Big|\mathbb{E}\big[N Z_t^{i,N}\big] - \mathbb{E}\big[\partial_\mu v(t, \mathbb{P}^0_{X_t^i})(X_t^i)\big]\Big|^2 \mathrm{d}t\Big)^{\frac{1}{2}},$$

where X^i has the same law than X, and with McKean-Vlasov dynamics as in (6.2.5) *but driven by W^i, $i = 1, \dots, N$. Then, they can be decomposed as*

$$E_N^{\mathrm{y}} \le \mathbb{E}\big[\mathcal{E}_N^{\mathrm{y}}\big] + \tilde{E}_N^{\mathrm{y}}, \quad E_N^{\mathrm{z}} \le \big\|\mathcal{E}_N^{\mathrm{z}}\big\|_2 + \tilde{E}_N^{\mathrm{z}},$$

where $\tilde{E}_N^{\mathrm{y}}$, $\tilde{E}_N^{\mathrm{z}}$ are the (weak) propagation of chaos errors defined by

$$\tilde{E}_N^{\mathrm{y}} := \sup_{0\le t\le T} \big|\mathbb{E}[v(t, \bar{\mu}(\boldsymbol{X}_t^N))] - \mathbb{E}[v(t, \mathbb{P}^0_{X_t})]\big|$$
$$\tilde{E}_N^{\mathrm{z}} := \frac{1}{N}\sum_{i=1}^N \Big(\int_0^T \Big|\mathbb{E}\big[\partial_\mu v(t, \bar{\mu}(\boldsymbol{X}_t^N))(X_t^{i,N})\big] - \mathbb{E}\big[\partial_\mu v(t, \mathbb{P}^0_{X_t^i})(X_t^i)\big]\Big|^2 \mathrm{d}t\Big)^{\frac{1}{2}},$$

From the conditional propagation of chaos result, which states that for any fixed $k \ge 1$, the law of $(X_t^{i,N})_{t\in[0,T]}^{i\in[\![1,k]\!]}$ converges toward the conditional law of $(X_t^i)_{t\in[0,T]}^{i\in[\![1,k]\!]}$, as N goes to infinity, we deduce that $\tilde{E}_N^{\mathrm{y}}, \tilde{E}_N^{\mathrm{z}} \to 0$. Furthermore, under additional assumptions on v, we can obtain a rate of convergence. Namely, if $v(t,.)$ is Lipschitz uniformly in $t \in [0,T]$, with Lipschitz constant $[v]$, we have

$$\tilde{E}_N^{\mathrm{y}} \le [v] \sup_{0\le t\le T} \Big(\mathbb{E}[\mathcal{W}_2(\bar{\mu}(\boldsymbol{X}_t^N), \mathbb{P}^0_{X_t})^2]\Big)^{\frac{1}{2}} = O\Big(N^{-\frac{1}{\max(d,4)}}\sqrt{1+\ln(N)1_{d=4}}\Big),$$
$$\textit{hence} \quad E_N^{\mathrm{y}} = O\Big(N^{-\frac{1}{\max(d,4)}}\sqrt{1+\ln(N)1_{d=4}}\Big), \tag{6.2.10}$$

where we use the rate of convergence of empirical measures in Wasserstein distance stated in [FG15] (see also Theorem 2.12 in [CD18b]), and since we have the standard estimate $\mathbb{E}[\sup_{0\le t\le T}|X_t|^{4q}] \le C(1+\|\mu_0\|_{2q}^{4q})$ by Assumption 6.2.2(iii). The rate of convergence in (6.2.10) *is consistent with the one found in Theorem 6.17 [CD18b] for mean-field control problem. Furthermore, if the function $\partial_\mu v(t,.)(.)$ is Lipschitz in (x,μ) uniformly in t, then by the rate of convergence in Theorem 2.12 of [CD18b], we have*

$$\tilde{E}_N^{\mathrm{z}} = O\big(N^{-\frac{1}{\max(d,4)}}\big(1+\ln(N)1_{d=4}\big)\big), \quad \textit{hence} \quad E_N^{\mathrm{z}} = O\big(N^{-\frac{1}{\max(d,4)}}\big(1+\ln(N)1_{d=4}\big)\big).$$

Remark 6.2.4 (Comparison with [GMS21])**.** *In the related paper [GMS21] the authors consider a pure common noise case, that is $\sigma = 0$ and restrict themselves to $\sigma_0(t, x_i, \bar{\mu}(\boldsymbol{x})) = \kappa I_d$ for $\kappa \in \mathbb{R}$. If we consider these assumptions in our smooth setting, we directly see that $\Delta Y_s^N = 0$ and $\Delta Z_s^N = 0$ $\mathbb{P}$ a.s. Indeed by* (6.2.6) *and* (6.3.2) *we notice that (Y_t^N, Z_t^N) and*

$$(\tilde{Y}_t^N := v(t, \bar{\mu}(\boldsymbol{X}_t^N)), \{\tilde{Z}_t^{i,N} := \frac{1}{N}\partial_\mu v(t, \bar{\mu}(\boldsymbol{X}_t^N))(X_t^{i,N}), \quad i = 1, \dots, N\}),$$

solve the same BSDE therefore by existence and pathwise uniqueness for Lipschitz BSDEs the result follows. Moreover, [GMS21] does not allow H to depend on y. Our approach allows to extend their findings to the case of idiosyncratic noises and in contrast to them we are able to choose a state-dependent volatility coefficient. Moreover we provide a convergence rate for the solution. However, we have to assume existence of a smooth solution for the master equation which is a restrictive assumption.

6.3 Proof of main results

6.3.1 Proof of Theorem 6.2.1

Step 1. Under the smoothness condition on v in Assumption 6.2.1, one can apply the standard Itô's formula in $(\mathbb{R}^d)^N$ to the process $\tilde{v}^N(t, \boldsymbol{X}_t^N) = v(t, \bar{\mu}(\boldsymbol{X}_t^N))$, and get

$$\begin{aligned}\tilde{v}^N(t, \boldsymbol{X}_t^N) = \; & \tilde{v}^N(T, \boldsymbol{X}_T^N) - \int_t^T \partial_t \tilde{v}^N(s, \boldsymbol{X}_s^N)\mathrm{d}s \\ & - \int_t^T B_N(s, \boldsymbol{X}_s^N).D_{\boldsymbol{x}}\tilde{v}^N(s, \boldsymbol{X}_s^N) + \frac{1}{2}\mathrm{tr}\big(\Sigma_N(s, \boldsymbol{X}_s^N)D_{\boldsymbol{x}}^2\tilde{v}^N(s, \boldsymbol{X}_s^N)\big)\mathrm{d}s \\ & - \sum_{i=1}^N \int_t^T \big(D_{x_i}\tilde{v}^N(s, \boldsymbol{X}_s^N)\big)^\intercal \sigma(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N))\mathrm{d}W_s^i \\ & - \sum_{i=1}^N \int_t^T \big(D_{x_i}\tilde{v}^N(s, \boldsymbol{X}_s^N)\big)^\intercal \sigma_0(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N))\mathrm{d}W_s^0,\end{aligned} \tag{6.3.1}$$

Now, by setting (recall (6.2.1)):

$$\begin{aligned}\tilde{Y}_t^N &:= v(t, \bar{\mu}(\boldsymbol{X}_t^N)) = \tilde{v}^N(t, \boldsymbol{X}_t^N), \\ \tilde{Z}_t^{i,N} &:= \frac{1}{N}\partial_\mu v(t, \bar{\mu}(\boldsymbol{X}_t^N))(X_t^{i,N}) = D_{x_i}\tilde{v}^N(t, \boldsymbol{X}_t^N), \quad i = 1, \dots, N, \quad 0 \le t \le T,\end{aligned}$$

and using the relation (6.2.2) satisfied by $\tilde{v}^N$ into (6.3.1), we have for all $0 \le t \le T$,

$$\begin{aligned}\tilde{Y}_t^N = \; & G\big(\bar{\mu}(\boldsymbol{X}_T^N)\big) + \frac{1}{N}\sum_{i=1}^N \int_t^T H_b(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N), \tilde{Y}_s^N, N\tilde{Z}_s^{i,N})\mathrm{d}s \\ & - \frac{1}{2N^2}\sum_{i=1}^N \int_t^T \mathrm{tr}\big(\Sigma(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N))\partial_\mu^2 v(s, \bar{\mu}(\boldsymbol{X}_s^N))(X_s^{i,N}, X_s^{i,N})\big)\mathrm{d}s \\ & - \sum_{i=1}^N \int_t^T (\tilde{Z}_s^{i,N})^\intercal \sigma\big(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N)\big)\mathrm{d}W_s^i - \sum_{i=1}^N \int_t^T (\tilde{Z}_s^{i,N})^\intercal \sigma_0\big(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N)\big)\mathrm{d}W_s^0.\end{aligned} \tag{6.3.2}$$

Step 2: Linearization. We set

$$\Delta Y_t^N := Y_t^N - \tilde{Y}_t^N, \quad \Delta Z_t^{i,N} := N(Z_t^{i,N} - \tilde{Z}_t^{i,N}), \quad i = 1, \dots, N, \quad 0 \le t \le T,$$

so that by (6.2.6)-(6.3.2),

$$\begin{aligned}\Delta Y_t^N =\ & \frac{1}{N}\sum_{i=1}^N \int_t^T \big[H_b(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N), Y_s^N, NZ_s^{i,N}) - H_b(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N), \tilde Y_s^N, N\tilde Z_s^{i,N})\big]\mathrm{d}s \\ & + \frac{1}{2N^2}\sum_{i=1}^N \int_t^T \mathrm{tr}\big(\Sigma(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N))\partial_\mu^2 v(s, \bar\mu(\boldsymbol{X}_s^N))(X_s^{i,N}, X_s^{i,N})\big)\mathrm{d}s \\ & - \frac{1}{N}\sum_{i=1}^N \int_t^T (\Delta Z_s^{i,N})^\intercal \sigma\big(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N)\big)\mathrm{d}W_s^i \\ & - \frac{1}{N}\sum_{i=1}^N \int_t^T (\Delta Z_s^{i,N})^\intercal \sigma_0\big(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N)\big)\mathrm{d}W_s^0, \qquad 0 \le t \le T. \end{aligned} \tag{6.3.3}$$

We now use the linearization method for BSDEs and rewrite the above equation as

$$\begin{aligned}\Delta Y_t^N =\ & \int_t^T \alpha_s \Delta Y_s^N \mathrm{d}s + \frac{1}{N}\sum_{i=1}^N \int_t^T \beta_s^i . \Delta Z_s^{i,N}\mathrm{d}s \\ & - \frac{1}{N}\sum_{i=1}^N \int_t^T (\Delta Z_s^{i,N})^\intercal \sigma\big(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N)\big)\mathrm{d}W_s^i \\ & - \frac{1}{N}\sum_{i=1}^N \int_t^T (\Delta Z_s^{i,N})^\intercal \sigma_0\big(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N)\big)\mathrm{d}W_s^0 \\ & + \frac{1}{2N^2}\sum_{i=1}^N \int_t^T \mathrm{tr}\big(\Sigma(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N))\partial_\mu^2 v(s, \bar\mu(\boldsymbol{X}_s^N))(X_s^{i,N}, X_s^{i,N})\big)\mathrm{d}s, \end{aligned} \tag{6.3.4}$$

with

$$\begin{cases} \alpha_s = \frac{1}{N}\sum_{i=1}^N \dfrac{H_b(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N), Y_s^N, N\tilde Z_s^{i,N}) - H_b(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N), \tilde Y_s^N, N\tilde Z_s^{i,N})}{\Delta Y_s^N}\mathbb{1}_{\Delta Y_s^N \neq 0} \\ \beta_s^i = \frac{H_b(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N), Y_s^N, NZ_s^{i,N}) - H_b(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N), Y_s^N, N\tilde Z_s^{i,N})}{|\Delta Z_s^{i,N}|^2}\Delta Z_s^{i,N}\mathbb{1}_{\Delta Z_s^{i,N}\neq 0} \end{cases} \tag{6.3.5}$$

for $i = 1, \dots, N$, and we notice by Assumption 6.2.2(iv) that the processes α and β^i are bounded by $[H_b]_1$. Under Assumption 6.2.2(ii), let us define the bounded processes $\lambda_s^i = \sigma^+(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N))\beta_s^i$, $s \in [0,T]$, $i = 1, \dots, N$, and introduce the change of probability measure $\mathbb{P}^\lambda$ with Radon-Nikodym density:

$$\frac{\mathrm{d}\mathbb{P}^\lambda}{\mathrm{d}\mathbb{P}} = \mathcal{E}_T^\lambda := \exp\Big(\sum_{i=1}^N \int_0^T \lambda_s^i \mathrm{d}W_s^i - \sum_{i=1}^N \frac{1}{2}\int_0^T |\lambda_s^i|^2 \mathrm{d}s\Big),$$

so that by Girsanov's theorem: $\widetilde W_t^i = W_t^i - \int_0^t \lambda_s^i \mathrm{d}s$, $i = 1, \dots, N$, and W^0 are independant Brownian motion under $\mathbb{P}^\lambda$. By applying Itô's Lemma to $e^{\int_0^s \alpha_s \mathrm{d}s}\Delta Y_t^N$ under $\mathbb{P}^\lambda$, we then obtain

$$\begin{aligned}\Delta Y_t^N =\ & \frac{1}{2N^2}\sum_{i=1}^N \int_t^T e^{\int_t^s \alpha_u \mathrm{d}u}\mathrm{tr}\big(\Sigma(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N))\partial_\mu^2 v(s, \bar\mu(\boldsymbol{X}_s^N))(X_s^{i,N}, X_s^{i,N})\big)\mathrm{d}s \\ & - \frac{1}{N}\sum_{i=1}^N \int_t^T e^{\int_t^s \alpha_u \mathrm{d}u}(\Delta Z_s^{i,N})^\intercal \sigma\big(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N)\big)\mathrm{d}\widetilde W_s^i \\ & - \frac{1}{N}\sum_{i=1}^N \int_t^T e^{\int_t^s \alpha_u \mathrm{d}u}(\Delta Z_s^{i,N})^\intercal \sigma_0\big(s, X_s^{i,N}, \bar\mu(\boldsymbol{X}_s^N)\big)\mathrm{d}W_s^0, \quad 0 \le t \le T. \end{aligned} \tag{6.3.6}$$

Step 3. Let us check that the stochastic integrals in (6.3.6), namely $\int \tilde{\mathcal{Z}}_s^{i,N}.\mathrm{d}\widetilde{W}_s^i$, and $\int \tilde{\mathcal{Z}}_s^{0,i,N}.\mathrm{d}W_s^0$ are "true" martingales under $\mathbb{P}^\lambda$, where $\tilde{Z}_s^{i,N} := e^{\int_0^s \alpha_u \mathrm{d}u}\sigma^\intercal\big(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N)\big)\Delta Z_s^{i,N}$, $\tilde{Z}_s^{0,i,N} := e^{\int_0^s \alpha_u \mathrm{d}u}\sigma_0^\intercal\big(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N)\big)\Delta Z_s^{i,N}$, $i = 1, \dots, N$, $0 \le s \le T$.

Indeed, for fixed $i \in [\![1, N]\!]$, recalling that α is bounded, and by the linear growth condition of σ_0 from Assumption 6.2.2(i), we have

$$\begin{aligned}\mathbb{E}^{\mathbb{P}^\lambda}\Big[\int_0^T |\tilde{Z}_s^{0,i,N}|^2 \mathrm{d}s\Big] \le\ & C\mathbb{E}^{\mathbb{P}^\lambda}\Big[\int_0^T \big(|\sigma_0(s,0,\delta_0)|^2 + |X_s^{i,N}|^2 + \|\bar{\mu}(\boldsymbol{X}_s^N)\|_2^2\big) \\ & \qquad \big(|NZ_s^{i,N}|^2 + |\partial_\mu v(s, \bar{\mu}(\boldsymbol{X}_s^N))(X_s^{i,N})|^2\big)\mathrm{d}s \\ \le\ & C\mathbb{E}\Big[\mathcal{E}_T^\lambda \int_0^T N^2(|\sigma_0(s,0,\delta_0)|^4 + |\boldsymbol{X}_s^N|^4)\mathrm{d}s\Big]\end{aligned}$$

where we use Bayes formula, the estimation (6.2.9), the growth condition on $\partial_\mu v(.)(.)$ in Assumption 6.2.1, and noting that $|X_s^{i,N}| \le |\boldsymbol{X}_s^N|$, $\|\bar{\mu}(\boldsymbol{X}_s^N)\|_2^2 = |\boldsymbol{X}_s^N|^2/N$. By Hölder inequality with q as in Assumption 6.2.2(iii), and $\frac{1}{p} + \frac{1}{q} = 1$, the above inequality yields

$$\mathbb{E}^{\mathbb{P}^\lambda}\Big[\int_0^T |\tilde{Z}_s^{0,i,N}|^2 \mathrm{d}s\Big] \le CN^2\Big(\mathbb{E}\big[|\mathcal{E}_T^\lambda|^p\big]\Big)^{\frac{1}{p}}\Big(\mathbb{E}\Big[\int_0^T (|\sigma_0(s,0,\delta_0)|^{4q} + |\boldsymbol{X}_s^N|^{4q})\mathrm{d}s\Big]\Big)^{\frac{1}{q}},$$

which is finite by (6.2.8), and since λ is bounded. This shows the square-integrable martingale property of $\int \tilde{\mathcal{Z}}_s^{0,i,N}.\mathrm{d}W_s^0$ under $\mathbb{P}^\lambda$. By the same arguments, we get the square-integrable martingale property of $\int \tilde{\mathcal{Z}}_s^{i,N}.\mathrm{d}\tilde{W}_s^i$ under $\mathbb{P}^\lambda$

Step 4: Estimation of $\mathcal{E}_N^{\mathrm{y}}$. By taking the $\mathbb{P}^\lambda$ conditional expectation in (6.3.6), we obtain

$$\Delta Y_t^N = \frac{1}{2N^2}\sum_{i=1}^N \mathbb{E}^{\mathbb{P}^\lambda}\Big[\int_t^T e^{\int_t^s \alpha_u \mathrm{d}u}\mathrm{tr}\big(\Sigma(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N))\partial_\mu^2 v(s, \bar{\mu}(\boldsymbol{X}_s^N))(X_s^{i,N}, X_s^{i,N})\big)\mathrm{d}s\big|\mathcal{F}_t\Big],$$

for all $t \in [0, T]$. Under the boundedness condition on $\Sigma = \sigma\sigma^\intercal$ in Assumption 6.2.2(ii), and on $\partial_\mu^2 v$ in Assumption 6.2.1, it follows immediately that

$$\mathcal{E}_N^{\mathrm{y}} = \sup_{0\le t\le T}|\Delta Y_t^N| \le \frac{C_y}{N}, \quad a.s.$$

where $C_y = \frac{T}{2}e^{[H_b]_1 T}L\|\sigma\|_\infty^2$, with $\|\sigma\|_\infty = \sup_{(t,x,\mu)\in[0,T]\times\mathbb{R}^d\times\mathcal{P}_2(\mathbb{R}^d)}|\sigma(t,x,\mu)|$.

6.3.2 Proof of Theorem 6.2.2

From (6.3.5), and under Assumption 6.2.3(i) and (iii), we see that

$$\begin{aligned}\alpha_s &= \frac{1}{N}\sum_{i=1}^N \frac{H_b(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N), Y_s^N, N\tilde{Z}_s^{i,N}) - H_b(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N), \tilde{Y}_s^N, N\tilde{Z}_s^{i,N})}{\Delta Y_s^N}\mathbb{1}_{\Delta Y_s^N \neq 0} \\ &= \frac{1}{N}\sum_{i=1}^N \frac{H_1(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N), Y_s^N) - H_1(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N), \tilde{Y}_s^N)}{\Delta Y_s^N}\mathbb{1}_{\Delta Y_s^N \neq 0} \\ &\quad + \frac{1}{N}\sum_{i=1}^N N\tilde{Z}_s^{i,N}.\frac{H_2(s, \bar{\mu}(\boldsymbol{X}_s^N), Y_s^N) - H_2(s, \bar{\mu}(\boldsymbol{X}_s^N), \tilde{Y}_s^N)}{\Delta Y_s^N}\mathbb{1}_{\Delta Y_s^N \neq 0}, \qquad (6.3.7)\end{aligned}$$

is bounded by $[H_1]_1 + [H_2]_1 L$, recalling $N\tilde{Z}_s^{i,N} = \partial_\mu v(s, \bar{\mu}(\boldsymbol{X}_s^N))(X_s^{i,N})$. As a consequence, the proof of Theorem 6.2.1 still applies. Then by (6.3.3)

$$\begin{aligned}
\Delta Y_t^N = &\int_t^T \alpha_s \Delta Y_s^N \mathrm{d}s + \frac{1}{N}\sum_{i=1}^N \int_t^T H_2(s, \bar{\mu}(\boldsymbol{X}_s^N), Y_s^N).\Delta Z_s^{i,N}\mathrm{d}s \\
&- \frac{1}{N}\sum_{i=1}^N \int_t^T (\Delta Z_s^{i,N})^\intercal \sigma\big(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N)\big)\mathrm{d}W_s^i \\
&- \frac{1}{N}\sum_{i=1}^N \int_t^T (\Delta Z_s^{i,N})^\intercal \sigma_0(s, \bar{\mu}(\boldsymbol{X}_s^N))\mathrm{d}W_s^0 \\
&+ \frac{1}{2N^2}\sum_{i=1}^N \int_t^T \mathrm{tr}\big(\Sigma(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N))\partial_\mu^2 v(s, \bar{\mu}(\boldsymbol{X}_s^N))(X_s^{i,N}, X_s^{i,N})\big)\mathrm{d}s.
\end{aligned}$$

By applying Itô's formula to $|\Delta Y_t^N|^2$ in (6.3.4) under $\mathbb{P}$

$$\begin{aligned}
&|\Delta Y_0^N|^2 + \frac{1}{N^2}\int_0^T \sum_{i=1}^N \big|\sigma^\intercal\big(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N)\big)\Delta Z_s^{i,N}\big|^2 \mathrm{d}s + |\sigma_0^\intercal(s, \bar{\mu}(\boldsymbol{X}_s^N))\sum_{j=1}^N \Delta Z_s^{j,N}|^2 \mathrm{d}s \\
&= 2\int_0^T \alpha_s |\Delta Y_s^N|^2 \mathrm{d}s + \frac{2}{N}\sum_{i=1}^N \int_0^T \Delta Y_s^N H_2(s, \bar{\mu}(\boldsymbol{X}_s^N), Y_s^N).\Delta Z_s^{i,N}\mathrm{d}s \\
&\quad - \frac{2}{N}\sum_{i=1}^N \int_0^T \Delta Y_s^N (\Delta Z_s^{i,N})^\intercal \sigma\big(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N)\big)\mathrm{d}W_s^i \\
&\quad - \frac{2}{N}\sum_{i=1}^N \int_0^T \Delta Y_s^N (\Delta Z_s^{i,N})^\intercal \sigma_0(s, \bar{\mu}(\boldsymbol{X}_s^N))\mathrm{d}W_s^0 \\
&\quad + \frac{1}{N^2}\sum_{i=1}^N \int_0^T \Delta Y_s^N \mathrm{tr}\big(\Sigma(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N))\partial_\mu^2 v(s, \bar{\mu}(\boldsymbol{X}_s^N))(X_s^{i,N}, X_s^{i,N})\big)\mathrm{d}s,
\end{aligned}$$

so by taking expectation under $\mathbb{P}$, and using the Cauchy-Schwarz inequality in $\mathbb{R}^d$

$$\begin{aligned}
&\frac{1}{N^2}\int_0^T \sum_{i=1}^N \mathbb{E}\Big[\big|\sigma^\intercal\big(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N)\big)\Delta Z_s^{i,N}\big|^2 + |\sigma_0^\intercal(s, \bar{\mu}(\boldsymbol{X}_s^N))\sum_{j=1}^N \Delta Z_s^{j,N}|^2\Big]\mathrm{d}s \\
&\le \mathbb{E}\Big[2\int_0^T |\alpha_s||\Delta Y_s^N|^2 \mathrm{d}s + 2\int_0^T \Big|\Delta Y_s^N H_2(s, \bar{\mu}(\boldsymbol{X}_s^N), Y_s^N).\sum_{i=1}^N \frac{\Delta Z_s^{i,N}}{N}\Big|\mathrm{d}s\Big] \\
&\quad + \frac{1}{N^2}\sum_{i=1}^N \int_0^T \mathbb{E}|\Delta Y_s^N \mathrm{tr}\big(\Sigma(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N))\partial_\mu^2 v(s, \bar{\mu}(\boldsymbol{X}_s^N))(X_s^{i,N}, X_s^{i,N})\big)|\mathrm{d}s, \\
&\le \mathbb{E}\Big[2\int_0^T |\alpha_s||\Delta Y_s^N|^2 \mathrm{d}s + 2\int_0^T \Big|\Delta Y_s^N H_2(s, \bar{\mu}(\boldsymbol{X}_s^N), Y_s^N)\Big|\Big|\sum_{i=1}^N \frac{\Delta Z_s^{i,N}}{N}\Big|\mathrm{d}s\Big] \\
&\quad + \frac{1}{N^2}\sum_{i=1}^N \int_0^T \mathbb{E}|\Delta Y_s^N \mathrm{tr}\big(\Sigma(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N))\partial_\mu^2 v(s, \bar{\mu}(\boldsymbol{X}_s^N))(X_s^{i,N}, X_s^{i,N})\big)|\mathrm{d}s, \\
&\le \mathbb{E}\Big[\vartheta\int_0^T |\Delta Y_s^N H_2(s, \bar{\mu}(\boldsymbol{X}_s^N), Y_s^N)|^2 \mathrm{d}s + \frac{1}{\vartheta N^2}\int_0^T \Big|\sum_{i=1}^N \Delta Z_s^{i,N}\Big|^2 \mathrm{d}s\Big] + \frac{\tilde{C}_z}{N^2},
\end{aligned}$$

by Young inequality for any $\vartheta > 0$, boundedness of α (see (6.3.7)), Σ, $\partial_\mu^2 v$ and Theorem 6.2.1, where $\tilde{C}_z = 2T([H_1]_1 + [H_2]_1 L)\bar{C}_y^2 + \bar{C}_y T\|\sigma\|_\infty^2 L$ and $\bar{C}_y = \frac{T}{2}e^{([H_1]_1 + [H_2]_1 L)T} L\|\sigma\|_\infty^2$. Thus,

by Assumption 6.2.3(ii), and by choosing $\vartheta = \frac{2}{c_0}$, it follows from the boundedness of H_2, and Theorem 6.2.1 that

$$\mathbb{E}\Big[\frac{1}{N}\sum_{i=1}^{N}\int_0^T |\sigma^\intercal\big(s, X_s^{i,N}, \bar{\mu}(\boldsymbol{X}_s^N)\big)\Delta Z_s^{i,N}|^2 \,\mathrm{d}s\Big] \;\leq\; \frac{\tilde{C}_z + \frac{2}{c_0}\bar{C}_y^2 T\|H_2\|_\infty^2}{N},$$

which ends the proof by recalling that $\|\sigma^+\|_\infty < +\infty$, using Cauchy-Schwarz inequality in $\mathbb{R}^N$ (in the form $\frac{1}{N}\sum_i^N \sqrt{|a_i|} \leq \sqrt{\frac{1}{N}\sum_i^N |a_i|}$) and Jensen inequality (in the form $\mathbb{E}[\sqrt{|X|}] \leq \sqrt{\mathbb{E}[|X|]}$).

Chapter 7

Solving mean-field PDEs with symmetric neural networks

This chapter is based on the paper [Ger+22]
M. Germain, M. Laurière, H. Pham, X. Warin. "DeepSets and their derivative networks for solving symmetric PDEs". In: *Journal of Scientific Computing* 91, 63 (8 April 2022).

This Chapter is a companion work of Chapter 6. Inspired from the finite dimensional PDEs stemming from Chapter 6, which have the property of being symmetric with respect to the permutation of their space arguments, we develop a numerical method to solve symmetric PDEs. The main tool is to consider symmetric neural networks like PointNet [Qi+17] and DeepSet [Zah+17]. Plugging the symmetry properties of the solution into the scheme itself allows us to solve high dimensional problems for which classical feedforward neural networks do not converge. As a consequence we provide a way to estimate the solution of Master Bellman equations in infinite dimension. However, some randomization has to be achieved in order to efficiently explore the Wasserstein space. We propose a first idea in this direction but a systematic treatment of this point is left to future research.

Abstract

Machine learning methods for solving nonlinear partial differential equations (PDEs) are hot topical issues, and different algorithms proposed in the literature show efficient numerical approximation in high dimension. In this paper, we introduce a class of PDEs that are invariant to permutations, and called *symmetric* PDEs. Such problems are widespread, ranging from cosmology to quantum mechanics, and option pricing/hedging in multi-asset market with exchangeable payoff. Our main application comes actually from the particles approximation of mean-field control problems. We design deep learning algorithms based on certain types of neural networks, named PointNet and DeepSet (and their associated derivative networks), for computing simultaneously an approximation of the solution and its gradient to symmetric PDEs. We illustrate the performance and accuracy of the PointNet/DeepSet networks compared to classical feedforward ones, and provide several numerical results of our algorithm for the examples of a mean-field systemic risk, mean-variance problem and a min/max linear quadratic McKean-Vlasov control problem.

7.1 Introduction

The numerical resolution of partial differential equations (PDEs) in high dimension is a major challenge in various areas of science, engineering, and finance. PDEs that appear in the applications are often non linear and of very high dimension (number of particles in physics, number of agents in large population control problems, number of assets and factors in financial markets, etc), and are subject to the so-called curse of dimensionality, which makes infeasible the implementation of classical grid methods and Monte-Carlo approaches.

A breakthrough with deep learning based-algorithms has been made in the last five years towards this computational challenge, and we mention the recent survey papers by [Bec+20] and [GPW22b]. The main interest in the use of machine learning techniques for PDEs is the ability of deep neural networks to efficiently represent high dimensional functions without using spatial grids, and with no curse of dimensionality (see e.g. [Hut+20]). Although the use of neural networks for solving PDEs is not new, the approach has been successfully revived with new ideas and directions. Moreover, recently developed open source libraries like Tensorflow and Pytorch offer an accessible framework to implement these algorithms.

In this paper, we introduce a class of PDEs that are invariant by permutation, and called here *symmetric* PDEs. Such PDEs occur naturally in the modelling of systems dealing with sets that are invariant by permutation of their elements. Applications range from models in general relativity and cosmology, to quantum mechanics and chemistry, see e.g. [Wyk08], [Smu11]. Symmetric PDEs also appear in the pricing/hedging of basket option and options on the maximum of multiple assets. Our main motivation for introducing this general class of symmetric PDEs comes from the control of large population of interacting indistinguishable agents, which leads in the asymptotic regime of infinite population to the theory of mean-field games (MFG) and mean-field type control, also called McKean-Vlasov (MKV) control. These topics have attracted an increasing and large interest since the seminal papers [LL07] and [HCM06] with important mathematical developments and numerous applications in various fields over the last decade. We refer to the two-volume monographs [CD18a]-[CD18b] for an exhaustive exposition of this research domain, where it is known that the solution to MFG or MKV control problem are characterized in terms of a Master equation or a Bellman equation, which are PDEs in the Wasserstein space of probability measures. It turns out that the finite-dimensional approximation of these equations are formulated as *symmetric* non linear PDEs, and the convergence of this approximation has been recently obtained in [GMS21], and [GPW22c] (for a rate of convergence), see also [Lac17] and [Dje20].

Symmetric PDEs are often in very high dimension, typically of the order of one thousand in the case of particles approximation of Master and Bellman equations, and it is tempting to apply

machine learning algorithms in this framework. For that purpose, we shall furthermore exploit the symmetric structure of the PDEs in order to design deep learning-based algorithms with a suitable class of neural networks. We first observe that the solution to symmetric PDEs is invariant by permutation (also called *exchangeable*), and we shall then consider a class of symmetric neural networks, named PointNet and DeepSets, aiming to approximate our solution. Such class of neural networks has been recently introduced in the machine learning community, see [Qi+17], [Zah+17], [BRT20], for dealing with tasks involving some invariant data sets, and it turns out that they provide much better accuracy than classical feedforward neural networks (NN in short) in the approximation of symmetric functions. Indeed, feedforward NN have too many degrees of freedom, and the optimization over parameters in (stochastic) gradient descent algorithm may be trapped away in the approximation of a symmetric function, as illustrated in several examples and comparison tests presented in this paper. We shall also introduce different classes of derivative symmetric network, named DeepDerSet and AD-DeepSet, for the approximation of the gradient of the solution to symmetric PDEs.

By relying on the class of symmetric NN, and their derivative networks, we next adapt the deep backward dynamic programming scheme [HPW20], [PWG21] for numerically solving symmetric PDEs, i.e., finding approximations of the solution and its gradient. We also explain in the case of mean-field control problem how our scheme provides an approximation for the solution to a Bellman equation in the Wasserstein space of probability measures. This yields alternative deep learning schemes for mean-field control problems to the ones recently designed in [GMW22], [CL22], [FZ20], or [Rut+20]. We test our algorithms on several examples arising from different McKean-Vlasov control problem, for which we have explicit or benchmarked solutions: a systemic risk model as in [CFS15], the classical mean-variance, i.e., Markowitz portfolio allocation problem, and a min/max linear quadratic mean-field control problem as in [SMLN15].

Outline of the paper. The rest of the paper is organized as follows. We introduce in Section 7.2 the class of *symmetric* PDEs with some examples, and show exchangeability properties of the solution and its gradient to such PDEs. Section 7.3 is devoted to the exposition of the class of symmetric neural networks, as well as its derivative networks, and we provide several comparison tests with respect to classical feedforward NN. We describe in Section 7.4 the deep learning schemes for solving symmetric PDEs, and finally provide several numerical examples in Section 7.5.

Notations. Given $N \in \mathbb{N}^*$, $\mathcal{X}^N$ denotes the set of all elements $\boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]}$ with coefficients x_i valued in $\mathcal{X}$ and $[\![1,N]\!] = \{1,\cdots,N\}$. When $\mathcal{X} = \mathbb{R}^d$, one usually identifies $(\mathbb{R}^d)^N$ with $\mathbb{R}^{d\times N}$ the set of $d \times N$-matrices with real-valued coefficients. $\mathbb{S}^N(\mathcal{X})$ is the set of $N \times N$-symmetric matrices with coefficients valued in $\mathcal{X}$, and is simply denoted by $\mathbb{S}^N$ when $\mathcal{X} = \mathbb{R}$. For a real-valued C^2 function φ defined on $(\mathbb{R}^d)^N$, its gradient $D\varphi(\boldsymbol{x}) = (D_{x_i}\varphi(\boldsymbol{x}))_{i\in[\![1,N]\!]}$ is valued in $(\mathbb{R}^d)^N$, while its Hessian $D^2\varphi(\boldsymbol{x}) = (D^2_{x_ix_j}\varphi(\boldsymbol{x}))_{i,j\in[\![1,N]\!]}$ is valued in $\mathbb{S}^N(\mathbb{S}^d)$.

We denote by $\mathfrak{S}_N$ the set of permutations on $\{1,\ldots,N\}$. For any $\boldsymbol{x} = (x_i)_{i\in[\![1,\mathbb{N}]\!]} \in \mathcal{X}^N$, $\pi \in \mathfrak{S}_N$, we denote by $\pi[\boldsymbol{x}] = (x_{\pi(i)})_{i\in[\![1,\mathbb{N}]\!]} \in \mathcal{X}^N$. For any $\boldsymbol{\Gamma} = (\Gamma_{ij})_{i,j\in[\![1,\mathbb{N}]\!]} \in \mathbb{S}^N(\mathcal{X})$, we denote by $\pi[\boldsymbol{\Gamma}] = (\Gamma_{\pi(i)\pi(j)})_{i,j\in[\![1,N]\!]} \in \mathbb{S}^N(\mathcal{X})$.

We say that a function φ defined on $\mathcal{X}^N$ is exchangeable to the order N on $\mathcal{X}$ if it is invariant by permutation, i.e., $\varphi(\boldsymbol{x}) = \varphi(\pi[\boldsymbol{x}])$, for any $\boldsymbol{x} \in \mathcal{X}^N$, and $\pi \in \mathfrak{S}_N$. We may simply say exchangeable when it is clear from the context. The notations $0_d, 1_d$ refer respectively to d-dimensional vectors full of 0 and 1. With two vectors $a, b \in \mathbb{R}^d$, the notation $a.b = \sum_{i=1}^d a_ib_i$ refers to the canonical scalar product.

7.2 Symmetric PDEs

We consider a so-called *symmetric* class of parabolic second-order partial differential equations (PDEs):

$$\left\{ \begin{array}{rcll} \partial_t v + F(t, \boldsymbol{x}, v, D_{\boldsymbol{x}} v, D^2_{\boldsymbol{x}} v) & = & 0, & (t, \boldsymbol{x}) \in [0,T) \times (\mathbb{R}^d)^N \\ v(T, \boldsymbol{x}) & = & G(\boldsymbol{x}), & \boldsymbol{x} \in (\mathbb{R}^d)^N, \end{array} \right. \tag{7.2.1}$$

where F is a real-valued function defined on $[0,T] \times (\mathbb{R}^d)^N \times \mathbb{R} \times (\mathbb{R}^d)^N \times \mathbb{S}^N(\mathbb{S}^d)$, G is defined on $(\mathbb{R}^d)^N$, and satisfying the permutation-invariance condition:

(HI) For any $t \in [0,T]$, $\boldsymbol{x} \in (\mathbb{R}^d)^N$, $y \in \mathbb{R}$, $\boldsymbol{z} \in (\mathbb{R}^d)^N$, $\boldsymbol{\gamma} \in \mathbb{S}^N(\mathbb{S}^d)$,

$$\begin{array}{rcl} F(t, \boldsymbol{x}, y, \boldsymbol{z}, \boldsymbol{\gamma}) & = & F(t, \pi[\boldsymbol{x}], y, \pi[\boldsymbol{z}], \pi[\boldsymbol{\gamma}]) \\ G(\boldsymbol{x}) & = & G(\pi[\boldsymbol{x}]), \quad \forall \pi \in \mathfrak{S}_N. \end{array}$$

We assume that PDE (7.2.1) is well-posed in the sense that there exists a unique classical solution satisfying a suitable growth condition.

We list below some examples of symmetric PDEs in the form (7.2.1). We start with an example of pricing in a "symmetric" multi-asset model.

Example 7.2.1 (*Multi-asset pricing*)**.** Let us consider a model with N risky assets of price process $\boldsymbol{X} = (X^1, \ldots, X^N)$ governed by

$$dX^i_t \;=\; \tilde{b}_i(\boldsymbol{X}_t)dt + \sum_{j=1}^N \sigma_{ij}(\boldsymbol{X}_t) dW^j_t,$$

where the diffusion coefficients satisfy the property: for all $\pi \in \mathfrak{S}_N$,

$$\sigma_{ij}(\pi[\boldsymbol{x}]) \;=\; \sigma_{\pi(i)\pi(j)}(\boldsymbol{x}), \quad \boldsymbol{x} = (x_i)_{i \in [\![1,N]\!]},\ i,j = 1, \ldots, N. \tag{7.2.2}$$

Notice that $\tilde{b}_i$ is the drift of the asset price under the historical probability measure, and does not appear in the pricing equations below. The symmetry condition (7.2.2) is satisfied for example when $\sigma_{ii}(\boldsymbol{x}) = \sigma(\boldsymbol{x})$, and $\sigma_{ij}(\boldsymbol{x}) = \tilde{\sigma}(\boldsymbol{x})$, $i,j = 1, \ldots, N$, $i \neq j$, with $\sigma, \tilde{\sigma}$ exchangeable functions. Another example is when $\sigma_{ii}(\boldsymbol{x}) = \sigma(x_i)$, and $\sigma_{ij}(\boldsymbol{x}) = \tilde{\vartheta}(x_i)\bar{\vartheta}(x_j)$, $i,j = 1, \ldots, N$, $i \neq j$, for some functions σ, $\tilde{\vartheta}$, $\bar{\vartheta}$ defined on $\mathbb{R}$, which means that all the assets have the same marginal volatility coefficient, and the correlation function between any pair of assets is identical. We consider an option of maturity T with payoff $G(X^1_T, \ldots, X^N_T)$, where G is an exchangeable function, for example:

$$G(\boldsymbol{x}) \;=\; \left\{ \begin{array}{ll} \big(\max(x_1, \ldots, x_N) - K\big)_+, & \text{(call on max)} \\ \big(\sum_{i=1}^N x_i - K\big)_+, & \text{(call on sum)}, \\ \sum_{i=1}^N 1_{x_i \geq K}, & \text{(sum of binary options)}, \end{array} \right.$$

for $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathbb{R}^N$. In a frictionless market with constant interest rate r, the option price $(t, \boldsymbol{x}) \in [0,T] \times \mathbb{R}^N \mapsto v(t, \boldsymbol{x})$ satisfies a linear PDE (7.2.1) with terminal condition given by the exchangeable function G and

$$F(t, \boldsymbol{x}, y, \boldsymbol{z}, \boldsymbol{\gamma}) \;=\; -ry + r\sum_{i=1}^N x_i z_i + \frac{1}{2}\sum_{i,j=1}^N \sigma^2_{ij}(\boldsymbol{x})\gamma_{ij},$$

for $t \in [0,T]$, $\boldsymbol{x} = (x_i)_{i \in [\![1,N]\!]} \in \mathbb{R}^N$, $y \in \mathbb{R}$, $\boldsymbol{z} = (z_i)_{i \in [\![1,\mathbb{N}]\!]} \in \mathbb{R}^N$, and $\boldsymbol{\gamma} = (\gamma_{ij})_{i,j \in [\![1,N]\!]} \in \mathbb{S}^N$. In the case of counterparty risk, the pricing of CVA leads to a quasi-linear PDE (7.2.1) with F in the form (see [HL12] for the details of the PDE derivation):

$$F(t, \boldsymbol{x}, y, \boldsymbol{z}, \boldsymbol{\gamma}) \;=\; \beta(y^+ - y) + r\sum_{i=1}^N x_i z_i + \frac{1}{2}\sum_{i,j=1}^N \sigma^2_{ij}(\boldsymbol{x})\gamma_{ij},$$

where $\beta > 0$ is the intensity of default. Another case of non-linearity occurs when lending rate $r > 0$ is smaller than borrowing rate $R > 0$, which leads to a super-replication price solution to (7.2.1) with F given by

$$F(t,\boldsymbol{x},y,\boldsymbol{z},\boldsymbol{\gamma}) \;=\; \sup_{b\in[r,R]}\big[-by+b\sum_{i=1}^N x_iz_i\big]+\frac{1}{2}\sum_{i,j=1}^N\sigma_{ij}^2(\boldsymbol{x})\gamma_{ij}.$$

In the above three cases, and under (7.2.2), the generator function F clearly satisfies the permutation-invariance condition in **(HI)**. □

The second example is actually our main motivation for considering symmetric PDEs, and comes from mean-field models.

Example 7.2.2 (*McKean-Vlasov control problem with common noise).* Let us consider N interacting indistinguishable agents with controlled state process $\boldsymbol{X} = (X^1,\ldots,X^N)$ valued in $(\mathbb{R}^d)^N$, and driven by

$$\begin{aligned} dX_t^i = \;& \beta(t,X_t^i,\bar\mu(\boldsymbol{X}_t),\alpha_t^i)dt+\sigma(t,X_t^i,\bar\mu(\boldsymbol{X}_t),\alpha_t^i)dW_t^i \\ &+\;\sigma_0(t,X_t^i,\bar\mu(\boldsymbol{X}_t))dW_t^0, \qquad 0\le t\le T,\; i=1,\ldots,N,\end{aligned}$$

where $\boldsymbol{x}=(x_i)_{i\in[\![1,N]\!]}\mapsto\bar\mu(\boldsymbol{x})=\frac{1}{N}\sum_{i=1}^N\delta_{x_i}$ is the empirical measure (exchangeable) function, W^i, $i=1,\ldots,N$, are independent Brownian motions representing idiosyncratic noises, and W^0 is a Brownian motion independent of $\boldsymbol{W}=(W^i)_{i\in[\![1,N]\!]}$, representing a common noise. Moreover, α^i is a control process (valued in some Polish space A) applied by the agent i who follows in a cooperative equilibrium a social planner aiming to minimize a social cost in the form

$$J(\alpha^1,\ldots,\alpha^N) \;=\; \frac{1}{N}\sum_{i=1}^N\mathbb{E}\Big[\int_0^T e^{-rt}f(X_t^i,\bar\mu(\boldsymbol{X}_t),\alpha_t^i)dt+e^{-rT}g(X_T^i,\bar\mu(\boldsymbol{X}_T))\Big].$$

The Bellman equation to this N-cooperative agents control problem is in the form (7.2.1) with a Hamiltonian function F given by

$$\begin{aligned} F(t,\boldsymbol{x},y,\boldsymbol{z},\boldsymbol{\gamma}) \;=\; & \sum_{i=1}^N\inf_{a\in A}\big[\beta(t,x_i,\bar\mu(\boldsymbol{x}),a).z_i+\frac{1}{2}\mathrm{tr}\big(\Sigma(t,x_i,\bar\mu(\boldsymbol{x}),a)\gamma_{ii}\big)+\frac{1}{N}f(x_i,\bar\mu(\boldsymbol{x}),a)\big] \\ & +\frac{1}{2}\sum_{i\neq j}\mathrm{tr}\big(\sigma_0(t,x_i,\bar\mu(\boldsymbol{x}))\sigma_0^\intercal(t,x_j,\bar\mu(\boldsymbol{x}))\gamma_{ij}\big)\;-ry,\end{aligned}$$

where $\Sigma=\sigma\sigma^\intercal+\sigma_0\sigma_0^\intercal$, and a terminal condition given by

$$G(\boldsymbol{x}) \;=\; \frac{1}{N}\sum_{i=1}^N g(x_i,\bar\mu(\boldsymbol{x})).$$

Such functions F and G clearly satisfy condition **(HI)**. Let us point out that in the limiting regime when the number N of agents goes to infinity, it is proved in [Lac17], [Dje20], [GMS21] that the solution to this cooperative-agents problem converges to the McKean-Vlasov control problem with state process $X=X^\alpha$ of dynamics

$$dX_t=\beta(t,X_t,\mathbb{P}^0_{X_t},\alpha_t)dt+\sigma(t,X_t,\mathbb{P}^0_{X_t},\alpha_t)dW_t+\sigma_0(t,X_t,\mathbb{P}^0_{X_t})dW_t^0, \tag{7.2.3}$$

and cost functional

$$J_{MKV}(\alpha) \;=\; \mathbb{E}\Big[\int_0^T e^{-rt}f(X_t,\mathbb{P}^0_{X_t},\alpha_t)dt+e^{-rT}g(X_T,\mathbb{P}^0_{X_T})\Big].$$

(Here $\mathbb{P}^0_{X_t}$ denotes the conditional law of X_t given the common noise W^0). Moreover, the corresponding Bellman equation in the Wasserstein space of square-integrable probability measures $\mathcal{P}_2(\mathbb{R}^d)$ is given by (see [PW17])

$$\left\{ \begin{array}{rcll} \partial_t v + \mathcal{F}(t,\mu,v,\partial_\mu v,\partial_x\partial_\mu v,\partial^2_\mu v) &=& 0, & (t,\mu)\in[0,T)\times\mathcal{P}_2(\mathbb{R}^d) \\ v(T,\mu) &=& \mathcal{G}(\mu), & \mu\in\mathcal{P}_2(\mathbb{R}^d), \end{array} \right. \tag{7.2.4}$$

where $\partial_\mu\varphi(\mu)(.):\mathbb{R}^d\to\mathbb{R}^d$, $\partial_x\partial_\mu\varphi(\mu)(.):\mathbb{R}^d\to\mathbb{S}^d$, $\partial^2_\mu\varphi(\mu)(.,.):\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{S}^d$, are the L-derivatives of a function φ on $\mathcal{P}_2(\mathbb{R}^d)$ (see [CD18a]) and

$$\begin{array}{rcl} \mathcal{F}(t,\mu,y,Z(.),\Gamma(.),\Gamma_0(.,.)) &=& -ry + \displaystyle\int_{\mathbb{R}^d} h(t,x,\mu,Z(x),\Gamma(x))\mu(dx) \\ && + \displaystyle\int_{\mathbb{R}^d\times\mathbb{R}^d} \frac{1}{2}\mathrm{tr}\big(\sigma_0(t,x,\mu)\sigma_0^\intercal(t,x',\mu)\Gamma_0(x,x')\big)\mu(dx)\mu(dx'), \\ \mathcal{G}(\mu) &=& \displaystyle\int_{\mathbb{R}^d} g(x,\mu)\mu(dx), \end{array}$$

with

$$h(t,x,\mu,z,\gamma) \;=\; \inf_{a\in A}\Big[\beta(t,x,\mu,a).z + \frac{1}{2}\mathrm{tr}\big(\Sigma(t,x,\mu,a)\gamma\big) + \; f(x,\mu,a)\Big].$$

□

We end this section by showing some exchangeability properties of the solution to the symmetric PDE (7.2.1). Let us introduce the notion of D-exchangeability where D stands for derivative.

Definition 7.2.1. *A function $(\boldsymbol{x},x)\in(\mathbb{R}^d)^N\times\mathbb{R}^d\mapsto \mathrm{z}(\boldsymbol{x},x)\in\mathcal{X}$ is D-exchangeable if for any fixed $x\in\mathbb{R}^d$, $\mathrm{z}(.,x)$ is exchangeable. Given a D-exchangeable function z, we denote by $\mathbf{z}$ the function defined on $(\mathbb{R}^d)^N$ by $\mathbf{z}(\boldsymbol{x}) = (\mathrm{z}(\boldsymbol{x},x_i))_{i\in[\![1,N]\!]}\in\mathcal{X}^N$.*

This definition is actually motivated by the exchangeability property of the solution to the PDE (7.2.1), and by a structural property on the gradient of an exchangeable function that is differentiable.

Lemma 7.2.1. *The solution v to the PDE* (7.2.1) *with F and G satisfying* **(HI)** *is exchangeable, i.e., for all $\pi\in\mathfrak{S}_N$,*

$$v(t,\boldsymbol{x}) \;=\; v(t,\pi[\boldsymbol{x}]), \quad (t,\boldsymbol{x})\in[0,T]\times(\mathbb{R}^d)^N.$$

Proof. Let $\pi\in\mathfrak{S}_N$. We set $v_\pi(t,\boldsymbol{x}) = v(t,\pi[\boldsymbol{x}])$, and observe that $\partial_t v(t,\pi[\boldsymbol{x}]) = \partial_t v_\pi(t,\boldsymbol{x})$, while

$$D_{\boldsymbol{x}}v(t,\pi[\boldsymbol{x}]) \;=\; \pi[D_{\boldsymbol{x}}v_\pi(t,\boldsymbol{x})], \quad D^2_{\boldsymbol{x}}v(t,\pi[\boldsymbol{x}]) \;=\; \pi[D^2_{\boldsymbol{x}}v_\pi(t,\boldsymbol{x})].$$

By writing the PDE (7.2.1) at $(t,\pi[\boldsymbol{x}])$, it follows under **(HI)** that v_π satisfies

$$\left\{ \begin{array}{rcll} \partial_t v_\pi + F(t,\boldsymbol{x},v_\pi,D_{\boldsymbol{x}}v_\pi,D^2_{\boldsymbol{x}}v_\pi) &=& 0, & (t,\boldsymbol{x})\in[0,T)\in(\mathbb{R}^d)^N \\ v_\pi(T,\boldsymbol{x}) &=& G(\boldsymbol{x}), & \boldsymbol{x}\in(\mathbb{R}^d)^N. \end{array} \right.$$

By uniqueness of the solution to PDE (7.2.1), we conclude that $v_\pi = v$, i.e., the exchangeability property of v. □

Lemma 7.2.2. *Let w be an exchangeable, and differentiable function on $(\mathbb{R}^d)^N$. Then there exists a D-exchangeable function z such that*

$$D_{x_i}w(\boldsymbol{x}) = \; \mathrm{z}(\boldsymbol{x},x_i), \quad i=1,\dots,N, \tag{7.2.5}$$

for all $\boldsymbol{x} = (x_i)_{\in[\![1,N]\!]}\in(\mathbb{R}^d)^N$, i.e., $Dw = \mathbf{z}$.

Proof. Since w is exchangeable, it is clear that for fixed $i \in [\![1, N]\!]$, and $x_i \in \mathbb{R}^d$,

$$\boldsymbol{x}_{-i} := (x_j)_{j\neq i} \in (\mathbb{R}^d)^{N-1} \mapsto D_{x_i} w(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_N) \quad \text{is exchangeable,}$$

and we shall then write:

$$\mathrm{z}^i(\boldsymbol{x}_{-i}, x) := D_{x_i} w(x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_N), \quad x \in \mathbb{R}^d .$$

By exchangeability of w, we also note that

$$\mathrm{z}^i(\boldsymbol{x}_{-i}, x) = \mathrm{z}^\ell(\boldsymbol{x}_{-i}, x), \quad \forall i, \ell \in [\![1, N]\!]. \tag{7.2.6}$$

Let us now define the function z on $(\mathbb{R}^d)^N \times \mathbb{R}^d$ by:

$$\mathrm{z}(\boldsymbol{x}, x) := \frac{1}{N} \sum_{p=0}^{N-1} (-1)^p \sum_{1\leq i_1<\dots<i_{p+1}\leq N} \sum_{\ell=1}^{N} \mathrm{z}^\ell((\underbrace{x, \dots, x}_{p \text{ times}}, x_j)_{j\neq i_1,\dots,i_{p+1}}, x).$$

for $\boldsymbol{x} = (x_1, \dots, x_N) \in (\mathbb{R}^d)^N$, and $x \in \mathbb{R}^d$, $(\underbrace{x, \dots, x}_{p \text{ times}}, (x_j)_{j\neq i_1,\dots,i_{p+1}})$ is the vector in $(\mathbb{R}^d)^{N-1}$ consisting of p components x, and the $N - p - 1$ components x_j, for $j \neq i_1, \dots, i_{p+1}$. By construction, it is clear that for fixed $x \in \mathbb{R}^d$, $\mathrm{z}(., x)$ is exchangeable, i.e., z is a D-exchangeable function. Let us now show (7.2.5), i.e., that for fixed $\boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N$,

$$\mathrm{z}(\boldsymbol{x}, x_i) = \mathrm{z}^i(\boldsymbol{x}_{-i}, x_i), \quad i \in [\![1, N]\!].$$

It suffices to check this property for $i = 1$. We set for $p = 0, \dots, N - 1$:

$$S^p := \sum_{1\leq i_1<\dots<i_{p+1}\leq N} \sum_{\ell=1}^{N} \mathrm{z}^\ell(x_1, (\underbrace{x_1, \dots, x_1}_{p \text{ times}}, x_j)_{j\neq i_1,\dots,i_{p+1}}).$$

and see that

$$\begin{cases} S^0 &= \displaystyle\sum_{\ell=1}^{N} \mathrm{z}^\ell(\boldsymbol{x}_{-1}, x_1) + \sum_{i_1=2}^{N} \sum_{\ell=1}^{N} \mathrm{z}^\ell((x_1, x_j)_{j\neq 1,i_1}, x_1) \\ S^1 &= \displaystyle\sum_{i_2=2}^{N} \sum_{\ell=1}^{N} \mathrm{z}^\ell((x_1, x_j)_{j\neq 1,i_2}, x_1) + \sum_{2\leq i_1<i_2\leq N} \sum_{\ell=1}^{N} \mathrm{z}^\ell((x_1, x_j)_{j\neq i_1,i_2}, x_1) \\ &\vdots \\ S^{N-2} &= \displaystyle\sum_{2\leq i_2<\dots<i_{N-1}\leq N} \sum_{\ell=1}^{N} \mathrm{z}^\ell((\underbrace{x_1, \dots, x_1}_{N-2 \text{ times}}, x_j)_{j\neq 1,i_2,\dots,i_{N-1}}, x_1) + \sum_{\ell=1}^{N} \mathrm{z}^\ell((x_1, \dots, x_1), x_1) \\ S^{N-1} &= \displaystyle\sum_{\ell=1}^{N} \mathrm{z}^\ell((x_1, \dots, x_1), x_1). \end{cases}$$

The telescopic sum then yields

$$\mathrm{z}((x_1, \dots, x_N), x_1) = \frac{1}{N} \sum_{p=0}^{N-1} (-1)^p S^p = \frac{1}{N} \sum_{\ell=1}^{N} \mathrm{z}^\ell((x_j)_{j\neq 1}, x_1) = \mathrm{z}^1((x_j)_{j\neq 1}, x_1),$$

where the last equality follows from (7.2.6). This shows the property (7.2.5). □

7.3 Symmetric neural networks

7.3.1 DeepSets and variants

In view of Lemma 7.2.1 and 7.2.2, we shall consider a class of neural networks (NN in short) that satisfy the exchangeability and D-exchangeability properties for approximating the solution (and its gradient) to the PDE (7.2.1).

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 result of 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 (see [CUH16]), or tanh. When ρ is the identity function, we simply write $\mathcal{L}_{d_1,d_2}$.

We then define

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

as the set of feedforward (or artificial) neural networks with input layer dimension d_0, output layer dimension k, and ℓ hidden layers with m neurons (or units). These numbers d_0,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.

A *symmetric neural network* function, denoted $\mathcal{U}\in\mathcal{S}^{\mathfrak{s},N,\rho}_{d,\ell,m,k,d'}$, is an $\mathbb{R}^{d'}$-valued exchangeable function to the order N on $\mathbb{R}^d$, in the form:

$$\mathcal{U}(\boldsymbol{x}) = \ \psi\big(\mathfrak{s}((\varphi(x_i))_{i\in[\![1,N]\!]})\big), \quad \text{for } \boldsymbol{x}=(x_i)_{i\in[\![1,N]\!]}\in(\mathbb{R}^d)^N, \tag{7.3.1}$$

where $\varphi\in\mathcal{N}^{\rho}_{d,\ell,m,k}$, $\psi\in\mathcal{N}^{\rho}_{k,\ell,m,d'}$ (here, for simplicity of notations, we assume that the number of hidden layers and neurons of φ and ψ are the same but in practical implementation, they may be different), and $\mathfrak{s}$ is a given $\mathbb{R}^k$-valued exchangeable function to the order N on $\mathbb{R}^k$, typically:

- Max-pooling (component-wise): $\mathfrak{s}(\boldsymbol{y}) = \max(y_i)_{i\in[\![1,N]\!]}$,
- Sum: $\mathfrak{s}(\boldsymbol{y}) = \sum_{i=1}^N y_i$, or average: $\mathfrak{s}(\boldsymbol{y}) = \frac{1}{N}\sum_{i=1}^N y_i$,

for $\boldsymbol{y}=(y_i)_{i\in[\![1,N]\!]}\in(\mathbb{R}^k)^N$. When $\mathfrak{s}$ is the max-pooling function, $\mathcal{S}^{\mathfrak{s},N,\rho}_{d,\ell,m,k,d'}$ is called *PointNet*, as introduced in [Qi+17], while for $\mathfrak{s}$ equals to the sum/average function, it is called *DeepSet*, see [Zah+17]. The architecture is described in Figure 7.1, and k can be interpreted as a number of features describing the geometry of the set of points $\{x_i\}_{i\in[\![1,N]\!]}$. For example in the context of mean-field control problem, k will be related to the moments for describing the law of the McKean-Vlasov SDE.

A given symmetric network function $\mathcal{U}\in\mathcal{S}^{\mathfrak{s},N,\rho}_{d,\ell,m,k,d'}$ is determined by the weight/bias parameters $\theta=(\theta^{(1)},\theta^{(2)})$ with $\theta^{(1)}=(\mathcal{W}^{(1)}_0,\beta^{(1)}_0,\dots,\mathcal{W}^{(1)}_\ell,\beta^{(1)}_\ell)$ defining the layer functions $\phi_0\dots,\phi_\ell$ of φ, and $\theta^{(2)}=(\mathcal{W}^{(2)}_0,\beta^{(2)}_0,\dots,\mathcal{W}^{(2)}_\ell,\beta^{(2)}_\ell)$ defining the layer functions $\psi_0\dots,\psi_\ell$ of ψ. The number of parameters is $M=M_1+M_2$, with $M_1=m(d+1)+m(m+1)(\ell-1)+(m+1)k$, $M_2=(k+1)m+m(m+1)(\ell-1)+(m+1)d'$, and we observe that it does not depend on the number N of inputs.

Remark 7.3.1 (*Time dependent symmetric network).* A time-dependent symmetric in space neural network can be constructed as

$$\mathcal{U}(t,\boldsymbol{x}) = \qquad \psi\big(t,\mathfrak{s}((\varphi(x_i))_{i\in[\![1,N]\!]})\big), \quad \text{for } t\in\mathbb{R}_+,\ \boldsymbol{x}=(x_i)_{i\in I}\in(\mathbb{R}^d)^N,$$

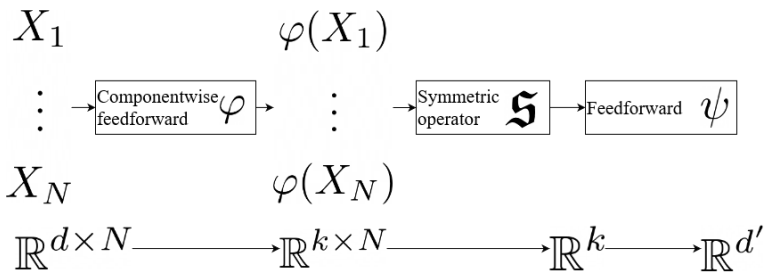


Figure 7.1: Architecture of a symmetric neural network.

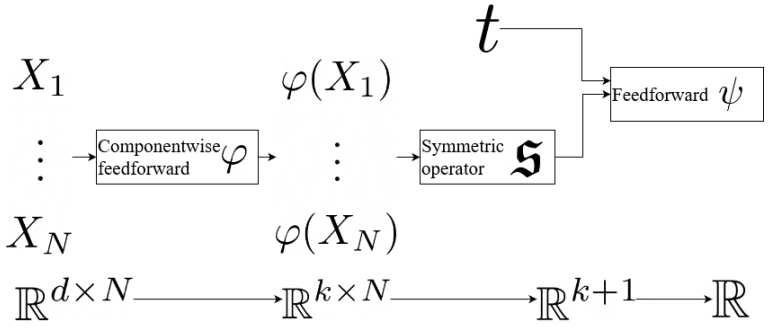


Figure 7.2: Architecture of time dependent symmetric network.

with φ a feedforward network from $\mathbb{R}^d$ into $\mathbb{R}^k$, and ψ is a feedforward from $\mathbb{R}^{k+1}$ into $\mathbb{R}^{d'}$, where we add time as an additional feature, see architecture in Figure 7.2.

□

A *D-symmetric neural network* function, denoted $\mathcal{Z} \in D\mathcal{S}^{\mathfrak{s},N,\rho}_{d,\ell,m,k,d'}$, is an $\mathbb{R}^{d'}$-valued D-exchangeable function in the form

$$\mathcal{Z}(\boldsymbol{x}, x) = \psi\big(\mathfrak{s}((\varphi(x_i))_{i\in[\![1,N]\!]}), x\big), \quad \text{for } \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N, x \in \mathbb{R}^d,$$

where $\varphi \in \mathcal{N}^{\rho}_{d,\ell,m,k}$, $\psi \in \mathcal{N}^{\rho}_{k+d,\ell,m,d'}$, and $\mathfrak{s}$ is a given $\mathbb{R}^k$-valued N-exchangeable function on $\mathbb{R}^k$. The number of parameters of a given $\mathcal{Z} \in D\mathcal{S}^{\mathfrak{s},\rho}_{d,\ell,m,k,d'}$ is $M' = M'_1 + M'_2$, with $M'_1 = m(d+1) + m(m+1)(\ell-1) + (m+1)k$, $M'_2 = m(k+d+1) + m(m+1)(\ell-1) + (m+1)d'$. We shall often take for $\mathfrak{s}$ the average function, and call $D\mathcal{S}^{\mathfrak{s},N,\rho}_{d,\ell,m,k,d'}$ as *DeepDerSet*. Its architecture is given in Figure 7.3. Given a D-symmetric neural network $\mathcal{Z}$, we denote by $\boldsymbol{\mathcal{Z}}$ the function defined on $(\mathbb{R}^d)^N$ by $\boldsymbol{\mathcal{Z}}(\boldsymbol{x}) = (\mathcal{Z}(\boldsymbol{x}, x_i))_{\in[\![1,N]\!]} \in (\mathbb{R}^d)^N$, and by misuse of notation, we may also call $\boldsymbol{\mathcal{Z}}$ as a D-symmetric NN. By construction, these networks respect the representation given by Lemma 7.2.2, by being defined as a D-exchangeable function applied component by component. In that way we are able to enforce the correct symmetries for representing both a symmetric function and its derivative, which will be useful in Section 7.4 and Section 7.5 when looking for the gradient of PDE solutions.

Alternatively, one can generate D-exchangeable functions as follows. Starting from a DeepSet element $\mathcal{U} \in \mathcal{S}^{\mathfrak{s},N,\rho}_{d,\ell,m,k,d'}$ as in (7.3.1) with $\mathfrak{s}$ the sum function, and network functions φ, ψ with smooth activation functions

$$D_{x_i}\mathcal{U}(\boldsymbol{x}) = \mathrm{D}\mathcal{U}(\boldsymbol{x}, x_i), \quad \boldsymbol{x} = (x_i)_{\in[\![1,N]\!]} \in (\mathbb{R}^d)^N,$$

with

$$\mathrm{D}\mathcal{U}(\boldsymbol{x}, x) := D\varphi(x) D\psi\big(\mathfrak{s}((\varphi(x_i))_{i\in I})\big), \quad x \in \mathbb{R}^d.$$

The set of D-exchangeable functions obtained from differentiation of DeepSet network functions is called *AD-DeepSet*, where AD stands for automatic differentiation, and Automatic refers to

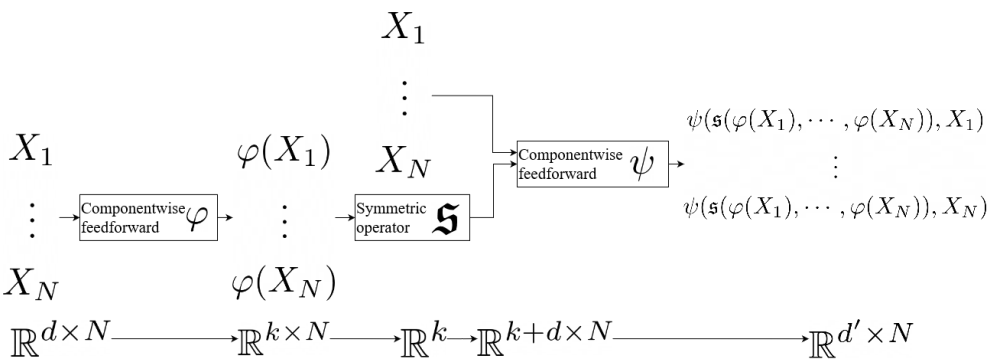


Figure 7.3: Architecture of DeepDerSet network

the implementation of the differentiation in software library, e.g. in TensorFlow. This alternative network for approximating the derivative of a differentiable symmetric function naturally respects the structure given by Lemma 7.2.2, by construction as the derivative of a symmetric DeepSet neural network.

Given a D-symmetric NN $\boldsymbol{\mathcal{Z}}$ in *DeepDerSet* or in *AD-DeepSet* with smooth activation functions, we denote by $D\boldsymbol{\mathcal{Z}}$ its differentiation.

As for the well-known universal approximation theorem [Hor91] for neural networks, we have a similar result for symmetric neural networks, which states that any exchangeable function can be arbitrarily approximated by a PointNet or DeepSet given enough neurons. More precisely, by combining Theorem 2.9 of [Wag+19] with Theorem 2 of [Hor91], we obtain the following approximation theorem for DeepSets.

Universal approximation for DeepSets networks. Let $\mathfrak{s}$ be the sum function. The set $\cup_{m=1}^{\infty}\mathcal{S}_{d,\ell,m,N+1,d'}^{\mathfrak{s},N,\rho}$ approximates any N-exchangeable continuous function w arbitrary well on any compact set of $K \subset \mathbb{R}^d$, once ρ is continuous, bounded and non-constant: for all $\varepsilon > 0$, $N \in \mathbb{N}^*$, there exists $\mathcal{U} \in \cup_{m=1}^{\infty}\mathcal{S}_{d,\ell,m,N+1,d'}^{\mathfrak{s},N,\rho}$ such that

$$\big|w(\boldsymbol{x}) - \mathcal{U}(\boldsymbol{x})\big| \leq \varepsilon \quad \forall \boldsymbol{x} \in K^N.$$

Note that a priori the latent space dimension k has to be chosen equal to $N+1$.

Alternatively, by combining Theorem 1 of [Qi+17] with Theorem 2 of [Hor91], we obtain the following one-dimensional approximation theorem for PointNet.

Universal approximation for PointNet networks. Let $\mathfrak{s}$ be the max function. The set $\cup_{m=1}^{\infty}\cup_{k=1}^{\infty}\mathcal{S}_{1,\ell,m,k,d'}^{\mathfrak{s},N,\rho}$ approximates any N-exchangeable Hausdorff continuous function w (seen as a function on sets) arbitrary well on any compact set of $K \subset \mathbb{R}$, once ρ is continuous, bounded, and non-constant: for all $\varepsilon > 0$, $N \in \mathbb{N}^*$, there exists $\mathcal{U} \in \cup_{m=1}^{\infty}\cup_{k=1}^{\infty}\mathcal{S}_{1,\ell,m,k,d'}^{\mathfrak{s},N,\rho}$ such that

$$\big|w(S) - \mathcal{U}(\boldsymbol{x})\big| \leq \varepsilon, \quad \forall S \subset K,\ S = \{x_1,\cdots,x_N\}.$$

Note here that a priori the latent space dimension k has to be chosen as large as needed.

7.3.2 Comparison tests

In this paragraph, we test the accuracy of the approximation of exchangeable functions by *DeepSet* or *PointNet*, and also the approximation of D-exchangeable functions by *DeepDerSet* or *AD-DeepSet*, and compare numerically with classical feedforward approximations.

Approximation of some simple functions

We first test the approximation of the following simple symmetric functions:

1. $f(x) = \exp(2\bar{x} + 3\bar{x}^3)$, with $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ (case 1)

2. $f(x) = \frac{1}{N}\sum_{i=1}^{N}\big[\sin(x_i)1_{x_i<0} + x_i 1_{x_i\geq 0}\big]$ (case 2)

3. $f(x) = \bar{x} + 2\bar{x}^2 + 3\bar{x}^3$, with $\bar{x} = \max\{x_i, i = 1, \ldots, N\}$ (case 3)

4. $f(x) = \cos(2\bar{x} + 3\bar{x}^2)$, with $\bar{x} = \sum_{i=1}^{N} x_i$ (case 4)

We use a symmetric neural network architecture as proposed in [Qi+17], [Zah+17]:

- First, a feedforward network φ with $\ell = 5$ hidden layers, and respectively $64, 64, 64, 128$ and 1024 neurons such that each dimension i, $i = 1, \ldots, N$, is treated with the same network in one dimension avoiding to break the symmetry.
- Two possible symmetric functions $\mathfrak{s}$ to the order N on $\mathbb{R}^k$ with $k = 1024$, the max-pooling (*PointNet*) and the sum function (*DeepSet*).
- At last, a feedforward network ψ from $\mathbb{R}^{1024}$ to $\mathbb{R}$ with $\ell = 2$ hidden layers, and respectively 512 and 256 neurons.

For the approximation with classical feedforward networks, we used three or four layers and a number of neurons constant per layer equal to $10 + d$, or $10 + 2d$ neurons.

We use the ADAM optimizer ([KB14]), with a batch size equal to 300 for solving the approximation problem with quadratic loss function:

$$\min_{\theta} \mathbb{E}[|f(X) - \mathcal{U}^{\theta}(X)|^2], \tag{7.3.2}$$

with training simulations from $X \sim \mathcal{N}(0_N, 1_N)$, and θ are the parameters of the network function $\mathcal{U}^{\theta}$. The number of epochs (corresponding to the number of gradient descent iterations) used is equal to 100. After epoch iterations of the stochastic gradient, the error (7.3.2) is estimated with 20000 simulations. If the error is below a threshold equal to 1e−5 the optimization is stopped, otherwise a counter for outer iterations is incremented. The number of outer iterations is blocked at epochExt = 1000 (meaning a maximal total number of stochastic gradient iterations equal to epoch × epochExt = 100000).

In Tables 7.1, 7.2, 7.3, we report the accuracy reached (Error) and the number of iterations (Iter.) used to obtain this given accuracy: then a threshold equal to 1e−5 means that the optimization has been successful and the relevant parameter is the number of iterations used. A number of iterations equal to expochExt = 1000 means that the optimization has not been successful and the error reached indicates how far we are from optimality. For the feedforward case, we report the best result ("minimum" in table) and the worse result ("maximum" in table) obtained changing the number of layers and the number of neurons used.

The initial learning rate is taken equal to 1e−3 for first outer simulation in cases 1 and 2 with a linear decay to 1e−5 for a number of outer iterations equal to 1000. For test case 3, the initial learning rate is taken equal to 1e−4 with a linear decay to 1e−5. The result obtained in Table 7.1 is similarly obtained with a large number of functions tested in dimension between N = 10 to 1000. It shows the following results:

- The classical feedforward, with dense layers, often permits to obtain optimally without forcing symmetry of the solution,
- Classical feedforward results do not depend a lot on the number of layers, the number of neurons tested and the activation function used,

	Symmetric				Feedforward			
	PointNet		DeepSet		Minimum		Maximum	
Activation	Error	Iter.	Error	Iter.	Error	Iter.	Error	Iter.
ReLU	0.008	1000	1e−5	10	1e−5	125	1e−5	166
tanh	0.016	1000	1e−5	288	1e−5	180	1e−5	308
ELU	0.015	1000	1e−5	176	1e−5	108	1e−5	130

Table 7.1: Approximation error (7.3.2) obtained for different networks on one run and number of iterations used depending on activation functions for approximation of the function f in case 1, dimension $N = 100$.

Case	activ	Symmetric				Feedforward			
		PointNet		DeepSets		Minimum		Maximum	
		Error	Iter.	Error	Iter.	Error	Iter.	Error	Iter.
2	ReLU	0.001	1000	1e−5	5	1e−5	992	0.002	1000
2	tanh	0.03	1000	1e−5	342	0.0018	1000	6× 1e−5	1000
3	ReLU	0.001	1000	0.23	1000	88	1000	432	1000
3	tanh	0.002	1000	65	1000	933	1000	969	1000

Table 7.2: Approximation error (7.3.2) obtained for different networks on one run and number of iterations used for approximation of the function f in cases 2 and 3, dimension $N = 100$, activation function ReLU.

- For symmetric approximations, DeepSets generally permits to get the best results and the ReLU activation function is the best out of the three tested.

In the sequel, we drop the ELU activation function on other cases as shown in Tables 7.2 for cases 2 and 3. Notice that case 3, involving a max function is the only one where PointNet approximation gives the best result among the other tested networks. On cases 2 and 3 in dimension 100, the DeepSets approximation outperforms the classical feedforward network for all the number of layers and neurons tested. However, results on case 3 are not very good for the PointNet approximation even with the ReLU activation function.

Results for test case 4 are given on Table 7.3 using an initial learning rate equal to 5e−5 and a decay linear to 5e−6 with the number of outer iterations. At last, considering case 4 in dimension $N = 1000$, when the function is quickly changing, we see that the classical feedforward network functions have difficulty to converge while the DeepSets network approximation converges. The latter turns out to be a very good candidate to some very high dimensional PDEs when there is symmetry in the solution.

Approximation of a function of t and x with symmetry in x

We test the accuracy of our time dependent symmetric neural network by considering the following two cases of functions:

	Symmetric				Feedforward			
	PointNet		DeepSets		Minimum		Maximum	
activ	Error	Iter.	Error	Iter.	Error	Iter.	Error	Iter.
ReLU	0.1910	1000	8.9e−5	1000	0.0045	1000	0.01	1000
tanh	0.19	1000	4e−5	1000	0.0006	1000	0.0012	1000

Table 7.3: Approximation error (7.3.2) obtained for different networks, activation functions for approximation of the function f in case 4 dimension 1000.

Case	DeepSets		Feedforward	
	Error	Iter.	Error	Iter.
1	1e−5	67	0.008	1000
2	1e−5	344	0.048	1000

Table 7.4: Approximation error (7.3.3) obtained for different networks on one run and number of iterations used for approximation of the function f in case 1 and 2, dimension $N = 100$.

1. $f(x) = \exp\left(\bar{x}(t + 2t^2) + 3t\bar{x}^3\right)$ with $\bar{x} = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} x_i$ (case 1)

2. $f(x) = t + \cos(t\bar{x})$, with $\bar{x} = \frac{1}{\sqrt{N}}\sum_{i=1}^{N} x_i$ (case 2)

The approximation is performed through the minimization problem

$$\min_{\theta} \mathbb{E}[|f(\tau, X) - \mathcal{U}^{\theta}(\tau, X)|^2], \tag{7.3.3}$$

with training simulations from $X \sim \mathcal{N}(0_N, 1_N)$, and an independent uniform law for τ on $[0, 1]$, and where $\mathcal{U}^{\theta}$ is a time-dependent DeepSet with parameters θ. We keep the same number of neurons and layers as in the previous section, and compare with a classical feedforward network composed of 3 layers of $d + 10$ neurons. In all experiments, we use a ReLU activation function.

In Table 7.4, we give the results obtained in dimension 100. Surprisingly, the feedforward approximation seems to have difficulties to approximate the case 1 although it is quite similar to case one in the previous section. As for the second case, the result is not so surprising as the case is quite similar to case 4 in previous section, where the feedforward network has difficulties to converge.

Gradient approximation

We now focus on the approximation of the derivative of an exchangeable function by means of $\mathcal{U}^{\theta}$ a *DeepDerSet*, a *AD-DeepSet*, or a classical feedforward network .

The minimization problem is now:

$$\min_{\theta} \mathbb{E}[\|Df(X) - \mathcal{U}^{\theta}(X)\|^2], \tag{7.3.4}$$

where the norm $\|\cdot\|$ is the Euclidean norm.

The comparison is performed on the following test functions:

1. $f(x) = \exp\left(\bar{x} + \bar{x}^3\right)(1_N + 3x^2)$ where $\bar{x} = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} x_i$ (case 1)

2. $f(x) = y$ where $y_i = 1_{x_i>0} + cos(x_i)1_{x_i<0}, \quad i = 1, \ldots, N$ (case 2)

3. $f(x) = \frac{1}{\sqrt{N}}\sin(\bar{x})1_N$, where $\bar{x} = \frac{1}{\sqrt{N}}\sum_{i=1}^{N} x_i$ (case 3)

We compare the classical feedforward approximation to our network approximation in Tables 7.5 and 7.6, using a maximal number of iterations equal to 5000. Clearly using a ReLU activation function is superior to the tanh activation function and the *DeepDerSet* gives the best approximation while the *AD-DeepSet* or the feedforward may have difficulties to approximate the functions accurately.

Case	N	Feedforward		DeepDerSet		AD-DeepSet	
		Error	Iter.	Error	Iter.	Error	Iter.
1	10	5e−4	5000	5e−4	5000	7e−3	5000
1	100	1e−5	300	1e−5	250	1e−5	50
2	10	0.03	5000	1e−5	550	2e−4	5000
2	100	0.12	5000	1e−5	450	1e−4	5000
3	10	1e−5	3800	1e−5	850	0.03	5000
3	100	1e−5	1850	1e−5	700	3e−3	5000

Table 7.5: Approximation error (7.3.4) with ReLU activation function obtained for different networks on one run and number of iterations used for approximation of the derivative of an exchangeable function.

Case	N	Feedforward		DeepDerSet		AD-DeepSet	
		Error	Iter.	Error	Iter.	Error	Iter.
1	10	5e−4	5000	1.7e−4	5000	3e−3	5000
1	100	1e−5	500	1e−5	100	1e−5	50
2	10	0.03	5000	1e−5	950	0.028	5000
2	100	0.12	5000	1e−5	700	0.56	5000
3	10	1.5e−5	5000	1e−5	1850	0.02	5000
3	100	1e−5	2100	1e−5	1350	4e−3	5000

Table 7.6: Approximation error (7.3.4) with tanh activation function obtained for different networks on one run and number of iterations used for approximation of the derivative of an exchangeable function.

7.4 Numerical schemes

We now adapt the deep backward dynamic programming (DBDP) schemes developed in [HPW20] and [PWG21] for solving nonlinear PDEs, by using symmetric neural networks and D-symmetric neural networks (instead of feedforward neural networks) for approximating the exchangeable solution v and its gradient $D_{\boldsymbol{x}}v$. We recall the main steps of the DBDP scheme, and distinguish the case of semi-linear and fully non-linear PDEs.

7.4.1 Semi-linear PDE

We first consider the case where the generator F in (7.2.1) may be decomposed into the form

$$F(t,\boldsymbol{x},y,\boldsymbol{z},\boldsymbol{\gamma}) \;=\; H(t,\boldsymbol{x},y,\boldsymbol{z}) + \sum_{i=1}^{N} b_i(t,\boldsymbol{x}).z_i + \frac{1}{2}\sum_{i,j=1}^{N} \mathrm{tr}\big(\Sigma_{ij}(t,\boldsymbol{x})\gamma_{ij}\big),$$

for $t \in [0,T]$, $\boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N$, $y \in \mathbb{R}$, $\boldsymbol{z} = (z_i)_{i\in[\![1,N]\!]} \in (\mathbb{R}^d)^N$, and $\boldsymbol{\gamma} = (\gamma_{ij})_{i,j\in[\![1,N]\!]} \in \mathbb{S}^N(\mathbb{S}^d)$. Here, H is a function on $[0,T]\times(\mathbb{R}^d)^N\times\mathbb{R}\times(\mathbb{R}^d)^N$ satisfying the permutation-invariance condition:

$$H(t,\boldsymbol{x},y,\boldsymbol{z}) \;=\; H(t,\pi[\boldsymbol{x}],y,\pi[\boldsymbol{z}]), \quad \forall\pi\in\mathfrak{S}_N,$$

the coefficients b_i, $i=1,\dots,N$, are $\mathbb{R}^d$-valued functions on $[0,T]\times(\mathbb{R}^d)^N$ satisfying the condition

$$b_i(t,\pi[\boldsymbol{x}]) = \; b_{\pi(i)}(t,\boldsymbol{x}), \quad \forall\pi\in\mathfrak{S}_N, \tag{7.4.2}$$

and the coefficients Σ_{ij}, $i,j=1,\dots,N$, are $d\times d$-matrix valued functions on $[0,T]\times(\mathbb{R}^d)^N$ in the form

$$\Sigma_{ij}(t,\boldsymbol{x}) = \; \sigma_{ij}\sigma_{ij}^{\intercal}(t,\boldsymbol{x}) + \sigma_{i0}\sigma_{j0}^{\intercal}(t,\boldsymbol{x}), \tag{7.4.3}$$

for some $d \times d$-matrix valued functions σ_{ij}, and $d \times q$-matrix valued functions σ_{i0}, satisfying the invariance property: for all $\pi \in \mathfrak{S}_N$,

$$\sigma_{ij}(t, \pi[\boldsymbol{x}]) \;=\; \sigma_{\pi(i)\pi(j)}(t, \boldsymbol{x}), \qquad \sigma_{i0}(t, \pi[\boldsymbol{x}]) \;=\; \sigma_{\pi(i)0}(t, \boldsymbol{x}). \tag{7.4.4}$$

In this case, the permutation-invariance condition **(HI)** on F is satisfied, and we observe that it includes Example 7.2.1 of multi-asset pricing with $H(t, \boldsymbol{x}, y, \boldsymbol{z}) = \beta(y^+ - y)$ (in the case of the CVA pricing), $b_i \equiv r$ and $\sigma_{i0} \equiv 0$. This also includes Example 7.2.2 of the McKean-Vlasov control problem under common noise with uncontrolled diffusion coefficient, where $\sigma_{ij}(t, \boldsymbol{x}) = \sigma(t, x_i, \bar{\mu}(\boldsymbol{x}))\delta_{ij}$, $\sigma_{i0}(t, \boldsymbol{x}) = \sigma_0(t, x_i, \bar{\mu}(\boldsymbol{x}))$, and

$$\begin{aligned} H(t, \boldsymbol{x}, y, \boldsymbol{z}) = -ry + \sum_{i=1}^N \inf_{a \in A} \big[\beta(t, x_i, \bar{\mu}(\boldsymbol{x}), a).z_i + \frac{1}{N} f(x_i, \bar{\mu}(\boldsymbol{x}), a)\big] \\ - \sum_{i=1}^N b_i(t, \boldsymbol{x}).z_i, \end{aligned} \tag{7.4.5}$$

for any function b_i satisfying (7.4.2).

We shall discuss more in detail the relevant choice of the drift coefficient b_i in Section 7.4.3.

The starting point of the numerical scheme is the probabilistic representation of the PDE (7.2.1) with F as in (7.4.1) in terms of a forward backward stochastic differential equation (FBSDE), as in [PP90]. In our context, the forward system is described by the process $\boldsymbol{X} = (X^1, \ldots, X^N)$ valued in $(\mathbb{R}^d)^N$ governed by the diffusion dynamics:

$$dX_t^i = \; b_i(t, \boldsymbol{X}_t)dt + \sum_{j=0}^N \sigma_{ij}(t, \boldsymbol{X}_t)dW_t^j, \tag{7.4.6}$$

where W^i, $i = 1, \ldots, N$, are independent d-dimensional Brownian motions, independent of the q-dimensional Brownian motion W^0. Given this forward diffusion process, we then consider the pair process $(Y, \boldsymbol{Z} = (Z^i)_{i \in [\![1,N]\!]})$ valued in $\mathbb{R} \times (\mathbb{R}^d)^N$ solution to the BSDE

$$\begin{aligned} G(\boldsymbol{X}_T) - Y_t + \int_t^T H(s, \boldsymbol{X}_s, Y_s, \boldsymbol{Z}_s)ds \\ - \sum_{i=1}^N \sum_{j=0}^N \int_t^T (Z_s^i)^\intercal \sigma_{ij}(s, \boldsymbol{X}_s)dW_s^j \;=\; 0, \qquad 0 \le t \le T, \end{aligned} \tag{7.4.7}$$

which is connected by Itô's formula to the solution of the PDE (7.2.1) via:

$$Y_t \;=\; v(t, \boldsymbol{X}_t), \qquad Z_t^i \;=\; D_{x_i} v(t, \boldsymbol{X}_t),\; i = 1, \ldots, N, \quad 0 \le t \le T.$$

We next consider a time discretization of this FBSDE on a time grid $\{t_k, k = 0, \ldots, N_T\}$, with $t_0 = 0$, $t_{N_T} = T$, $\Delta t_k := t_{k+1} - t_k > 0$, by defining the Euler scheme $\{\boldsymbol{X}_k^{N_T} = (X_k^{i,N_T})_{i \in [\![1,N]\!]}$, $k = 0, \ldots, N_T\}$ associated to the forward diffusion process $\{\boldsymbol{X}_t = (X_t^i)_{i \in [\![1,N]\!]}, 0 \le t \le T\}$, which is used for the training simulations, together with the increments of the Brownian motions: $\Delta W_k^j := W_{t_{k+1}}^j - W_{t_k}^j$, $k = 0, \ldots, N_T - 1$, $j = 0, \ldots, N$, of our numerical backward scheme. The DBDP algorithm reads then as follows:

The output of the DBDP scheme provides approximations $\widehat{\mathcal{U}}_k(\boldsymbol{x})$ of $v(t_k, \boldsymbol{x})$, and $\widehat{\boldsymbol{\mathcal{Z}}}_k(\boldsymbol{x})$ of $D_{\boldsymbol{x}} v(t_k, \boldsymbol{x})$, $k = 0, \ldots, N_T - 1$, for values of $\boldsymbol{x} \in (\mathbb{R}^d)^N$ that are well-explored by the training simulations of $\boldsymbol{X}_k^{N_T}$. We refer to [PWG21] (see their section 3.1) for a discussion on the choice of the algorithm hyperparameters.

Remark 7.4.1. *We stress that the neural networks do not take time as an input. Adding time would not make any difference because the training is done locally, time step per time step. So the*

Algorithm 8: DBDP scheme with symmetric NN

Initialization: Initialize from the exchangeable function: $\widehat{\mathcal{U}}_{N_T}(\cdot) = G(\cdot)$
for $k = N_T - 1, \dots, 0$ **do**
minimize over symmetric NN $\mathcal{U}_k$, and D-symmetric NN $\mathcal{Z}_k$, the quadratic loss function

$$J_1(\mathcal{U}_k, \mathcal{Z}_k) = \mathbb{E}\Big|\widehat{\mathcal{U}}_{k+1}(\boldsymbol{X}_{k+1}^{N_T}) - \mathcal{U}_k(\boldsymbol{X}_k^{N_T}) + H\big(t_k, \boldsymbol{X}_k^{N_T}, \mathcal{U}_k(\boldsymbol{X}_k^{N_T}), \boldsymbol{\mathcal{Z}}_k(\boldsymbol{X}_k^{N_T})\big)\Delta t_k - \sum_{i=1}^{N}\sum_{j=0}^{N}\big(\mathcal{Z}_k(\boldsymbol{X}_k^{N_T}, X_k^{i,N_T})\big)^\intercal \sigma_{ij}\big(t_k, \boldsymbol{X}_k^{N_T}\big)\Delta W_k^j\Big|^2,$$

and update $(\widehat{\mathcal{U}}_k, \widehat{\mathcal{Z}}_k)$ as the solution to this minimization problem.
end

neural networks approximating $\mathcal{U}_k$ and $\mathcal{Z}_k$ would be trained with only t_k as an input and hence they would not be able to learn the dependence on time. However, at time $k < N_T$, we initialize the parameters of $\mathcal{U}_k$ and $\mathcal{Z}_k$ respectively with the parameters of the neural networks for $\mathcal{U}_{k+1}$ and $\mathcal{Z}_{k+1}$, which have been trained at the previous iteration. This gives a good initial guess for the neural networks at time t_k and leads to more efficient training.

7.4.2 Fully nonlinear PDE

We consider more generally the fully non-linear PDE (7.2.1) with a symmetric generator F satisfying **(HI)**. We adapt the machine learning scheme in [PWG21] for solving fully nonlinear PDEs by exploiting furthermore the exchangeability property of the solution by using again symmetric neural networks as in the semi-linear case.

We fix some arbitrary drift and diffusion coefficients b_i, σ_{ij}, $i = 1, \dots, N$, $j = 0, \dots, N$, satisfying invariance properties as in (7.4.2)-(7.4.4) (in practice, they should be chosen depending on the studied problem as for the semi-linear case, see a general discussion in Section 3.1 in [PWG21], and an application in Section 7.5.3), and introduce the forward diffusion system $\boldsymbol{X}$ as in (7.4.6) and its discrete-time Euler scheme $\boldsymbol{X}^{N_T}$. We then consider the triple process $(Y, \boldsymbol{Z} = (Z^i)_{i\in[\![1,N]\!]}, \boldsymbol{\Gamma} = (\Gamma^{ij})_{i,j\in[\![1,N]\!]})$ valued in $\mathbb{R}\times(\mathbb{R}^d)^N \times \mathbb{S}^N(\mathbb{S}^d)$ solution to the BSDE

$$G(\boldsymbol{X}_T) - Y_t + \int_t^T F_{b,\sigma}(s, \boldsymbol{X}_s, Y_s, \boldsymbol{Z}_s, \boldsymbol{\Gamma}_s)ds - \sum_{i=1}^{N}\sum_{j=0}^{N}\int_t^T (Z_s^i)^\intercal \sigma_{ij}(s, \boldsymbol{X}_s)dW_s^j = 0, \quad 0 \le t \le T,$$

with

$$F_{b,\sigma}(t, \boldsymbol{x}, y, \boldsymbol{z}, \boldsymbol{\gamma}) := F(t, \boldsymbol{x}, y, \boldsymbol{z}, \boldsymbol{\gamma}) - \sum_{i=1}^{N} b_i(t, \boldsymbol{x}).z_i - \frac{1}{2}\sum_{i,j=1}^{N}\mathrm{tr}\big(\Sigma_{ij}(t, \boldsymbol{x})\gamma_{ij}\big),$$

and Σ_{ij} as in (7.4.3). It is connected by Itô's formula to the fully non-linear PDE (7.2.1) via the representation: $Y_t = v(t, \boldsymbol{X}_t)$, $\boldsymbol{Z}_t = D_{\boldsymbol{x}}v(t, \boldsymbol{X}_t)$, $\boldsymbol{\Gamma}_t = D^2_{\boldsymbol{x}}v(t, \boldsymbol{X}_t)$, $0 \le t \le T$.

Assuming that G is smooth, the algorithm is designed in Algorithm 9.

7.4.3 The case of mean-field PDEs

We consider in this section the case where the PDE (7.2.1) is the particles approximation of a McKean-Vlasov control problem with a Bellman equation (7.2.4) in the Wasserstein space of

Algorithm 9: Fully nonlinear DPBD scheme with symmetric NN

Initialization: Initialize from the exchangeable function: $\widehat{\mathcal{U}}_{N_T}(\cdot) = G(\cdot)$ and the D-exchangeable function $\widehat{\boldsymbol{\mathcal{Z}}}_{N_T}(\cdot) = DG(\cdot)$.

for $k = N_T - 1, \ldots, 0$ **do**

minimize over symmetric NN $\mathcal{U}_k$, and D-symmetric NN $\mathcal{Z}_k$, the quadratic loss function

$$\begin{aligned} J_2(\mathcal{U}_k, \mathcal{Z}_k) = \mathbb{E}\Big| & \widehat{\mathcal{U}}_{k+1}(\boldsymbol{X}_{k+1}^{N_T}) - \mathcal{U}_k(\boldsymbol{X}_k^{N_T}) \\ & + F_{b,\sigma}\big(t_k, \boldsymbol{X}_k^{N_T}, \mathcal{U}_k(\boldsymbol{X}_k^{N_T}), \boldsymbol{\mathcal{Z}}_k(\boldsymbol{X}_k^{N_T}), D\widehat{\boldsymbol{\mathcal{Z}}}_{k+1}(\boldsymbol{X}_{k+1}^{N_T})\big)\Delta t_k \\ & - \sum_{i=1}^{N}\sum_{j=0}^{N} \big(\mathcal{Z}_k(\boldsymbol{X}_k^{N_T}, X_k^{i,N_T})\big)^\intercal \sigma_{ij}\big(t_k, \boldsymbol{X}_k^{N_T}\big)\Delta W_k^j\Big|^2, \end{aligned}$$

and update $(\widehat{\mathcal{U}}_k, \widehat{\mathcal{Z}}_k)$ as the solution to this minimization problem. Here $\mathrm{D}\widehat{\boldsymbol{\mathcal{Z}}}_{k+1}$ is the automatic differentiation of the D-symmetric NN $\widehat{\boldsymbol{\mathcal{Z}}}_{k+1}$ computed previously at the time step $k+1$.

end

probability measures as described in Example 7.2.2. To simplify the presentation, we consider that there is only control on the drift coefficient $\beta(t, x, \mu, a)$ but no control on the diffusion coefficient $\sigma(t, x, \mu)$ and $\sigma_0(t, x, \mu)$ of the McKean-Vlasov equation (7.2.3). In this case, recall that when the solution $v(t, \mu)$ to this Bellman equation is smooth, an optimal control is given in feedback form by (see [PW17]):

$$\alpha_t^* \;=\; \hat{a}(t, X_t^*, \mathbb{P}^0_{X_t^*}, \partial_\mu v(t, \mathbb{P}^0_{X_t^*})(X_t^*)),$$

where $\hat{a}(t, x, \mu, z)$ is an argmin of $a \in A \mapsto \beta(t, x, \mu, a).z + f(x, \mu, a)$, and $X^* = X^{\alpha^*}$ is the optimal McKean-Vlasov state process.

Approximation of the optimal control by forward induction of the scheme. As proven in [GPW22c], the solution $(\boldsymbol{X}, Y, \boldsymbol{Z})$ to the FBSDE (7.4.6)-(7.4.7) provides an approximation with a rate of convergence $1/N$, when N goes to infinity, of the solution v to (7.2.4), and its L-derivative: $Y_t \simeq v(t, \bar{\mu}(\boldsymbol{X}_t))$, $NZ_t^i \simeq \partial_\mu v(t, \bar{\mu}(\boldsymbol{X}_t))(X_t^i)$. The drift coefficients b_i of the forward particles system $\boldsymbol{X}$ should be chosen in order to generate from training simulations a suitable exploration of the state space for getting a good approximation of the optimal feedback control. In practice, in a first step, one can choose $b_i(t, \boldsymbol{x}) = \beta(t, x_i, \bar{\mu}(\boldsymbol{x}), a_0)$, for some arbitrary value $a_0 \in A$ of the control. After a first implementation of Algorithm 8, we thus have an approximation of $\partial_\mu v(t, \mu)(x)$ at time $t = t_k$, and $\mu = \bar{\mu}(\boldsymbol{x})$, by $N\widehat{\mathcal{Z}}_k(\boldsymbol{x}, x)$. Notice however that we solved the PDE along the law of the forward training process, which is different from the optimally controlled process law, except at the initial time t_0, where we then get an approximation of the optimal feedback control with

$$(x, \bar{\mu}(\boldsymbol{x})) \longmapsto \; \hat{a}(t_0, x, \bar{\mu}(\boldsymbol{x}), N\widehat{\mathcal{Z}}_0(\boldsymbol{x}, x)).$$

Next, by defining an updated initial drift coefficient as

$$\hat{b}_i(t_0, \boldsymbol{x}) := \; \beta\big(t_0, x_i, \bar{\mu}(\boldsymbol{x}), \hat{a}(t_0, x_i, \bar{\mu}(\boldsymbol{x}), N\widehat{\mathcal{Z}}_0(\boldsymbol{x}, x_i))\big), \;\; \text{for } \boldsymbol{x} = (x_i)_{i\in[\![1,N]\!]},\; i = 1, \ldots, N,$$

and considering the N-particle discrete-time system $\{\hat{\boldsymbol{X}}_k^{N_T} = (\hat{X}_k^{i,N_T})_{i\in[\![1,N]\!]}, k = 0, \ldots, N_T\}$, starting from i.i.d. samples X_0^i, $i = 1, \ldots, N$ distributed according to some distribution μ_0 on $\mathbb{R}^d$, and with dynamics

$$\begin{aligned} \hat{X}_1^{i,N_T} &= X_0^i + \hat{b}_i(t_0, \boldsymbol{X}_0)\Delta t_0 + \sigma(t_0, X_0^i, \bar{\mu}(\boldsymbol{X}_0))\Delta W_0^i, \\ \hat{X}_{k+1}^{i,N_T} &= \hat{X}_k^{i,N_T} + b_i(t_k, \boldsymbol{X}_k^{N_T})\Delta t_k + \sigma(t_k, \hat{X}_k^{i,N_T}, \bar{\mu}(\hat{\boldsymbol{X}}_k^{N_T}))\Delta W_k^i, \end{aligned}$$

for $k = 1, \dots, N_T - 1$, we obtain an approximation of the distribution of the optimal particle mean-field process at time t_1. Applying the algorithm again between t_1 and t_{N_T} then allows to compute an approximation of the optimal feedback control $\hat{a}(t_1, x, \bar{\mu}(\boldsymbol{x}), \widehat{\mathcal{Z}}_1(\boldsymbol{x}, x))$ at time t_1 and to update the simulation of $\hat{X}_2^{i,N_T}$. By induction, we can compute the optimal feedback control at every time step through N_T executions of the scheme.

Approximation of the solution by randomization of the training simulations. Algorithm 8 provides actually an approximation of $v(t, \mu)$ (resp. $\partial_\mu v(t, \mu)(x)$) at time t_k, and for empirical measures $\mu = \bar{\mu}(\boldsymbol{x})$, by $\widehat{\mathcal{U}}_k(\boldsymbol{x})$ (resp. $N\widehat{\mathcal{Z}}_k(\boldsymbol{x}, x)$). Thus, in order to get an approximation of $v(t_k, .)$ (resp. $\partial_\mu v(t_k, .)(x)$) on the whole Wasserstein space $\mathcal{P}_2(\mathbb{R}^d)$, we need a suitable exploration of $\bar{\mu}(\boldsymbol{X}_k^{N_T})$ when using the training simulations $\boldsymbol{X}_k^{N_T}$, $k = 0, \dots, N_T$. For that purpose, some randomization can first be implemented by randomizing the initial law μ_0 of the forward process. By sampling μ_0 in a compact set K of $\mathcal{P}_2(\mathbb{R}^d)$ for each batch element, such as a family of Gaussian measures for instance, our algorithm will be able to learn the value function $v(0, \mu)$ and its Lions derivative $\partial_\mu v(0, \mu)$ on K. Therefore, instead of solving the PDE several times for each initial law we can run the algorithm only once. This can be useful if we have an uncertainty in the initial law of the problem we aim to solve. It corresponds to learning the solution $v(t_k, \overline{\mu_{k,\ell}})$ on a family of empirical measures corresponding to forward processes $X^{i,(\ell),N_T}$, $i = 1, \dots, N$, with initial laws $\mu_0^\ell \in K$. Relying on the generalization properties of neural networks, we expect to approximate the value function at time $t_0 = 0$ on K. Furthermore, if the goal is to obtain an approximation of the PDE solution at any time step t_k, the task is more complex. A randomization needs to be performed at each time step t_k by sampling $\boldsymbol{X}_k^{N_T}$ according to a Gaussian mixture ν_k with random parameters. We then apply Algorithm 8, and expect to learn the solution over measures with regular densities. The updated method is presented in Algorithm 10. If the state space exploration is efficient, the feedback control will be directly available with only one execution of the algorithm, contrarily to the previously described procedure with N_T executions. We should explore the Wasserstein space well enough to learn the value function and its derivative on the unknown law of the optimal process.

7.5 Numerical results

In the different test cases, for the approximation of the solution v by means of symmetric neutral networks, we used DeepSets.

7.5.1 A toy example of symmetric PDE in very high dimension

We consider a symmetric semi-linear PDE:

$$\begin{cases} \partial_t v + b.D_{\boldsymbol{x}} v + \frac{1}{2}\mathrm{tr}(\sigma\sigma^\intercal D_{\boldsymbol{x}}^2 v) + f(\boldsymbol{x}, v, \sigma^\intercal D_{\boldsymbol{x}} v) \;=\; 0, \quad (t, \boldsymbol{x}) \in [0, T) \times \mathbb{R}^N \\ v(T, \boldsymbol{x}) \;=\; \cos(\bar{x}), \quad \text{with } \bar{x} = \sum_{i=1}^N x_i, \text{ for } \boldsymbol{x} = (x_1, \dots, x_N) \in \mathbb{R}^N, \end{cases}$$

with $b = 0.2/N$, $\sigma = \frac{I_N}{\sqrt{N}}$,

$$f(\boldsymbol{x}, y, \boldsymbol{z}) = (\cos(\overline{x}) + 0.2\sin(\overline{x}))\, e^{\frac{T-t}{2}} - \frac{1}{2}(\sin(\overline{x})\cos(\overline{x})e^{T-t})^2 + \frac{1}{2N}(y(1_N \cdot \boldsymbol{z}))^2.$$

so that the PDE solution is exchangeable and given by

$$v(t, \boldsymbol{x}) = \; \cos(\overline{x}) \exp\big(\frac{T-t}{2}\big).$$

We solve this PDE in dimension $N = 1000$ by using the deep backward scheme (DBDP) in [HPW20] with 60 time steps, and estimate $U_0 = v(0, 1_N)$ and $Z_0 = D_{\boldsymbol{x}} v(0, 1_N)$. For the approximation of v, and its gradient $D_{\boldsymbol{x}} v$, we test with three classes of networks:

Algorithm 10: DBDP scheme with symmetric NN and exploration of Wasserstein space

Initialization: Initialize from the exchangeable function: $\widehat{\mathcal{U}}_{N_T}(\cdot) = G(\cdot)$
for $k = N_T - 1, \dots, 0$ **do**
define random variables

$$L \sim U(1, L_{max}),\ \varphi_l \sim U(0,1),\ \mu_i \sim U(-\mu_{max}, \mu_{max}),\ (\theta_i)^2 \sim U(0, \sigma^2_{max})$$

define a random Gaussian mixture ν_k of random density

$$\frac{\sum_{l=1}^{L} \varphi_l \mathcal{N}(\mu_i, \theta_i^2)}{\sum_{l=1}^{L} \varphi_l}$$

define N i.i.d. particles X_k^{i,N_T} with law ν_k for $i = 1, \cdots, N$,
perform one Euler-Maruyama scheme step

$$X_{k+1}^{i,N_T} = X_k^{i,N_T} + b_i(t_k, \boldsymbol{X}_k^{N_T})\Delta t_k + \sigma\big(t_k, X_k^{i,N_T}, \bar{\mu}(\boldsymbol{X}_k^{N_T})\big)\Delta W_k^i,$$

minimize over symmetric NN $\mathcal{U}_k$, and D-symmetric NN $\mathcal{Z}_k$, the quadratic loss function (with H as in (7.4.5)):

$$\begin{aligned} J_1(\mathcal{U}_k, \mathcal{Z}_k) = \mathbb{E}\Big| & \widehat{\mathcal{U}}_{k+1}(\boldsymbol{X}_{k+1}^{N_T}) - \mathcal{U}_k(\boldsymbol{X}_k^{N_T}) \\ & + H\big(t_k, \boldsymbol{X}_k^{N_T}, \mathcal{U}_k(\boldsymbol{X}_k^{N_T}), \boldsymbol{\mathcal{Z}}_k(\boldsymbol{X}_k^{N_T})\big)\Delta t_k \\ & - \sum_{i=1}^{N} \big(\mathcal{Z}_k(\boldsymbol{X}_k^{N_T}, X_k^{i,N_T})\big)^\intercal \sigma\big(t_k, X_k^{i,N_T}, \bar{\mu}(\boldsymbol{X}_k^{N_T})\big)\Delta W_k^i \\ & - \sum_{i=1}^{N} \big(\mathcal{Z}_k(\boldsymbol{X}_k^{N_T}, X_k^{i,N_T})\big)^\intercal \sigma_0\big(t_k, X_k^{i,N_T}, \bar{\mu}(\boldsymbol{X}_k^{N_T})\big)\Delta W_k^0\Big|^2, \end{aligned}$$

and update $(\widehat{\mathcal{U}}_k, \widehat{\mathcal{Z}}_k)$ as the solution to this minimization problem.
end

Analytical		(i) AD-DeepSets		(ii) DeepDerSet		(iii) Feedforward	
U_0	Z_0	U_0	Z_0	U_0	Z_0	U_0	Z_0
0.9272	-1.3632	0.9289	-1.2973	0.90140	-1.304	0.6896	-1e-7

Table 7.7: PDE resolution in dimension 1000 with DBDP scheme [HPW20].

(i) DeepSet $\mathcal{U}$ for v, and AD-DeepSet D$\mathcal{U}$ for $D_{\boldsymbol{x}}v$ (DeepSets derivative case).

(ii) DeepSet for v, and DeepDerSet for $D_{\boldsymbol{x}}v$ (DeepDerSet case)

(iii) Feedforward for v and $D_{\boldsymbol{x}}v$ (Feedforward case)

For each of theses case, we use ReLU activation functions for all the networks, and for the feedforward network, we choose 3 layers of 1010 neurons.

Remark 7.5.1. An alternative to Case (i) is to consider an AD-DeepSet D$\mathcal{U}_1$ for $D_{\boldsymbol{x}}v$ with $\mathcal{U}_1$ another DeepSet independent of the one $\mathcal{U}$ used for v. □

We report the solution in Table 7.7.

We observe that the results with the feedforward network are not good. This is due to the fact that the feedforward network is not able to approximate correctly the final condition whatever the initial learning rate and the number of epochExt are taken, as already shown in Table 7.3. In contrast, we see that the AD-DeepSets and DeepDerSet networks give good results but only when the initial learning rate is taken small enough (here we took 1e−5). Finally, we have tested the Deep BSDE method in [HJE18] with the variation proposed in [CWNMW19] using a network reported in section 7.3.2. The results are unstable and so we do not report them. A direct use of [HJE18] method with a network per time step is impossible to test due the size of the problem but results in lower dimension also indicate some instability directly linked to the initialization of the network.

7.5.2 A mean-field control problem of systemic risk

We consider a mean-field model of systemic risk introduced in [CFS15]. This model was introduced in the context of mean field games but here we consider a cooperative version. The limiting problem (when the number of banks is large) of the social planner (central bank) is formulated as follows. The log-monetary reserve of the representative bank is governed by the mean-reverting controlled McKean-Vlasov dynamics

$$dX_t = \big[\kappa(\mathbb{E}[X_t] - X_t) + \alpha_t\big]\,\mathrm{d}t + \sigma dW_t, \quad X_0 \sim \mu_0,$$

where $\alpha = (\alpha_t)_t$ is the control rate of borrowing/lending to a central bank that aims to minimize the functional cost

$$J(\alpha) \;=\; \mathbb{E}\Big[\int_0^T \tilde{f}(X_t, \mathbb{E}[X_t], \alpha_t)\,\mathrm{d}t + \tilde{g}(X_T, \mathbb{E}[X_T])\Big] \;\to\quad V_0 \;=\; \inf_\alpha J(\alpha), \qquad (7.5.1)$$

where the running and terminal costs are given by

$$\tilde{f}(x, \bar{x}, a) \;=\; \frac{1}{2}a^2 - qa(\bar{x} - x) + \frac{\eta}{2}(\bar{x} - x)^2, \quad \tilde{g}(x, \bar{x}) \;=\; \frac{c}{2}(x - \bar{x})^2,$$

for some positive constants q, η, $c > 0$, with $q^2 \le \eta$.

The value function v to the mean-field type control problem (7.5.1) is solution to the Bellman (semi-linear PDE) equation (7.2.4) with σ constant, $\sigma_0 \equiv 0$, $r = 0$, and

$$\begin{aligned} h(t, x, \mu, z, \gamma) &= \inf_{a \in \mathbb{R}} \big\{ \big[\kappa(\mathbb{E}_\mu[\xi] - x) + a\big] z + \frac{1}{2}a^2 - qa(\mathbb{E}_\mu[\xi] - x) \big\} + \frac{\sigma^2}{2}\gamma + \frac{\eta}{2}\big(\mathbb{E}_\mu[\xi] - x\big)^2 \\ &= (\kappa + q)(\mathbb{E}_\mu[\xi] - x)z + \frac{\sigma^2}{2}\gamma + \frac{\eta - q^2}{2}\big(\mathbb{E}_\mu[\xi] - x\big)^2 - \frac{z^2}{2}, \end{aligned}$$

and $\mathcal{G}(\mu) = \frac{c}{2}\mathrm{Var}(\mu) := \frac{c}{2}\mathbb{E}_\mu|\xi - \mathbb{E}_\mu[\xi]|^2$ is the variance of the distribution μ (up to $c/2$). Here, we use the notation: $\mathbb{E}_\mu[\varphi(\xi)] := \int \varphi(x)\mu(dx)$.

The finite-dimensional approximation of (7.5.1) with N-bank model corresponds to the symmetric Bellman semi-linear PDE on $[0,T]\times\mathbb{R}^N$:

$$\partial_t v^N + \sum_{i=1}^N (\kappa+q)\big(\bar{x}-x_i\big)\partial_{x_i} v^N + \frac{\sigma^2}{2}\Delta_{\boldsymbol{x}} v^N + \frac{\eta-q^2}{2N}\sum_{i=1}^N \big(\bar{x}-x_i\big)^2 - \frac{N}{2}\sum_{i=1}^N |\partial_{x_i} v^N|^2 = 0, \tag{7.5.2}$$

for $\boldsymbol{x} = (x_1,\dots,x_N)\in\mathbb{R}^N$, where we set $\bar{x} = \frac{1}{N}\sum_{i=1}^N x_i$, and $\Delta_{\boldsymbol{x}} = \sum_{i=1}^N \partial^2_{x_i x_i}$ is the Laplacian operator. We numerically solve (7.5.2) with Algorithm 8 described in Section 7.4. The algorithm is trained with the forward process in $\mathbb{R}^N$:

$$X^i_{k+1} = X^i_k + \sigma\Delta W^i_k, \quad X^i_0 \sim \mu_0, \;\; k = 0,\dots,N_T-1, \quad i = 1,\dots,N.$$

The choice of a null drift for this training process is intuitively justified by the fact that the objective in (7.5.1) is to incite the log-monetary reserve of the banks to be close to the average of all the other banks, hence we formally expect their drift to be close to zero.

We test our algorithm by increasing N, and compare with the explicit solution of the limiting linear-quadratic McKean-Vlasov control problem (7.5.1), which is solved via the resolution of a Riccati equation (see [BP19]), and is analytically given by

$$v(t,\mu) = K_t\mathrm{Var}(\mu) + \sigma^2\int_t^T K_s\,\mathrm{d}s,$$

where

$$K_t = -\frac{1}{2}\Big[\kappa+q-\sqrt{\Delta}\frac{\sqrt{\Delta}\sinh(\sqrt{\Delta}(T-t)) + (\kappa+q+c)\cosh(\sqrt{\Delta}(T-t))}{\sqrt{\Delta}\cosh(\sqrt{\Delta}(T-t)) + (\kappa+q+c)\sinh(\sqrt{\Delta}(T-t))}\Big],$$

with $\sqrt{\Delta} = \sqrt{(\kappa+q)^2+\eta-q^2}$, and

$$\int_t^T K_s\,\mathrm{d}s = \frac{1}{2}\ln\Big[\cosh(\sqrt{\Delta}(T-t)) + \frac{\kappa+q+c}{\sqrt{\Delta}}\sinh(\sqrt{\Delta}(T-t))\Big] - \frac{1}{2}(\kappa+q)(T-t).$$

We have tested various approximation symmetric networks and different resolution methods. We list the methods that fail to solve the problem in high dimension:

- First we tried the global resolution method in [EHJ17] by using the network described in paragraph 7.3.2 for v. In this case, we could not obtain exploitable results.
- Then we decided to use the local resolution method [HPW20] with a DeepDerSet approximation approach for $\mathcal{Z}$. We found that the method give accurate results in dimension below 100 but with a variance increasing with the dimension. Results were impossible to exploit in dimension 1000. We thus decided to not report the results.
- At last we tested the local resolution methods [HPW20] with classical feedforward networks using tanh or ReLU activation functions. Two variants were tested with $N = 500$: the first one using a network for v^N (we stress the dependence of the solution to the PDE on N) and another network for Dv^N giving values not exploitable, and a second version using a single network for v^N and using automatic differentiation to approximate Dv^N giving a very high bias and a high standard deviation.

Therefore we only report the case where we use a DeepSet network for $\mathcal{U}$ and a second AD-DeepSet network to estimate $\mathcal{Z}$ or a single DeepSet network for $\mathcal{U}$ which is differentiated to approximate $\mathcal{Z}$.

N_T	Dimension N	Averaged	Std	Relative error
15	10	0.259	0.0029	0.11
15	100	0.2871	0.0016	0.018
15	500	0.2866	0.00179	0.019
15	1000	0.2877	0.00238	0.016
30	10	0.265	0.004	0.09
30	100	0.2892	0.2892	0.010
30	500	0.2897	0.00153	0.009
30	1000	0.2899	0.00146	0.0084
60	10	0.2655	0.0045	0.092
60	100	0.2894	0.0012	0.010
60	500	0.2894	0.0027	0.010
60	1000	0.2916	0.0014	0.0025

Table 7.8: Systemic risk with ReLU activation function, a DeepSet network for $\mathcal{U}$ and a second AD-DeepSet network to estimate $\mathcal{Z}$.

N_T	Dimension N	Averaged	Std	Relative error
15	10	0.2530	0.0074	0.1346
15	100	0.27968	0.0051	0.043
15	500	0.2938	0.0067	0.0049
15	1000	0.3084	0.0253	0.054
30	10	0.2494	0.0074	0.1471
30	100	0.2756	0.00677	0.057
30	500	0.2885	0.0127	0.013
30	1000	0.2860	0.009	0.02
60	10	0.2519	0.0037	0.138
60	100	0.28253	0.0047	0.033
60	500	0.28329	0.0108	0.03
60	1000	0.2881	0.0043	0.014

Table 7.9: Systemic risk with ReLU activation function, a single DeepSet network for $\mathcal{U}$ which is differentiated to approximate $\mathcal{Z}$.

We test the tanh and ReLU activation function on this test case using the parameters $\sigma = 1$, $\kappa = 0.6$, $q = 0.8$, $c = 2$, $\eta = 2$, $T = 1$. We report v^N estimated with different values of N_T and N at time $t = 0$ and $x = 0$ so using $\mu_0 = \delta_0$ on the figures below. The theoretical solution obtained when N goes to infinity is 0.29244.
We use a batch size equal to 200, a number of gradient iteration equal to 30000 for the resolution to project the terminal condition on the network and 6000 gradient iterations for other resolutions. The initial learning rate is taken equal to $1e-4$ at the first resolution and $5e-5$ for other resolutions. The learning is taken decreasing linearly with gradient iterations to $5e-6$.

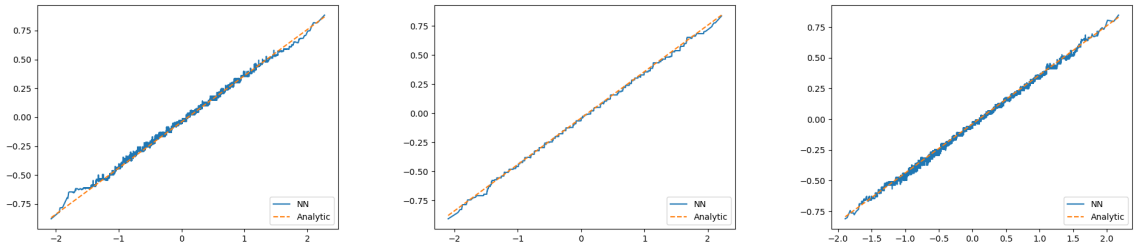
On figure 7.8, we give the results obtained with ReLU activation function using a DeepSet network for $\mathcal{U}$ and a second AD-DeepSet network to approximate $\mathcal{Z}$. The convergence is steady as N_T grows and as the dimension grows leading to a very accurate result for $N_T = 60$ and $N = 1000$.

Using a ReLU activation function, a single network for $\mathcal{U}$ which is differentiated to approximate $\mathcal{Z}$, we get the results in figure 7.9. The convergence is still steady but results are not as good as in table 7.8.

The replace the ReLU activation function by a tanh one using two networks and the results are given in table 7.10. The convergence is not steady and increasing to much N or N_T worsen to results : it shows the importance of the activation function in this method. At last we do not

N_T	N	Averaged	Std	Relative error
15	10	0.2678	0.0061	0.08
15	100	0.28858	0.0144	0.013
15	500	0.2491	0.027	0.14
15	1000	0.27401	0.0127	0.063
30	10	0.2725	0.0052	0.068
30	100	0.2959	0.0161	0.012
30	500	0.2577	0.01568	0.118
30	1000	0.320	0.0030	0.096
60	10	0.2739	0.0049	0.063
60	100	0.2924	0.0309	0.0001
60	500	0.3158	0.00297	0.079
60	1000	0.2210	0.004	0.24

Table 7.10: Systemic with tanh activation function, a DeepSet network for $\mathcal{U}$ and a second AD-DeepSet network to estimate $\mathcal{Z}$.



$N_T = 10$ $N_T = 20$ $N_T = 40$

Figure 7.4: Resolution on $[0.5, 1]$ in dimension $N = 500$: analytic Lions derivatives versus $N\mathcal{Z}$ estimated by the network. DeepSet network for $\mathcal{U}$, AD-DeepSet for $\mathcal{Z}$. ReLU activation function.

report the test obtained using a ReLU activation function for the first network and a tanh one for the second network given results far better than in table 7.10 but not as good as in tables 7.8 and 7.9. We also test the accuracy of our algorithm for approximating the L-derivative of the solution, which is here explicitly given by

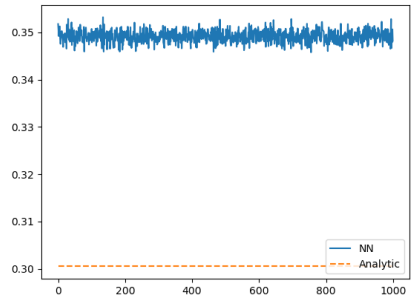
$$\partial_\mu v(t,\mu)(x) = 2K_t(x - \mathbb{E}_\mu[\xi]).$$

For this purpose, using N_T steps, we solve the same problem on $[t, T]$, starting at $t = \frac{T}{2}$ with a distribution μ_0 equal to real distribution of the solution of (7.5.1) taken at date t. After training, we plot $x \mapsto N\widehat{\mathcal{Z}}(\boldsymbol{X}_t, x)$, where $\boldsymbol{X}_t \sim \mu_0^{\otimes N}$, and compare to the analytic solution: $x \mapsto \partial_\mu v(t,\mu_0)(x)$. Some graphs are reported in figure 7.4, which shows the accuracy of the approximation.

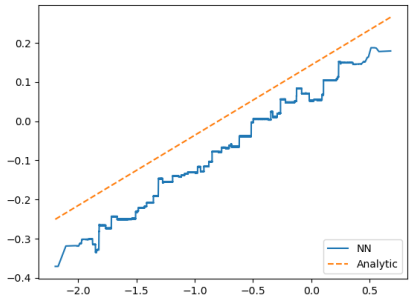
As mentioned in Section 7.4.3, in theory, the proposed methodology should learn the solution for any initial law μ_0 in the space of measures so that we should be able to solve the problem in infinite dimension. We test our algorithm by sampling μ_0 in the following way: for a sample j, we pick up a mean $\hat{M} \in [-1, 1]$ and a standard deviation $\sigma \in [0.2, 1]$ with an uniform law. Then $X_0^{i,j} \sim \mathcal{N}(\hat{M}, \sigma^2)$ $i = 1, \dots, N$ and as before we use the forward process:

$$X_{k+1}^{i,j} = X_k^{i,j} + \sigma \Delta W_k^{i,j}, \quad k = 0, \dots, N_T - 1, \quad i = 1, \dots, N.$$

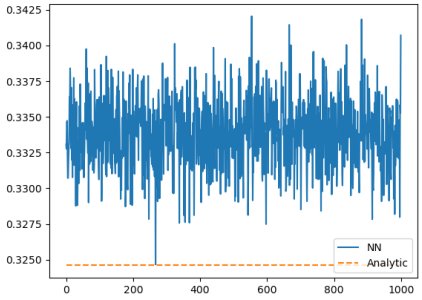
After the training part, we try to recover the initial solution and the initial Lions derivative for a given μ_0 following a gaussian law. Results are given on figure 7.5. The Lions derivative is relatively correctly calculated but the initial value can get an error around 15%.



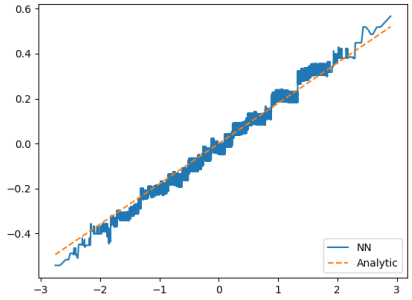
Solution for $\mu_0 = \mathcal{N}(-0.8, 0.09)$



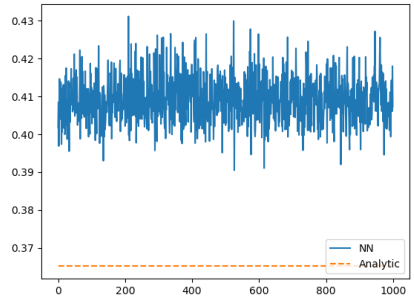
Lions derivative for $\mu_0 = \mathcal{N}(-0.8, 0.09)$



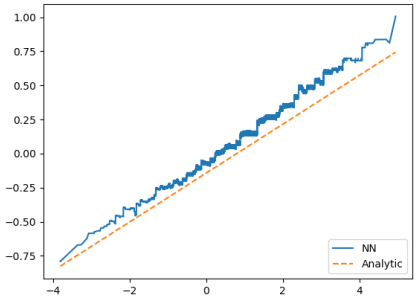
Solution for $\mu_0 = \mathcal{N}(0., 0.36)$



Lions derivative for $\mu_0 = \mathcal{N}(0., 0.36)$

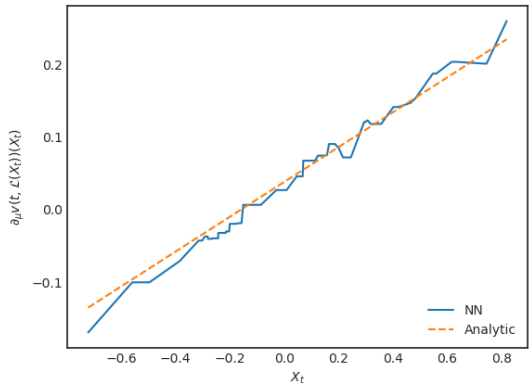


Solution for $\mu_0 = \mathcal{N}(0.8, 0.81)$

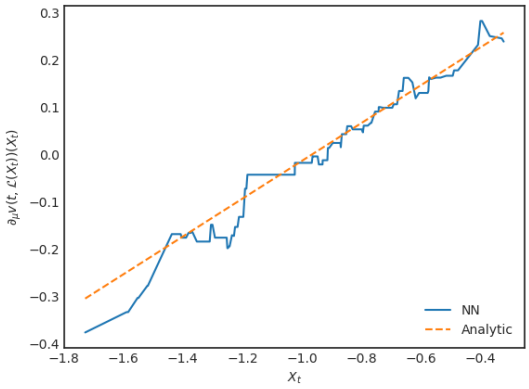


Lions derivative for $\mu_0 = \mathcal{N}(0.8, 0.81)$

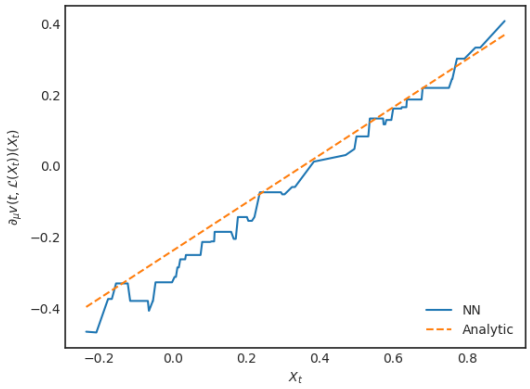
Figure 7.5: Solution and Lions derivative after a single training, with $N = 500$, $N_T = 30$, with ReLU activation function, a single DeepSet network for $\mathcal{U}$ which is differentiated to approximate $\mathcal{Z}$. For the solution, the x-axis corresponds to the sample number and the y-axis is the value of the estimated solution. For the Lions derivative, the x-axis is the state space and the y-axis is the value of the derivative.



$t = 0.2$



$t = 0.5$



$t = 0.7$

Figure 7.6: Analytic Lions derivative versus $N\mathcal{Z}$ estimated by the network. Dimension $N = 300$, number of time steps $N_T = 30$. We use a DeepSet network for $\mathcal{U}$ with ReLU activation functions, and $\mathcal{Z}$ its automatic derivative.

More generally, if we want to solve the PDE at each time step in the Wasserstein space, we can use Algorithm 10. In order to illustrate the exploration of the Wasserstein space we plot in Figure 7.6 the graphs of $(X^i, N\hat{\mathcal{Z}}(\boldsymbol{X}, X^i))$, $i = 1, \ldots, N$, vs $X^i \mapsto \partial_\mu v(t, \mathbb{P}_{X^i})(X^i)$, when X_t^i $\rightsquigarrow$ random mixture of Gaussian laws, for $N = 300$, $N_T = 30$. We observe that we are able to estimate correctly the Lions derivative of the solution (and therefore the optimal control) on several probability measures through a randomized training. Concerning the solution itself, we observe similar behavior as in Figure 7.5 with an error of order 10-15% so we do not show the plots. Further numerical studies are left to future research to improve the estimation of the solution with the randomization procedure.

7.5.3 Mean-variance problem

We consider the celebrated Markowitz portfolio selection problem where an investor can invest at any time t an amount α_t in a risky asset (assumed for simplicity to follow a Black-Scholes model with constant rate of return β and volatility $\nu > 0$), hence generating a wealth process $X = X^\alpha$ with dynamics

$$dX_t = \alpha_t \beta dt + \alpha_t \nu dW_t, \quad 0 \le t \le T, \; X_0 = x_0 \in \mathbb{R}\,.$$

The goal is then to minimize over portfolio control α the mean-variance criterion:

$$J(\alpha) = \lambda \mathrm{Var}(X_T^\alpha) - \mathbb{E}[X_T^\alpha],$$

where $\lambda > 0$ is a parameter related to the risk aversion of the investor. Due to the presence of the variance term Var in the criterion, the Markowitz problem falls into the class of McKean-Vlasov control problems, and the associated value function v satisfies the Bellman equation (7.2.4) on $[0,T] \times \mathcal{P}_2(\mathbb{R})$ with $r = 0$, $\sigma_0 \equiv 0$,

$$\begin{cases} h(x,\mu,z,\gamma) = \inf_{a\in\mathbb{R}} \left[za\beta + \frac{1}{2}\gamma a^2 \nu^2 \right] = -\frac{R}{2}\frac{z^2}{\gamma}, \quad z \in \mathbb{R}, \gamma > 0, \\ \mathcal{G}(\mu) = \lambda \mathbb{E}_\mu\big|\xi - \mathbb{E}_\mu[\xi]\big|^2 - \mathbb{E}_\mu[\xi], \quad \mu \in \mathcal{P}_2(\mathbb{R}), \end{cases}$$

where we set $R := \beta^2/\nu^2$.

The associated finite-dimensional PDE with N particles is given by

$$\begin{cases} \partial_t v^N - \frac{R}{2} \sum_{i=1}^N \dfrac{(D_{x_i} v^N)^2}{D^2_{x_i} v^N} = 0, \quad t \in [0,T), \boldsymbol{x} = (x_1, \ldots, x_N) \in (\mathbb{R}^d)^N, \\ v^N(T, \boldsymbol{x}) = \mathcal{G}(\bar{\mu}(\boldsymbol{x})). \end{cases} \tag{7.5.3}$$

We refer to [IP19] for the McKean-Vlasov approach to Markowitz mean-variance problems (in a more general context), and we recall that the solution to the Bellman equation is given by

$$\begin{aligned} v(t,\mu) &= \lambda e^{-R(T-t)} \mathbb{E}_\mu\big|\xi - \mathbb{E}_\mu[\xi]\big|^2 - \mathbb{E}_\mu[\xi] - \frac{1}{4\lambda}\big[e^{R(T-t)} - 1\big] \\ \partial_\mu v(t,\mu)(x) &= 2\lambda e^{-R(T-t)}(x - \mathbb{E}_\mu[\xi]) - 1, \quad \partial_x \partial_\mu v(t,\mu)(x) = 2\lambda e^{-R(T-t)} \end{aligned} \tag{7.5.4}$$

and in particular $V_0 := \inf_\alpha J(\alpha) = v(0, \delta_{x_0}) = -x_0 - \frac{1}{4\lambda}[e^{RT} - 1]$. Moreover, the optimal portfolio strategy is given by

$$\begin{aligned} \alpha_t^* &= \hat{a}(t, X_t^*, \mathbb{E}[X_t^*]) := -\frac{\beta}{\nu^2}\Big[X_t^* - \mathbb{E}[X_t^*] - \frac{e^{R(T-t)}}{2\lambda}\Big] \\ &= -\frac{\beta}{\nu^2}\Big[X_t^* - x_0 - \frac{e^{RT}}{2\lambda}\Big], \quad 0 \le t \le T, \end{aligned} \tag{7.5.5}$$

where $X^* = X^{\alpha^*}$ is the optimal wealth process.

We test our Algorithm 9 described in Section 7.4.2 with the training of the forward process

$$X_{k+1}^{i,N,\pi} = X_k^{i,N,\pi} + \frac{R}{2\lambda}\Delta t_k + \frac{\sqrt{R}}{2\lambda}\Delta W_k^i, \quad X_0^i = x_0, \ \ k = 0, \ldots, N_T - 1, \quad i = 1, \ldots, N,$$

which is the time discretization of the wealth process for a constant portfolio strategy $\alpha_t = \beta/(2\nu^2\lambda)$, which is known to be optimal for the exponential utility function $U(x) = -e^{-2\lambda x}$. This corresponds to the choice of $b_i = R/(2\lambda)$ and $\sigma_{ij} = \sqrt{R}/(2\lambda)$. Here, notice that $\partial_\mu G(\mu)(x) = 2\lambda(x - \mathbb{E}_\mu[\xi]) - 1$, and we then use for the initialization at terminal step N_T, the DeepDerSet function $\mathcal{Z}_{N_T}((x_i)_i, x) = 2\lambda\big(x - \frac{1}{N}\sum_i x_i\big) - 1$ (corresponding to the average function $\mathfrak{s}((x_i)_i) = \frac{1}{N}\sum_i x_i$), which yields the automatic differentiation $\mathrm{D}\mathcal{Z}_{N_T}((x_i)_i, x) = 2\lambda(1 - \frac{1}{N})$.

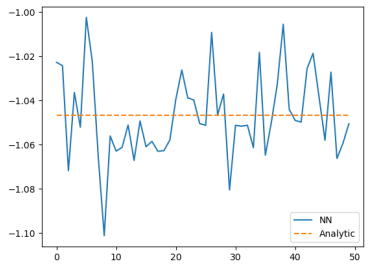
We choose the parameters $\beta = 0.15$, $\nu = 0.35$, $\lambda = 1$, and the quantile at 99.9% for the truncation in scheme [PWG21], and report the results in Table 7.11. The optimization parameters are the same as in the semilinear case, except the batch size taken equal to 50 and the number of gradient iterations after first step taken equal to 4000. We use a ReLU Deepset for $\mathcal{U}$ and a AD-Deepset with a tanh activation function for $\mathcal{Z}$. Remark that in this case it is not possible to use a ReLU activation function for the second network.

Moreover, we test the accuracy of the control approximation. We solve the PDE from $T/2$ to T starting with the optimal distribution of the wealth at $T/2$, which is given by:

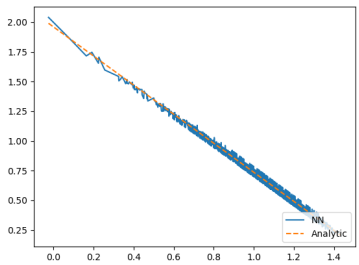
$$\log\Big(\frac{X_{\frac{T}{2}}}{\eta(T/2)} - x_0 - e^{\frac{RT}{2\lambda}}\Big) \sim \mathcal{N}(0, \kappa(T/2)),$$

N_T	Dimension N	Averaged	Std	Relative error
10	10	-1.0561	0.001	0.005
10	100	-1.0522	0.0008	0.0017
20	10	-1.0570	0.0008	0.006
20	100	-1.0520	0.0007	0.0015
30	10	-1.0578	0.0011	0.007
30	100	-1.0535	0.0021	0.0029

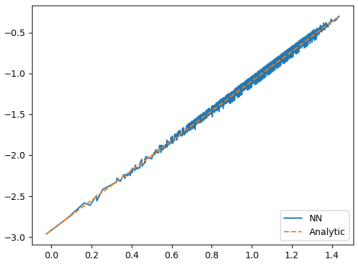
Table 7.11: Estimate of $\mathbb{E}\big[v^N(0, X_0^1, \dots, X_0^N)\big]$ with a deterministic initial condition $X_0 = 1$, $T = 1$, $\sigma = 1$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is -1.0504058 when $N, N_T \to +\infty$.



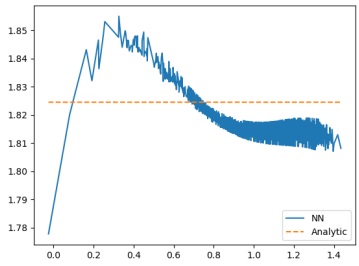
Solution



Control



Derivative



Second order derivative

Figure 7.7: Solution and control obtained on the mean variance case at $\frac{T}{2}$ in dimension 100 with 20 time steps comparing analytic solution to the calculated one (NN). Truncation factor equal to 0.999.

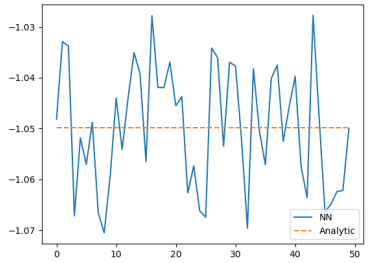
with

$$\eta(t) = -e^{\frac{R(T-t)}{2\lambda}}, \quad \kappa^2(t) = \log\Big(\frac{e^{R(T-t)}(e^{RT} - e^{R(T-t)})}{4\lambda^2\eta(t)^2} + 1\Big),$$

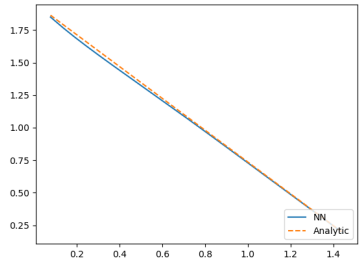
and we calculate the solution obtained at date $T/2$ and the control obtained solving the PDE (7.5.3) that we can compare to the analytical solution given by (7.5.4) and (7.5.5). After training, using $n_s = 50$ samples of $X \in (\mathbb{R}^N)^{n_s}$ following the law of $X_{\frac{T}{2}}$, we calculate the control obtained for each sample in each of the dimension. After sorting X in a one dimensional array, We plot the result obtained on Figures 7.7–7.8. For the solution, the x-axis corresponds to the sample number and the y-axis is the value of the estimated solution. For the other plots, the x-axis is the state space and the y-axis is the value of the corresponding function.

7.5.4 A min/max Linear quadratic mean-field control problem

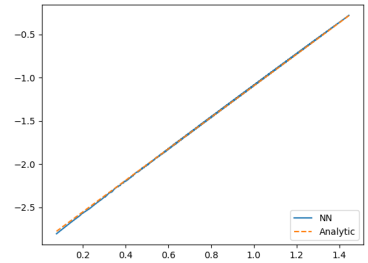
We consider a mean-field model in which the dynamics is linear and the running cost is quadratic in the position, the control and the expectation of the position. The terminal cost is encourages



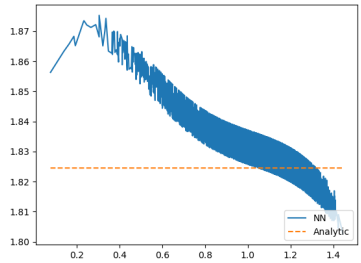
Solution



Control



Derivative



Second order derivative

Figure 7.8: Solution and control obtained on the mean variance case at $\frac{T}{2}$ in dimension 300 with 20 time steps comparing analytic solution to the calculated one (NN). Truncation factor: 0.999.

to be close to one of two targets. This type of model is inspired by the min-LQG problem of [SMLN15]. More precisely, we consider the following controlled McKean-Vlasov dynamics

$$\mathrm{d}X_t = \big[AX_t + \bar{A}\mathbb{E}[X_t] + B\alpha_t\big]\,\mathrm{d}t + \sigma\,\mathrm{d}W_t, \quad X_0 \sim \mu_0,$$

where $\alpha = (\alpha_t)_t$ is the control, and the agent aims to minimize the functional cost

$$J(\alpha) \;=\; \mathbb{E}\Big[\int_0^T f(X_t, \mathbb{E}[X_t], \alpha_t)\,\mathrm{d}t + g(X_T)\Big] \;\to\; V_0 \;=\; \inf_\alpha J(\alpha), \tag{7.5.6}$$

where the running and terminal costs are given by

$$f(x, \bar{x}, a) \;=\; \frac{1}{2}\left(Qx^2 + \bar{Q}(x - S\bar{x})^2 + Ra^2\right), \quad g(x) \;=\; \min\left\{|x-\xi_1|^2, |x-\xi_2|^2\right\},$$

for some non-negative constants Q, $\bar{Q}$, S, R, and two real numbers ξ_1 and ξ_2.

The value function to the mean-field type control problem (7.5.6) is solution to the Bellman (semi-linear PDE) equation (7.2.4) with $r = 0$, and

$$\begin{aligned} h(x,\mu,z,\gamma) &= \inf_{a\in\mathbb{R}}\left\{\big[Ax + \bar{A}\mathbb{E}_\mu[\xi] + Ba\big]z + \frac{1}{2}\left(Qx^2 + \bar{Q}(x - S\mathbb{E}_\mu[\xi])^2 + Ra^2\right)\right\} + \frac{\sigma^2}{2}\gamma \\ &= \big[Ax + \bar{A}\mathbb{E}_\mu[\xi]\big]z - \frac{B^2}{2R}z^2 + \frac{1}{2}\left(Qx^2 + \bar{Q}(x - S\mathbb{E}_\mu[\xi])^2\right) + \frac{\sigma^2}{2}\gamma, \end{aligned}$$

where the minimizer in the above inf is given by $a = -\frac{B}{R}z$, and the terminal condition $\mathcal{G}(\mu) = \mathbb{E}_\mu \min\left\{|\xi-\xi_1|^2, |\xi-\xi_2|^2\right\}$ is the expected minimal distance to one of the targets ξ_1, ξ_2.

For the sake of illustration, we present several test cases. The targets are at $\xi_1 = 0.25$ and $\xi_2 = 1.75$. Here we used $A = \bar{A} = 0$, $B = 1$, $Q = 0, \bar{Q} = S = R = 1$, and a time horizon $T = 0.5$. The initial distribution μ_0 is a Gaussian $\mathcal{N}(x_0, \vartheta_0^2)$. We consider the following test cases:

1. $\sigma = 0.3, x_0 = 1, \vartheta_0 = 0.2$,

Case	Benchmark	Global
1	0.2256	0.2273(0.004)
2	0.2085	0.2098(0.006)
3	0.1734	0.1742(0.005)
4	0.2276	0.2300(0.009)

Table 7.12: Min-LQC example reference solutions : benchmark solution estimated by finite difference scheme and Algorithm 1 in [CL22] with $N = 10000$, $N_T = 50$, 10 neurons and 3 hidden layers, tanh activation function, average on 10 runs.

Case	$N = 100, N_T = 30$	$N = 100, N_T = 60$	$N = 500, N_T = 30$	$N = 500, N_T = 60$
1	0.2370(0.013)	0.2382(0.012)	0.2446(0.013)	0.2495(0.09)
2	0.2088(0.002)	0.2092(0.001)	0.2106(0.003)	0.2105(0.003)
3	0.1774(0.007)	0.1784(0.005)	0.1819(0.005)	0.1785(0.008)
4	0.2279(0.005)	0.2264(0.005)	0.2292(0.006)	0.2274(0.006)

Table 7.13: Min-LQC example with DPBD scheme using ReLU activation functions with a DeepSet network for $\mathcal{U}$ and a second AD-DeepSet network to estimate $\mathcal{Z}$, average on 10 runs, standard deviation in parenthesis.

2. $\sigma = 0.5, x_0 = 0.625, \vartheta_0 = \sqrt{0.2}$,
3. $\sigma = 0.3, x_0 = 0.625, \vartheta_0 = \sqrt{0.2}$,
4. $\sigma = 0.3, x_0 = 0.625, \vartheta_0 = \sqrt{0.4}$.

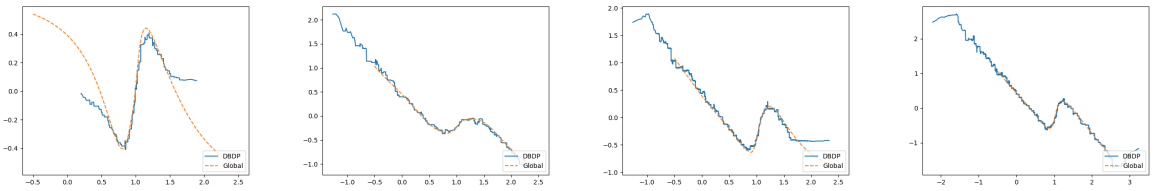
References are given in table 7.12: they are calculated by the PDE method in [ACD10] (in the context of mean field games; see [AL15] for the adaptation to the PDE system arising in mean field control) with step size in space and time of size 10^{-3}, and the neural network method referred to as Algorithm 1 in [CL22] with $N = 10000$ and $N_T = 50$.

In table 7.13, we give the results obtained with different time discretization and dimension for the DPBD scheme using ReLU activation functions with a DeepSet network for $\mathcal{U}$ and a second AD-DeepSet network to estimate $\mathcal{Z}$. Results are very good except for test case 1 where a small bias appears.

In table 7.14, we give the same results using a single network. Here the results are very good for all test cases. Using two networks, the algorithm certainly face difficulties to approximate the derivatives near maturities which is not required using a single network.

Case	$N = 100, N_T = 30$	$N = 100, N_T = 60$	$N = 500, N_T = 30$	$N = 500, N_T = 60$
1	0.2289(0.0006)	0.2271(0.001)	0.2290(0.0004)	0.2271(0.0008)
2	0.2083(0.0008)	0.2086(0.0007)	0.2097(0.0008)	0.2089(0.0004)
3	0.1740(0.001)	0.1740(0.001)	0.1742(0.0004)	0.1729(0.0007)
4	0.2276(0.001)	0.2310(0.003)	0.2282(0.0008)	0.2278(0.001)

Table 7.14: Min-LQC example with DBDP scheme using ReLU activation functions and a single DeepSet network for $\mathcal{U}$ which is differentiated to approximate $\mathcal{Z}$, average on 10 run, standard deviation in parenthesis.



Case 1 Case 2 Case 3 Case 4

Figure 7.9: Control calculated at $t = 0$ for Min-LQC examples: comparison DBDP using a single DeepSet network with $N_T = 50$, $N = 500$ and global approximation.

Chapter 8

A level-set approach to the control of state-constrained McKean-Vlasov equations: application to renewable energy storage and portfolio selection

This chapter is based on the paper [GPW21]
M. Germain, H. Pham, and X. Warin. “A level-set approach to the control of state-constrained McKean-Vlasov equations: application to renewable energy storage and portfolio selection". In:arXiv:2112.11059, submitted to *Numerical Algebra, Control and Optimization*, special issue *Stochastic Analysis, Mathematical Finance, and Related Fields*.

This last Chapter develops a framework to consider state constrained mean-field control. In this context we establish a method to write down mean-field control with constraints on the law of the state. As a consequence, it contains in particular the case of almost sure constraints which are considered in the mean field games literature [CC18; CCC18]. The constraint is handled thanks to a representation result with a level-set approach and an auxiliary unconstrained deterministic problem in infinite dimension. This problem can be solved numerically by an adaptation of the scheme from [CL22]. An alternative could have been to consider the resolution of the master Bellman equation characterizing the auxiliary problem, which is left to future research.

Abstract

We consider the control of McKean-Vlasov dynamics (or mean-field control) with probabilistic state constraints. We rely on a level-set approach which provides a representation of the constrained problem in terms of an unconstrained one with exact penalization and running maximum or integral cost. The method is then extended to the common noise setting. Our work extends (Bokanowski, Picarelli, and Zidani, SIAM J. Control Optim. 54.5 (2016), pp. 2568–2593) and (Bokanowski, Picarelli, and Zidani, Appl. Math. Optim. 71 (2015), pp. 125–163) to a mean-field setting.

The reformulation as an unconstrained problem is particularly suitable for the numerical resolution of the problem, that is achieved from an extension of a machine learning algorithm from (Carmona, Laurière, arXiv:1908.01613 to appear in Ann. Appl. Prob., 2019). A first application concerns the storage of renewable electricity in the presence of mean-field price impact and another one focuses on a mean-variance portfolio selection problem with probabilistic constraints on the wealth. We also illustrate our approach for a direct numerical resolution of the primal Markowitz continuous-time problem without relying on duality.

8.1 Introduction

The control of McKean-Vlasov dynamics, also known as mean-field control problem, has attracted a lot of interest over the last years since the emergence of the mean-field game theory. There is now an important literature on this topic addressing on one hand the theoretical aspects either by dynamic programming approach (see [LP14; PW17; PW18; CP19]), or by maximum principle (see [CD15]), and on the other hand the numerous applications in economics and finance, and we refer to the two-volume monographs [CD18a; CD18b] for an exhaustive and detailed treatment of this area.

In this paper, we aim to study control of McKean-Vlasov dynamics under the additional presence of state constraints in law. The consideration of probabilistic constraints (usually in expectation or in target form) for standard stochastic control has many practical applications, notably in finance with quantile and CVaR type constraints, and is the subject of many papers, we refer to [ST02; BEI10; Gel+13; CYZ20; PTZ21; Bal+21] for an overview.

There exists some recent works dealing with mean-field control under some specific law state constraints. For example, the paper [CW19] solves mean-field control with delay and smooth expectation terminal constraint (and without dependence with respect to the law of the control). In the case of mean field games, state constraints are considered by [CC18; CCC18; FH20; GM21; AM21]. In these cited works the state belongs to a compact set, which corresponds to a particular case of our constraints in distribution. Related literature includes the recent work [BDK20] which studies a mean-field target problem where the aim is to find the initial laws of a controlled McKean-Vlasov process satisfying a law constraint, but only at terminal time. The paper [Dau20] also studies these terminal constraint in law for the control of a standard diffusion process. Next, it has been extended in [Dau21] to a running law constraint for the control of a standard diffusion process with McKean-Vlasov type cost through the control of a Fokker-Planck equation. Several works also consider directly the optimal control of Fokker-Planck equations in the Wasserstein space with terminal or running constraints, such as [Bon19; BF21] through Pontryagin principle, in the deterministic case without diffusion.

In this paper, we consider general running (at discrete or continuous time) and terminal constraints in law, and extend the level-set approach [BPZ15; BPZ16] (see also [ABZ13] in the deterministic case) to our mean-field setting. This enables us to reformulate the constrained McKean-Vlasov control problem into an unconstrained mean-field control problem with an auxiliary state variable, and a running path-dependent supremum cost or alternatively a non path-dependent integral cost over the constrained functions. Such equivalent representations of the control problem with exact penalization turns out to be quite useful for an efficient numerical

resolution of the original constrained mean-field control problem. We shall actually adapt the machine learning algorithm in [CL22] for solving two applications in renewable energy storage and in portfolio selection.

The outline of the paper is organized as follows. Section 8.2 develops the level-set approach in our constrained mean-field setting with supremum term. We present in Section 8.3 the alternative level-set formulation with integral term, and discuss when the optimization over open-loop controls yields the same value than the optimization over closed-loop controls. This will be useful for numerical purpose in the approximation of optimal controls. The method is then extended in Section 8.4 to the common noise setting. Finally, we present in Section 8.5 the applications and numerical tests.

8.2 Mean-field control with state constraints

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space on which is defined a d-dimensional Brownian motion W with associated filtration $\mathbb{F} = (\mathcal{F}_t)_t$ augmented with $\mathbb{P}$-null sets. We assume that $\mathcal{F}_0$ is "rich enough" in the sense that any probability measure μ on $\mathbb{R}^d$ can be represented as the distribution law of some $\mathcal{F}_0$-measurable random variable. This is satisfied whenever the probability space $(\Omega, \mathcal{F}_0, \mathbb{P})$ is atomless.

We consider the following cost and dynamics:

$$J(X_0, \alpha) = \mathbb{E}\Big[\int_0^T f\big(s, X_s^\alpha, \alpha_s, \mathbb{P}_{(X_s^\alpha, \alpha_s)}\big)\, \mathrm{d}s + g\big(X_T^\alpha, \mathbb{P}_{X_T^\alpha}\big)\Big]$$
$$X_t^\alpha = X_0 + \int_0^t b\big(s, X_s^\alpha, \alpha_s, \mathbb{P}_{(X_s^\alpha, \alpha_s)}\big)\, \mathrm{d}s + \int_0^t \sigma\big(s, X_s^\alpha, \alpha_t, \mathbb{P}_{(X_s^\alpha, \alpha_s)}\big)\, \mathrm{d}W_s, \quad (8.2.1)$$

where $\mathbb{P}_{(X_s^\alpha, \alpha_s)}$ is the joint law of (X_s^α, α_s) under $\mathbb{P}$ and X_0 is a given random variable in $L^2(\mathcal{F}_0, \mathbb{R}^d)$. The control α belongs to a set $\mathcal{A}$ of $\mathbb{F}$-progressively measurable processes with values in a set $A \subseteq \mathbb{R}^q$. The coefficients b and σ are measurable functions from $[0,T] \times \mathbb{R}^d \times A \times \mathcal{P}_2(\mathbb{R}^d \times A)$ into $\mathbb{R}^d$ and $\mathbb{R}^{d\times d}$, where $\mathcal{P}_2(E)$ is the set of square integrable probability measures on the metric space E, equipped with the 2-Wasserstein distance $\mathcal{W}_2$. We make some standard Lipschitz conditions on b, σ in order to ensure that equation (8.2.1) is well-defined and admits a unique strong solution, which is square-integrable. The function f is a real-valued measurable function on $[0,T] \times \mathbb{R}^d \times A \times \mathcal{P}_2(\mathbb{R}^d \times A)$, while g is a measurable function on $\mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^d)$, and we assume that f and g satisfy some linear growth condition which ensures that the functional in (8.2.1) is well-defined.

Furthermore, the law of the controlled McKean-Vlasov process X is constrained to verify

$$\Psi(t, \mathbb{P}_{X_t^\alpha}) \leq 0, \quad 0 \leq t \leq T, \quad (8.2.2)$$

where $\Psi = (\Psi^l)_{1\leq l\leq k}$ is a given function from $[0,T] \times \mathcal{P}_2(\mathbb{R}^d)$ into $\mathbb{R}^k$. Here, the multi-dimensional constraint $\Psi(t,\mu) \leq 0$ has to be understood componentwise, i.e., $\Psi^l(t,\mu) \leq 0$, $l = 1, \cdots, k$. The problem of interest is therefore

$$V := \inf_{\alpha\in\mathcal{A}} \big\{ J(X_0, \alpha) : \Psi(t, \mathbb{P}_{X_t^\alpha}) \leq 0, \ \forall\, t \in [0,T] \big\}.$$

By convention the infimum of the empty set is $+\infty$. When needed, we will sometimes use the notation V^Ψ to emphasize the dependence of the value function on Ψ. Clearly, $\Psi \leq \Psi'$ (meaning that for each component $\Psi^l \leq \Psi'^l$, $l = 1 \cdots k$) implies $V^\Psi \leq V^{\Psi'}$.

Remark 8.2.1. *This very general type of constraints includes for instance:*

- *Controlled McKean-Vlasov process X constrained to stay inside a non-empty closed set $\mathcal{K}_t \subseteq \mathbb{R}^d$ with probability larger than a threshold $p_t \in [0,1]$, namely*

$$\mathbb{P}(X_t^\alpha \in \mathcal{K}_t) \geq p_t, \ \forall\, t \in [0,T],$$

with $\Psi : (t,\mu) \mapsto p_t - \mu(\mathcal{K}_t)$. With $p_t = 1$, $\forall t \in [0,T]$ it yields almost sure constraints.

- *Almost sure contraints on the state,* $X_t^\alpha \in \mathcal{K}_t,\ \forall t \in [0,T]\ \mathbb{P}$ *a.s., with*

$$\Psi : (t,\mu) \mapsto \int_{\mathbb{R}^d} d_{\mathcal{K}_t}(x)\ \mu(\mathrm{d}x),$$

  where $d_{\mathcal{K}_t}$ *is the distance function to the non-empty closed set* $\mathcal{K}_t$*.*

- *The case of a Wasserstein ball constraint around a benchmark law* η_t *in the form* $\mathcal{W}_2(\mathbb{P}_{X_t^\alpha}, \eta_t) \leq \delta_t$ *with*

$$\Psi : (t,\mu) \mapsto \mathcal{W}_2(\mu, \eta_t) - \delta_t.$$

  This is the constraint considered in [PJ21] at terminal time.

- *A terminal constraint in law* $\varphi(\mathbb{P}_{X_T^\alpha}) \leq 0$ *as in [Dau20] with*

$$\Psi : (t,\mu) \mapsto \varphi(\mu)\mathbb{1}_{t=T}.$$

- *Terminal constraint in law* $\mathbb{P}_{X_T^\alpha} \in \mathbb{K} \subset \mathcal{P}_2(\mathbb{R}^d)$ *as in [BDK20] with*

$$\Psi : (t,\mu) \mapsto (1 - \mathbb{1}_{\mu\in\mathbb{K}})\mathbb{1}_{t=T}.$$

- *The case of discrete time constraints* $\phi(t_i, \mathbb{P}_{X_{t_i}^\alpha}) \leq 0$ *for* $t_1 < \cdots < t_k$ *with*

$$\Psi : (t,\mu) \mapsto \phi(t,\mu)\mathbb{1}_{t\in\{t_1,\cdots,t_k\}}.$$

Even though this problem seems much more involved than the standard stochastic control problem with state constraints investigated in [BPZ16], thanks to an adequate reformulation, it turns out that we can adapt the main ideas from this paper to our framework and construct similarly an unconstrained auxiliary problem (in infinite dimension).

8.2.1 A target problem and an associated control problem

Given $z \in \mathbb{R}$, and $\alpha \in \mathcal{A}$, define a new state variable

$$Z_t^{z,\alpha} := z - \mathbb{E}\Big[\int_0^t f\big(s, X_s^\alpha, \alpha_s, \mathbb{P}_{(X_s^\alpha,\alpha_s)}\big)\ \mathrm{d}s\Big] = z - \int_0^t \widehat{f}\big(s, \mathbb{P}_{(X_s^\alpha,\alpha_s)}\big)\ \mathrm{d}s, \quad 0 \leq t \leq T, \tag{8.2.3}$$

where $\widehat{f}$ is the function defined on $[0,T] \times \mathcal{P}_2(\mathbb{R}^d \times A)$ by $\widehat{f}(t,\nu) = \int_{\mathbb{R}^d\times A} f\big(t,x,a,\nu\big)\ \nu(\mathrm{d}x,\mathrm{d}a)$. We also denote by $\widehat{g}$ the function defined on $\mathcal{P}_2(\mathbb{R}^d)$ by $\widehat{g}(\mu) = \int_{\mathbb{R}^d} g(x,\mu)\mu(\mathrm{d}x)$.

Lemma 8.2.1. *The value function admits the* ***deterministic target problem*** *representation*

$$V = \inf\{z \in \mathbb{R}\ \mid \exists\ \alpha \in \mathcal{A} \text{ s.t. } \widehat{g}(\mathbb{P}_{X_T^\alpha}) \leq Z_T^{z,\alpha},\ \Psi(s, \mathbb{P}_{X_s^\alpha}) \leq 0,\ \forall\ s \in [0,T]\}.$$

Proof. We first observe from the definition of V in (8.2.2) that it can be rewritten as

$$V = \inf\{z \in \mathbb{R}\ \mid \exists\ \alpha \in \mathcal{A} \text{ s.t. } J(X_0,\alpha) \leq z,\ \Psi(t, \mathbb{P}_{X_t^\alpha}) \leq 0,\ \forall\ t \in [0,T]\}.$$

Next, by noting that the cost functional is written as

$$J(X_0,\alpha) = \int_0^T \widehat{f}\big(t, \mathbb{P}_{(X_t^\alpha,\alpha_t)}\big)\ \mathrm{d}t + \widehat{g}(\mathbb{P}_{X_T^\alpha}),$$

the result then follows immediately by the definition of $Z^{z,\alpha}$ in (8.2.3). □

We want to link this representation to the **zero-level set** of the solution of an auxiliary unconstrained control problem. Define the **auxiliary unconstrained deterministic** control problem:

$$\mathcal{Y}^{\Psi} : z \in \mathbb{R} \mapsto \inf_{\alpha \in \mathcal{A}} \Big[\{\widehat{g}(\mathbb{P}_{X_T^\alpha}) - Z_T^{z,\alpha}\}_+ + \sum_{l=1}^{k} \sup_{s\in[0,T]} \{\Psi^l(s, \mathbb{P}_{X_s^\alpha})\}_+ \Big], \tag{8.2.4}$$

with the notation $\{x\}_+ = \max(x, 0)$ for the positive part. We see that $\mathcal{Y}^{\Psi}(z) \geq 0$.

By classical estimates on McKean-Vlasov equations we can obtain continuity and growth conditions on $\mathcal{Y}^{\Psi}$. The proof of Proposition 8.2.1 is given in Section 8.2.3.

Proposition 8.2.1. *$\mathcal{Y}^{\Psi}$ verifies*

1. *$\mathcal{Y}^{\Psi}$ is 1-Lipschitz. For any $z, z' \in \mathbb{R}$*
$$|\mathcal{Y}^{\Psi}(z) - \mathcal{Y}^{\Psi}(z')| \leq |z - z'|.$$
2. *$\mathcal{Y}^{\Psi}$ is non-increasing. Thus if $\mathcal{Y}^{\Psi}(z_0) = 0$ then $\mathcal{Y}^{\Psi}(z) = 0$ for all $z \geq z_0$.*

Define the infimum of the zero level-set

$$\mathcal{Z}^{\Psi} := \inf\{z \in \mathbb{R} \mid \mathcal{Y}^{\Psi}(z) = 0\}. \tag{8.2.5}$$

We prove a first result linking the auxiliary control problem with the original constrained problem. Solving this easier problem provides bounds on the value function, by making the constraint function vary.

Theorem 8.2.1. 1. *If for some $z \in \mathbb{R}$ $\exists\ \alpha \in \mathcal{A}$ s.t. $\widehat{g}(\mathbb{P}_{X_T^\alpha}) \leq Z_T^{z,\alpha}$, $\Psi(s, \mathbb{P}_{X_s^\alpha}) \leq 0$, $\forall\ s \in [0, T]$ then $\mathcal{Y}^{\Psi}(z) = 0$.*
2. *If V^{Ψ} is finite then $\mathcal{Y}^{\Psi}(V^{\Psi}) = 0$. Thus $\mathcal{Z}^{\Psi} \leq V^{\Psi}$.*
3. *Define $1_k = (1, \cdots, 1) \in \mathbb{R}^k$. We have the upper bound*
$$V^{\Psi} \leq \inf_{\varepsilon > 0} \mathcal{Z}^{\Psi + \varepsilon 1_k}.$$

To sum up, when $V^{\Psi} < +\infty$, Theorem 8.2.1 provides the bounds

$$\mathcal{Z}^{\Psi} \leq V^{\Psi} \leq \inf_{\varepsilon > 0} \mathcal{Z}^{\Psi + \varepsilon 1_k}.$$

The proof of Theorem 8.2.1 is given in Section 8.2.3.

Remark 8.2.2. *In the easier case where optimal controls exist for the auxiliary problem, as assumed in [BPZ16], and when Ψ is continuous, similar arguments as in [BPZ16] (and Section 8.4) directly prove that $\mathcal{Z}^{\Psi} = V^{\Psi}$ and that an optimal control α^* associated to the auxiliary problem $\mathcal{Y}^{\Psi}(V)$ is optimal for the original problem. However some difficulties arise when trying to remove this assumption.*

Remark 8.2.3. *If there exists $\varepsilon_0 > 0$ such that $V^{\Psi + \varepsilon_0 1_k} < \infty$ then $\mathcal{Z}^{\Psi + \varepsilon_0 1_k} \leq V^{\Psi + \varepsilon_0 1_k} < \infty$ by Theorem 8.2.1. Thus the right-hand side of the previous inequality is finite.*

On the other hand, if we consider for instance a one-dimensional terminal constraint in law $\varphi(\mathbb{P}_{X_T^\alpha}) \leq 0$, it is represented with

$$\Psi : (t, \mu) \mapsto \varphi(\mu) \mathbb{1}_{t=T},$$

and we see that the constraint $\Psi(t, \mu) + \varepsilon \leq 0$ would never be verified for any $t < T$ and any $\varepsilon > 0$, hence $V^{\Psi + \varepsilon} = \infty$.

In view of the above example in Remark 8.2.3, we introduce a modified constraint function in order to deal with discrete time constraints, and also with a.s. constraints. Given a constraint function $\Psi(t,\mu)$, we define

$$\overline{\Psi}_\kappa(t,\mu) := \Psi(t,\mu) - \kappa \sum_{l=1}^{k} \mathbb{1}_{\{\Psi^l(t,\mu)\leq 0\}} e_l, \tag{8.2.6}$$

with $\kappa > 0$ and e_l the l-th component of the canonical basis of $\mathbb{R}^k$. Then it is immediate to see that

$$V^\Psi = V^{\overline{\Psi}_\kappa},\ \mathcal{Y}^\Psi = \mathcal{Y}^{\overline{\Psi}_\kappa},\ \mathcal{Z}^\Psi = \mathcal{Z}^{\overline{\Psi}_\kappa}.$$

Remark 8.2.4. *Notice that by taking $\varepsilon_0 < \kappa$, and assuming that $V^\Psi < \infty$, we have $\mathcal{Z}^{\overline{\Psi}_\kappa+\varepsilon_0 1_k} < \infty$. Indeed, by applying Theorem 8.2.1 to $\overline{\Psi}_\kappa$, we have $\mathcal{Z}^{\overline{\Psi}_\kappa+\varepsilon_0 1_k} \leq V^{\overline{\Psi}_\kappa+\varepsilon_0 1_k}$. Moreover, by observing that an admissible control for the original problem V^Ψ is also admissible for the auxiliary problem with constraint function $\overline{\Psi}_\kappa + \varepsilon_0 1_k$, by definition of $\overline{\Psi}_\kappa$, this implies that $V^{\overline{\Psi}_\kappa+\varepsilon_0 1_k} < \infty$.*

8.2.2 Representation of the value function

Now we prove under some assumptions on the constraints the continuity property $\mathcal{Z}^{\overline{\Psi}_\kappa} = \inf_{\varepsilon>0} \mathcal{Z}^{\overline{\Psi}_\kappa+\varepsilon 1_k}$ in order to obtain a characterization of the original value function V^Ψ. The result relies on convexity arguments.

Lemma 8.2.2. *$(z,\varepsilon) \in \mathbb{R}\times\mathbb{R} \mapsto \mathcal{Y}^{\Psi+\varepsilon 1_k}(z)$ is jointly convex.*

The proof of Lemma 8.2.2 is given in Section 8.2.3.

Proposition 8.2.2. *$\mathcal{Y}^\Psi$ being convex, positive and non-increasing, if $\mathcal{Z}^\Psi < \infty$ then $\mathcal{Y}^\Psi$ is decreasing on $(-\infty, \mathcal{Z}^\Psi]$ then $\mathcal{Y}^\Psi(z) = 0$ on $[\mathcal{Z}^\Psi, \infty)$.*

Proof. By contradiction, if $\mathcal{Y}^\Psi(a) = \mathcal{Y}^\Psi(b) > 0$ with $a < b$ then by monotonicity $\mathcal{Y}^\Psi([a,b]) = \{\mathcal{Y}^\Psi(a)\}$ and $0 \in \partial\mathcal{Y}^\Psi(a)$ thus $\mathcal{Y}^\Psi(x) \geq \mathcal{Y}^\Psi(a) > 0\ \forall x \in \mathbb{R}$ which is not the case because $\mathcal{Z}^\Psi < \infty$. As a consequence, $\mathcal{Y}^\Psi$ is decreasing. Then by continuity of $\mathcal{Y}^\Psi$ and definition of $\mathcal{Z}^\Psi$ we obtain $\mathcal{Y}^\Psi(\mathcal{Z}^\Psi) = 0$. □

Theorem 8.2.2. *Assume that $V^\Psi < \infty$. Then we have the representation*

$$\mathcal{Z}^\Psi = V^\Psi.$$

Moreover ε-optimal controls α^ε for the auxiliary problem $\mathcal{Y}^\Psi(V^\Psi)$ are ε-admissible ε-optimal controls for the original problem in the sense that

$$J(X_0,\alpha^\varepsilon) \leq V^\Psi + \varepsilon,\ \sup_{0\leq s\leq T} \Psi(s, \mathbb{P}_{X_s^{\alpha^\varepsilon}}) \leq \varepsilon.$$

Proof of Theorem 8.2.2. We prove the continuity of $\mathcal{Z}^{\overline{\Psi}_\kappa}$ along the curve $\mathcal{Z}^{\overline{\Psi}_\kappa+\varepsilon 1_k}$ for $\varepsilon \in \mathbb{R}$ where $\overline{\Psi}_\kappa$ is defined in (8.2.6).

Let $\kappa > 0$ and $\varepsilon_0 < \kappa$. By Remark 8.2.4, we know that $\mathcal{Z}^{\overline{\Psi}_\kappa+\varepsilon_0 1_k} < \infty$. We consider the optimization problem

$$\Phi : \varepsilon \in \mathbb{R} \mapsto \inf_z\ z + \chi^\Xi(\varepsilon, z),$$

where χ^Ξ is the indicator function of the non-empty admissible set $\Xi = \{(\varepsilon,z) \in \mathbb{R}^2 \mid \mathcal{Y}^{\overline{\Psi}_\kappa+\varepsilon 1_k}(z) = 0\} = \{(\varepsilon,z) \in \mathbb{R}^2 \mid \mathcal{Y}^{\overline{\Psi}_\kappa+\varepsilon 1_k}(z) \leq 0\}$, namely

$$\chi^{\Xi}(\varepsilon,z)=\begin{cases}0 & \text{if } (\varepsilon,z)\in\Xi\\ +\infty & \text{otherwise.}\end{cases}$$

Note that $\mathcal{Z}^{\overline{\Psi}_\kappa+\varepsilon 1_k}=\inf\{z\mid \mathcal{Y}^{\overline{\Psi}_\kappa+\varepsilon 1_k}(z)=0\}=\Phi(\varepsilon)$. By Proposition 8.2.2, $\mathcal{Y}^{\overline{\Psi}_\kappa+\varepsilon 1_k}(z)$ is jointly convex thus Ξ is convex. Hence, $(\varepsilon,z)\mapsto z+\chi^{\Xi}(\varepsilon,z)$ is jointly convex. Now $\Phi(\varepsilon)$ is convex as the marginal of a jointly convex function. As a consequence, $\varepsilon\in\mathbb{R}\mapsto\Phi(\varepsilon)$ is continuous in zero by noticing that $\varepsilon\in(-\infty,\varepsilon_0)\mapsto\Phi(\varepsilon)\leq\mathcal{Z}^{\overline{\Psi}_\kappa+\varepsilon_0 1_k}<+\infty$ and applying Lemma 2.1 from [ET99]. As a consequence $\mathcal{Z}^{\overline{\Psi}_\kappa}=\inf_{\varepsilon>0}\mathcal{Z}^{\overline{\Psi}_\kappa+\varepsilon 1_k}$. Therefore by Theorem 8.2.1 applied to $\overline{\Psi}_\kappa$, we obtain $\mathcal{Z}^{\overline{\Psi}_\kappa}=V^{\overline{\Psi}_\kappa}$. Then recalling that $\mathcal{Z}^{\Psi}=\mathcal{Z}^{\overline{\Psi}_\kappa}$, $V^{\Psi}=V^{\overline{\Psi}_\kappa}$, the result follows.
Concerning the controls, take $\varepsilon>0$, and consider an ε-optimal control $\alpha^\varepsilon\in\mathcal{A}$ such that

$$\{\widehat{g}(\mathbb{P}_T^{\alpha^\varepsilon})-Z_T^{\mathcal{Z}^\Psi,\alpha^\varepsilon}\}_+ +\sum_{l=1}^k\sup_{s\in[0,T]}\{\Psi^l(s,\mathbb{P}_{X_s^{\alpha^\varepsilon}})\}_+\leq\varepsilon.$$

The two terms on the l.h.s. being non-negative, they both are smaller than ε and thus

$$\widehat{g}(\mathbb{P}_T^{\alpha^\varepsilon})\leq Z_T^{\mathcal{Z}^\Psi,\alpha^\varepsilon}+\varepsilon,\ \text{and}\ \Psi^l(s,\mathbb{P}_{X_s^{\alpha^\varepsilon}})\leq\varepsilon,\ \forall\ s\in[0,T],\ \forall l=1,\cdots,k.$$

Hence

$$J(X_0,\alpha^\varepsilon)\leq\mathcal{Z}^\Psi+\varepsilon=V^\Psi+\varepsilon$$

and

$$\Psi(s,\mathbb{P}_{X_s^{\alpha^\varepsilon}})\leq\varepsilon,\ \forall\ s\in[0,T].$$

□

8.2.3 Proofs

Proof of Proposition 8.2.1. 1) By the inequalities $|\inf_u A(u)-\inf_u B(u)|\leq\sup_u|A(u)-B(u)|$, $|\sup_u A(u)-\sup_u B(u)|\leq\sup_u|A(u)-B(u)|$we obtain for any $z,z'\in\mathbb{R}$

$$\begin{aligned}
&|\mathcal{Y}^\Psi(z)-\mathcal{Y}^\Psi(z')|\\
&=|\inf_{\alpha\in\mathcal{A}}\Big[\{\widehat{g}(\mathbb{P}_{X_T^\alpha})-Z_T^{z,\alpha}\}_+ +\sum_{l=1}^k\sup_{s\in[t,T]}\{\Psi^l(s,\mathbb{P}_{X_s^\alpha})\}_+\Big]\\
&-\inf_{\alpha\in\mathcal{A}}\Big[\{\widehat{g}(\mathbb{P}_{X_T^\alpha})-Z_T^{z',\alpha}\}_+ +\sum_{l=1}^k\sup_{s\in[t,T]}\{\Psi^l(s,\mathbb{P}_{X_s^\alpha})\}_+\Big]|\\
&\leq\sup_{\alpha\in\mathcal{A}}|\{\widehat{g}(\mathbb{P}_{X_T^\alpha})-Z_T^{z,\alpha}\}_+-\{\widehat{g}(\mathbb{P}_{X_T^\alpha})-Z_T^{z',\alpha}\}_+ +\sum_{l=1}^k\sup_{s\in[t,T]}\{\Psi^l(s,\mathbb{P}_{X_s^\alpha})\}_+-\sum_{l=1}^k\sup_{s\in[t,T]}\{\Psi^l(s,\mathbb{P}_{X_s^\alpha})\}_+|\\
&\leq\sup_{\alpha\in\mathcal{A}}|Z_T^{z,\alpha}-Z_T^{z',\alpha}|=|z-z'|,
\end{aligned}$$

by 1-Lipschitz continuity of $x\mapsto\{x\}_+$.

2) Denote by

$$L^\Psi(z,\alpha)=\{\widehat{g}(\mathbb{P}_{X_T^\alpha})-Z_T^{z,\alpha}\}_+ +\sum_{l=1}^k\sup_{s\in[0,T]}\{\Psi^l(s,\mathbb{P}_{X_s^\alpha})\}_+,$$

so that $\mathcal{Y}^\Psi(z)=\inf_{\alpha\in\mathcal{A}}L^\Psi(z,\alpha)$. Then, it is clear that

$$z \leq z' \implies L^{\Psi}(z', \alpha) \leq L^{\Psi}(z, \alpha)$$

hence by minimizing, the same monotonicity property holds also for the value function

$$z \leq z' \implies \mathcal{Y}^{\Psi}(z') \leq \mathcal{Y}^{\Psi}(z).$$

□

Proof of Theorem 8.2.1. 1) $\exists\ \alpha \in \mathcal{A},\ \widehat{g}(\mathbb{P}_{X_T^\alpha}) \leq Z_T^{z,\alpha}$ and $\Psi(s, \mathbb{P}_{X_s^\alpha}) \leq 0,\ \forall\ s \in [0,T]$. Therefore

$$\{\widehat{g}(\mathbb{P}_{X_T^\alpha}) - Z_T^{z,\alpha}\}_+ + \sum_{l=1}^{k} \sup_{s\in[0,T]} \{\Psi^l(s, \mathbb{P}_{X_s^\alpha})\}_+ = 0$$

and by non-negativity of $\mathcal{Y}$ we obtain $\mathcal{Y}^{\Psi}(z) = 0$

2) By continuity of $\mathcal{Y}$ (Proposition 8.2.1) and 1), we obtain $\mathcal{Y}^{\Psi}(V^{\Psi}) = 0$ by taking admissible ε-optimal controls for the original problem and taking the limit $\varepsilon \to 0$. By definition of $\mathcal{Z}^{\Psi}$ the property is established.

3) We assume that exists $\varepsilon_0 > 0$ such that $\mathcal{Z}^{\Psi+\varepsilon_0 1_k} < +\infty$. If it is not the case then $\inf_{\varepsilon>0} \mathcal{Z}^{\Psi+\varepsilon 1_k} = +\infty$ and the inequality is verified. Let $0 < \varepsilon < \varepsilon_0$ satisfying $\mathcal{Z}^{\Psi+\varepsilon 1_k} < \infty$. By continuity of $\mathcal{Y}$ in the z variable (Proposition 8.2.1), $\mathcal{Y}^{\Psi+\varepsilon 1_k}(\mathcal{Z}^{\Psi+\varepsilon 1_k}) = 0$. Then by definition of $\mathcal{Y}^{\Psi+\varepsilon 1_k}$, for $0 < \varepsilon' \leq \varepsilon, \exists\ \alpha^{\varepsilon'} \in \mathcal{A}$ such that

$$\{\widehat{g}(\mathbb{P}_T^{\alpha^{\varepsilon'}}) - Z_T^{\mathcal{Z}^{\Psi+\varepsilon 1_k}, \alpha^{\varepsilon'}}\}_+ + \sum_{l=1}^{k} \sup_{s\in[0,T]} \{\Psi^l(s, \mathbb{P}_{X_s^\alpha}) + \varepsilon\}_+ \leq \varepsilon'.$$

The two terms on the l.h.s. being non-negative, they both are smaller than ε' and thus

$$\widehat{g}(\mathbb{P}_T^{\alpha^{\varepsilon'}}) \leq Z_T^{\mathcal{Z}^{\Psi+\varepsilon 1_k}, \alpha^{\varepsilon'}} + \varepsilon',\ \text{and}\ \Psi^l(s, \mathbb{P}_{X_s^\alpha}) \leq \varepsilon' - \varepsilon \leq 0,\ \forall\ s \in [0,T],\ \forall l = 1, \cdots, k.$$

Hence

$$J(\alpha^{\varepsilon'}) \leq \mathcal{Z}^{\Psi+\varepsilon 1_k} + \varepsilon'$$

and

$$\Psi(s, \mathbb{P}_{X_s^\alpha}) \leq 0,\ \forall\ s \in [0,T].$$

Therefore by arbitrariness of ε' verifying $0 < \varepsilon' < \varepsilon$ we conclude that $V^{\Psi} \leq \mathcal{Z}^{\Psi+\varepsilon 1_k}$. By arbitrariness of ε verifying $0 < \varepsilon < \varepsilon_0$ it follows

$$V^{\Psi} \leq \inf_{\varepsilon\in(0,\varepsilon_0)} \mathcal{Z}^{\Psi+\varepsilon 1_k} = \inf_{\varepsilon>0} \mathcal{Z}^{\Psi+\varepsilon 1_k},$$

where the last equality comes from the non-increasing property of $\mathcal{Z}^{\Psi+\varepsilon 1_k}$ w.r.t. ε. □

Proof of Lemma 8.2.2. 1. Let $0 \le \lambda \le 1$. Then for $z, z', \varepsilon, \varepsilon' \in \mathbb{R}$

$$
\begin{aligned}
&L^{\Psi+\lambda\varepsilon 1_k+(1-\lambda)\varepsilon' 1_k}(\lambda z + (1-\lambda)z', \alpha) \\
&= \{\lambda\Big(\widehat{g}(\mathbb{P}_{X_T^\alpha}) + \int_0^T \widehat{f}\big(s, \mathbb{P}_{(X_s^\alpha, \alpha_s)}\big)\, \mathrm{d}s - z\Big) + (1-\lambda)\Big(\widehat{g}(\mathbb{P}_{X_T^\alpha}) + \int_0^T \widehat{f}\big(s, \mathbb{P}_{(X_s^\alpha, \alpha_s)}\big)\, \mathrm{d}s - z'\Big)\}_+ \\
&+ \sum_{l=1}^k \sup_{s\in[0,T]} \{\lambda\Psi^l(s, \mathbb{P}_{X_s^\alpha}) + \lambda\varepsilon + (1-\lambda)\Psi^l(s, \mathbb{P}_{X_s^\alpha}) + (1-\lambda)\varepsilon'\}_+ \\
&\le \lambda\{\widehat{g}(\mathbb{P}_{X_T^\alpha}) + \int_0^T \widehat{f}\big(s, \mathbb{P}_{(X_s^\alpha, \alpha_s)}\big)\, \mathrm{d}s - z\}_+ + (1-\lambda)\{\widehat{g}(\mathbb{P}_{X_T^\alpha}) + \int_0^T \widehat{f}\big(s, \mathbb{P}_{(X_s^\alpha, \alpha_s)}\big)\, \mathrm{d}s - z'\}_+ \\
&+ \sum_{l=1}^k \sup_{s\in[0,T]} \lambda\{\Psi^l(s, \mathbb{P}_{X_s^\alpha}) + \varepsilon\}_+ + (1-\lambda)\{\Psi^l(s, \mathbb{P}_{X_s^\alpha}) + \varepsilon'\}_+ \\
&\le \lambda L^{\Psi+\varepsilon 1_k}(z, \alpha) + (1-\lambda)L^{\Psi'+\varepsilon' 1_k}(z', \alpha)
\end{aligned}
$$

by convexity of $x \mapsto \{x\}_+$. By minimizing over the controls, the result follows. □

8.2.4 Potential extension towards dynamic programming

If one wants to use dynamic programming in order to solve the auxiliary control problem, it requires to write it down under a Markovian dynamic formulation. Define

$$
X_s^{t,\xi,\alpha} = \xi + \int_t^s b\big(u, X_u^{t,\xi,\alpha}, \alpha_u, \mathbb{P}_{(X_u^{t,\xi,\alpha}, \alpha_u)}\big)\, \mathrm{d}u + \int_t^s \sigma(u, X_u^{t,\xi,\alpha}, \alpha_u, \mathbb{P}_{(X_u^{t,\xi,\alpha}, \alpha_u)})\, \mathrm{d}W_u,
$$

for $t \in [0,T]$, and $\xi \in L^2(\mathcal{F}_t, \mathbb{R}^d)$, and notice that we have the flow property

$$
X_r^{t,\xi,\alpha} \;=\; X_r^{s, X_s^{t,\xi,\alpha}, \alpha}, \quad \mathbb{P}_{X_r^{t,\xi,\alpha}} \;=\; \mathbb{P}_{X_r^{s, X_s^{t,\xi,\alpha}, \alpha}}, \; \forall\, 0 \le s \le r \le T,
$$

coming from existence and pathwise uniqueness in (8.2.1). We thus consider the cost function

$$
J(t,\xi,\alpha) := \mathbb{E}\Big[\int_t^T f\big(s, X_s^{t,\xi,\alpha}, \alpha_s, \mathbb{P}_{(X_s^{t,\xi,\alpha}, \alpha_s)}\big)\, \mathrm{d}s + g\big(X_T^{t,\xi,\alpha}, \mathbb{P}_{X_T^{t,\xi,\alpha}}\big)\Big],
$$

and the value function

$$
V(t,\xi) := \inf_{\alpha\in\mathcal{A}} \{J(t,\xi,\alpha) \mid \Psi(s, \mathbb{P}_{X_s^{t,\xi,\alpha}}) \le 0, \; \forall\, s \in [t,T]\}.
$$

Then we introduce the auxiliary state variable

$$
Z_r^{t,\xi,z,\alpha} := z - \mathbb{E}\Big[\int_t^r f\big(s, X_s^{t,\xi,\alpha}, \alpha_s, \mathbb{P}_{(X_s^{t,\xi,\alpha}, \alpha_s)}\big)\, \mathrm{d}s\Big] = z - \int_t^r \widehat{f}\big(s, \mathbb{P}_{(X_s^{t,\xi,\alpha}, \alpha_s)}\big)\, \mathrm{d}s, \quad t \le r \le T,
$$

and the auxiliary value function is given by

$$
\begin{aligned}
\mathcal{Y}^\Psi(t,\xi,z) &= \inf_{\alpha\in\mathcal{A}} \Big[\{\widehat{g}(\mathbb{P}_{X_T^{t,\xi,\alpha}}) - Z_{t,\xi,z}^\alpha(T)\}_+ + \sup_{s\in[t,u]} \{\Psi^l(s, \mathbb{P}_{X_s^{t,\xi,\alpha}})\}_+\Big)\Big] \\
&=: \inf_{\alpha\in\mathcal{A}} L^\Psi(t,\xi,z,\alpha). \qquad (8.2.7)
\end{aligned}
$$

We can treat the non-Markovian formulation of this problem by introducing as in [BPZ15] an additional state variable $Y_u^{t,\xi,\alpha,m} = \Big(\sum_{l=1}^k \sup_{s\in[t,u]}\{\Psi^l(s, \mathbb{P}_{X_s^{t,\xi,\alpha}})\}_+\Big) \vee m \ge 0$ for $u \ge t$ with $m \in \mathbb{R}$ and the value function

$$
\overline{\mathcal{Y}}^\Psi(t,\xi,z,m) = \inf_{\alpha\in\mathcal{A}} \Big[\{\widehat{g}(\mathbb{P}_{X_T^{t,\xi,\alpha}}) - Z_{t,\xi,z}^\alpha(T)\}_+ + Y_T^{t,\xi,\alpha,m}\Big] =: \inf_{\alpha\in\mathcal{A}} \overline{L}^\Psi(t,\xi,z,m,\alpha).
$$

The two problems are related by

$$\mathcal{Y}^{\Psi}(t,\xi,z) = \overline{\mathcal{Y}}^{\Psi}(t,\xi,z,\sum_{l=1}^{k}\{\Psi^l(t,\mathbb{P}_{X_t^{t,\xi,\alpha}})\}_+).$$

With this formulation, the problem (8.2.7) becomes a Mayer-type Markovian optimal control problem in the augmented state space $[0,T]\times L^2(\mathcal{F}_0,\mathbb{R}^d)\times\mathbb{R}\times\mathbb{R}$. As mentioned in [BPZ15], this procedure is used for instance for hedging lookback options in finance, see e.g. [GHLT14]. Now the infimum of the zero level-set is given by

$$\mathcal{Z}^{\Psi}(t,\xi) := \inf\{z\in\mathbb{R} \mid \overline{\mathcal{Y}}^{\Psi}(t,\xi,z,0)=0\}.$$

Indeed note that $\overline{\mathcal{Y}}^{\Psi}(t,\xi,z,m)=0 \iff m\le 0$ and $\overline{\mathcal{Y}}^{\Psi}(t,\xi,z,0)=0$.

The Lipschitz and convexity properties of the value function are proven exactly as in Proposition 8.2.1 but we detail here the continuity in space and in the running maximum variable m.

Assumption 8.2.1. *Ψ, f, g, b, σ are Lipschitz continuous uniformly with respect to to other variables. Namely exists $[\Psi],[f],[g],[b],[\sigma],L>0$ and locally bounded functions $h,\ell,\mathfrak{L}:[0,+\infty)\mapsto[0,+\infty)$ such that for any $t\in[0,T]$, $x,x'\in\mathbb{R}^d,\mu\in\mathcal{P}_2(\mathbb{R}^d),\nu,\nu'\in\mathcal{P}_2(\mathbb{R}^d\times A)$, $a\in A$*

$$\begin{aligned}
|\Psi(t,\mu)-\Psi(t,\mu')| &\le [\Psi]\mathcal{W}_2(\mu,\mu')\\
|f(t,x,a,\nu)-f(t,x',a,\nu')| &\le [f](|x-x'|+\mathcal{W}_2(\nu,\nu'))\\
|g(x,\mu)-g(x,\mu')| &\le [g](|x-x'|+\mathcal{W}_2(\mu,\mu'))\\
|b(t,x,a,\nu)-b(t,x',a,\nu')| &\le [b](|x-x'|+\mathcal{W}_2(\nu,\nu'))\\
|\sigma(t,x,a,\nu)-\sigma(t,x',a,\nu')| &\le [\sigma](|x-x'|+\mathcal{W}_2(\nu,\nu'))\\
|b(t,0,a,\delta_0\otimes\mu)|+|\sigma(t,0,a,\delta_0\otimes\mu)|+|f(t,0,a,\delta_0\otimes\mu)| &\le L\\
|f(t,x,a,\nu)| &\le h(\|\nu\|_2)(1+|x|^2)\\
|g(x,\mu)| &\le \ell(\|\mu\|_2)(1+|x|^2)\\
|\Psi(t,\mu)| &\le \mathfrak{L}(\|\mu\|_2).
\end{aligned}$$

Proposition 8.2.3. *Under Assumption 8.2.1 $\overline{\mathcal{Y}}^{\Psi}$ is Lipschitz continuous: there exists $C>0$ such that for any $t\in[0,T]$, $\xi,\xi'\in L^2(\mathcal{F}_t,\mathbb{R}^d),m,m'\in\mathbb{R}$*

$$|\overline{\mathcal{Y}}^{\Psi}(t,\xi,z,m)-\overline{\mathcal{Y}}^{\Psi}(t,\xi',z',m')| \le |z-z'|+|m-m'|+C\sqrt{\mathbb{E}|\xi-\xi'|^2}.$$

Proof of Proposition 8.2.3. By the inequalities $|\inf_u A(u)-\inf_u B(u)|\le\sup_u|A(u)-B(u)|$, $|\sup_u A(u)-\sup_u B(u)|\le\sup_u|A(u)-B(u)|$, and $|a\vee b-c\vee d|\le|a-c|\vee|b-d|\le|a-c|+|b-d|$

we obtain for any $\xi, \xi' \in L^2(\mathcal{F}_t, \mathbb{R}^d)$ (if Ψ is not continuous consider $\xi = \xi'$)

$$
\begin{aligned}
&|\overline{\mathcal{Y}}^{\Psi}(t,\xi,z,m) - \overline{\mathcal{Y}}^{\Psi}(t,\xi',z',m')| \\
&\leq \sup_{\alpha\in\mathcal{A}} |\{\widehat{g}(\mathbb{P}_{X_T^{t,\xi,\alpha}}) - Z_T^{t,\xi,z,\alpha}\}_+ - \{\widehat{g}(\mathbb{P}_{X_T^{t,\xi',\alpha}}) - Z_T^{t,\xi',z',\alpha}\}_+ + \\
&\sup_{s\in[t,T]} \{\Psi(s,\mathbb{P}_{X_s^{t,\xi,\alpha}})\}_+ \vee m - \sup_{s\in[t,T]} \{\Psi(s,\mathbb{P}_{X_s^{t,\xi',\alpha}})\}_+ \vee m'| \\
&\leq \sup_{\alpha\in\mathcal{A}} (|\widehat{g}(\mathbb{P}_{X_T^{t,\xi,\alpha}}) - \widehat{g}(\mathbb{P}_{X_T^{t,\xi',\alpha}})| + |Z_T^{t,\xi,z,\alpha} - Z_T^{t,\xi',z',\alpha}| + |\sup_{s\in[t,T]} \{\Psi(s,\mathbb{P}_{X_s^{t,\xi,\alpha}})\}_+ - \sup_{s\in[t,T]} \{\Psi(s,\mathbb{P}_{X_s^{t,\xi',\alpha}})\}_+| \\
&+ |m - m'|) \\
&\leq \sup_{\alpha\in\mathcal{A}} \Big(|\mathbb{E}[g(X_T^{t,\xi,\alpha}, \mathbb{P}_{X_T^{t,\xi,\alpha}}) - g(X_T^{t,\xi',\alpha}, \mathbb{P}_{X_T^{t,\xi',\alpha}})]| + |z - z'| \\
&+ \Big| \mathbb{E}\Big[\int_t^T f\big(s, X_s^{t,\xi,\alpha}, \alpha_s, \mathbb{P}_{(X_s^{t,\xi,\alpha},\alpha_s)}\big)\, \mathrm{d}s - \int_t^T f\big(s, X_s^{t,\xi',\alpha}, \alpha_s, \mathbb{P}_{(X_s^{t,\xi',\alpha},\alpha_s)}\big)\, \mathrm{d}s \Big] \Big| \Big) \\
&+ \sup_{\alpha\in\mathcal{A}} \sup_{s\in[t,T]} |\{\Psi(s,\mathbb{P}_{X_s^{t,\xi,\alpha}})\}_+ - \{\Psi(s,\mathbb{P}_{X_s^{t,\xi',\alpha}})\}_+| + |m - m'| \\
&\leq [\widehat{g}] \sup_{\alpha\in\mathcal{A}} (\mathbb{E}|X_T^{t,\xi,\alpha} - X_T^{t,\xi',\alpha}| + \mathcal{W}_2(\mathbb{P}_{X_T^{t,\xi,\alpha}}, \mathbb{P}_{X_T^{t,\xi',\alpha}})) + |z - z'| + |m - m'| + [\Psi] \sup_{\alpha\in\mathcal{A}} \sup_{s\in[t,T]} \mathcal{W}_2(\mathbb{P}_{X_s^{t,\xi,\alpha}}, \mathbb{P}_{X_s^{t,\xi',\alpha}}) \\
&+ T[f] \sup_{\alpha\in\mathcal{A}} \{\mathbb{E}[\sup_{s\in[t,T]} |X_s^{t,\xi,\alpha} - X_s^{t,\xi',\alpha}|] + \sup_{s\in[t,T]} \mathcal{W}_2(\mathbb{P}_{X_s^{t,\xi,\alpha}}, \mathbb{P}_{X_s^{t,\xi',\alpha}})\},
\end{aligned}
$$

by Lipschitz continuity of $\Psi, x \mapsto \{x\}_+$. We recall the estimates

$$
\begin{aligned}
\sup_{s\in[t,T]} \mathcal{W}_2(\mathbb{P}_{X_s^{t,\xi,\alpha}}, \mathbb{P}_{X_s^{t,\xi',\alpha}}) &= \sqrt{\sup_{s\in[t,T]} \mathcal{W}_2(\mathbb{P}_{X_s^{t,\xi,\alpha}}, \mathbb{P}_{X_s^{t,\xi',\alpha}})^2} \leq C\sqrt{\mathbb{E}|\xi - \xi'|^2} \\
\mathbb{E}[\sup_{s\in[t,T]} |X_s^{t,\xi,\alpha} - X_s^{t,\xi',\alpha}|] &\leq C\mathbb{E}|\xi - \xi'| \leq C\sqrt{\mathbb{E}|\xi - \xi'|^2},
\end{aligned}
$$

obtained by standard arguments (see e.g. the proof of Proposition 3.3 in [CP19]). Then the result follows.

□

Proposition 8.2.4 (Law invariance properties)**.** *Under Assumption 8.2.1, we have law invariance of $\overline{\mathcal{Y}}^{\Psi}$ and $\mathcal{Z}^{\Psi}$, namely if ξ, η are $\mathcal{F}_t$-adapted square integrable with the same law, for any $(t,z,m) \in [0,T] \times \mathbb{R} \times \mathbb{R}$*

$$
\begin{aligned}
\overline{\mathcal{Y}}^{\Psi}(t,\xi,z,m) &= \overline{\mathcal{Y}}^{\Psi}(t,\eta,z,m) \\
\mathcal{Z}^{\Psi}(t,\xi) &= \mathcal{Z}^{\Psi}(t,\eta).
\end{aligned}
$$

Therefore we can define the lifted functions y^{Ψ}, z^{Ψ} on $[0,T] \times \mathcal{P}_2(\mathbb{R}^d) \times \mathbb{R}$ (respectively $[0,T] \times \mathcal{P}_2(\mathbb{R}^d)$ by $y^{\Psi}(t,\mathbb{P}_{\xi},z,m) := \overline{\mathcal{Y}}^{\Psi}(t,\xi,z,m)$ and $z^{\Psi}(t,\mathbb{P}_{\xi},z,m) := \mathcal{Z}^{\Psi}(t,\xi,z,m)$.

Proof. Apply the same arguments as in Theorem 3.5. from [Cos+20] to the unconstrained Markovian value function $\mathcal{Y}^{\Psi}$ on the extended state space. In particular use the continuity of $\mathcal{Y}^{\Psi}$ from Proposition 8.2.1 and notice for a given control α that in Step 1 of Theorem 3.5. from [Cos+20] the equality in law

$$
\begin{aligned}
&((X_s^{t,\xi,\alpha})_{s\in[t,T]}, (Z_s^{t,\xi,z,\alpha})_{s\in[t,T]}, (Y_u^{t,\xi,\alpha,m})_{s\in[t,T]}, (\alpha_s)_{s\in[t,T]}) \\
&\overset{\mathcal{L}}{=} ((X_s^{t,\eta,\beta})_{s\in[t,T]}, (Z^{\beta,t,\eta,z})_{s\in[t,T]}, (Y_u^{t,\eta,\beta,m})_{s\in[t,T]}, (\beta_s)_{s\in[t,T]}),
\end{aligned}
$$

holds true with a_s defined in Lemma B.2. from [Cos+20] (verifying in particular the equality in law $\alpha_s = a_s(\xi, U_\xi)$) and $\beta_s = a_s(\eta, U_\eta)$ where U_η (respectively U_ξ) is a $\mathcal{F}_t$-adapted uniform

random variable on $[0,1]$ independent of η (respectively ξ). Then use the definition (8.2.5) to obtain the same law invariance property for $\mathcal{Z}^\Psi$ too. □

Theorem 8.2.1 and Theorem 8.2.2 are still valid in the the dynamic case, by applying the exact same arguments. More precisely for any $(t,\xi) \in [0,T] \times L^2(\mathcal{F}_t, \mathbb{R}^d)$, if $V^\Psi(t,\xi) < \infty$ then

$$\mathcal{Z}^\Psi(t,\xi) \leq V^\Psi(t,\xi) \leq \inf_{\varepsilon>0} \mathcal{Z}^{\Psi+\varepsilon 1_k}(t,\xi).$$

Similarly, arguments like in Theorem 8.2.2 prove that

$$\mathcal{Z}^\Psi(t,\xi) = V^\Psi(t,\xi),$$

if $V^\Psi(t,\xi) < \infty$.

If the value function is law invariant (see Proposition 8.2.4) and Theorem 8.2.2 holds true, we expect y to be formally (by combining arguments from [BPZ15; CP19]) characterized by a Master Bellman equation in Wassertein space with oblique derivative boundary conditions.

8.3 An alternative auxiliary problem

We study the constrained McKean-Vlasov control problem

$$V := \inf_{\alpha\in\mathcal{A}} \big\{ J(X_0,\alpha) : \Psi(t,\mathbb{P}_{X_t^\alpha}) \leq 0,\ \forall\, t\in[0,T],\ \varphi(\mathbb{P}_{X_T^\alpha}) \leq 0 \big\},$$

where we now assume that the running constraint Ψ is continuous (hence, no discrete time constraints, see Remark 8.4.1), and with a terminal constraint function φ. We now consider an alternative auxiliary control problem as in [BPZ16]:

$$w(z) := \inf_{\alpha\in\mathcal{A}} \Big[\{\widehat{g}(\mathbb{P}_{X_T^\alpha}) - Z_T^{z,\alpha}\}_+ + \sum_{l=1}^k \int_0^T \{\Psi^l(s,\mathbb{P}_{X_s^\alpha})\}_+ \,\mathrm{d}s + \{\varphi(\mathbb{P}_{X_T^\alpha})\}_+ \Big]. \tag{8.3.1}$$

Compared to the control problem (8.2.4) of the previous section, the penalization term of the constrained function Ψ is in integral form instead of a supremum form. It follows that this problem is not path-dependent, and we shall show that it also provides a similar representation of the value function by its zero level set:

$$V = \inf\{z\in\mathbb{R} : w(z) = 0\},$$

but under the additional assumption that optimal controls do exist. Actually, we prove this result in the more general case with common noise in the next section.

The mean-field control problem (8.3.1) is Markovian with respect to the state variables $(X_t^\alpha, \mathbb{P}_{X_t^\alpha}, Z_t^{z,\alpha})$, and it is known from [Cos+20] that the infimum over open-loop controls α in $\mathcal{A}$ can be taken equivalently over randomized feedback policies, i.e. controls α in the form: $\alpha_t = \mathfrak{a}(t, X_t^\alpha, \mathbb{P}_{X_t^\alpha}, Z_t^{z,\alpha}, U)$, for some deterministic function $\mathfrak{a}$ from $[0,T]\times\mathbb{R}^d\times\mathcal{P}(\mathbb{R}^d)\times\mathbb{R}\times[0,1]$ into A, where U is an $\mathcal{F}_0$-measurable uniform random variable on $[0,1]$.

Let us now discuss conditions under which the infimum in (8.3.1) can be taken equivalently over (deterministic) feedback policies, i.e. for controls α in the form: $\alpha_t = \mathfrak{a}(t, X_t^\alpha, \mathbb{P}_{X_t^\alpha}, Z_t^{z,\alpha})$, for some deterministic function $\mathfrak{a}$ from $[0,T]\times\mathbb{R}^d\times\mathcal{P}(\mathbb{R}^d)\times\mathbb{R}$ into A. This will be helpful for numerical purpose in Section 8.5. We assume on top of Assumption 8.2.1 that the running cost f, the drift b and the volatility coefficient σ do not depend on the law of the control process. We also assume that the running cost $f = f(t,x,\mu)$ does not depend on the control argument. The terminal constraint function φ should also verify the same assumptions as the terminal cost function g, namely Lipschitz continuity and local boundedness (see Assumption 8.2.1).

In this case, the corresponding dynamic auxiliary problem of (8.3.1) is written as

$$\begin{aligned}
\mathrm{w}(t,\mu,z) &= \inf_{\alpha\in\mathcal{A}} \Big[\{\widehat{g}(\mathbb{P}_{X_T^{t,\xi,\alpha}}) - Z_T^{t,\xi,z,\alpha}\}_+ + \sum_{l=1}^{k}\int_0^T \{\Psi^l(s,\mathbb{P}_{X_s^{t,\xi,\alpha}})\}_+ \,\mathrm{d}s + \{\varphi(\mathbb{P}_{X_T^{t,\xi,\alpha}})\}_+\Big] \quad (8.3.2)\\
X_r^{t,\xi,\alpha} &= \xi + \int_t^r b\big(s, X_s^{t,\xi,\alpha}, \alpha_s, \mathbb{P}_{X_s^{t,\xi,\alpha}}\big)\,\mathrm{d}s + \int_t^r \sigma\big(s, X_s^{t,\xi,\alpha}, \alpha_t, \mathbb{P}_{X_s^{t,\xi,\alpha}}\big)\,\mathrm{d}W_s, \quad \xi\sim\mu,\\
Z_r^{t,\xi,z,\alpha} &= z - \int_t^r \bar{f}\big(s, \mathbb{P}_{X_s^{t,\xi,\alpha}}\big)\,\mathrm{d}s, \quad r\geq t,
\end{aligned}$$

where $\bar{f}$ is the function defined on $[0,T]\times\mathcal{P}_2(\mathbb{R}^d)$ by $\bar{f}(t,\mu) = \int_{\mathbb{R}^d} f\big(t,x,\mu\big)\,\mu(\mathrm{d}x)$. Note that we have applied Theorem 3.5 from [Cos+20] to obtain the law invariance of the auxiliary value function which can be written as a function of the measure μ. From Theorem 3.5, Proposition 5.6. 2), and equation (5.17) in [Cos+20] (see also Remark 5.2. from [CP19] and Section 6 in [PW18]) we see that the Bellman equation for problem (8.3.2) is:

$$\begin{cases}
\partial_t \mathrm{w}(t,\mu,z) + \mathbb{E}[\inf_{a\in A}\{b\big(t,\xi,a,\mu\big)\partial_\mu \mathrm{w}(t,\mu,z)(\xi) - \bar{f}(t,\mu)\partial_z \mathrm{w}(t,\mu,z)\\
\quad +\frac{1}{2}\mathrm{Tr}(\sigma\sigma^\top\big(t,\xi,a,\mu\big)\partial_x\partial_\mu \mathrm{w}(t,\mu,z)(\xi))\}] + \sum_{l=1}^k\{\Psi^l(t,\mu)\}_+ = 0 \text{ for } (t,\mu,z)\in[0,T]\times\mathcal{P}_2(\mathbb{R}^d)\times\mathbb{R}\\
\mathrm{w}(T,\mu,z) = \{\widehat{g}(\mu) - z\}_+ + \{\varphi(\mu)\}_+ \text{ for } (\mu,z)\in\mathcal{P}_2(\mathbb{R}^d)\times\mathbb{R}\,.
\end{cases} \quad (8.3.3)$$

By assuming that w is a smooth solution to this Bellman equation, and when the infimum in

$$\inf_{a\in A}\{b\big(t,x,a,\mu\big)\partial_\mu \mathrm{w}(t,\mu,z)(x) - \bar{f}(t,\mu)\partial_z \mathrm{w}(t,\mu,z) + \frac{1}{2}\mathrm{Tr}(\sigma\sigma^\top\big(t,x,a,\mu\big)\partial_x\partial_\mu \mathrm{w}(t,\mu,z)(x))\}$$

is attained for some measurable function $\hat{\mathfrak{a}}(t,x,\mu,z)$ on $[0,T]\times\mathbb{R}^d\times\mathcal{P}(\mathbb{R}^d)\times\mathbb{R}$, we get an optimal control for (8.3.1) given in feedback form by $\alpha_t^* = \hat{\mathfrak{a}}(t, X_t^{\alpha^*}, \mathbb{P}_{X_t^{\alpha^*}}, Z_t^{z,\alpha^*})$, $0\leq t\leq T$, which shows that one can restrict in (8.3.1) to deterministic feedback policies.

8.4 Extension to the common noise setting

We briefly discuss how the state constraints can be extended to mean-field control problems with common noise. In this case, in contrast with the previous section, we need to assume the existence of optimal control for the auxiliary unconstrained problem. It is similar to the assumption made by [BPZ16]. Let W^0 be a p-dimensional Brownian motion independent of W, and denote by $\mathbb{F}^0 = (\mathcal{F}_t^0)_t$ the filtration generated by W^0. We consider the following cost and dynamics:

$$\begin{aligned}
J(\alpha) &= \mathbb{E}\Big[\int_0^T f\big(t, X_t^\alpha, \alpha_t, \mathbb{P}^{W^0}_{(X_t^\alpha,\alpha_t)}\big)\,\mathrm{d}t + g\big(X_T^\alpha, \mathbb{P}^{W^0}_{X_T}\big)\Big]\\
\mathrm{d}X_t^\alpha &= b\big(t, X_t^\alpha, \alpha_t, \mathbb{P}^{W^0}_{(X_t^\alpha,\alpha_t)}\big)\,\mathrm{d}t + \sigma\big(t, X_t^\alpha, \alpha_t, \mathbb{P}^{W^0}_{(X_t^\alpha,\alpha_t)}\big)\,\mathrm{d}W_t + \sigma^0\big(t, X_t^\alpha, \alpha_t, \mathbb{P}^{W^0}_{(X_t^\alpha,\alpha_t)}\big)\,\mathrm{d}W_t^0.
\end{aligned}$$

where $\mathbb{P}^{W^0}_{(X_t^\alpha,\alpha_t)}$ is the joint conditional law of (X_t^α,α_t) given W^0. The control process α belongs to a set $\mathcal{A}$ of $\mathbb{F}$-progressively measurable processes with values in a set $A\subset\mathbb{R}^q$.

The controlled McKean-Vlasov process X is constrained to verify $\Psi(t,\mathbb{P}^{W_0}_{X_t^\alpha})\leq 0$ and $\varphi(\mathbb{P}^{W_0}_{X_T^\alpha})\leq 0$. The proofs still follow the arguments from [BPZ16] but are slightly more involved than in Section 8.2 due to the additional noise appearing in the conditional law with respect to the common noise. We refer to [PW17; DPT19] for the dynamic programming approach to these problems. The problem of interest is

$$V^0 = \inf_{\alpha\in\mathcal{A}}\{J(\alpha)\mid \Psi(t,\mathbb{P}^{W_0}_{X_t^\alpha})\leq 0,\ \forall\, t\in[0,T],\ \varphi(\mathbb{P}^{W_0}_{X_T^\alpha})\leq 0\ \}.$$

8.4.1 Representation by a stochastic target problem and an associated control problem

Given $z \in \mathbb{R}$, $\alpha \in \mathcal{A}$, and $\beta \in L^2(\mathbb{F}^0, \mathbb{R}^p)$, the set of $\mathbb{R}^p$-valued $\mathbb{F}^0$-adapted processes β s.t. $\mathbb{E}[\int_0^T |\beta_t|^2 \mathrm{d}t] < \infty$, define

$$Z_t^{z,\alpha,\beta} := z - \int_0^t \widehat{f}(s, \mathbb{P}^{W^0}_{(X_s^\alpha, \alpha_s)})\, \mathrm{d}s + \int_0^t \beta_s\, \mathrm{d}W_s^0, \quad 0 \le t \le T. \tag{8.4.1}$$

Lemma 8.4.1. *The value function admits the* ***stochastic target problem*** *representation*

$$V^0 = \inf\{z \in \mathbb{R} \mid \exists\, (\alpha, \beta) \in \mathcal{A} \times L^2(\mathbb{F}^0, \mathbb{R}^p) \text{ s.t. } \widehat{g}(\mathbb{P}^{W^0}_{X_T^\alpha}) \le Z_T^{z,\alpha,\beta}, \\ \Psi(t, \mathbb{P}^{W^0}_{X_t^\alpha}) \le 0,\ \forall\, t \in [0,T],\ \varphi(\mathbb{P}^{W^0}_{X_T^\alpha}) \le 0,\ \mathbb{P} \text{ a.s.}\}.$$

Lemma 8.4.1 is proven in Section 8.4.2.

Define the **auxiliary unconstrained** control problem

$$\mathcal{U}(z) := \inf_{(\alpha,\beta) \in \mathcal{A} \times L^2(\mathbb{F}^0, \mathbb{R}^p)} \mathbb{E}\Big[\{\widehat{g}(\mathbb{P}^{W^0}_{X_T^\alpha}) - Z_T^{z,\alpha,\beta}\}_+ + \sum_{l=1}^k \int_0^T \{\Psi^l(s, \mathbb{P}^{W^0}_{X_s^\alpha})\}_+\, \mathrm{d}s + \{\varphi(\mathbb{P}^{W^0}_{X_T^\alpha})\}_+\Big] \tag{8.4.2}$$

for $z \in \mathbb{R}$. We notice that $\mathcal{U}(z) \ge 0$.

Proposition 8.4.1. *$\mathcal{U}$ is 1-Lipschitz. For any $z, z' \in \mathbb{R}$*

$$|\mathcal{U}(z) - \mathcal{U}(z')| \le |z - z'|.$$

Proposition (8.4.1) is proven exactly as (8.2.1).

Assumption 8.4.1. *Problem* (8.4.2) *admits an optimal control for any $z \in \mathbb{R}$ and the constraint function $(t, \mu) \in [0,T] \times \mathcal{P}_2(\mathbb{R}^d) \mapsto \Psi(t, \mu)$ is continuous.*

Remark 8.4.1. *Please note that the integral penalization in* (8.4.2) *does not allow to consider discrete times constraints (except at terminal time) because the contribution to the integral would be null and the constraint function Ψ would be discontinuous in time. We could consider discrete time constraints in the objective of the auxiliary problem by adding a sum of functions of $\mathbb{P}^{W^0}_{X_{t_i}^\alpha}$ for some $(t_i)_i \in [0,T]$ but it would lose its standard Bolza form.*

Define $\mathcal{Z} = \inf\{z \in \mathbb{R} \mid \mathcal{U}(z) = 0\}$.

Theorem 8.4.1. *1. If $\exists\, (\alpha, \beta) \in \mathcal{A} \times L^2(\mathbb{F}^0, \mathbb{R}^p)$, $\widehat{g}(\mathbb{P}^{W^0}_{X_T^\alpha}) \le Z_T^{z,\alpha,\beta}$, $\Psi(s, \mathbb{P}^{W^0}_{X_s^\alpha}) \le 0,\ \forall\, s \in [0,T]$, and $\varphi(\mathbb{P}^{W^0}_{X_T^\alpha}) \le 0,\ \mathbb{P}$ a.s. then $\mathcal{U}(z) = 0$. Hence $\mathcal{Z} \le V^0$.*

2. The value function verifies $V^0 \le \mathcal{Z}$ thus $V^0 = \mathcal{Z}$. Moreover optimal controls for the problem $\mathcal{U}(\mathcal{Z}) = 0$ are optimal for the original problem.

Theorem 8.4.1 is proven in Section 8.4.2.

8.4.2 Proofs in the common noise framework

Proof of Lemma 8.4.1. We first observe that

$$V^0 = \inf\{z \in \mathbb{R} \mid \exists\, \alpha \in \mathcal{A} \text{ s.t. } \mathbb{E}\Big[\int_0^T \widehat{f}(s, \mathbb{P}^{W^0}_{(X_s^\alpha, \alpha_s)})\, \mathrm{d}s + \widehat{g}(\mathbb{P}^{W^0}_{X_T^\alpha})\Big] \le z, \\ \Psi(s, \mathbb{P}^{W^0}_{X_s^\alpha}) \le 0,\ \forall\, s \in [0,T],\ \mathbb{P}^0 \text{ a.s.}\}.$$

We need to prove that for $z \in \mathbb{R}$

$$\exists\, (\alpha,\beta) \in \mathcal{A}\times L^2(\mathbb{F}^0,\mathbb{R}^p) \text{ s.t. } \widehat{g}(\mathbb{P}^{W^0}_{X^\alpha_T}) \le Z^{z,\alpha,\beta}_T,\ \Psi(s,\mathbb{P}^{W^0}_{X^\alpha_s}) \le 0,\ \forall\, s\in[0,T],\ \varphi(\mathbb{P}^{W^0}_{X^\alpha_T}) \le 0,\ \mathbb{P}^0 \text{ a.s.}, \tag{8.4.3}$$

and

$$\exists\, \alpha \in \mathcal{A} \text{ s.t. } \mathbb{E}\Big[\int_0^T \widehat{f}(s,\mathbb{P}^{W^0}_{(X^\alpha_s,\alpha_s)})\,\mathrm{d}s + \widehat{g}(\mathbb{P}^{W^0}_{X^\alpha_T})\Big] \le z,\ \Psi(s,\mathbb{P}^{W^0}_{X^\alpha_s}) \le 0,\ \forall\, s\in[0,T],\ \varphi(\mathbb{P}^{W^0}_{X^\alpha_T}) \le 0,\ \mathbb{P}^0 \text{ a.s.}, \tag{8.4.4}$$

are equivalent. It is immediate to see that (8.4.3) $\implies$ (8.4.4) by taking the expectation and noticing that the Itô integral is a true martingale. Conversely, assuming (8.4.4), the martingale representation theorem provides a process $\widehat{\beta}$ such that

$$\begin{aligned} z &\ge \mathbb{E}\Big[\int_0^T \widehat{f}(s,\mathbb{P}^{W^0}_{(X^\alpha_s,\alpha_s)})\,\mathrm{d}s + \widehat{g}(\mathbb{P}^{W^0}_{X^\alpha_T})\Big] \\ &= \int_0^T \widehat{f}(s,\mathbb{P}^{W^0}_{(X^\alpha_s,\alpha_s)})\,\mathrm{d}s + \widehat{g}(\mathbb{P}^{W^0}_{X^\alpha_T}) - \int_0^T \widehat{\beta}_s\,\mathrm{d}W^0_s. \end{aligned}$$

Thus by (8.4.1)

$$Z^{z,\alpha,\widehat{\beta}}_T \ge \widehat{g}(\mathbb{P}^{W^0}_{X^\alpha_T}),\ \mathbb{P}^0 \text{ a.s.},$$

and we see that (8.4.4) $\implies$ (8.4.3). Then the result follows. □

Proof of Theorem 8.4.1. 1) $\exists\, (\alpha,\beta) \in \mathcal{A}\times L^2(\mathbb{F}^0,\mathbb{R}^p)$, $\widehat{g}(\mathbb{P}^{W^0}_{X^\alpha_T}) \le Z^{z,\alpha,\beta}_T$, $\Psi(s,\mathbb{P}^{W^0}_{X^\alpha_s}) \le 0,\ \forall\, s \in [0,T]$ and $\varphi(\mathbb{P}^{W^0}_{X^\alpha_T}) \le 0,\ \mathbb{P}^0$ a.s.. Therefore

$$\{\widehat{g}(\mathbb{P}^{W^0}_{X^\alpha_T}) - Z^{z,\alpha,\beta}_T\}_+ + \sum_{l=1}^k \int_0^T \{\Psi^l(s,\mathbb{P}^{W^0}_{X^\alpha_s})\}_+\,\mathrm{d}s + \{\varphi(\mathbb{P}^{W^0}_{X^\alpha_T})\}_+ = 0,\ \mathbb{P}^0 \text{ a.s.}$$

and by non-negativity of $\mathcal{U}$ we obtain $\mathcal{U}(z) = 0$. Then with optimal controls α^*, β^* we obtain $\mathcal{U}(V^0) = 0$. By definition of $\mathcal{Z}$ the property is established.

2) By 1) and the continuity given by Proposition 8.4.1, we obtain $\mathcal{U}(\mathcal{Z}) = 0$. Then by Assumption 8.4.1 $\exists\, (\alpha,\beta) \in \mathcal{A}\times L^2(\mathbb{F}^0,\mathbb{R}^p)$ such that

$$\mathbb{E}^0\Big[\{\widehat{g}(\mathbb{P}^{W^0}_{X^\alpha_T}) - Z^{\mathcal{Z},\alpha,\beta}_T\}_+ + \sum_{l=1}^k \int_0^T \{\Psi(s,\mathbb{P}^{W^0}_{X^\alpha_s})\}_+\,\mathrm{d}s + \{\varphi(\mathbb{P}^{W^0}_{X^\alpha_T})\}_+\Big] = 0.$$

The three terms on the l.h.s. being non-negative, they are in fact null $\mathbb{P}$ a.s. Thus

$$(\mathbb{P}^{W^0}_{X^\alpha_T}, Z^{\mathcal{Z},\alpha,\beta}_T) \in \mathrm{Epi}(\widehat{g}),\ \Psi(s,\mathbb{P}^{W^0}_{X^\alpha_s}) \le 0\ \forall\, s\in[0,T],\ \text{and}\ \varphi(\mathbb{P}^{W^0}_{X^\alpha_T}) \le 0 \quad \mathbb{P} \text{ a.s.}$$

by continuity of Ψ and of $s \mapsto \mathbb{P}^{W^0}_{X^\alpha_s}$, which means $V^0 \le \mathcal{Z}$. By 1) it yields $V^0 = \mathcal{Z}$. As a consequence the previous proof provides an optimal control α for the original problem. □

8.5 Applications and numerical tests

We design several machine learning methods to solve this problem. We discretize the problem in time, parametrize the control by a neural network and directly minimize the cost. When the constraints are almost sure, we can sometimes enforce them by choosing an appropriate neural network architecture, for instance in storage problems. A more adaptive alternative is to solve the unconstrained auxiliary problem. We propose an extension of the first algorithm from [CL22] to achieve this task. Thus we obtain a machine learning method able to solve state constrained mean field control problems.

8.5.1 Algorithms

We solve the auxiliary problem in the simpler case without common noise with a first algorithm. We fix a relevant line segment K of $\mathbb{R}$ on which we are going to explore the potential values of the problem. For instance we know that the value is greater than the value of the unconstrained problem V therefore it is useless to compute the auxiliary value function for $z \leq V$. We discretize the problem in time on the grid $t_k := \frac{kT}{N}$. We call $\Delta t := \frac{kT}{N}$ and the Brownian increment $\Delta W_i := W_{t_{i+1}} - W_{t_i}$. For $j = 1, \cdots, N$, ΔW_i^j (respectively X_0^j) correspond to samples from N independent Brownian motions W^j (respectively from N independent random variables with law μ_0). For training we discretize K by using N points. We choose ε as a small parameter, typically of order 10^{-6}. We refer to [BGZ21] for results on the numerical approximation of level sets with a given threshold in the context of constrained deterministic optimal control. We propose the following extension of the Method 1 from [CL22]. It is tested in Subsection 8.5.2. It can indeed also be used to solve unconstrained problem.

Remark 8.5.1. *We point out that adding an additional parameter $\Lambda > 0$ in front of the constraint function does not modify the representation results. In that case we solve the following auxiliary problem*

$$\mathcal{Y}_\Lambda^\Psi := z \in \mathbb{R} \mapsto \inf_{\alpha \in \mathcal{A}} \Big[\{\widehat{g}(\mathbb{P}_{X_T^\alpha}) - Z_T^{z,\alpha}\}_+ + \Lambda \sum_{l=1}^{k} \int_0^T \{\Psi^l(s, \mathbb{P}_{X_s^\alpha})\}_+ \, \mathrm{d}s + \Lambda \varphi(\mathbb{P}_{X_T^\alpha}) \Big].$$

We discretize the problem in time and use a neural network by time step, since a single network taking time as input is usually not sufficient enough for complex problems, as shown in [War21b]. In view of the discussion about closed-loop controls in Section 8.3, the neural network representing the control at each time step takes as inputs the current states X and $Z_i^{z,\alpha}$ where z is taken on a discretization of K. The method is described in Algorithm 11 with an example in Section 8.5.2. Solving (8.3.3) with the approach of [Ger+22] would provide another numerical method for mean-field control with state constraints. The extension to the common noise case is given in Algorithm 12 where the neural network for the control at each time step t_i takes in addition as input the current value of the common noise $W_{t_i}^0$. Notice that in general, the control may depend on the past values of the common noise, which could be taken into account in the neural network by taking as inputs the past increments of the common noise $\Delta W_0^0, \ldots, \Delta W_{i-1}^0$, where $\Delta W_i^0 = W_{t_{i+1}}^0 - W_{t_i}^0$. The neural network for the auxiliary control β at each time t_i takes as inputs the current state $Z_i^{z,\alpha}$ and the current value of the common noise. An illustration is given in Section 8.5.3.

8.5.2 Mean-variance problem with state constraints

We consider the celebrated Markowitz portfolio selection problem where an investor can invest at any time t an amount α_t in a risky asset (assumed for simplicity to follow a Black-Scholes model with constant rate of return r and volatility $\sigma > 0$), hence generating a wealth process $X = X^\alpha$ with dynamics

$$\mathrm{d}X_t = \alpha_t r \, \mathrm{d}t + \alpha_t \sigma \, \mathrm{d}W_t, \quad 0 \leq t \leq T, \; X_0 = x_0 \in \mathbb{R}\,.$$

The goal is then to minimize over portfolio control α the mean-variance criterion :

$$\inf_\alpha \; J(\alpha) = \lambda \mathrm{Var}(X_T^\alpha) - \mathbb{E}[X_T^\alpha] \tag{8.5.1}$$

where $\lambda > 0$ is a parameter related to the risk aversion of the investor. We will add to this standard problem a conditional expectation constraint in the form

$$E[X_t^\alpha \mid X_t^\alpha \leq \theta] \geq \delta, \text{ if } \mathbb{P}(X_t^\alpha \leq \theta) > 0,$$

Algorithm 11: Algorithm to solve mean-field control with probabilistic constraints

For a discretization $z_1 < \cdots < z_M$ of K, minimize over neural networks $(\alpha_i)_{i\in 0,\cdots,N_T-1}$: $\mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^q$ the loss function

$$\sum_{m=1}^{M} w_\Lambda(z_m)$$

with w_Λ defined by

$$w_\Lambda(z) := \mathbb{E}\Big[\{\frac{1}{N}\sum_{l=1}^{N} g\Big(X^l_{N_T}, \frac{1}{N}\sum_{j=1}^{N}\delta_{X^j_{N_T}}\Big) - Z^{z,\alpha}_{N_T}\}_+ + \Lambda \sum_{m=1}^{k}\sum_{i=1}^{N_T}\{\Psi^m\Big(t_i, \frac{1}{N}\sum_{j=1}^{N}\delta_{X^j_i}\Big)\}_+$$

$$+ \Lambda\{\varphi\Big(\frac{1}{N}\sum_{j=1}^{N}\delta_{X^j_{N_T}}\Big)\}_+\Big].$$

`/* Auxiliary problem */`

and for $i = 0, \cdots, N_T - 1$, $j = 1, \cdots, N$

$$X^j_{i+1} = X^j_i + b\big(t_i, X^j_i, \alpha_i(X^j_i, Z^{z,\alpha}_i), \overline{\mu}_i\big)\Delta t + \sigma\big(t_i, X^j_i, \alpha_i(X^j_i, Z^{z,\alpha}_i), \overline{\mu}_i\big)\Delta W^j_i, \quad X^j_0 \sim \mu_0$$

$$Z^{z,\alpha}_{i+1} = Z^{z,\alpha}_i - \frac{1}{N}\sum_{l=1}^{N} f\big(t_i, X^l_i, \alpha_i(X^l_i, Z^{z,\alpha}_i), \overline{\mu}_i\big)\ \Delta t, \quad Z^{z,\alpha}_0 = z$$

$$\overline{\mu}_i = \frac{1}{N}\sum_{j=1}^{N}\delta_{(X^j_i, \alpha_i(X^j_i, Z^{z,\alpha}_i))}$$

`/* Particle approximations */`

Define α^* as the solution to this minimization problem.
Then, compute $V_0 = \inf\{z_i,\ i \in [\![1, M]\!] \mid w_\Lambda(z_i) \le \varepsilon\}$ with $\alpha = \alpha^*$ in the dynamics.

`/* Recovering the cost of the original problem */`

Return the value V_0 and the optimal controls $\hat{\alpha}_i : x \mapsto \alpha^*_i(x, Z^{V_0,\alpha^*}_i)$ for $i = 0, \cdots, N_T - 1$.

`/* Recovering the control of the original problem */`

Algorithm 12: Algorithm to solve mean-field control with probabilistic constraints and common noise

For a discretization $z_1 < \cdots < z_M$ of K, minimize over neural networks $(\alpha_i)_{i\in 0,\cdots,N_T-1}$: $\mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^p \to \mathbb{R}^q$ and $(\beta_i)_{i\in 0,\cdots,N_T-1}$: $\mathbb{R}\times\mathbb{R}^p \to \mathbb{R}^p$ the loss function

$$\sum_{m=1}^{M} w_\Lambda(z_m)$$

with w_Λ defined by

$$w_\Lambda(z) := \mathbb{E}\Big[\{\frac{1}{N}\sum_{l=1}^{N} g\Big(X^l_{N_T}, \frac{1}{N}\sum_{j=1}^{N}\delta_{X^j_{N_T}}\Big) - Z^{z,\alpha,\beta}_{N_T}\}_+ + \Lambda\sum_{m=1}^{k}\sum_{i=1}^{N_T}\{\Psi^m\Big(t_i, \frac{1}{N}\sum_{j=1}^{N}\delta_{X^j_i}\Big)\}_+ \Delta t$$

$$+ \Lambda\{\varphi\Big(\frac{1}{N}\sum_{j=1}^{N}\delta_{X^j_{N_T}}\Big)\}_+\Big].$$

`/* Auxiliary problem */`

and for $i = 0,\cdots,N_T-1$, $j = 1,\cdots,N$

$$X^j_{i+1} = X^j_i + b\big(t_i, X^j_i, \alpha_i(X^j_i, Z^{z,\alpha,\beta}_i, W^0_{t_i}), \overline{\mu}_i\big)\Delta t + \sigma\big(t_i, X^j_i, \alpha_i(X^j_i, Z^{z,\alpha,\beta}_i, W^0_{t_i}), \overline{\mu}_i\big)\Delta W^j_i$$

$$+ \sigma_0\big(t_i, X^j_i, \alpha_i(X^j_i, Z^{z,\alpha,\beta}_i, W^0_{t_i}), \overline{\mu}_i\big)\Delta W^0_i, \quad X^j_0 \sim \mu_0$$

$$Z^{z,\alpha,\beta}_{i+1} = Z^{z,\alpha,\beta}_i - \frac{1}{N}\sum_{l=1}^{N} f\big(t_i, X^l_i, \alpha_i(X^l_i, Z^{z,\alpha,\beta}_i, W^0_{t_i}), \overline{\mu}_i\big)\ \Delta t + \beta_i(Z^{z,\alpha,\beta}_i, W^0_{t_i})\ \Delta W^0_i, \quad Z^{z,\alpha,\beta}_0 = z$$

$$\overline{\mu}_i = \frac{1}{N}\sum_{j=1}^{N}\delta_{(X^j_i, \alpha_i(X^j_i, Z^{z,\alpha,\beta}_i, W^0_{t_i}))}$$

`/* Particle approximations */`

Define (α^*, β^*) as the solution to this minimization problem.

Then, compute $V_0 = \inf\{z_i,\ i \in [\![1, M]\!] \mid w_\Lambda(z_i) \le \varepsilon\}$ with $\alpha = \alpha^*$ and $\beta = \beta^*$ in the dynamics.

`/* Recovering the cost of the original problem */`

Return the value V_0 and the optimal controls $\hat{\alpha}_i : x \mapsto \alpha^*_i(x, Z^{V_0,\alpha^*,\beta^*}_i, W^0_{t_i})$ for $i = 0,\cdots,N_T-1$.

`/* Recovering the control of the original problem */`

with $\delta < \theta$, which can be reformulated as

$$0 \geq (\delta - E[X_t^\alpha \mid X_t^\alpha \leq \theta])\mathbb{P}(X_t^\alpha \leq \theta).$$

The auxiliary deterministic unconstrained control problem is therefore

$$\mathcal{Y}_\Lambda(z) := \inf_{\alpha \in \mathcal{A}} \left[\{\lambda \mathrm{Var}(X_T^\alpha) - \mathbb{E}[X_T^\alpha] - z\}_+ + \Lambda \int_0^T \{(\delta - E[X_s^\alpha \mid X_s^\alpha \leq \theta])\mathbb{P}(X_s^\alpha \leq \theta)\}_+ \, \mathrm{d}s \right]$$

with the dynamics $\mathrm{d}X_s^\alpha = \alpha_s r \, \mathrm{d}s + \alpha_s \sigma \, \mathrm{d}W_s$, which corresponds to the constraint function $\Psi(t,\mu) \mapsto (\delta - E_\mu[\xi \mid \xi \leq \theta])\mu((-\infty, \theta])$. We have the representation $J(\alpha^*) = \mathcal{Z} = \inf\{z \in \mathbb{R} \mid \mathcal{Y}_\Lambda(z) = 0\}$. Indeed we see that the null control is admissible with the modified constraint $E[X_t^\alpha \mid X_t^\alpha \leq \theta]\mathbb{P}(X_t^\alpha \leq \theta) = 0 \geq (\delta + \varepsilon)\mathbb{P}(X_t^\alpha \leq \theta) = 0, \ \forall t \in [0,T]$ for any $0 < \varepsilon < \theta - \delta$ because $x_0 \geq \theta$ hence $\mathbb{P}(X_t^\alpha \leq \theta) = 0$ so we can apply Theorem 8.2.2. For practical application, other constraints could be considered like almost sure constraints on the portfolio weights as in [War21a]. Instead of the dualization method used by [LLP20], constraints on the law of the tracking error with respect to a reference portfolio could be enforced.

For numerical tests we take $r = 0.15, \ \sigma = 0.35, \ \lambda = 1$. We choose $x_0 = 1, \theta = 0.9, \ \delta = 0.8$ and solve

$$\begin{aligned} \inf_\alpha \ & J(\alpha) = \lambda \mathrm{Var}(X_T^\alpha) - \mathbb{E}[X_T^\alpha] \\ & \mathrm{d}X_t = \ \alpha_t r \, \mathrm{d}t + \alpha_t \sigma \, \mathrm{d}W_t, \\ & (0.8 - E[X_t^\alpha \mid X_t^\alpha \leq 0.9])\mathbb{P}(X_t^\alpha \leq 0.9) \leq 0, \ \forall t \in [0,T]. \end{aligned} \tag{8.5.2}$$

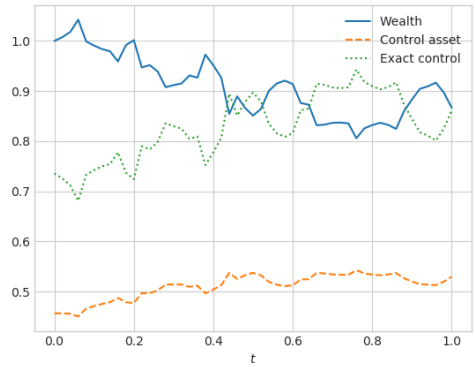
We compare the controls from Algorithm 11 with the exact optimal ones in the unconstrained case for which we have an analytical value. We also solve without constraints for comparison and plot the final time histograms. We solve the unconstrained case with algorithm 11 and the one from [CL22] for comparison. We take 50 time steps for the time discretization and a batch size of 20000. We use an feedforward architecture with two hidden layers of 15 neurons. We perform 15000 gradient descent iterations thanks to the Tensorflow library. The true value $v = J(\alpha^*)$ is -1.05041 without constraints. We also have the upper bound $-1.$ for the value in the constrained case, corresponding to the identically null control and wealth process $X_t = 1 \ \forall t \in [0,T]$. With constraint we choose $K = [-1.047, -1.041]$, without constraint we take $K = [-1.07, -1.03]$, discretized by regular grids with 25 points.

In Figure 8.2 we observe the shift of the distribution of the final wealth thanks to the constraint (on the left) with less probable large losses but also less probable large gains. Indeed Figure 8.4 confirms that the conditional expectation constraint is verified when we solve the corresponding problem through our level set approach. We see in Figure 8.3 that the more Λ is large the more the auxiliary value function becomes affine before reaching zero. Additional results are presented in Table 8.1.

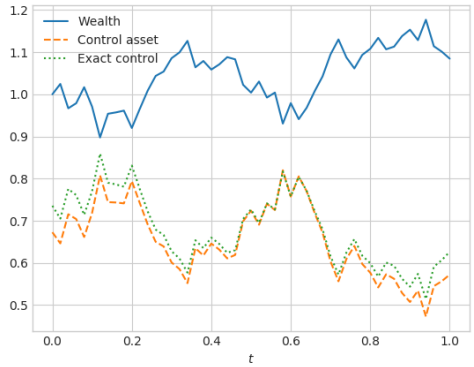
Our method can also handle directly the primal of the mean-variance problem, that is to maximize over portfolio control α the expected terminal wealth under a terminal variance constraint:

$$\begin{aligned} \inf_\alpha \ & \bar{J}(\alpha) = -\mathbb{E}[X_T^\alpha] \\ & \mathrm{d}X_t = \ \alpha_t r \, \mathrm{d}t + \alpha_t \sigma \, \mathrm{d}W_t, \\ & \mathrm{Var}(X_T^\alpha) \leq \vartheta. \end{aligned} \tag{8.5.3}$$

which give the same optimal control as Problem (8.5.1) under the correspondence $\lambda = \sqrt{\frac{\exp(\sigma^{-2} r^2 T) - 1}{4\vartheta}}$. This problem allows us to consider a constrained problem with an analytical solution. In this

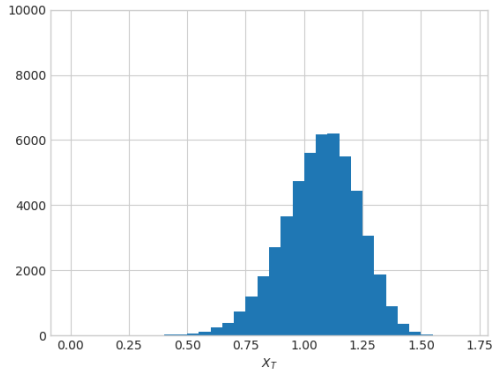


Problem (8.5.2)

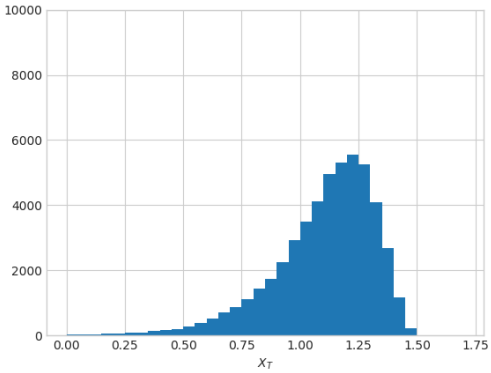


No constraint, problem (8.5.1)

Figure 8.1: Sample path of the controlled process X_t^α, with the analytical optimal control (for the unconstrained case) and the computed control. On the left figure we don't have the true control but plot the unconstrained one for comparison. Here $\Lambda = 100$

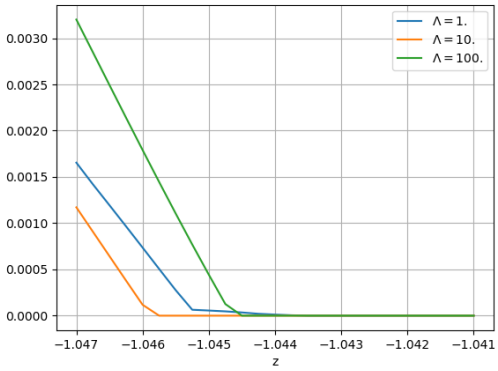


Problem (8.5.2)

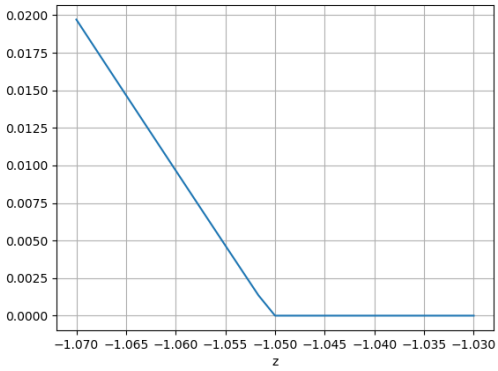


No constraint, problem (8.5.1)

Figure 8.2: Histogram of X_T^α for 50000 samples. Here $\Lambda = 100$.

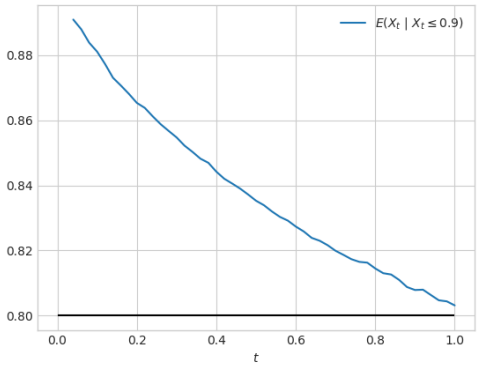


Problem (8.5.2)

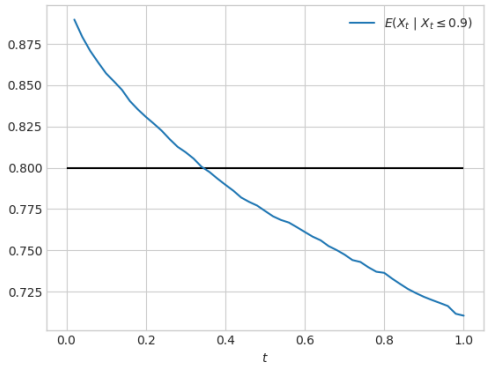


No constraint, problem (8.5.1)

Figure 8.3: Auxiliary value function $\mathcal{Y}_\Lambda(z)$ for several values of Λ in the constrained case, auxiliary value function $\mathcal{Y}(z)$ in the unconstrained case

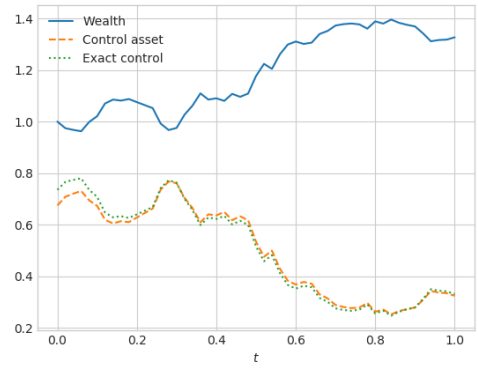


Problem (8.5.2)



No constraint, problem (8.5.1)

Figure 8.4: Conditional expectation $E[X_t^\alpha \mid X_t^\alpha \leq 0.9]$ estimated with 50000 samples. The black line corresponds to $\delta = 0.8$. Here $\Lambda = 100$



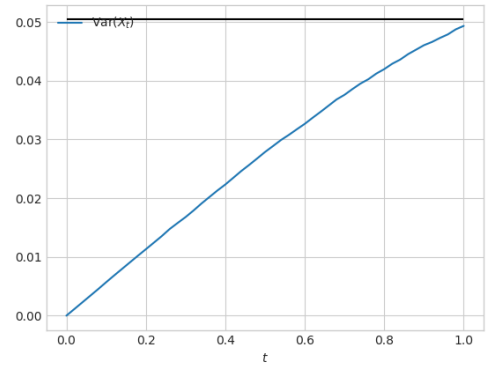


Figure 8.5: Sample trajectory of the controlled process X_t^α and the control for problem (8.5.3) (left). Variance $\mathrm{Var}(X_t)$ estimated with 50000 samples for problem (8.5.3) (right) with $\Lambda = 10$

case $\mathrm{Var}(X_T^{\alpha^*}) = \vartheta$ thus $J(\alpha^*) = \lambda \mathrm{Var}(X_T^{\alpha^*}) + \bar{J}(\alpha^*) = \lambda\vartheta + \bar{J}(\alpha^*)$. See Remark 2.5 from [DFNP19]. For comparison with Problem (8.5.1) we thus report $\lambda\vartheta + \bar{J}(\alpha^*)$ for Problem (8.5.3) in Table 8.1 and choose $\vartheta = \frac{\exp(\sigma^{-2}r^2T)-1}{4\lambda^2} = 0.0504$. In this case the auxiliary deterministic unconstrained control problem is now

$$\mathcal{U}_\Lambda(z) = \inf_{\alpha \in \mathcal{A}} \Big[\{-\mathbb{E}[X_T^\alpha] - z\}_+ + \Lambda \{\mathrm{Var}(X_T^\alpha) - \vartheta\}_+ \Big]$$
$$\mathrm{d}X_t = \alpha_t r \, \mathrm{d}t + \alpha_t \sigma \, \mathrm{d}W_t,$$

which corresponds to the constraint function $\Psi(t,\mu) \mapsto (\mathrm{Var}(\mu) - \vartheta)_+ \mathbb{1}_{t=T}$ and the modified constraint function $\overline{\Psi}_\eta(t,\mu) \mapsto (\mathrm{Var}(\mu) - \vartheta)_+ \mathbb{1}_{t=T} - \eta \mathbb{1}_{t<T}$ (see Remark 8.2.3). Theorem 8.2.2 still applies as far as the null control is admissible with the modified constraint $(\mathrm{Var}(\mu) - \vartheta)_+ \mathbb{1}_{t=T} + \varepsilon - \eta \mathbb{1}_{t<T} \leq 0$ for any $0 < \varepsilon < \eta$ and any $t \in [0,T]$.

Figure 8.5 shows that we recover the optimal control for the problem and that the terminal variance constraint is satisfied. We see in Figure 8.6 that similarly as in Figure 8.3, for large values of Λ the auxiliary value function is affine before reaching zero. In this case the exact solution is -1.10 which is very close to the point in which the affine part reaches zero.

In Table 8.1 we observe that our method gives a small variance for the results over several runs. In the case where an analytical solution is known, the value of the control problem is computed accurately with less than 0.5% of relative error. The expectation and variance of the

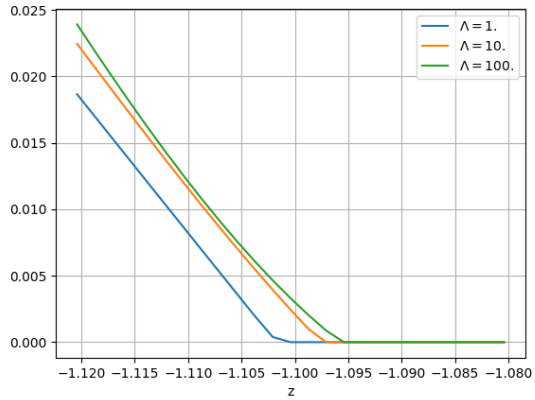


Figure 8.6: Auxiliary value function $\mathcal{U}_\Lambda(z)$ for several values of Λ

Problem	Average	Std	True val.	Error	$\mathbb{E}[X_T^{\alpha^*}]$	True $\mathbb{E}[X_T^{\alpha^*}]$	$\mathrm{Var}(X_T^{\alpha^*})$	True $\mathrm{Var}(X_T^{\alpha^*})$
(8.5.2)	-1.045	0.0005	?	?	1.07	?	0.027	?
(8.5.3)	-1.048	0.0017	-1.050	0.22	1.10	1.10	0.049	0.050
(8.5.1)	-1.050	0.0009	-1.050	0.07	1.10	1.10	0.050	0.050
(8.5.1) [CL22]	-1.052	0.0022	-1.050	0.13	1.10	1.10	0.053	0.050

Table 8.1: Estimate of the solution with maturity $T = 1$. Average and standard deviation observed over 10 independent runs are reported, with the relative error (in %). We also report the terminal expectation and variance of the approximated optimally controlled process for a single run. '?' means that we don't have a reference value. For problem (8.5.1) we take $\Lambda = 10$ and for problem (8.5.2) we choose $\Lambda = 100$

terminal value of the optimally controlled process are also very close to their theoretical values. In the case of a conditional expectation constraint, even though we don't have an exact solution we notice that the value is close to the unconstrained value hence since our solution is admissible, we expect to be near optimality. On the unconstrained problem (8.5.1) our scheme and the one from [CL22] give similar results.

8.5.3 Optimal storage of wind-generated electricity

We consider N wind turbines with N associated batteries. Define the productions P_t^i, storage levels X_t^i, storage injection α_t^i for which we provide a typical range[1]. We consider the following constraints

$$\begin{cases} 0 \le X_t^i \le X_{\max} \longrightarrow \text{limited storage capacity (1kWh} - \text{10 MWh)} \\ \underline{\alpha} \le \alpha_t^i \le \overline{\alpha} \longrightarrow \text{limited injection/withdrawal capacity (10 kW} - \text{10MW)} \end{cases}$$

with $X_{\max} \ge 0$, $\underline{\alpha} \le 0 \le \overline{\alpha}$. Define the spot price of electricity S_t without wind power, $\widetilde{S}_t$ the price with wind production. Selling a quantity $P_t^i - \alpha_t^i$ on the market, producer i obtains a revenue $\widetilde{S}_t(P_t^i - \alpha_t^i)$ (if $P_t^i - \alpha_t^i < 0$ the producer is buying from the market) where the market price is affected by linear price impact

$$\widetilde{S}_t = S_t - \frac{\Theta(N)}{N} \sum_{i=1}^{N} (P_t^i - \alpha_t^i),$$

modeling the impact of intermittent renewable production on the market. Θ is positive, non-decreasing and bounded. We call $\Theta_\infty = \lim_{N\to\infty} \Theta(N) < \infty$. We consider $N + 2$ indepen-

[1] https://css.umich.edu/factsheets/us-grid-energy-storage-factsheet

dent Brownian motions $W_t^0, B_t^0, W_t^1, \cdots, W_t^N$ and the following dynamics for the producers $i = 1, \cdots, N$ state processes

$$\begin{cases} \mathrm{d}X_t^i = \alpha_t^i \, \mathrm{d}t \\ \mathrm{d}P_t^i = \iota(\phi P_{\max} - P_t^i) \, \mathrm{d}t + \sigma_p(P_t \wedge \{P_{\max} - P_t^i\})_+(\rho \, \mathrm{d}W_t^0 + \sqrt{1-\rho^2} \, \mathrm{d}W_t^i) \\ \mathrm{d}F(t,T) = F(t,T)\sigma_f e^{-a(T-t)}\mathrm{d}B_t^0 \\ S_t = F(t,t). \end{cases}$$

In the production dynamics, the common noises W_t^0, B_t^0 corresponds to the global weather and the market price randomness whereas the idiosyncratic noises W_t^i for $i > 1$ model the local weather, independent from one wind turbine to another. We call $\mathbb{F}^0$ the filtration generated by W_t^0, B_t^0. The productions P_t^i are bounded processes and the price S_t is positive. Of course the modified price $\widetilde{S}_t$ in the presence of renewable producers can become negative, as empirically observed in some overproduction events. However it stays bounded by below in our model. Producer i gain function to maximize is

$$J^i(\alpha_1, \cdots, \alpha_N) = \mathbb{E}\Big[\int_0^T \{S_t(P_t^i - \alpha_t^i) - \frac{\Theta(N)}{N}(P_t^i - \alpha_t^i) \sum_{j=1}^N (P_t^j - \alpha_t^j)\} \, \mathrm{d}t \Big].$$

The related mean field control problem for a central planner is therefore

$$\begin{aligned} - \inf_{\alpha \in \mathcal{A}} \mathbb{E}\Big[\int_0^T \{-S_t(P_t - \alpha_t) + \Theta_\infty(P_t - \alpha_t)\mathbb{E}[P_t - \alpha_t | \mathbb{F}^0]\} \, \mathrm{d}t \Big] \\ \mathrm{d}X_t = \alpha_t \, \mathrm{d}t \\ \mathrm{d}P_t = \iota(\phi P_{\max} - P_t) \, \mathrm{d}t + \sigma_p(P_t \wedge \{P_{\max} - P_t\})_+(\rho \, \mathrm{d}W_t^0 + \sqrt{1-\rho^2} \, \mathrm{d}W_t^1) \\ \mathrm{d}F(t,T) = F(t,T)\sigma_f e^{-a(T-t)}\mathrm{d}B_t^0 \\ S_t = F(t,t) \\ 0 \le X_t \le X_{\max} \ \mathbb{P} \text{ a.s.} \end{aligned}$$

Here the state is $(X_t, P_t, S_t) \in \mathbb{R}^3$ hence the distribution of the state lives in $\mathcal{P}_2(\mathbb{R}^3)$. The set $\mathcal{A}$ corresponds to progressively measurable controls with values in the compact set $[\underline{\alpha}, \overline{\alpha}]$. A similar problem is solved by [ABTM20] without any storage constraints by Pontryagin principle. With constraints but without mean-field interaction, a close problem is solved by [PV20]. For instance $X_{\max} = 0$ corresponds to the much simpler problem without storage nor control of the valuation of a wind power park. See also [Cur+21; War21b]. To represent the almost sure constraint $0 \le X_t \le X_{\max}$ we choose as constrained function

$$\Psi : \mu \in \mathcal{P}_2(\mathbb{R}^3) \mapsto \int_{\mathbb{R}} \{(-x)_+^2 + (x - X_{\max})_+^2\} \, \mu_1(\mathrm{d}x),$$

where μ_1 is the first marginal law of the measure μ.

The auxiliary unconstrained control problem is therefore

$$\begin{aligned} w(z) := - \inf_{\alpha, \beta^{0,1}, \beta^{0,2} \in \mathcal{A} \times L^2 \times L^2} \mathbb{E}\Big[\{ \int_0^T \mathbb{E}[-S_t(P_t - \alpha_t) + \Theta_\infty(P_t - \alpha_t)\mathbb{E}[P_t - \alpha_t | \mathbb{F}^0] | \mathbb{F}^0] \, \mathrm{d}t - \quad (8.5.4) \\ - \int_0^T \beta_s^{0,1} \, \mathrm{d}W_s^0 - \int_0^T \beta_s^{0,2} \, \mathrm{d}B_s^0 \}_+ + \frac{1}{\epsilon} \int_0^T \mathbb{E}[(-X_s)_+^2 + (X_s - X_{\max})_+^2] \, \mathrm{d}s \Big] \end{aligned}$$

$$\begin{aligned} \mathrm{d}X_t = \alpha_t \, \mathrm{d}t \\ \mathrm{d}P_t = \iota(\phi P_{\max} - P_t) \, \mathrm{d}t + \sigma_p(P_t \wedge \{P_{\max} - P_t\})_+ \, (\rho \, \mathrm{d}W_t^0 + \sqrt{1-\rho^2} \, \mathrm{d}W_t^1) \\ \mathrm{d}F(t,T) = F(t,T)\sigma_f e^{-a(T-t)}\mathrm{d}B_t^0 \\ S_t = F(t,t) \end{aligned}$$

where ϵ is a small term used to force the a.s. constraints.

We consider the standard stochastic control benchmark with only common noise for the production ($\rho = 1$). It corresponds to a single very large wind farm where all wind turbines produce the same power. The problem degenerates as

$$
\begin{aligned}
- \inf_{\alpha \in \mathcal{A}} \mathbb{E}\Big[\int_0^T \{(-S_t + \Theta_\infty(P_t - \alpha_t))(P - \alpha_t)\} \, \mathrm{d}t \Big] \\
\mathrm{d}X_t &= \alpha_t \, \mathrm{d}t \\
\mathrm{d}P_t &= \iota(\phi P_{\max} - P_t) \, \mathrm{d}t + \sigma_p (P_t \wedge \{P_{\max} - P_t\})_+ \, \mathrm{d}W_t^0 \\
\mathrm{d}F(t,T) &= F(t,T)\sigma_f e^{-a(T-t)} \mathrm{d}B_t^0 \\
S_t &= F(t,t) \\
0 \le X_t &\le X_{\max} \ \mathbb{P} \text{ a.s.}
\end{aligned} \tag{8.5.5}
$$

and equation (8.5.4) gives

$$
\begin{aligned}
w(z) := - \inf_{\alpha,\beta \in \mathcal{A} \times L^2} \mathbb{E}\Big[((Y^{\alpha,\beta} - z)_+ + \frac{1}{\epsilon} \int_0^T \mathbb{E}[(-X_s)_+^2 + (X_s - X_{\max})_+^2] \, \mathrm{d}s \Big] \\
\mathrm{d}X_t &= \alpha_t \, \mathrm{d}t \\
\mathrm{d}P_t &= \iota(\phi P_{\max} - P_t) \, \mathrm{d}t + \sigma_p (P_t \wedge \{P_{\max} - P_t\})_+ \, \mathrm{d}W_t^0 \\
\mathrm{d}F(t,T) &= F(t,T)\sigma_f e^{-a(T-t)} \mathrm{d}B_t^0 \\
S_t &= F(t,t)
\end{aligned}
$$

where

$$
Y^{\alpha,\beta} = \int_0^T (-S_t + \Theta_\infty(P_t - \alpha_t))(P_t - \alpha_t) \, \mathrm{d}t - \int_0^T \beta_s^{0,1} \, \mathrm{d}W_s^0 - \int_0^T \beta_s^{0,2} \, \mathrm{d}B_s^0. \tag{8.5.6}
$$

The solution of our optimization problem is then $z^* = \sup\{z \mid \hat{w}(z) = 0\}$ where $\hat{w}(z) := -w(z)$. Remark now that (8.5.6) will be estimated discretizing the integral $\int_0^T \beta_s^{0,1} \, \mathrm{d}W_s^0$ and $\int_0^T \beta_s^{0,2} \, \mathrm{d}B_s^0$ using an Euler scheme for the underlying processes and therefore $\hat{w}(z)$ will be above 0 except for low values of z due to the variance of the $Y^{\alpha*,\beta}$ estimator that cannot be reduced to 0.
In order to reduce the variance of $Y^{\alpha,\beta}$, we propose to modify $Y^{\alpha,\beta}$ as follows :

$$
\begin{aligned}
Y^{\alpha,\beta} = \int_0^T (-S_t + \Theta_\infty(P_t - \alpha_t))(P_t - \alpha_t) \, \mathrm{d}t - \int_0^T (-S_t + \Theta_\infty(P_t - \hat{\alpha}_t))(P_t - \hat{\alpha}_t) \, \mathrm{d}t + \\
\mathbb{E}[\int_0^T (-S_t + \Theta_\infty(P - \hat{\alpha}_t))(P_t - \hat{\alpha}_t) \, \mathrm{d}t] - \int_0^T \beta_s^{0,1} \, \mathrm{d}W_s^0 - \int_0^T \beta_s^{0,2} \, \mathrm{d}B_s^0
\end{aligned}
$$

where $\hat{\alpha}_t$ is the rough estimation of the optimal **deterministic** command maximizing the gain. We take $T = 40$, $N_T = 40$ time steps, $X_{\max} = 1$, $X_0 = 0.5$, $P_0 = 0.12$, $F(0,t) = 30 + 5\cos(\frac{2\pi t}{N}) + \cos(\frac{2\pi t}{7})$, $\sigma_f = 0.3$, $a = 0.16$, $\iota = 0.2$, $\sigma_p = 0.2$, $\phi = 0.3$, $P_{\max} = 0.2$, $-0.2 \le \alpha \le 0.2$, $\Theta(N) = 10$. The network depends on P_t, S_t, X_t and z where z takes some deterministic values on a grid with the same spacing. The global curve is therefore approximated by a single run.
The grid is taken from 107 to 127 with a spacing of 0.5. The neural networks have two hidden layers with 14 neurons on each layer. We take a ϵ parameter equal to 10^{-4}. The number of gradient iterations is set to 50000 with a learning rate equal to 2×10^{-3} and every 100 iterations a more accurate estimate of $\hat{w}$ is calculated.
We give the $\hat{w}$ function on figure 8.7. Using Dynamic Programming with the StOpt library [Gev+18], we get an optimal value equal to 117.28 while a direct optimization of (8.5.5) using some neural networks as in [War21b], [CL22] we get a value of 117.11. Encouraged by Remark 8.5.1, Figure 8.3, Figure 8.6 and the related comments, we empirically estimate the value function by the point where the linear part of the auxiliary function reaches zero when $\Lambda = \frac{1}{\varepsilon}$ is

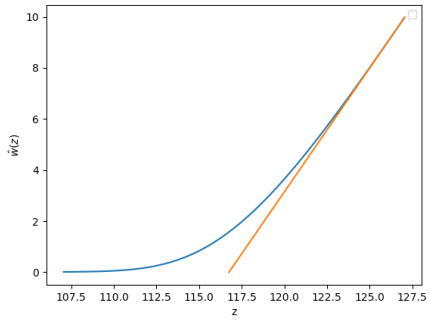


Figure 8.7: $\hat{w}$ function value for the storage problem

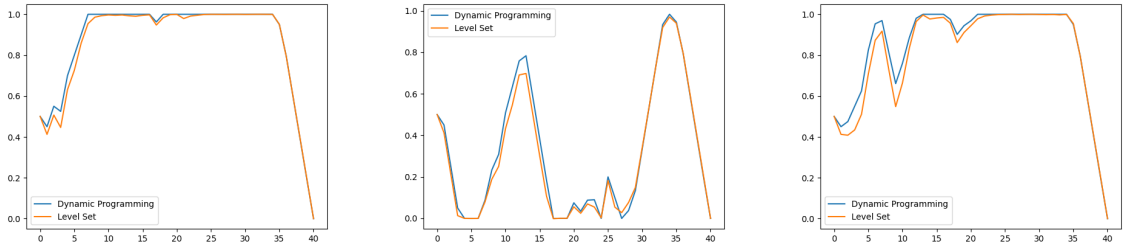


Figure 8.8: Storage trajectories with the Level Set and Dynamic Programming method.

sufficiently large. The estimated value is 116.75, close to the reference solutions. On figure 8.8, we compare trajectories obtained by Dynamic Programming and by the Level Set approach : they are accurately calculated.

Conclusion

We have developed several schemes in order to solve nonlinear PDEs in moderate and high dimension, thanks to neural networks. We also considered the more difficult case of mean-field control. Our methods show a good empirical performance and some first results have been obtained on the theoretical analysis side.

We have notably described the approximation error of the multistep and deep backward dynamic programming schemes in terms of the number of layers and neurons needed to reach an error of size $O(1/N)$, that is the discretization error. However this study is achieved thanks to the GroupSort activation function. One should try to obtain similar results for Relu or tanh neural networks in order to comply with the architectures used in practice in our tests. The main difficulty in this study is to consider regular enough approximations (Sobolev regularity, convexity, or boundedness) to have quantitative approximation results. The regularity then has to be preserved when the time step vanishes which motivated the introduction of GroupSort neural networks in our study. Future work could also be dedicated to the fully nonlinear setting for which we have not been able to perform a theoretical analysis of our methods.

Concerning the numerical approximation of PDEs on the Wasserstein space, we obtained the convergence speed of a finite dimensional PDE approximation. The resulting scheme with symmetric neural networks is able to solve the PDE along one trajectory of the forward process. That is why we performed some first tests to randomize the training samples and explore the space of probability measures. But further research is needed to fully understand how to proceed in order to efficiently perform the exploration and solve the PDE in the whole domain. Moreover, we require a lot of regularity for the target solution, which is not satisfied in most cases. Hence it would be interesting to look for an alternative result in a more realistic setting.

We also introduced probabilistic state constraints for mean-field control thanks to a level-set approach making use of an auxiliary unconstrained problem. Our method allows us to consider an exact penalization and is suitable for a numerical implementation thanks to machine learning. To the best of our knowledge we are the first authors to consider this problem with general constraints. Future research could introduce Pontryagin principle or HJB equations based algorithms, following the (theoretical) works [Bon19; Dau21]. The numerical resolution of the Master Bellman equation arising from the unconstrained auxiliary problem is another possibility. A comparison of these methods with our level-set approach would be very interesting.

Bibliography

[AA+18] A. Al-Aradi et al. “Solving Nonlinear and High-Dimensional Partial Differential Equations via Deep Learning”. In: *arXiv:1811.08782* (2018).

[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.

[ABTM20] C. Alasseur, I. Ben Taher, and A. Matoussi. “An Extended Mean Field Game for Storage in Smart Grids”. In: *Journal of Optimization Theory and Applications* 184 (2020), 644–670.

[ABZ13] A. Altarovici, O. Bokanowski, and H. Zidani. “A general Hamilton-Jacobi framework for non-linear state-constrained control problems”. In: *ESAIM: COCV* 19.2 (2013), pp. 337–357.

[ACD10] Y. Achdou and I. Capuzzo-Dolcetta. “Mean Field Games: Numerical Methods”. In: *SIAM Journal on Numerical Analysis* 48 (Jan. 2010). DOI: 10.1137/090758477.

[AFL20] A. Angiuli, J.-P. Fouque, and M. Laurière. “Unified Reinforcement Q-Learning for Mean Field Game and Control Problems”. In: *arXiv:2006.13912* (2020).

[AKS19] B. Anahtarcı, C. Deha Karıksız, and N. Saldi. “Fitted Q-Learning in Mean-field Games”. In: *arXiv:1912.13309* (2019).

[AL15] Y. Achdou and M. Laurière. “On the system of partial differential equations arising in mean field type control”. In: *Discrete Contin. Dyn. Syst.* 35.9 (2015), pp. 3879–3900.

[ALG19] C. Anil, J. Lucas, and R. Grosse. “Sorting Out Lipschitz Function Approximation”. In: *Proceedings of the 36th ICML*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. 2019, pp. 291–301.

[AM21] S. Sadeghi Arjmand and G. Mazanti. “Nonsmooth mean field games with state constraints”. In: *arXiv:2110.15713* (2021).

[Bac17] F. Bach. “Breaking the Curse of Dimensionality with Convex Neural Networks”. In: *Journal of Machine Learning Research* 18.19 (2017), pp. 1–53.

[Bac+21] A. Bachouch et al. “Deep neural networks algorithms for stochastic control problems on finite horizon: numerical computations”. In: *Methodol. Comput. Appl. Probab* (2021).

[Bal+21] A. Balata et al. “Statistical learning for probability-constrained stochastic optimal control”. In: *European Journal of Operational Research* 290.2 (2021), pp. 640–656.

[BCJ19] S. Becker, P. Cheridito, and A. Jentzen. “Deep optimal stopping”. In: *J. Mach. Learn. Res.* 20 (2019), pp. 1–25.

[BD07] C. Bender and R. Denk. “A forward scheme for backward SDEs”. In: *Stochastic Processes and their Applications* 117.12 (2007), pp. 1793 –1812.

[BDK20] B. Bouchard, B. Djehiche, and I. Kharroubi. "Quenched Mass Transport of Particles Toward a Target". In: *Journal of Optimization Theory and Applications* 186.2 (2020), pp. 345–374. DOI: 10.1007/s10957-020-01704-.

[Bec+19] S. Becker et al. "Solving high-dimensional optimal stopping problems using deep learning". In: *arXiv:1908.01602* (2019).

[Bec+20] C. Beck et al. "An overview on deep learning-based approximation methods for partial differential equations". In: *arXiv:2012.12348* (2020).

[Bec+21] C. Beck et al. "Deep splitting method for parabolic PDEs". In: *SIAM Journal on Scientific Computing* 43.5 (2021).

[BEI10] B. Bouchard, R. Elie, and C. Imbert. "Optimal Control under Stochastic Target Constraints". In: *SIAM Journal on Control and Optimization* 48.5 (2010), pp. 3501–3531.

[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.

[Ber+21] C. Bertucci et al. "Economic Modelling of the Bitcoin Mining Industry". In: *SSRN: https://ssrn.com/abstract=3907822* (2021).

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

[BF21] B. Bonnet and H. Frankowska. "Necessary Optimality Conditions for Optimal Control Problems in Wasserstein Spaces". In: *Appl Math Optim* (2021).

[BFY13] A. Bensoussan, J. Frehse, and P. Yam. *Mean field games and mean field type control theory*. Springer Briefs in Mathematics. Springer, 2013.

[BFY15] A. Bensoussan, J. Frehse, and P. Yam. "The Master equation in mean-field theory". In: *J. de Math. Pures et Appliquées* 103.6 (2015), pp. 1441–1474.

[BGS15] G. Balazs, A. György, and C. Szepesvari. "Near-optimal max-affine estimators for convex regression". In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Guy Lebanon and S. V. N. Vishwanathan. Vol. 38. 2015, pp. 56–64.

[BGZ21] O. Bokanowski, N. Gammoudi, and H. Zidani. "Optimistic Planning Algorithms For State-Constrained Optimal Control Problems". preprint. July 2021.

[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.

[Bon19] B. Bonnet. "A Pontryagin Maximum Principle in Wasserstein spaces for constrained optimal control problems". In: *ESAIM: COCV* 25 (2019), p. 52.

[Bou+17] B. Bouchard et al. "Numerical approximation of BSDEs using local polynomial drivers and branching processes". In: *Monte Carlo Methods and Applications* 23.4 (2017), pp. 241–263.

[BP19] M. Basei and H. Pham. "A weak martingale approach to linear-quadratic McKean-Vlasov stochastic control problem". In: *Journal of Optimization Theory and Applications* 181.2 (2019), pp. 347–382.

[BPZ15] O. Bokanowski, A. Picarelli, and H. Zidani. "Dynamic Programming and Error Estimates for Stochastic Control Problems with Maximum Cost". In: *Appl Math Optim* 71 (2015), pp. 125–163.

[BPZ16] O. Bokanowski, A. Picarelli, and H. Zidani. "State-Constrained Stochastic Optimal Control Problems via Reachability Approach". In: *SIAM Journal on Control and Optimization* 54.5 (2016), pp. 2568–2593.

[BRT20] B. Bloem-Reddy and Y.W. Teh. "Probabilistic symmetries and invariant neural networks". In: *Journal of Machine Learning Research* 21 (2020), pp. 1–61.

[BS91] G. Barles and P.E. Souganidis. "Convergence of approximation schemes for fully nonlinear second order equations". In: *Asymptotic Analysis* 4.3 (1991), pp. 271–283.

[BT04] B. Bouchard and N. Touzi. "Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations". In: *Stoch. Process. Appl.* 111.2 (2004), pp. 175 –206.

[BTB16] D. Bauso, H. Tembine, and T. Başar. "Opinion Dynamics in Social Networks through Mean-Field Games". In: *SIAM Journal on Control and Optimization* 54.6 (2016), pp. 3225–3257.

[Buc+17] R. Buckdahn et al. "Mean-field stochastic differential equations and associated PDEs". In: *Ann. Probab.* 45.2 (Mar. 2017), pp. 824–878.

[Bur+20] M. Burzoni et al. "Viscosity solutions for controlled McKean-Vlasov jump-diffusions". In: *SIAM Journal on Control and Optimization* 58.3 (2020), pp. 1676–1699.

[Car+19] P. Cardaliaguet et al. *The Master equation and the convergence problem in mean-field games*. Vol. 201. Annals of mathematics studies. Princeton University Press, 2019.

[CC18] P. Cannarsa and R. Capuani. "Existence and Uniqueness for Mean Field Games with State Constraints". In: *PDE Models for Multi-Agent Phenomena*. Ed. by Cardaliaguet P., Porretta A., and Salvarani F. Vol. 28. Springer INdAM Series. Springer, Cham, 2018.

[CCC18] P. Cannarsa, R. Capuani, and P. Cardaliaguet. "Mean Field Games with state constraints: from mild to pointwise solutions of the PDE system". In: *arXiv:1812.11374* (2018).

[CCD15] J.-F. Chassagneux, D. Crisan, and F. Delarue. "A probabilistic approach to classical solutions of the master equation for large population equilibria". In: *to appear in Memoirs of the AMS* (2015).

[CD13] R. Carmona and F. Delarue. "Probabilistic Analysis of Mean-Field Games". In: *SIAM Journal on Control and Optimization* 51.4 (2013), pp. 2705–2734. DOI: 10.1137/120883499. URL: https://doi.org/10.1137/120883499.

[CD15] R. Carmona and F. Delarue. "Forward–backward stochastic differential equations and controlled McKean–Vlasov dynamics". In: *The Annals of Probability* 43.5 (2015), pp. 2647 –2700.

[CD18a] R. Carmona and F. Delarue. *Probabilistic Theory of Mean Field Games: vol. I, Mean Field FBSDEs, Control, and Games,* Springer, 2018.

[CD18b] R. Carmona and F. Delarue. *Probabilistic Theory of Mean Field Games: vol. II, Mean Field FBSDEs, Control, and Games,* Springer, 2018.

[CD18c] R. Carmona and F. Delarue. *Probabilistic Theory of Mean Field Games with Applications vol I. and II.* Vol. 83. Probability Theory and Stochastic Modelling. Springer, 2018.

[CDL13] R. Carmona, F. Delarue, and A. Lachapelle. "Control of McKean–Vlasov dynamics versus mean field games". In: *Mathematics and Financial Economics* 7 (2013), 131–166.

[CFS15] R. Carmona, J.P. Fouque, and L. Sun. "Mean field games and systemic risk". In: *Commun. Math. Sci.* 13.4 (2015), pp. 911–933.

[Cha+21] J.-F. Chassagneux et al. "A learning scheme by sparse grids and Picard approximations for semilinear parabolic PDEs". In: *arXiv:2102.12051* (2021).

[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.

[CL18] P. Cardaliaguet and C.-A. Lehalle. "Mean field game of controls and an application to trade crowding". In: *Mathematics and Financial Economics* 12.3 (2018), pp. 335–363. DOI: https://doi.org/10.1007/s11579-017-0206-z.

[CL22] R. Carmona and M. Laurière. "Convergence analysis of machine learning algorithms for the numerical solution of mean-field control and games: II The finite horizon case". In: *to appear in the Annals of Applied Probability* (2022).

[CLT19] R. Carmona, M. Laurière, and Z. Tan. "Model-Free Mean-Field Reinforcement Learning: Mean-Field MDP and Mean-Field Q-Learning". In: *arXiv:1910.12802* (2019).

[Cos+20] A. Cosso et al. "Optimal control of path-dependent McKean-Vlasov SDEs in infinite dimension". In: *arXiv:2012.14772* (2020).

[CP19] A. Cosso and H. Pham. "Zero-sum stochastic differential games of generalized McKean–Vlasov type". In: *Journal de Mathématiques Pures et Appliquées* 129 (2019), pp. 180 –212.

[CS17] P. Chan and R. Sircar. "Fracking, Renewables, and Mean Field Games". In: *SIAM Review* 59.3 (2017), pp. 588–615.

[CUH16] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2016. URL: http://arxiv.org/abs/1511.07289.

[Cur+21] N. Curin et al. "A deep learning model for gas storage optimization". In: *arXiv:2102.01980* (2021).

[CW19] L. Chen and J. Wang. "Maximum principle for delayed stochastic mean-field control problem with state constraint". In: *Advances in Difference Equations* 348 (2019).

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

[CYZ20] Y.-L. Chow, X. Yu, and C. Zhou. "On Dynamic Programming Principle for Stochastic Control under Expectation Constraints". In: *Journal of Optimization Theory and Applications* 185 (2020), 803–818.

[Dau20] S. Daudin. "Optimal Control of Diffusion Processes with Terminal Constraint in Law". In: *arXiv:2012.10707* (2020).

[Dau21] S. Daudin. "Optimal control of the Fokker-Planck equation under state constraints in the Wasserstein space". In: *arXiv:2109.14978* (2021).

[DFNP19] C. De Franco, J. Nicolle, and H. Pham. "Bayesian learning for the Markowitz portfolio selection problem". In: *International Journal of Theoretical and Applied Finance* 22.07 (2019), p. 1950037.

[DHS11] J Duchi, E Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159. URL: http://jmlr.org/papers/v12/duchi11a.html.

[Dje20] F Djete. "Extended mean-field control problem: a propagation of chaos result". In: *arXiv:2006.12996* (2020).

[DLM20] J. Darbon, G. Langlois, and T. Meng. "Overcoming the curse of dimensionality for some Hamilton–Jacobi partial differential equations via neural network architectures". In: *Res Math Sci* 7 (2020).

[DPT19] F. Djete, D. Possamai, and X. Tan. "McKean-Vlasov optimal control: the dynamic programming principle". In: *arXiv:1907.08860* (2019).

[DPT94] M.W.M. Dissanayake and N. Phan-Thien. "Neural network-based approximations for solving partial differential equations". In: *Commun. Numer. Methods Eng.* 10.3 (1994), pp. 195–201.

[E+19] 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: *Journal of Scientific Computing* 79 (2019), 1534–1571.

[EHJ17] W. E, J. Han, and A. Jentzen. "Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations". In: *Communications in Mathematics and Statistics* 5.4 (2017), pp. 349–380.

[ET99] I. Ekeland and R. Témam. *Convex Analysis and Variational Problems*. Society for Industrial and Applied Mathematics, 1999.

[EY18] W. E and B. Yu. "The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems". In: *Commun. Math. Stat.* 6 (2018), pp. 1–12.

[FG15] N. Fournier and A. Guillin. "On the rate of convergence in the Wasserstein distance of the empirical measure". In: *Probability Theory and Related Fields* 162 (2015), pp. 707–738.

[FH20] G. Fu and U. Horst. "Mean-Field Leader-Follower Games with Terminal State Constraint". In: *SIAM Journal on Control and Optimization* 58.4 (2020), pp. 2078–2113.

[FMW21] S. Fécamp, J. Mikael, and X. Warin. "Deep learning for discrete-time hedging in incomplete markets". In: *Journal of Computational Finance* 25.2 (2021), 51–85.

[FTT19] M. Fujii, A. Takahashi, and M. Takahashi. "Asymptotic expansion as prior knowledge in deep learning method for high dimensional BSDEs". In: *Asia Pacific Financial Markets* 26.3 (2019), pp. 391–408.

[FTT20] O. Féron, P. Tankov, and L. Tinsi. "Price Formation and Optimal Trading in Intraday Electricity Markets with a Major Player". In: *Risks* 8.4 (2020). ISSN: 2227-9091.

[FTT21] O. Féron, P. Tankov, and L. Tinsi. "Price formation and optimal trading in intraday electricity markets". In: *Math Finan Econ* (2021).

[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.

[FZ20] J-.P. Fouque and Z. Zhang. "Deep Learning Methods for Mean Field Control Problems with Delay". In: *Frontiers in Applied Mathematics and Statistics* 6 (2020). DOI: https://doi.org/10.3389/fams.2020.00011.

[GAS20] C. Gräser and P. A. Alathur Srinivasan. "Error bounds for PDE-regularized learning". In: *arXiv:2003.06524* (2020).

[Gel+13] A. Geletu et al. "Advances and applications of chance-constrained approaches to systems optimisation under uncertainty". In: *International Journal of Systems Science* 44.7 (2013), pp. 1209–1232.

[Ger+22] M. Germain et al. "DeepSets and derivative networks for solving symmetric PDEs". In: *Journal of Scientific Computing* 91.63 (2022).

[Gev+18] H. Gevret et al. *STochastic OPTimization library in C++*. 2018. URL: https://hal.archives-ouvertes.fr/hal-01361291.

[GG21a] E. Gobet and M. Grangereau. "Extended McKean-Vlasov optimal stochastic control applied to smart grid management". In: *hal-02181227* (2021).

[GG21b] E. Gobet and M. Grangereau. "Federated stochastic control of numerous heterogeneous energy storage systems". In: *hal-03108611* (2021).

[GHLT14] A. Galichon, P. Henry-Labordère, and N. Touzi. "A stochastic control approach to no-arbitrage bounds given marginals, with an application to lookback options". In: *The Annals of Applied Probability* 24.1 (2014), pp. 312 –336.

[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 (2005), pp. 2172–2202.

[GM05] E. Gobet and R. Munos. "Sensitivity analysis using Itô-Malliavin calculus and martingales, and application to stochastic optimal control". In: *SIAM J. Control Optim.* 43.5 (2005), pp. 1676–1713.

[GM21] J. Graber and S. Mayorga. "A note on mean field games of controls with state constraints: existence of mild solutions". In: *arXiv:2109.11655* (2021).

[GMS21] W. Gangbo, S. Mayorga, and A. Swiech. "Finite Dimensional Approximations of Hamilton–Jacobi–Bellman Equations in Spaces of Probability Measures". In: *SIAM Journal on Mathematical Analysis* 53.2 (2021), pp. 1320–1356.

[GMW22] M. Germain, J. Mikael, and X. Warin. "Numerical resolution of McKean-Vlasov FBSDEs using neural networks". In: *Methodology and Computing in Applied Probability* (2022). DOI: https://doi.org/10.1007/s11009-022-09946-1.

[GPW21] M. Germain, H. Pham, and X. Warin. "A level-set approach to the control of state-constrained McKean-Vlasov equations: application to renewable energy storage and portfolio selection". In: *arXiv:2112.11059* (2021).

[GPW22a] M. Germain, H. Pham, and X. Warin. "Approximation Error Analysis of Some Deep Backward Schemes for Nonlinear PDEs". In: *SIAM Journal on Scientific Computing* 44.1 (2022), A28–A56.

[GPW22b] M. Germain, H. Pham, and X. Warin. "Neural networks based algorithms for stochastic control and PDEs in finance". In: *arXiv:2101.08068 to appear in Machine Learning And Data Sciences For Financial Markets: A Guide To Contemporary Practices*. Ed. by A. Capponi and C.A. Lehalle. Cambridge University Press, 2022.

[GPW22c] M. Germain, H. Pham, and X. Warin. "Rate of convergence for particle approximation of PDEs in Wasserstein space". In: *to appear in Journal of Applied Probability 59.4* (2022).

[Gro+18] P. Grohs et al. "A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equation". In: *to appear in Memoirs of the American mathematical society* (2018).

[GS15] W. Gangbo and A. Swiech. "Existence of a solution to an equation arising from the theory of mean-field games". In: *Journal of Differential equations* 259.11 (2015), pp. 6573–6643.

[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).

[Gu+20] H. Gu et al. "Q-Learning Algorithm for Mean-Field Controls, with Convergence and Complexity Analysis". In: *arXiv:2002.04131* (2020).

[Guo+20] X. Guo et al. "A General Framework for Learning Mean-Field Games". In: *arXiv:2003.06069* (2020).

[GW20] K. Glau and L. Wunderlich. "The deep parametric PDE method: application to option pricing". In: *arXiv:2012.06211* (2020).

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

[HCM06] M. Huang, P. Caines, and R. Malhamé. "Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle". In: *Communication in Information and Systems* 3 (2006), pp. 221–252.

[HE16] J. Han and W. E. "Deep learning approximation for stochastic control problems". In: *Deep Reinforcement Learning Workshop* (2016).

[HH21] J. Han and R. Hu. "Recurrent Neural Networks for Stochastic Control Problems with Delay". In: *arXiv:2101.01385* (2021).

[HHL20] J. Han, R. Hu, and J. Long. "Convergence of deep fictitious play for stochastic differential games". In: *arXiv:2008.05519* (2020).

[HJE18] J. Han, A. Jentzen, and W. E. "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences of America* 115.34 (2018), pp. 8505–8510.

[HK20] M. Hutzenthaler and T. Kruse. "Multilevel Picard Approximations of High-Dimensional Semilinear Parabolic Differential Equations with Gradient-Dependent Nonlinearities". In: *SIAM J. Numer. Anal.* 58.2 (2020), pp. 929–961.

[HL12] P. Henry-Labordère. "Counterparty risk valuation: a marked branching diffusion approach". In: *Hal-00677348* (2012).

[HL17] P. Henry-Labordere. "Deep Primal-Dual Algorithm for BSDEs: Applications of Machine Learning to CVA and IM". In: *Available at SSRN: https://ssrn.com/abstract=3071506* (2017).

[HL+19] P. Henry-Labordere et al. "Branching diffusion representation of semilinear PDEs and Monte Carlo approximation". In: *Ann. Inst. Henri Poincaré Probab. Stat.* 55.1 (2019), pp. 184–210.

[HL20] J. Han and J. Long. "Convergence of the Deep BSDE Method for Coupled FBSDEs". In: *Probability, Uncertainty and Quantitative Risk* 5.1 (2020), pp. 1–33.

[Hor91] K. Hornik. "Approximation Capabilities of Multilayer Feedforward Networks". In: *Neural Networks* 4 (1991), pp. 251–257.

[HPW20] C. Huré, H. Pham, and X. Warin. "Deep backward schemes for high-dimensional nonlinear PDEs". In: *Mathematics of Computation* 89.324 (July 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 Networks* 3(5) (1990), pp. 551–560.

[Hu19] R. Hu. "Deep fictitious play for stochastic differential games". In: *arXiv:1903.09376* (2019).

[Hur19] C. Huré. "Numerical Methods and Deep Learning for Stochastic Control Problems and Partial Differential Equations". PhD thesis. Université Paris Diderot (Paris 7), Sorbonne Paris Cité, June 2019. URL: `https://tel.archives-ouvertes.fr/tel-02331441`.

[Hur+21] C. Huré et al. "Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis". In: *SIAM J. Numer. Anal.* 59.1 (2021), 525–557.

[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).

[Hut+20] M. Hutzenthaler et al. "A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equation". In: *SN partial differential equations and applications* 1.10 (2020), pp. 1–34.

[IP19] A. Ismail and H. Pham. "Robust Markowitz mean-variance portfolio selection under ambiguous covariance matrix". In: *Mathematical Finance* 29.174-207 (2019).

[Ji+20a] S. Ji et al. "Solving stochastic optimal control problem via stochastic maximum principle with deep learning method". In: *arXiv:2007.02227* (2020).

[Ji+20b] S. Ji et al. "Three Algorithms for Solving High-Dimensional Fully Coupled FBSDEs Through Deep Learning". In: *IEEE Intelligent Systems* 35.3 (2020), pp. 71–84. DOI: `10.1109/MIS.2020.2971597`.

[JL21] Y. Jiang and J. Li. "Convergence of the Deep BSDE method for FBSDEs with non-Lipschitz coefficients". In: *arXiv:2101.01869* (2021).

[JO19] A. Jacquier and M. Oumgari. "Deep curve-dependent PDEs for affine rough volatility". In: *arXiv:1906.02551* (2019).

[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.

[KJY20] Y. Khoo, J.Lu, and L. Ying. "Solving parametric PDE problems with artificial neural networks". In: *European Journal of Applied Mathematics* (2020), pp. 1–15.

[KLW21] I. Kharroubi, T. Lim, and X. Warin. "Discretization and Machine Learning Approximation of BSDEs with a Constraint on the Gains-Process". In: *Monte Carlo Methods and Applications* 27.1 (2021), pp. 27–55.

[KSS20] S. Kremsner, A. Steinicke, and M. Szölgyenyi. "A deep neural network algorithm for semilinear elliptic PDEs with applications in insurance mathematics". In: *arXiv:2010.15757* (2020).

[Lac17] D. Lacker. "Limit Theory for Controlled McKean–Vlasov Dynamics". In: *SIAM Journal on Control and Optimization* 55.3 (2017), pp. 1641–1672.

[Lee+21] W. Lee et al. "Controlling Propagation of Epidemics via Mean-Field Control". In: *SIAM Journal on Applied Mathematics* 81.1 (2021), pp. 190–207.

[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 (2006), pp. 889–916.

[LL06a] J.-M. Lasry and P.-L. Lions. "Jeux à champ moyen. I – Le cas stationnaire". In: *Comptes Rendus Mathematique - C R MATH* 343 (Nov. 2006), pp. 619–625. DOI: 10.1016/j.crma.2006.09.019.

[LL06b] J.-M. Lasry and P.-L. Lions. "Jeux à champ moyen. II – Horizon fini et contrôle optimal". In: *Comptes Rendus. Mathématique. Académie des Sciences, Paris* 10 (Nov. 2006). DOI: 10.1016/j.crma.2006.09.018.

[LL07] J.-M. Lasry and P.-L. Lions. "Mean field games". In: *Japanese Journal of Mathematics* 2 (2007), pp. 229–260.

[LLP20] W. Lefebvre, G. Loeper, and H. Pham. "Mean-Variance Portfolio Selection with Tracking Error Penalization". In: *Mathematics* 8.11 (2020).

[LM21] W. Lefebvre and E. Miller. "Linear-Quadratic Stochastic Delayed Control and Deep Learning Resolution". In: *Journal of Optimization Theory and Applications* 191 (2021), 134–168.

[LP14] M. Laurière and O. Pironneau. "Dynamic programming for mean-field type control". In: *Comptes Rendus Mathematique* 352.9 (2014), pp. 707–713.

[LT19] M. Laurière and L. Tangpi. "Backward propagation of chaos". In: *arXiv:1911.06835* (2019).

[LT20] M. Laurière and L. Tangpi. "Convergence of large population games to mean field games with interaction through the controls". In: *arXiv:2004.08351* (2020).

[LXL19] J. Liang, Z. Xu, and P. Li. "Deep Learning-Based Least Square Forward-Backward Stochastic Differential Equation Solver for High-Dimensional Derivative Pricing". In: *arXiv:1907.10578* (2019).

[MZ19] J. Müller and M. Zeinhofer. "Deep Ritz revisited". In: *arXiv:1912.03937* (2019).

[NAO21] B. Negyesi, K. Andersson, and C. W. Oosterlee. "The One Step Malliavin scheme: new discretization of BSDEs implemented with deep learning regressions". In: *arXiv:2110.05421* (2021).

[NR20] N. Nüsken and L. Richter. "Solving high-dimensional Hamilton-Jacobi-Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space". In: *arXiv:2005.05409* (2020).

[Pas+19] A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[Pha00] H. Pham. "On quadratic hedging in continuous time". In: *Mathematical Methods of Operations Research* 51 (2000), 315–339.

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

[Pin99a] A. Pinkus. "Approximation theory of the MLP model". In: *Acta Numerica* 8 (1999), pp. 143–195.

[Pin99b] A. Pinkus. "Approximation theory of the MLP model in neural networks". In: *Acta numerica* 8 (1999), pp. 143–195.

[PJ21] S. M. Pesenti and S. Jaimungal. "Portfolio Optimisation within a Wasserstein Ball". In: *Available at SSRN: https://ssrn.com/abstract=3744994* (2021).

[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.

[PTZ21] L. Pfeiffer, X. Tan, and Y.-L. Zhou. "Duality and Approximation of Stochastic Optimal Control Problems under Expectation Constraints". In: *SIAM Journal on Control and Optimization* 59.5 (2021), pp. 3231–3260.

[PV20] A. Picarelli and T. Vargiolu. "Optimal management of pumped hydroelectric production with state constrained optimal control". In: *Journal of Economic Dynamics and Control* (2020), p. 103940.

[PW17] H. Pham and X. Wei. "Dynamic programming for optimal control of stochastic McKean-Vlasov dynamics". In: *SIAM J. Control Optim.* 55.2 (2017), pp. 1069–1101.

[PW18] H. Pham and X. Wei. "Bellman equation and viscosity solutions for mean-field stochastic control problem". In: *ESAIM: COCV* 24.1 (2018), pp. 437–461.

[PWG21] H. Pham, X. Warin, and M. Germain. "Neural networks-based backward scheme for fully nonlinear PDEs". In: *SN Partial Differential Equations and Applications* 2.16 (2021).

[Qi+17] C. Ruizhongtai Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 77–85.

[Rai18] M. Raissi. "Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations". In: *arXiv:1804.07010* (2018).

[RM51] H. Robbins and S. Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407.

[RPK19] M. Raissi, P. Perdikaris, and G.E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *J. Comput. Phys.* 378 (2019), pp. 686 –707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045.

[RSZ20] C. Reisinger, W. Stockinger, and Y. Zhang. "A posteriori error estimates for fully coupled McKean-Vlasov forward-backward SDEs". In: *arXiv:2007.07731* (2020).

[RSZ21] C. Reisinger, W. Stockinger, and Y. Zhang. "A fast iterative PDE-based algorithm for feedback controls of nonsmooth mean-field control problems". In: *arXiv:2108.06740* (2021).

[RT17] Z. Ren and X. Tan. "On the convergence of monotone schemes for path-dependent PDEs". In: *Stochastic Processes and their Applications* 127.6 (2017), pp. 1738–1762. ISSN: 0304-4149.

[Rut+20] L. Ruthotto et al. "A machine learning framework for solving high-dimensional mean field game and mean field control problems". In: *Proc. Natl. Acad. Sci. USA* 117.17 (2020), pp. 9183–9193. ISSN: 0027-8424. DOI: 10.1073/pnas.1922204117.

[Sak20] T. Sakuma. "Application of deep quantum neural networks to finance". In: *arXiv:2011.07319v1* (2020).

[SDEK20] Y. Shin, J. Darbon, and G. Em Karniadakis. "On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs". In: *arXiv:2004.01806* (2020).

[SMLN15] R. Salhab, R. P. Malhamé, and J. Le Ny. "A dynamic game model of collective choice in multi-agent systems". In: *2015 IEEE 54th Annual Conference on Decision and Control (CDC)*. 2015, pp. 4444–4449.

[Smu11] J. Smulevici. "On the area of the symmetry orbits of cosmological spacetimes with toroidal or hyperbolic symmetry". In: *Analysis and PDE* 4.2 (2011), pp. 191–245.

[SS18] J. Sirignano and K. Spiliopoulos. "DGM: A deep learning algorithm for solving partial differential equations". In: *Journal of Computational Physics* 375 (2018), pp. 1339–1364.

[ST02] H. M. Soner and N. Touzi. "Stochastic Target Problems, Dynamic Programming, and Viscosity Solutions". In: *SIAM Journal on Control and Optimization* 41.2 (2002), pp. 404–424.

[SVSS20] M. Sabate Vidales, D. Siska, and L. Szpruch. "Solving path dependent PDEs with LSTM networks and path signatures". In: *arXiv:2011.10630v1* (2020).

[SWA21a] A. Séguret, C. Wan, and C. Alasseur. "A mean field control approach for smart charging with aggregate power demand constraints". In: *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. 2021, pp. 01–05.

[SWA21b] A. Séguret, C. Wan, and C. Alasseur. "Computation and implementation of an optimal mean field control for smart charging". In: *arXiv:2110.00332* (2021).

[SZ19] Y. F. Saporito and J. Zhang. "Stochastic Control with Delayed Information and Related Nonlinear Master Equation". In: *SIAM Journal on Control and Optimization* 57.1 (2019), pp. 693–717.

[SZ20] Y. F. Saporito and Z. Zhang. "PDGM: a Neural Network Approach to Solve Path-Dependent Partial Differential Equations". In: *arXiv:2003.02035* (2020).

[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.

[Tan13] X. Tan. "A splitting method for fully nonlinear degenerate parabolic PDEs". In: *Electronic Journal of Probability* 18 (2013).

[TSB21] U. Tanielian, M. Sangnier, and G. Biau. "Approximating Lipschitz continuous functions with GroupSort neural networks". In: Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS). PMLR: Volume 130, 2021.

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

[Wag+19] E. Wagstaff et al. "On the Limitations of Representing Functions on Sets". In: ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. 2019, pp. 6487–6494.

[War12] X. Warin. "Gas Storage Hedging". In: *Numerical methods in finance*. Ed. by R. Carmona et al. Springer, 2012, pp. 421–445.

[War18a] X. Warin. "Monte Carlo for high-dimensional degenerated Semi Linear and Full Non Linear PDEs". In: *arXiv:1805.05078* (2018).

[War18b] X. Warin. "Nesting Monte Carlo for high-dimensional non-linear PDEs". In: *Monte Carlo Methods Appl.* 24.4 (2018), pp. 225–247.

[War21a] X. Warin. "Deep learning for efficient frontier calculation in finance". In: *arXiv:2101.02044, to appear in Journal of Computational Finance* (2021).

[War21b] X. Warin. "Reservoir optimization and Machine Learning methods". In: *arXiv:2106.08097* (2021).

[Wyk08] S. Van Wyk. "Partial differential equations and quantum mechanics". In: *Computer solution in Physics* (2008), pp. 99–139.

[WZ20] C. Wu and J. Zhang. "Viscosity solutions to parabolic master equations and McKean–Vlasov SDEs with closed-loop controls". In: *Annals of Applied Probability* 30.2 (Apr. 2020), pp. 936–986.

[Yar17] D. Yarotsky. "Error bounds for approximations with deep ReLU networks". In: *Neural Networks* 94 (2017), pp. 103–114.

[Zah+17] M. Zaheer et al. "Deep Sets". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 3391–3401.

[Zar01] T. Zariphopoulou. "A solution approach to valuation with unhedgeable risks". In: *Finance and Stochastics* 5 (2001), pp. 61–82.

[Zei12] M. D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *arXiv:1212.5701* (2012).

[Zha04] J. Zhang. "A numerical scheme for BSDEs". In: *Ann. Appl. Probab.* 14.1 (2004), pp. 459–488.

[Zha17] J. Zhang. *Backward stochastic differential equations: from linear to fully nonlinear theory*. Vol. 86. Probability theory and stochastic modelling. Springer, 2017.

[ZL00] X. Y. Zhou and D. Li. "Continuous-Time Mean-Variance Portfolio Selection: A Stochastic LQ Framework". In: *Applied Mathematics and Optimization* 42 (2000), 19–33.

List of Figures

List of Tables