Numerical methods and deep learning for stochastic control problems and partial differential equations

Come Huré

Université Sorbonne Paris Cité
Université Paris Diderot

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

Thèse de doctorat

en Mathématiques Appliquées

Présentée par
Côme Huré

Méthodes Numériques et Réseaux de Neurones pour le Contrôle Stochastique et les Équations aux Dérivées Partielles

---

Numerical Methods and Deep Learning for Stochastic Control Problems and Partial Differential Equations

---

Sous la direction de **Frédéric Abergel** et **Huyên Pham**

Soutenue le *27 juin 2019* devant le jury composé des professeurs:

Frédéric Abergel	BNP Paribas - AM	Directeur de thèse
Jean-François Chassagneux	Université Paris Diderot	Examinateur
Romuald Élie	Université Paris-Est	Rapporteur
Emmanuel Gobet	École Polytechnique	Examinateur
Charles-Albert Lehalle	Capital Fund Management	Examinateur
Gilles Pagès	Université Pierre et Marie Curie	Président du jury
Huyên Pham	Université Paris Diderot	Directeur de thèse
John Schoenmakers	Weierstrass Institute	Rapporteur

Remerciements

Ces quelques lignes sont pour moi l'occasion d'exprimer ma gratitude envers tous ceux qui m'ont accompagné dans cette thèse. J'espère aussi qu'elles divertiront les plus courageux qui sont venus me soutenir soutenir: notamment les malheureux qui n'ont pas révisé le cours de contrôle stochastique et les EDPs avant de venir, et ceux dans le fond qui trouvent le temps long mais qui n'osent pas sortir le journal.

Je tiens avant tout à remercier mes deux directeurs de thèse: Frédéric Abergel et Huyên Pham, avec qui se fut enrichissant de travailler avec vous pendant ces 3+1 dernières années. J'ai énormément appris à vos côtés. Frédéric, c'était une vraie chance de te connaître. C'était un plaisir d'aller te voir de temps-en-temps à l'École Centrale pour te parler des grands malheurs que je rencontrais dans mes premières années de recherche, et de constater que j'avais toujours ton soutien à la fin des rendez-vous: je repartais souvent requinqué, avec le sourire, et l'impression naïve mais éphémère que tous les problèmes étaient enfin réglés. Huyên, merci beaucoup pour votre présence tout au long de cette thèse. Je vous remercie pour votre disponibilité: vous avez toujours pris le temps de répondre à mes questions, même quand je toquais à votre porte après 19h alors que vous pensiez sûrement pouvoir travailler tranquillement à cette heure là. Je vous suis très reconnaissant pour les projets passionnants sur lesquelles vous m'avez proposé de travailler; et sur lesquels je me suis beaucoup épanoui pendant ma thèse.

Je remercie sincèrement mes rapporteurs: Romuald Élie et John Schoenmakers, pour avoir accepté de relire attentivement mon manuscrit. Je suis honoré de la présence dans mon jury de Jean-François Chassagneux, Emmanuel Gobet, Charles-Albert Lehalle, Gilles Pagès et de mes deux rapporteurs. Vos travaux et nos discussions m'ont beaucoup marqué.

Merci aux membres du LPSM avec qui j'ai partagé de grands moments. Je remercie en particulier les doctorants de mon bureau à p7, le bureau *Serpentard*, dans lequel le choixpeau arguably tend à placer le top du top des doctorants. Il y a d'abord les anciens: Clément, Adrien et 婷婷, avec qui j'ai eu le plaisir de partager ma première année de thèse; David Krief, qui m'a supporté 3 ans; 俊超; 易洋, qui ne me refuse jamais une partie de Go, surtout quand je pose le Goban au milieu de son bureau et que je rapproche les pierres; 厚志, au labo 7j/7; Johann qui, je l'espère, a fini par se remettre de son séjour au Limmathof; Enzo, qui a très peur de ne pas voir son nom apparaître dans cette liste; William, toujours en costume; Mamadou; Shanqiu; Remy; et Médéric. Je tiens aussi à remercier les doctorants que le choixpeau n'a pas jugé judicieux de mettre chez Serpentard. Dans le bureau *Serdaigle*: merci à Cyril; Lucas, "quand est-ce que tu retournes à New York?"; Laure, que je suspecte encore beaucoup d'aimer secrètement la finance; Assaf, dont l'absence à ma soutenance de thèse est pardonnée parce ça le tuerait de se lever avant 11h; Guillaume, et sa balle de tennis; Sothea; et Hiroshi. Dans le bureau *Gryffondor*, je remercie Luca, que j'ai eu le plaisir de bien connaître à Zurich; Fabio 4kyu, pour nos conversations quotidiennes sur la course à pieds; Vu Lan; Bogdan, qui aime bien le silence; Clément, qui ne supporte pas le silence; Yann, qui est un excellent câlin provider; Benjamin, dont une regression avec régularisation LASSO montrerait facilement que ses passages

fréquents en chaussettes expliquent à eux seuls la propreté du sol dans le labo; Mi-Song; Arturo; 晓利, la plus discrète. Dans le bureau *Pouffesoufle*, il y a Ziad, qui a passé les 6 premiers mois de sa thèse à jurer que jamais, jamais au grand jamais, il ne s'abaisserait à faire du numérique, et qui passe maintenant ses journées à débugger ses milliers de lignes de C++; Marc; S. (qui préfère rester anonyme); Barbara; Clément; ainsi que Georg. Je remercie Nathalie et Valérie et Amina qui ont fait un travail remarquable pour m'accompagner dans toutes mes démarches administratives, et sur qui j'ai pu compter même quand je faisais n'importe quoi avec les ordres de mission.

Merci aux doctorants de p6 que je croisais quelques fois au groupes de travail, en particulier: 雅婷; et Thibaut, avec qui j'ai passé de très bons moments ~~dans les restaurants d'altitude~~ à Métabief. Je remercie les membres de la Chair of quantitative finance à CentraleSupelec; en particulier XiaoFei, avec qui c'était un plaisir de consacrer les vendredis midi à la préparation des TDs, autour de bons repas chinois ou japonais. J'en profite enfin pour remercier les participants du CEMRACS2017, notamment les membres de mon projet: Isaque, Alessandro, et Mathieu.

Au cours de ma scolarité, j'ai rencontré beaucoup de professeurs qui ont su me transmettre la passion pour les maths. Je remercie en particulier Sébatien Chevalier, mon professeur au lycée, qui a été le premier à m'enseigner la rigueur mathématique; ainsi que René Martel, mon professeur en mathématiques spéciales, pour son enseignement divin, sa justesse et sa gentillesse. Merci à Pierre, bluffant en maths et encore plus en kite-surf; Yohan, sur qui l'on compte pour développer l'AI de demain; et Quentin, qui fait toujours preuve d'une grande patience lorsqu'on le fait tourner en rond pendant des heures à la recherche du meilleur restaurant.

Je remercie ma famille et mes amis qui m'ont accompagné et soutenu, et en particulier mes parents; Stéphane; Caroline; Éléonore; Hortense; et le chat.

Abstract

The present thesis deals with numerical schemes to solve Markov Decision Problems (MDPs), partial differential equations (PDEs), quasi-variational inequalities (QVIs), backward stochastic differential equations (BSDEs) and reflected backward stochastic differential equations (RBSDEs). The thesis is divided into three parts.

The first part focuses on methods based on quantization, local regression and global regression to solve MDPs. Firstly, we present a new algorithm, named *Qknn*, and study its consistency. A time-continuous control problem of market-making is then presented, which is theoretically solved by reducing the problem to a MDP, and whose optimal control is accurately approximated by *Qknn*. Then, a method based on Markovian embedding is presented to reduce McKean-Vlasov control problem with partial information to standard MDP. This method is applied to three different McKean-Vlasov control problems with partial information. The method and high accuracy of Qknn is validated by comparing the performance of the latter with some finite difference-based algorithms and some global regression-based algorithm such as regress-now and regress-later.

In the second part of the thesis, we propose new algorithms to solve MDPs in high-dimension. Neural networks, combined with gradient-descent methods, have been empirically proved to be the best at learning complex functions in high-dimension, thus, leading us to base our new algorithms on them. We derived the theoretical rates of convergence of the proposed new algorithms, and tested them on several relevant applications.

In the third part of the thesis, we propose a numerical scheme for PDEs, QVIs, BSDEs, and RBSDEs. We analyze the performance of our new algorithms, and compare them to other ones available in the literature (including the recent one proposed in [EHJ17]) on several tests, which illustrates the efficiency of our methods to estimate complex solutions in high-dimension.

Keywords: Deep learning, neural networks, Stochastic control, Markov Decision Process, nonlinear PDEs, QVIs, optimal stopping problem BSDEs, RBSDEs, McKean-Vlasov control, performance iteration, value iteration, hybrid iteration, global regression, local regression, regress-later, quantization, limit order book, pure-jump controlled process, algorithmic-trading, market-making, high-dimension.

List of the five papers being part of this thesis:

- F. Abergel, C. Huré, and H. Pham. "Algorithmic trading in a microstructural limit order book model". In: *arXiv:1705.01446* (2017), submitted to Quantitative Finance.
- A. Balata, C. Huré, M. Laurière, H. Pham, and I. Pimentel. "A class of finite-dimensional numerically solvable McKean-Vlasov control problems". In: *ESAIM Proceedings and surveys* 65 (2019).
- A. Bachouch, C. Huré, N. Langrené, and H. Pham. "Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications". In: *arXiv:1812.05916* (2018), in revision to Methodology and Computing in Applied Probability.
- C. Huré, H. Pham, A. Bachouch, and N. Langrené. "Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis". In: *arXiv:1812.04300* (2018), submitted to SIAM Journal on Numerical Analysis.
- C. Huré, H. Pham, and X. Warin. "Some machine learning schemes for high-dimensional nonlinear PDEs". In: *arXiv:1902.01599* (2019), submitted to Mathematics of Computation, American Mathematical Society.

Résumé

La thèse porte sur les schémas numériques pour les problèmes de décisions Markoviennes (MDPs), les équations aux dérivées partielles (EDPs), les équation différentielles stochastiques rétrogrades (EDSRs), ainsi que les équations différentielles stochastiques rétrogrades réfléchies (EDSRs réfléchies). La thèse se divise en trois parties.

La première partie porte sur des méthodes numériques pour résoudre les MDPs, à base de quantification et de régression locale ou globale. Un problème de market-making est proposé: il est résolu théoriquement en le réécrivant comme un MDP; et numériquement en utilisant le nouvel algorithme. Dans un second temps, une méthode de Markovian embedding est proposée pour réduire des problèmes de type McKean-Vlasov avec information partielle à des MDPs. Cette méthode est mise en œuvre sur trois différents problèmes de type McKean-Vlasov avec information partielle, qui sont par la suite numériquement résolus en utilisant des méthodes numériques à base de régression et de quantification.

Dans la seconde partie, on propose de nouveaux algorithmes pour résoudre les MDPs en grande dimension. Ces derniers reposent sur les réseaux de neurones, qui ont prouvé en pratique être les meilleurs pour apprendre des fonctions en grande dimension. La consistance des algorithmes proposés est prouvée, et ces derniers sont testés sur de nombreux problèmes de contrôle stochastique, ce qui permet d'illustrer leurs performances.

Dans la troisième partie, on s'intéresse à des méthodes basées sur les réseaux de neurones pour résoudre les EDPs, EDSRs et EDSRs réfléchies. La convergence des algorithmes proposés est prouvée; et ces derniers sont comparés à d'autres algorithmes récents de la littérature sur quelques exemples, ce qui permet d'illustrer leurs très bonnes performances.

Mots-clés: Réseaux de neurones, contrôle stochastique, MDPs, EDPs, IQVs, Temps d'arrêt optimaux, EDPRs, EDPRs réfléchies, contrôle de type McKean-Vlasov, régression globale, regress-later, regression locale, quantification, carnet d'ordres, trading algorithmique, market-making, grande-dimension.

Liste des cinq papiers inclus dans cette thèse:

- F. Abergel, C. Huré, and H. Pham. "Algorithmic trading in a microstructural limit order book model". In: *arXiv:1705.01446* (2017), soumis à Quantitative Finance.
- A. Balata, C. Huré, M. Laurière, H. Pham, and I. Pimentel. "A class of finite-dimensional numerically solvable McKean-Vlasov control problems". In: *ESAIM Proceedings and surveys* 65 (2019).
- A. Bachouch, C. Huré, N. Langrené, and H. Pham. "Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications". In: *arXiv:1812.05916* (2018), en révision chez Methodology and Computing in Applied Probability.
- C. Huré, H. Pham, A. Bachouch, and N. Langrené. "Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis". In: *arXiv:1812.04300* (2018), soumis à SIAM Journal on Numerical Analysis.
- C. Huré, H. Pham, and X. Warin. "Some machine learning schemes for high-dimensional nonlinear PDEs". In: *arXiv:1902.01599* (2019), soumis à Mathematics of Computation, American Mathematical Society.

Contents

Chapter 1

Introduction

The thesis is divided into three parts that can be read independently. In the first part, we propose algorithms based on quantization and regression to solve Markov Decision Processes (MDPs). We then present two applications: a problem of market-making and a McKean-Vlasov-type stochastic control problem[1]. In the second part of the thesis, we present a class of numerical schemes based on neural-networks to solve MDPs. Consistence results are derived, which show the convergence of our new algorithms. We then test them on several interesting examples found in the literature. In the third part of the thesis, we present neural-network-based numerical schemes to solve PDEs, QVIs, BSDEs and RBSDEs. Consistence results of our estimates are derived, and tests are performed which illustrate the great behaviors of our algorithms compared to other recent algorithms available in the literature.

Contents

[1] i.e. stochastic control problems in which the coefficients of the dynamic of the controlled process depend on the law of the controlled process itself

1.1 Quantization and regression methods for MDPs. Applications to market-making and McKean-Vlasov-type control problem with partial information

1.1.1 Quantization and regression methods for Markov decision processes (MDPs).

After reviewing literature about Markov decision processes (MDPs) that are usually solved by regression methods in practice, we proposed a new algorithm based on local regression and quantization, which we refer to as *Qknn* in this thesis.

1.1.1.1 Presentation of the MDPs

Let us consider a stochastic control problem with finite horizon and discrete time, i.e. we fix a finite horizon $N \in \mathbb{N} \setminus \{0\}$, and consider the (controlled process) $X^\alpha = (X_n^\alpha)_{n=0}^N$, taking values in $\mathcal{X} \subset \mathbb{R}^d$, with $d \in \mathbb{N}$ being the dimension of the problem, and s.t. it admits the following dynamics:

$$\left\{ \begin{array}{rcl} X_0^\alpha & = & x_0 \in \mathbb{R}^d, \\ X_{n+1}^\alpha & = & F_n(X_n^\alpha, \alpha_n, \varepsilon_{n+1}), \quad \text{for } n = 0, \dots, N-1, \end{array} \right.$$

where:

- $(\varepsilon_n)_{n=1}^N$ is a sequence of i.i.d. r.v., which takes values in a Borelian set $(E, \mathcal{B}(E))$, and defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We denote by $\mathbb{F} = (\mathcal{F}_n)_{n=0}^N$ the σ-algebra generated by $(\varepsilon_n)_{n=1}^N$ ($\mathcal{F}_0$ is the trivial σ-algebra).
- The control α is a $\mathbb{F}$-measurable process taking values in $\mathbb{A} \subset \mathbb{R}^q$.
- For $n = 0, \dots, N-1$, the F_n are measurable functions from $\mathbb{R}^d \times \mathbb{R}^q \times E$ to $\mathbb{R}^d$.

Let $\beta \geq 0$ be a discount factor, $f : \mathbb{R}^d \times \mathbb{R}^q \to \mathbb{R}$ be the instantaneous cost, and $g : \mathbb{R}^d \to \mathbb{R}$ be the terminal cost. We define $J(\alpha)$, the cost functional associated to the control α, as follows:

$$J(\alpha) \quad = \quad \mathbb{E}\left[\sum_{n=0}^{N-1} \beta^n f(X_n^\alpha, \alpha_n) + g(X_N^\alpha) \right]. \tag{1.1.1}$$

The control α is *admissible* if $J(\alpha)$ is well-defined, and we denote by $\mathcal{C}$ the set of the admissible controls. The control problem is to find

$$V_0(x_0) := \inf_{\alpha \in \mathcal{C}} J(\alpha), \tag{1.1.2}$$

and characterize (at least) one optimal control $\alpha^* \in \mathcal{C}$, which minimizes the cost functional, i.e. $V_0(x_0) =: J(\alpha^*)$.

In this thesis, we refer to the control problems with finite horizon and discrete time as Markov Decision Processes with finite horizon, or shortly as MDPs.

The first difficulty for solving (1.1.2) is that the infimum is taken on the set of the admissible processes $\mathcal{C}$, which can very often be of infinite dimension in practice. Before simplifying (1.1.2) using dynamic programming principle, let us first introduce some basic notations: we denote by $\{P^a(x, dx'),\, a \in \mathbb{A},\, x \in \mathcal{X}\}$ the family of transition probabilities associated to the controlled Markov chain (1.1.1.1), i.e.

$$P^a(x, dx') \quad = \quad \mathbb{P}\left[F(x, a, \varepsilon_1) \in dx'\right],$$

and moreover, we define

$$P^a\varphi(x) = \int \varphi(x')P^a(x,dx') \quad = \quad \mathbb{E}\left[\varphi\big(F(x,a,\varepsilon_1)\big)\right],$$

for all measurable function $\varphi : \mathcal{X} \to \mathbb{R}$.

With these notations, for all functions φ on $\mathcal{X}$, and for all control $\alpha \in \mathcal{C}$, we have:

$$\mathbb{E}[\varphi(X^\alpha_{n+1})|\mathcal{F}_n] \quad = \quad P^{\alpha_n}\varphi(X^\alpha_n), \quad \forall\, n \in \mathbb{N}.$$

By programming dynamic principle, $V_n(.)$, the value function at time n, for $n = 0, \dots, N$, is characterized as solution to the following backward equation[2]

$$\left\{\begin{array}{rcl} V_N(x) & = & g(x), \quad \forall x \in \mathcal{X}, \\ Q_n(x,a) & = & f(x,a) + \beta P^a V_{n+1}(x), \quad \forall x \in \mathcal{X},\ a \in \mathbb{A}, \\ V_n(x) & = & \inf\limits_{a\in\mathbb{A}} Q_n(x,a), \qquad \text{for } n = N-1, \dots, 0, \end{array}\right. \tag{1.1.3}$$

and moreover, α^*_n, the optimal feedback control at time n, is characterized as follows:

$$\alpha^*_n(x) \in \operatorname*{argmin}_{a\in\mathbb{A}} Q_n(x,a). \tag{1.1.4}$$

The community of the Reinforcement Learning (RL) usually refers to the function Q_n as the "Q-value function" or the "state-action function", and V_n as the value function at time n.

Remark 1.1.1. *A very general class of continuous/discrete with timeinfinite/finite horizon control problems can be reduced to (or approximated by) the scheme (1.1.3). For example, let us consider the following continous-time control problem with infinite horizon:*

$$\mathscr{V}(x) := \sup_{\alpha\in\mathcal{C}} \mathbb{E}\left[\int_0^\infty e^{-\lambda t} f(X^\alpha_t)dt\right],$$

with $\lambda > 0$, and where we assume that the controlled process X^α, for a control α follows the dynamic below:

$$dX^\alpha_t = b(t, X_t, \alpha_t)dt + \sigma(t, X_t, \alpha_t)dW_t,$$

where the drift b and the volatility σ are regular enough. Let us first discretize in time the problem: let us take a step size $h \ll 1$, and discretize the time as follows:

$$\mathscr{V}(x) \approx \mathscr{V}_h(x) := \sup_{\alpha\in\mathcal{C}} \mathbb{E}\left[\sum_{n=0}^\infty h e^{-\lambda nh} f(X^\alpha_{nh})\right],$$

where the dynamic of the controlled Markov chain $(X^\alpha_{nh})_{n\in\mathbb{N}}$ reads:

$$\left\{\begin{array}{rcl} X^\alpha_0 & = & X_0 \\ X^\alpha_{h(n+1)} & = & b(nh, X^\alpha_{nh}, \alpha_{nh})h + \sigma(nh, X_{nh}, \alpha_{nh})\sqrt{h}\varepsilon_{n+1}, \end{array}\right.$$

with $(\varepsilon_n)_{n\in\mathbb{N}}$ a sequence of i.i.d. r.v. following the reduced Gaussian law $\mathcal{N}(0,1)$.
We recognize a MDP with infinite horizon. Under standards assumptions[3], one can show that the value function $\mathscr{V}_h$ is the unique fixed point of the following operator

$$\begin{array}{rcl} \mathcal{T} : \mathcal{C}^0(\mathbb{R}) & \longrightarrow & \mathcal{C}^0(\mathbb{R}) \\ v & \longmapsto & \left(x \longmapsto \sup\limits_{a\in\mathbb{A}} \left\{r(x) + e^{-\lambda h}\mathbb{E}\left[v(X^a_h)|X_0 = x\right]\right\}\right), \end{array}$$

[2] This scheme is often referred to as Bellman equation in the literature of stochastic control.
[3] in particular, assuming that $\mathcal{T}$ is contracting.

where we denote $r(x) := hf(x)$. *Using a Picard fixed point theorem argument, we can approximate the fixed point of* $\mathcal{T}$ *by considering the* N*-th iterate of* $\mathcal{T}$, *starting from the zero function. Hence* V_0, *defined in the scheme* (2.1.5) *below, converges toward the fixed point of* $\mathcal{T}$ *when* N *goes to infinity.*

$$\begin{cases} V_N &= 0 \\ V_n(x) &= \sup_{a\in\mathbb{A}} \left\{ r(x) + e^{-\lambda h}\mathbb{E}\left[V_{n+1}(X_h^a)|X_0 = x\right] \right\}, \quad \textit{for all state } x \end{cases}$$

We finally recognize the backward scheme (1.1.3), *with* $\beta = e^{-\lambda h}$, $g = 0$, $f = r$.

Remark 1.1.2. *There exist more RL algorithms such as the Temporal Difference Learning and the Deep Q-learning, to solve control problem with infinite horizon, for which the value function is fixed point of an operator. We refer to [SB98] for an exhaustive introductions to them, [MKSR15] for a concrete application of Deep Q-learning to learn how to play ATARI at a better level than the best human players, and [Sil+16], [Sil+17] for a concrete application of Monte Carlo tree search on the game of Go, to learn how to manage with a large control space, and manage to play better than professional Go players.*

The problem (1.1.3) is theoretically easier to solve than (1.1.2) since the infimum is now taken over $\mathbb{A}$ which is a subset of the finite-dimensional space $\mathbb{R}^q$, and moreover we have the nice characterization (1.1.4) of the optimal control that is associated to (1.1.3). Unfortunately, in most of the concrete applications, we still meet two difficulties when using the scheme (1.1.3):

PB1 the conditional expectation P^a, that appears at each time step to compute Q_n in (1.1.3), may not have any closed-form formula and need to be approximated.

PB2 the infimum which appears in the definition of V_n in (1.1.3) may not admit any closed-form formula, and the approximation of the latter[4] may be time-consuming when the control space is large.

1.1.1.2 Algorithms available in the literature to solve the Bellman equation

In the first part of the thesis, we decided to investigate the quantization and regression based methods to solve the Bellman equation (1.1.3).

Regress-Now on a basis of functions The regress-now method to solve MDPs consists in the three steps procedure:

(i) Randomize the control, i.e. simulating the control as an exogenous process, so that the latter becomes a component of the state variable

(ii) Regress the value function at time $n+1$ on a basis functions at time n to estimate the Q-value at time n.

(iii) Optimize the Q-value at time n to estimate the value function at time n.

We refer to [KLP14], [CL10] and the recent paper [LM18] for more details on the regress-now method to solve control problems.

One of the main difficulty with the regress-now method is the choice of the basis functions: when the dimension of the control problem is low, usual basis functions (such as the polynomial family) works well; but when the dimension is getting high, it becomes more difficult or almost impossible to find good basis functions to regress the value functions on without any over-fitting or under-fitting. An idea, developped in [BSST18], consists in adapting the basis functions $\mathcal{B}_n$ on which we regress V_{n+1}, the value function at time $n+1$, by adding the functions that looks like V_{n+1} in $\mathcal{B}_n$. Doing so, the estimate of the conditional expectation is better in the case where the MDP admits a "continuity" of the value function w.r.t. time, i.e. the MDP is a reduction of a time-continuous control problem on which the value function is continuous w.r.t. the time component.

Regress-Later The regress-later method consists in the following three steps procedure:

[4] using optimization algorithms such as the gradient descent-type methods.

(i) **interpolate** the value function at time $n+1$ on basis functions

(ii) regress the interpolation [5], and estimate the Q-value at time n.

(iii) Optimize the Q-value at time n to get an estimate of the value function at time n

We refer to [BP17] for a detailed presentation of the regress-later method applied to solve appliquée MDPs. Note that this method is particularly efficient in the (not so rare) case where there are some closed-form formula for the computation of the conditional expectation of the basis functions. In this case, the regression step is almost instantaneous, and the agent does not have any error coming from the regression step to look after, i.e. make sure that the error does not blow up. In the case where there is no closed-form formula for the computation of the conditional expectations of the chosen basis functions, one can proceed to an approximation of the latter using e.g. quantization-based methods. Regress-later is more recent than regress-now, and regularly outperforms the latter. We refer to [Ala+19] for a comparison of regress-later and regress-now algorithms on a concrete example of micro-grid management, in which one can see that regress-later seems to give better estimates of the optimal strategy.

Motivation

It is surprising to see that most of the algorithms available in the literature for control problems are based on global regression methods. When dimension is high (more than 10), global regression is probably the only tool that numerically proved to work well; but when dimension is low, one should also consider the local regression methods to build algorithms. We refer to [BKS10] for some details on the local regression for MDPs.
Also, quantization methods appears to be very well-suited to solve control problems, as shown in [PPP04a].

In the first part of the thesis, we want to mix the local regression and quantization techniques; which seems to be adapted to some control problems where the dimension is not too high[6] and where the noise can be quantized.

1.1.1.3 Contributions (Chapter 4)

We propose the *Qknn* algorithm, based on Quantization of the exogenous noise $\varepsilon_n, n = 0, \dots, N-1$ and local regression ($k$nearest neighbors algorithm) to approximate the conditional expectations. As we can see in the pseudo-code of Qknn, written in Algorithm 1.1: at each time step n, the agent chooses a finite family of points called "training points" on which to estimate the value function at time n. The computation of the conditional expectation is then done by quantizing the exogenous noise, i.e. approximating the noise r.v. ε_{n+1} by a discrete r.v. $\hat{\varepsilon}_{n+1}$ which can take a finite number of state $e_1, \dots, e_\ell$ with probability $p_1, \dots, p_\ell$. $\hat{\varepsilon}_{n+1}$ shall be chosen such that the $\mathbb{L}^2$ norm of the difference between ε_{n+1} and $\hat{\varepsilon}_{n+1}$ is small enough[7].

We refer to Section 3.4.2 page 76 in Chapter 3 for a detailed introduction of Qknn algorithm. In this chapter, we also derive a consistence result for Qknn (see Theorem 3.4.1 page 78). We present below a simplified version of the latter:

Theorem 1.1.1. *Take $K = M^{2+d}$ points for the quantification of the exogenous noise ε_n, $n = 1, \dots, N$. Under some regularity assumptions, there exists a constant $C > 0$ such that for all $n = 0, ..., N-1$,*

$$\|\hat{V}_n^Q(X_n) - V_n(X_n)\|_2 \;\leq\; C \sum_{k=n+1}^{N} \left(\varepsilon_k^{proj} + \varepsilon_k^Q\right) + \mathcal{O}\left(\frac{1}{M^{1/d}}\right),$$

when $M \to +\infty$, and where $\varepsilon_k^Q := \|\hat{\varepsilon}_k - \varepsilon_k\|_2$ is a quantization error, and

$$\varepsilon_n^{proj} := \sup_{a \in A} \|\mathrm{Proj}_{n+1}\left(F(X_n, a, \hat{\varepsilon}_n)\right) - F(X_n, a, \hat{\varepsilon}_n)\|_2$$

[5] and not the value function at time $n+1$!

[6] The algorithm has been successfully tested on the market-making problem, introduced in Section 1.1.2, which is of dimension 12.

[7] We refer e.g. to [GL00] for a proof of existence of such a r.v.

Algorithm 1.1 Qknn Algorithm

Input:

- Γ_k, $k = 0, \dots, N$: grids of training points in $\mathbb{R}^d$,
- $\Gamma = \{e_1, \dots, e_L\}$, $(p_\ell)_{1 \le \ell \le L}$: the L-optimal grid for the quantization of the i.i.d. exogenous noise $(\varepsilon_i)_{i=0}^{N-1}$, and its associated weights.

1: Assign N: $\hat{V}^{\mathrm{Q}}_{\Delta t}(t_N, z) = g(z), \quad \forall z \in \Gamma_N$.
2: **for** $n = N-1, \dots, 0$ **do**
3: Estimate the value function at time n:

$$\widehat{V}^{\mathrm{Q}}_{\Delta t}(t_n, z) = \max_{a \in A} \left[f(z,a)\,\Delta t + \sum_{\ell=1}^{L} p_\ell \widehat{V}^{\mathrm{Q}}_{\Delta t}\Big(t_{n+1}, \mathrm{Proj}_{\Gamma_{n+1}}\big(G_{\Delta t}(z,a,e_\ell)\big)\Big)\right], \quad \forall z \in \Gamma_n.$$

4: **end for**

Output:

– $\big(\hat{\alpha}(t_n, z)\big)_{z \in \Gamma_n, 0 \le n \le N-1}$: estimate of the optimal strategy;
– $\big(\hat{V}^{\mathrm{Q}}_{\Delta t}(0, z)\big)_{z \in \Gamma_0}$: estimate of the value function;

a projection error when decision a is taken at time n.

Theorem 1.1.1 states that the $\mathbb{L}^2$ norm of the difference between the Qknn-estimate of the value function and the value function is bounded by the quantization and the projection errors made at each time step $n = 0, \dots, N-1$. These two errors cancel themselves respectively when the number of points taken for the quantization and the number of points chosen for the grids go to infinity.

1.1.2 Application of the MDPs to market-making (Chapter 3)

In Chapter 3, we are interested in solving a market-making problem. We solve it theoretically by rewriting it as a MDP, and numerically using *Qknn*. We introduce below the market-making problem, and present our theoretical and numerical contributions.

1.1.2.1 Presentation of the market-making control problem

We consider a financial instrument which can be traded in a financial market through an order book, which is an electronic list of buy and sell orders available at the different price levels. All the participants in the market observe the order book and can at any time:

- buy or sell the instrument at the best price (we refer to this order as "market order")
- propose to buy or sell the instrument at certain prices (we refer to this order as "limit order")
- cancel their limit orders before the latter are executed (we refer to this order as "cancel order")

A market-maker is a participant in the market that can emit some limit orders in the order book, and cancel her orders. The main job of the market-maker is to liquefy the market by emitting buy and sell limit orders at different prices. A strategy followed by the market-maker, which is referred to as market-making strategy, consists in emitting or canceling buy and sell limit orders, while maximizing a certain quantity such as the wealth of the market-maker.

Remark 1.1.3. *One must differentiate limit and market orders. A participant* proposes *to buy or sell an instrument at a certain price when emitting limit orders, whereas she buys or sells it at the best price when emitting a market order. Unlike the market orders, there is no transaction fees for limit orders, since they liquidate the market.*

Model et mathematical representation of the problem

Let us take $K \geq 1$. We represent the order book by its K first limits on the ask and bid sides, as proposed in [Abe+16]. We denote pa_t the *best-ask* at time t, which is the cheapest price a participant is willing to sell an instrument at time t, and by pb_t the *best-bid* at time t, which is the most expensive price a participant is willing to buy an instrument at time t. The order book is then represented as the couple $(\underline{a}_t, \underline{b}_t) = (a_t^1, ..., a_t^K, b_t^1, \dots, b_t^K)$ where

- a_t^i is the number of sell limit orders i ticks away from pb_t.
- b_t^i is the number of buy limit orders i ticks away from pa_t.

The order book can receive market, limit or cancel orders at any time, which modify the value of $(\underline{a}_t, \underline{b}_t)$. We model the arrivals of these events as point processes.

We consider a market-maker that observes the order book, and can choose at any time to send limit or cancel orders in the order book. We denote by $\mathbb{A}$ the set of all the admissible market-making strategies, which are the $\mathbb{F}$-predictable processes, where $\mathbb{F}$ is the filtration generated by the order arrivals processes, which represent the positions of the market-maker on the order book, i.e. where are the limit orders of the market maker in the order book. In Chapter 3, we denote by Z the "controlled order book", which is the controlled process that gathers up $(\underline{a}_t, \underline{b}_t) = (a_t^1, ..., a_t^K, b_t^1, \dots, b_t^K)$ and the positions of the market-maker. Z is a process that lies on a discrete space that we call E, since the prices the participants can buy or sell the instrument are organized by levels.

The goal of the market-maker is to find the optimal strategy that maximizes the following quantity

$$V(t,z) = \sup_{\alpha \in \mathbb{A}} \mathbb{E}_{t,z}^{\alpha} \left[\int_t^T f\big(\alpha_s, Z_s\big)\, \mathrm{d}s + g\big(Z_T\big) \right], \quad (t,z) \in [0,T] \times E,$$

where g is the terminal reward (for example the terminal wealth of the market-maker plus a penalization term of the terminal inventory); and where f is the instantaneous reward (for example the risk aversion of the market-maker).

1.1.2.2 Contributions

Theoretical contributions:

In the Section 3.3.2 of Chapter 3, we show[8] that the time-continuous market-making control problem (where the controlled process is Z and the value function is V) can be reformulated as a MDP. From the reformulation, we can derive[9] a characterization of V as the unique fixed-point of the operator $\mathcal{T}$ defined as follows:

$$\begin{array}{rccc} \mathcal{T}: & \mathcal{C}^0([0,T] \times E) & \longrightarrow & \mathcal{C}^0([0,T] \times E) \\ & v & \longmapsto & (t,x) \mapsto \sup\limits_{a \in A_z} \Big\{ r(t,z,a) + \mathbb{E}\left[v(T_{n+1}, Z_{n+1}) | T_n = t, Z_n = z, a \right] \Big\}, \end{array}$$

where r is the instantaneous reward defined as:

$$r(t,z,a) := -f\big(z,a\big) e^{-\lambda(z)(T-t)} (T-t) \mathbf{1}_{t>T} + f\big(z,a\big) \frac{1 - e^{-\lambda(z)(T-t)}}{\lambda(z)} + e^{-\lambda(z)(T-t)} g(z) \mathbf{1}_{t \leq T},$$

with λ being the intensity of the order arrivals in the order book. More precisely, we show in Section 3.3 of Chapter 3 the following theorem:

Theorem 1.1.2. *$\mathcal{T}$ a unique fixed point v, and we have $v = V$.*

Numerical contributions:

[8] see Proposition 3.3.1 page 70
[9] see Theorem 3.3.1 page 72

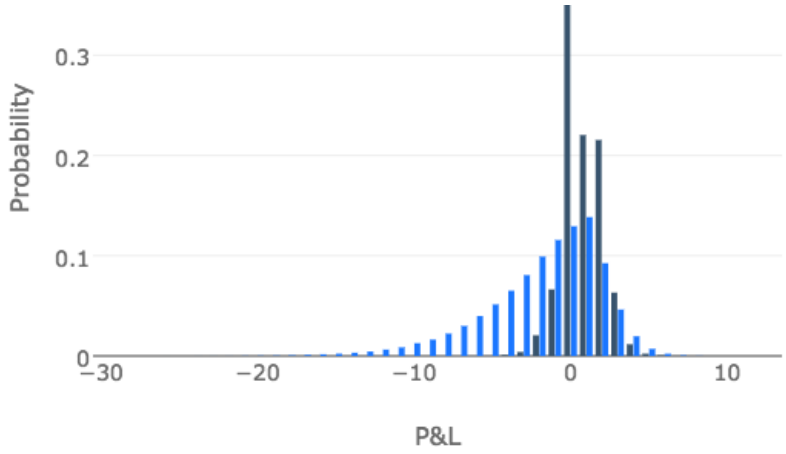


Figure 1.1 – Histogram of the terminal wealth of the market-maker when the latter follows a naive strategy (light blue), and the Qknn-estimated optimal strategy (dark blue). Notice that the wealth is better when following the Qknn-estimated optimal strategy, by reducing the losses.

Remark 1.1.4. *The state space of the market-making is discrete, sparse, large and lies on a space of dimension 12. The regress-now and regress-later methods presented above do not seem to be adapted for this problem since it is hard to choose good basis functions. Local regression that does not rely on basis functions looks to be a wiser choice.*

In Section 3.4.4 of Chapter 3, we present the numerical results when using the Qknn algorithm on the market-making problem. Qknn seems to be efficient to estimate the value function. In particular, we observe in Figure 1.1 that the estimate of the optimal strategy does better than a naive market-making strategy, which should be seen as a good benchmark. In Figure 1.1, we plot the histogram of the terminal wealth of the market-maker, when the latter follows the Qknn-estimated optimal strategy and the naive strategy: observe that the losses are reduced when using the optimal strategy. Other plots that illustrate the good behavior of Qknn in this problem are available in Section 3.4.4 page 82.

1.1.3 Application of the MDPs to McKean-Vlasov type control problems (Chapter 4)

In Chapter 4, we propose a theoretical method based on Markovian embedding to reduce McKean-Vlasov control problems with partial information to standard finite-dimensional control problems. Numerical results are presented for different McKean-Vlasov problems, where regress-later, regress-now, Qknn and finite difference-based methods are tested and compared. We introduce below the McKean-Vlasov problem with partial information framework, and present our theoretical and numerical contributions.

1.1.3.1 Presentation of McKean-Vlasov type control problems

We consider a McKean-Vlasov type control problem (MKV) with partial information and common noise: given two independent Brownian motions B and W^0, defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, let us consider a stochastic process X which dynamic reads:

$$\mathrm{d}X_s = b(X_s, \mathbb{P}_{X_s}^{W^0}, \alpha_s)\,\mathrm{d}s + \sigma(X_s, \mathbb{P}_{X_s}^{W^0}, \alpha_s)\,\mathrm{d}B_s + \sigma_0(X_s, \mathbb{P}_{X_s}^{W^0}, \alpha_s)\,\mathrm{d}W_s^0, \quad X_0 = x_0 \in \mathbb{R}^n,$$

where $\mathbb{P}_{X_s}^{W^0}$ is the distribution of X_s conditioned by $\mathcal{F}_s^0$ where $\mathbb{F}^0 = (\mathcal{F}_t^0)_t$ is the filtration generated by W^0, and where the control α is a $\mathbb{F}^0$-progressive process which takes its values on a Polish space A. The measurability condition on the control means that the agent partially observe X, i.e. she only observes the common noise.

We make the standard Lipschitz condition on the coefficients $b(x,\mu,a)$, $\sigma(x,\mu,a)$, $\sigma_0(x,\mu,a)$ with respect to (x,μ) in $\mathbb{R}^n \times \mathcal{P}_2(\mathbb{R}^n)$, uniformly in $a \in A$, where $\mathcal{P}_2(\mathbb{R}^n)$ is the set of all probability measures on $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n))$ with a finite second-order moment, endowed with the 2-Wasserstein metric $\mathcal{W}_2$. This ensures the well-posedness of the controlled MKV stochastic differential equation (SDE) (4.1.1). The cost functional, with finite horizon T associated to the control α is given by:

$$J(\alpha) = \mathbb{E}\Big[\int_0^T f(X_t, \mathbb{P}^{W^0}_{X_t}, \alpha_t)dt + g(X_T, \mathbb{P}^{W^0}_{X_T})\Big],$$

and the goal is to maximize it over all the admissible controls in $\mathcal{A}$:

$$V_0 := \sup_{\alpha\in\mathcal{A}} J(\alpha).$$

1.1.3.2 Contributions (Chapter 4)

Motivation The Chapter 4 has two main goals: the first one is to show how to reduce McKean-Vlasov type control problems with partial information as standard and finite-dimensional control problem. The second goal is to test and compare some algorithms, including *Qknn*, on various McKean-Vlasov type examples.

Question 1: How can we reduce the MKV control problems to standard control problems?

Answer to Question 1:

Let us make the following two assumptions:

Assumptions on the dependence of the coefficients w.r.t. x. We consider that the drift and the volatility in X are linear w.r.t. the state variable x, i.e.

$$\left\{\begin{array}{rcl} b(x,\mu,a) & = & b_0(\mu,a) + b_1(\mu,a)x, \\ \sigma(x,\mu,a) & = & \vartheta_0(\mu,a) + \vartheta_1(\mu,a)x, \\ \sigma_0(x,\mu,a) & = & \gamma_0(\mu,a) + \gamma_1(\mu,a)x, \end{array}\right.$$

We also assume that the terminal and instantaneous rewards are polynomial w.r.t. x, i.e.

$$\begin{aligned} f(x,\mu,a) &= f_0(\mu,a) + \sum_{k=1}^p f_k(\mu,a)x^k, \\ g(x,\mu) &= g_0(\mu) + \sum_{k=1}^p g_k(\mu)x^k, \end{aligned}$$

with $p \geq 1$.

Assumptions about the dependence of the coefficients w.r.t. μ: We assume in the following that the coefficients depend on μ only through its p first moments, i.e.,

$$\left\{\begin{array}{lllll} b_0(\mu,a) = \bar{b}_0(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a), & & & b_1(\mu,a) = \bar{b}_1(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a) \\ \vartheta_0(\mu,a) = \bar{\vartheta}_0(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a), & & & \vartheta_1(\mu,a) = \bar{\vartheta}_1(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a) \\ \gamma_0(\mu,a) = \bar{\gamma}_0(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a), & & & \gamma_1(\mu,a) = \bar{\gamma}_1(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a) \\ f_k(\mu,a) = \bar{f}_k(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a), & & g_k(\mu) = \bar{g}_k(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p), & & k=0,\ldots,p, \end{array}\right.$$

where, for $\mu \in \mathcal{P}_p(\mathbb{R})$, we denote

$$\bar{\mu}_k = \int x^k \mu(dx), \quad k = 1,\ldots,p.$$

Let Y be the following process

$$Y_t^{(k)} = \mathbb{E}[X_t^k | W^0], \quad k = 1,\ldots,p.$$

ρ	RLMC	CR	Q	Bench
0.2	8.73	8.98	8.69	8.77
0.4	8.02	8.25	7.91	8.06
0.6	6.93	6.97	6.68	6.79
0.8	4.86	4.82	4.62	4.67

$c = 100$ and $\eta = 10$.

c	RLMC	CR	Qknn	Bench
1	7.88	7.87	7.86	7.88
5	8.22	8.23	8.21	8.23
25	9.69	9.76	9.61	9.62
50	11.08	11.27	10.94	10.97

$\rho = 0.5$ and $\eta = 100$.

Table 1.1 – Estimates of the value function for the systemic risk problem for different parameters; using regress-later (RLMC),regress-now (CR), Qknn, and a finite difference based algorithm. The most accurate algorithm is the one providing the smallest values: Qknn is never beaten!

For the sake of simplicity in notations, we assume $n = 1$, thus obtaining the following dynamic for Y by Itô formula:

$$\begin{aligned} dY_t^{(k)} &= B_k(Y_t^{(1)}, Y_t^{(2)}, \dots, Y_t^{(p)}, \alpha_t)dt + \Sigma_k(Y_t^{(1)}, Y_t^{(2)}, \dots, Y_t^{(p)}, \alpha_t)dW_t^0, \\ Y_0^{(k)} &= x_0^k, \quad k = 1, \dots, p, \end{aligned}$$

where we denote $y_0 = 1$, $y_{-1} = 0$,

$$\begin{aligned} B_k(y_1, y_2, \dots, y_p, a) &= k\bar{b}_0(y_1, \dots, y_p, a)y_{k-1} + k\bar{b}_1(y_1, \dots, y_p, a)y_k \\ &\quad + \frac{k(k-1)}{2}(\bar{\vartheta}_0(y_1, \dots, y_p, a))^2 y_{k-2} + \frac{k(k-1)}{2}(\bar{\vartheta}_1(y_1, \dots, y_p, a))^2 y_k \\ &\quad + k(k-1)\bar{\vartheta}_0(y_1, \dots, y_p, a)\bar{\vartheta}_1(y_1, \dots, y_p, a)y_{k-1} \\ &\quad + \frac{k(k-1)}{2}(\bar{\gamma}_0(y_1, \dots, y_p, a))^2 y_{k-2} + \frac{k(k-1)}{2}(\bar{\gamma}_1(y_1, \dots, y_p, a))^2 y_k \\ &\quad + k(k-1)\bar{\gamma}_0(y_1, \dots, y_p, a)\bar{\gamma}_1(y_1, \dots, y_p, a)y_{k-1}, \quad k = 1, \dots, p, \\ \Sigma_k(y_1, y_2, \dots, y_p, a) &= k\left(\bar{\gamma}_0(y_1, \dots, y_p, a)y_{k-1} + \bar{\gamma}_1(y_1, \dots, y_p, a)y_k\right), \quad k = 1, \dots, p, \end{aligned}$$

and the cost functional Y reads:

$$J(\alpha) = \mathbb{E}\Big[\int_0^T \bar{f}(Y_t^{(1)}, Y_t^{(2)}, \dots, Y_t^{(p)}, \alpha_t)dt + \bar{g}(Y_T^{(1)}, Y_T^{(2)}, \dots, Y_T^{(p)})\Big].$$

The McKean-Vlasov type controlled problem with partial information is then reduced to a finite-dimensional control problem where the controlled process is now $(Y^{(1)}, Y^{(2)}, \dots, Y^{(p)})$, which is an answer to Question 1.

Numerical applications to solve MKV controlled problem We refer to Section 4.4 of Chapter 4 for examples of McKean-Vlasov controlled problem with partial information which are numerically solved using *Qknn*, regress-now, regress-later, and methods based on finite differences. The numerical results illustrate the accuracy of *Qknn*.

We consider a portfolio liquidation problem, a portfolio selection problem (see Section 4.4.1.2 page 125); and a systematic risk problem (see Section 4.4.2 page 125). The numerical results are available respectively in 4.1, 4.2 et 4.3 page 128, 131 et 134. To illustrate the efficiency of Qknn, we decided to present here Table 1.1 which is a part of Table 4.3 that can be found in Section 4.4.2.2. In Table 1.1, we tested different algorithms using different parameters to solve the systemic risk problem. The algorithm which gives the smallest result is the most accurate one, and as we can see, Qknn always provides the best estimates.

1.2 Neural networks for MDPs

The framework in the second part of this thesis is the same as the one of the first part. We present briefly the framework below, and refer to Section 2.1.1.1 page 30 for more details.

Fixing the horizon N, we want to get an estimate of the value function

$$V_n(x) := \inf_{\alpha \in \mathcal{C}} \mathbb{E}\left[\sum_{k=n}^{N-1} f(X_k^\alpha, \alpha_k) + g(X_N^\alpha) \middle| X_n = x \right],$$

associated to the MDP $X^\alpha = (X_n^\alpha)_{n=0}^N$, which dynamic reads

$$\left\{ \begin{array}{rcl} X_0^\alpha & = & x_0 \in \mathbb{R}^d, \\ X_{n+1}^\alpha & = & F_n(X_n^\alpha, \alpha_n, \varepsilon_{n+1}), \quad \text{for } n = 0, \dots, N-1, \end{array} \right.$$

where $(\varepsilon_n)_{n=1}^N$ is a sequence of i.i.d. noise generating a filtration $\mathbb{F} = (\mathcal{F}_n)_{n=0}^N$; and where $\mathcal{C}$ is the set of admissible controls.
By dynamic programming principle, V_n, the value function at time n, for $n = 0, \dots, N$, is characterized as solution to the following backward equation[10]:

$$\left\{ \begin{array}{rcl} V_N(x) & = & g(x), \quad \forall x \in \mathcal{X}, \\ Q_n(x,a) & = & f(x,a) + P^a V_{n+1}(x), \quad \forall (x,a) \in \mathcal{X} \times \mathbb{A}, \\ V_n(x) & = & \inf_{a \in \mathbb{A}} Q_n(x,a), \quad \text{for } n = N-1, \dots, 0. \end{array} \right. \tag{1.2.1}$$

In the second part of this thesis, we propose to solve MDPs using neural networks. Neural networks being well-suited for high-dimension, we expect our new schemes to be efficient for high-dimensional control problems (i.e. $d \gg 1$). In the next section, we introduce neural networks and gradient descent-based methods, on which our algorithms rely to learn functions of interest.

1.2.1 Representation of functions by neural networks & training

Let $L \geq 1$. We define a neural network with L layers as the parameterized function Φ

$$x \in \mathcal{X} \longmapsto \Phi(z;\theta) \in \mathbb{R}^o,$$

with $o \in \mathbb{N}$, and $\theta \in \Theta \subset \mathbb{R}^p$ standing for the parameters of Φ, such that Φ is the composition of L "simple" parameterized functions[11]. More precisely, $\Phi = \Phi_L$, where Φ_L is defined by induction as follows:

$$\left\{ \begin{array}{rcl} \Phi_1(x) & = & \sigma(A_1 x + b_1) \\ \Phi_{n+1} & = & \sigma(A_{n+1}\Phi_n + b_n) \quad \text{for } 1 \leq n \leq L-1, \end{array} \right.$$

where σ is a monotonic function[12]; where $A_1, \dots, A_L$ are matrices, and $b_1, \dots, b_L$ vectors, whose dimensions are such that the matrix multiplications and the sums of vectors are well-defined; and where, for all q-dimensional vector y, we used the notation $\sigma(y) = (\sigma(y_1), \dots, \sigma(y_q))^\top$.

The matrices $A_1, \dots, A_L$ are referred to as the *weights* of the neural networks, the $b_1, \dots, b_L$ as their *bias*, and the weights and bias are the set of parameters for the neural network that we denote by θ above.
Generally speaking, after representing a desired function by a neural network Φ, we want to learn the optimal parameters such that Φ approximates the latter. In the second part of this thesis, we need to approximate functions that can minimize objective functions. We will introduce the gradient descent-based algorithms that are well-designed to minimize such objective functions in the next section.

Gradient-descent algorithms

Let us consider the following situation where one has to optimize w.r.t. θ an expectation of the form

$$\mathscr{J}(\theta) = \mathbb{E}[G(X;\theta)], \quad \text{with } X \sim \mu, \tag{1.2.2}$$

[10] Scheme (1.2.1) is often referred to as the Bellman equation in the literature of the stochastic control.
[11] These simple functions are called "layers" in the literature of deep learning.
[12] σ is often called *activation function* in the deep learning literature. Classical examples of activation functions are: the ReLu function: $x \mapsto \max(x, 0)$,
the ELU function: $x \mapsto \left\{ \begin{array}{ll} \exp(x) - 1 & \text{if } x \leq 0 \\ x & \text{otherwise} \end{array} \right.$, or the sigmoid function: $x \mapsto \frac{1}{1+\exp(-x)}$.

where μ is a general probability measure, and where G is a differentiable function such that derivative and expectation symbols are interchangeable. As shown in Example 1.2.1, this situation is met when we train the neural networks to learn the optimal control and the value function of any control problem.

Example 1.2.1. *Let us propose a representation of the optimal control at time n by a neural network $A(.;\beta_n)$ and the value function at time n by a neural network $\Phi(.;\theta_n)$:*

Relying on scheme (1.2.1), the optimal parameter β_n^* that approximates the optimal control n is such that:

$$\beta_n^* = \underset{\beta}{\operatorname{argmin}}\, \mathbb{E}\left[V_{n+1}(X_{n+1}^{\beta}) + f(X_n, A(X_n;\beta))\right],$$

and the optimal parameter θ_n^* whose $\Phi(.;\theta^*)$ neural network approximates V_n, the value function at time n, is such that:

$$\theta_n^* = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}\left[\left(V_{n+1}(X_{n+1}^{\beta_n^*}) + f(X_n, A(X_n;\beta_n^*)) - \Phi(X_n;\theta)\right)^2\right],$$

where, we denoted by X_{n+1}^{β} the state at time $n+1$ when the feedback control $A(.;\beta)$ is taken at time n. These two problems are of the same form as (1.2.2).

Gradient-descent The naive method to optimize $\mathscr{J}$ is to apply a deterministic gradient-descent-type algorithm, for which a basic pseudo-code is written in Algorithm 1.2

Algorithm 1.2 Gradient-descent algorithm

Input:
- Learning parameter η;
- Threshold $\varepsilon \ll 1$;
- Training set $\{X_m, 1 \leq m \leq M\}$ with $M \gg 1$;
- $\theta \in \mathbb{R}^q$;

1: **while** $|\hat{\partial}_\theta \mathscr{J}(\theta)| > \varepsilon$ **do**
2: Compute

$$\hat{\partial}_\theta \mathscr{J}(\theta) := \hat{\mathbb{E}}[\partial_\theta V(X;\theta)],$$

where $\hat{\mathbb{E}}$ is the expectation under the empirical measure $\frac{1}{M}\sum_{m=1}^{M} \delta_{X_m}$;
3: ▷ $\hat{\partial}_\theta \mathscr{J}(\theta)$ is an estimate of the derivative of $\mathscr{J}$ at point θ.
4: Re-assign $\theta := \theta - \eta\, \hat{\partial}_\theta \mathscr{J}(\theta)$;
5: **end while**

Output: optimal parameter θ^*;

Estimate $\hat{\partial}_\theta \mathscr{J}(\theta)$ gives the direction to take to optimize θ by reducing the gradient, but its computation is heavy and time-consuming, which is why Algorithm 1.2 is intractable. In order to alleviate the computational time, one should consider a more stochastic estimate of the derivative, as proposed in the stochastic gradient descent (SGD) algorithms. We wrote a pseudo-code of simple SGD-based method in Algorithm 1.3. The SGD is much faster since the stochastic estimate of the derivative $\hat{\partial}_\theta \mathscr{J}(\theta)$ is computed almost instantaneously[13].

The stochastic gradient descent algorithms are designed to optimize $\mathscr{J}$, and the success of neural networks to learn functions of interest arguably comes from them. The most recent algorithms used to optimize $\mathscr{J}$ are based on SGD. Adagrad, Adadelta, RMSprop and Adam, are recent algorithms that numerically proved to improve the SGD, mainly by choosing wisely the learning rate in an adaptive way.

Remark 1.2.1. *In this thesis, we always used Adam algorithm, natively implemented in TensorFlow, to train neural networks.*

[13] We rely here on the fact that it takes more or less the same time to compute the output of a neural network or its derivatives at one point, when using reverse backpropagation to compute the derivative. All the deep learning languages such as Tensorflow or PyTorch use this method to compute the derivatives of neural networks.

Algorithm 1.3 Stochastic Gradient Descent (SGD)

Input:
– Learning rate η;
– Threshold $\varepsilon \ll 1$;
– Number of iterations $M \gg 1$;
– Initial parameter $\theta \in \mathbb{R}^q$;

1: **for** $m = 0, \dots, M$ **do**
2: Draw $X \sim \mu$;
3: Compute

$$\hat{\partial}_\theta \mathscr{J}(\theta) := \partial_\theta V(X;\theta);$$

▷ $\hat{\partial}_\theta \mathscr{J}(\theta)$ is an estimate of the derivative of $\mathscr{J}$ at point θ.

4: Re-assign $\theta := \theta - \eta\, \hat{\partial}_\theta \mathscr{J}(\theta)$;
5: **end for**

Output: optimal parameter θ^*;

1.2.2 Neural networks for MDPs

In this section, we propose a short review of the RL literature to solve (1.1.2), relying on neural networks to learn the optimal controls and the value functions at time $n = 0, \dots, N-1$.

There are two main families of algorithms to solve (1.1.1) using neural network. The first one, which we refer to as the "direct methods", contains the algorithms that only represent the optimal control by neural networks, and highly relies on the recent improvement made to learn optimal parameters on very deep neural networks, i.e. a neural networks with a large number of layers, (see Section 1.2.1). The second family, which we refer to as "Actor-critic" in the sequel, contains all the algorithms that represent the optimal control and the Q-value function by neural networks; and rely on the scheme (1.1.3) to estimate the value function.

1.2.2.1 Direct methods

The main idea for the algorithms in this family is to represent the controls α_n at time n, for $n = 0, \dots, N-1$; and then regard $J(\alpha)$ as a parameterized expectation (with lot of parameters!) that remains to be optimized, as shown in Section 1.2.1. To be more precise, for all $n = 0, \dots, N-1$, we represent the control α_n of the process (X_n^α) at time n, by a neural network

$$\alpha_n = A(.;\beta_n), \tag{1.2.3}$$

where $\beta_n \in \mathbb{R}^q$ is the parameter. We then plug these representations in (2.1.1), so that the control problem (1.1.2) reads:

$$V_0(x_0) = \inf_{\beta_0,\dots,\beta_n \in \mathbb{R}^q} \mathbb{E}\left[\sum_{n=0}^{N-1} f(X_n, A_n(X_n,\beta_n)) + g(X_N)\right], \tag{1.2.4}$$

where the dynamic of (X_n) is controlled using the neural network representation of the controls (1.2.3). Note that the process should be written (X_n^β), but we omit the superscript β for the sake of simplicity. The r.h.s. of (1.2.4) is a parameterized expectation that can be solved, as shown in Section 1.2.1.

Comments on the direct methods

✓ There is no representation of the value function by neural network, which implies that there is no approximation and estimation errors to look after coming from the estimation of the value functions at time $n = N-1, \dots, 0$.

✗ The method is very unstable, especially when N is getting large and/or the optimal control is a complex function. Indeed: if we represent each optimal control at time n using neural networks

with 2 layers, then $\mathscr{J}$ is a large neural network with $2 \times N$ layers that is hard to train. Deep neural networks remain very difficult to train, even for the most recent SGD-based algorithms; but tricks such as highway and resnet architectures are available to deal with them.

1.2.2.2 Actor-critic

In the second family of algorithms, we represent the Q-value and the control by neural networks, and we rely on (1.1.3) to derive estimate of the value functions at time $n = N-1, \dots, 0$. For all $n = 0, \dots, N-1$, we represent the control by a neural network as follows

$$\alpha_n = A(X_n; \beta_n),$$

and represent the Q-value by a neural network as follows

$$Q(X_n, \alpha_n) = Q(X_n, A(X_n; \beta_n); \theta_n).$$

The learning of the optimal parameters β_n^* and θ_n^*, for $n = 0, \dots, N-1$ is performed backward, as shown in the pseudo-code Algorithm 1.4. One can see that the optimal parameter for the Q-value is learnt at the same time as the optimal parameter for the optimal control, i.e. one cannot learn one without learning the other![14]

Algorithm 1.4 Actor-critic

Input:
– Learning rate η;
– Threshold $\varepsilon > 0$;
– Initial parameter $\theta_n, \beta_n \in \mathbb{R}^q$, for $n = 0, \dots, N-1$;

1: Assign $V_N = g$;
2: **for** $n = N-1, \dots, 0$ **do**
3: **while** $\max(\Delta\theta_n, \Delta\beta_n) > \varepsilon$ **do**
4: Minimize w.r.t. θ_n the quantity:

$$\mathbb{E}\left[(f(X_n, A(X_n; \beta_n)) + Q(X_{n+1}, A(X_{n+1}; \beta_{n+1}^*); \theta_{n+1}^*) - Q(X_n, A(X_n; \beta_n); \theta_n))^2\right],$$

and re-assign θ_n as its argmin ;

5: Re-assign $\Delta\theta_n$ as the difference between the new and the old value of θ_n;
6: Minimize w.r.t. β_n the quantity:

$$\mathbb{E}\left[f(X_n, A(X_n; \beta_n)) + Q(X_{n+1}, A(X_{n+1}; \beta_{n+1}^*); \theta_n)\right],$$

and re-assign β_n as its argmin ;

7: Re-assign $\Delta\beta_n$ as the difference between the new and the old value of β_n;
8: **end while**
9: **end for**

Output: optimal parameters $(\theta_n^*)_{n=0}^{N-1}, (\beta_n^*)_{n=0}^{N-1}$ for the approximation of the Q-value and the optimal control at time $n = 0, \dots, N-1$;

Comments on the Actor-critic methods

✓ The training of the neural networks for the optimal controls and the value functions at time $n = 0, \dots, N-1$ is done backward; and moreover we train the parameters of the optimal control and the value function at time n by using the trained one at time $n+1$.

✓ If the MDP comes from a time-continuous control problem, where the continuity of the value function and the optimal control w.r.t. the time holds, then one can improve the quality of the

[14] As we will see later, an important feature of the new algorithm that we design in the second part of the thesis is that the training of the neural networks for the optimal control and the Q-value are uncoupled.

training by using the *pre-training* trick[15]: i.e. initialize the parameters of the control and the Q-value at time n with the optimal ones that are already computed at time $n+1$, and reduce the learning rate η so that the training will lightly and perfectly optimize the parameters.

✗ Relying on an approximation of the Q-value to estimate the value function implies that one has to deal with estimation and approximation errors for the Q-value approximation at time $n = N-1, \dots, 0$. One shall make sure that this error does not blow up when running the backward procedure of the algorithms.

1.2.3 Contributions (Chapters 5 and 6)

Motivations: As we just saw, there exist already several algorithms to solve control problems. As every algorithms, they have drawbacks and we propose to fix some, in particular we want to improve the fact that the training procedure to learn the optimal parameters of the control and the Q-value must be done at the same time for the Actor-critic algorithms; and that the dynamic programming principle is ignored in the direct methods.

Question 2a: How can we improve the direct methods by relying on the dynamic programming principle?

Question 2b: How can we improve the Actor-critic methods by decoupling the training of the optimal parameters for the optimal control and the Q-value at time $n = N-1, \dots, 0$?

We propose new algorithms in the second part of this thesis: some belong to the direct methods family and answer to Question 2a, and others to the actor-critic family, and answer to Question 2b.

Algorithms to answer Question 2a We propose NNcontPI and ClassifPI algorithms to answer to Question 1a. We shortly present algorithm NNcontPI here, and refer to page 189 of Chapter 6 for the pseudo-code of ClassifPI algorithm.
As it can be seen in Algorithm 1.5, the main idea behind NNcontPI is to represent the optimal control at time n by a neural network $A(.;\beta_n)$, and then learn the optimal parameter β_n^* in a backward way, relying on the dynamic programming principle and "training" measures $(\mu_n)_{n=0}^{N-1}$: assume that the optimal parameter β_k^* is already learnt at time $k = n+1, \dots, N-1$, we have

$$\hat{\beta}_n \in \underset{\beta\in\mathbb{R}^q}{\operatorname{argmin}}\, \mathbb{E}\left[f\big(X_n, A(X_n;\beta)\big) + \sum_{k=n+1}^{N-1} f\big(X_k^\beta, \hat{a}_k\big(X_k^\beta\big)\big) + g\big(X_N^\beta\big) \right]$$

where X_n is drawn under the training measure μ_n and where $\big(X_k^\beta\big)_{k=n+1}^N$ is defined by induction:

$$\begin{cases} X_n & \sim & \mu_n, \\ X_{n+1}^\beta & = & F\big(X_n, A\big(X_n;\beta\big), \varepsilon_{n+1}\big), \\ X_{k+1}^\beta & = & F\big(X_k^\beta, \hat{a}_k\big(X_k^\beta;\beta\big), \varepsilon_{k+1}\big), \quad \text{for } k = n+1, \dots, N-1. \end{cases}$$

Comments on NNcontPI

✓ The training of the neural networks is done in a backward way; and the optimal control at time n is learnt based on the optimal parameters that have already been learnt at times $n+1, \dots, N-1$. A nice consequence of this feature is that one can use the pretraining trick to highly improve the time and quality of the training.

✓✗ The learning of the optimal parameter β_n at time n relies on the training measures $\mu_k, k = n, \dots, N-1$, chosen by the agent. In the case where the agent knows where to drive the process in order to maximize her utility function, then she can choose the training measures to be such as the density is high where the process is likely to go when optimally driven. In the case where the agent does not know where the optimally driven process is likely to go, she can rely on some

[15] The pre-training trick is also referred to as *transfer learning* trick in the literature.

bootstrapping methods, as explained in Section 4.3.3. The choice of the training measure might be a difficult task (especially when the dimension of the control space is high) when the agent has no idea of where to drive the process.

✗ The training of the neural network for the control at time n, using NNcontPI and ClassifPI, requires the re-simulation of the trajectories from time n to time N, as it can be seen in Algorithm 1.5; which implies that these methods are very volatile, especially when N is large, or when the rewards f and g are complex functions.

Algorithm 1.5 NNContPI

Input: Training measures $(\mu_n)_{n=0}^{N-1}$;

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

2: Compute

$$\hat{\beta}_n \in \underset{\beta\in\mathbb{R}^q}{\operatorname{argmin}}\, \mathbb{E}\left[f\big(X_n, A(X_n;\beta)\big) + \sum_{k=n+1}^{N-1} f\big(X_k^\beta, \hat{a}_k(X_k^\beta)\big) + g\big(X_N^\beta\big)\right]$$

where $X_n \sim \mu_n$ and where $\big(X_k^\beta\big)_{k=n+1}^{N}$ is defined by induction:

$$\begin{cases} X_{n+1}^\beta &= F\big(X_n, A(X_n;\beta), \varepsilon_{n+1}\big) \\ X_{k+1}^\beta &= F\big(X_k^\beta, \hat{a}_k(X_k^\beta;\beta), \varepsilon_{k+1}\big), \quad \text{for } k = n+1, \dots, N-1. \end{cases}$$

for $m = 1, \dots, M$;

3: Assign $\hat{a}_n = A(.;\hat{\beta}_n)$; ▷ $\hat{a}_n$ is an estimate of the optimal control at time n

4: **end for**

Output: $(\hat{a}_n)_{n=0}^{N-1}$, estimates of the optimal controls at time n, for $n = 0, \dots, N-1$;

We present in Chapter 5 consistence results for our algorithm NNcontPI, w.r.t. the size of the training and the considered neural network space.

Theorem 1.2.1 (Convergence of NNContPI)**.** *Assume that there exist some feedback optimal controls $(a_k^{\text{opt}})_{k=n}^{N-1}$, and denote by V_n the value functions at time $n = 0, \dots, N$. Let $X_n \sim \mu$. We then have, as $M \to \infty$*[16]*:*

$$\mathbb{E}\big[\hat{V}_n^M(X_n) - V_n(X_n)\big] = \mathcal{O}\left(\frac{\rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}}{\sqrt{M}} + \sup_{n\le k\le N-1} \inf_{A\in\mathcal{A}_M} \mathbb{E}\left[|A(X_k) - a_k^{\text{opt}}(X_k)|\right]\right),$$

Moreover, we have when $M \to \infty$[17]*:*

$$\mathbb{E}_M\big[\hat{V}_n^M(X_n) - V_n(X_n)\big] = \mathcal{O}_{\mathbb{P}}\left(\rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}\sqrt{\frac{\log(M)}{M}} + \sup_{n\le k\le N-1} \inf_{A\in\mathcal{A}_M} \mathbb{E}\left[|A(X_k) - a_k^{\text{opt}}(X_k)|\right]\right),$$

where $\mathbb{E}_M$ is the expectation conditioned by the training set used to estimate the optimal controls $(\hat{a}_k^M)_{n\le k\le N-1}$.

[16] The notation $x_M = \mathcal{O}(y_M)$ when $M \to \infty$, means that the ratio $|x_M|/|y_M|$ is bounded when M goes to infinity.

[17] The notation $x_M = \mathcal{O}_{\mathbb{P}}(y_M)$ when $M \to \infty$ means that there exists $c > 0$ such as $\mathbb{P}\big(|x_M| > c|y_M|\big) \to 0$ when M goes to infinity.

Since $\hat{V}_n^M \geq V_n$ always holds, Theorem 1.2.1 states that the value function at time n, computed using the estimates of the optimal controls at times $k = n, \dots, N-1$, is close to the value function at time n; and the mean of the difference between these two quantities goes to 0 when the size of the training set goes to infinity and the space of considered neural networks gets large. This result is interesting in particular because it takes into account the size of the training set used to run the SGD in the training procedure.

The algorithms NNcontPI and ClassifPI are tested numerically in several examples in Chapter 6: ClassifPI provided good results in the Valuation of Energy Storage problem, presented in Section 6.3.3 page 200, but is beaten by Qknn which is well-designed for this problem (see Table 6.2 page 203).

Algorithmes to answer Question 2b We propose the algorithms Hybrid-Now, Hybrid-Later and Classif-Hybrid to answer to Question 2b. We refer to Algorithms 6.4 and 6.6, respectively, pages 192 and 209 for the pseudo-codes of Hybrid-Later and Classif-Hybrid, and choose to only present here the Hybrid-Now algorithm, for which the pseudo-code is written in Algorithm 1.6.
In the Hybrid-Now algorithm, we represent the optimal control and the value function[18] at time n by two (independent) neural networks. As we can see from Algorithm 1.6, the main idea in Hybrid-Now for the training is to learn first the optimal control at time n by minimizing the cost functional; and then rely on a martingale principle for the optimally driven process to learn the value function at time n. Just like the NNcontPI algorithm, Hybrid-Now is based on training measures $(\mu_n)_{n=0}^{N-1}$ that have to be chosen by the agent, such that the density of the distribution μ_n should by high in the region where the optimally driven process is likely to go.

Algorithm 1.6 Hybrid-Now

Input: training distributions $(\mu_n)_{n=0}^{N-1}$;

Set $\hat{V}_N = g$;
for $n = N-1, \dots, 0$ **do**
  Compute:

$$\hat{\beta}_n \in \underset{\beta \in \mathbb{R}^q}{\operatorname{argmin}} \mathbb{E}\Big[f\big(X_n, A(X_n;\beta)\big) + \hat{V}_{n+1}(X_{n+1}^{\beta}) \Big]$$

  where $X_n \sim \mu$, and $X_{n+1}^{\beta} = F\big(X_n, A(X_n;\beta)\big), \varepsilon_{n+1})$;
  Assign $\hat{a}_n = A(.;\hat{\beta}_n)$; ▷ $\hat{a}_n$ is an estimate of the optimal control at time n;
  Compute

$$\hat{\theta}_n \in \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \mathbb{E}\left[\Big((f(X_n, \hat{a}_n(X_n)) + \hat{V}_{n+1}(X_{n+1}^{\hat{\beta}_n}) - \Phi(X_n;\theta) \Big)^2 \right];$$

  Assign $\hat{V}_n = \Phi(.;\hat{\theta}_n)$;
  ▷ $\hat{V}_n$ is an estimate of the value function at time n
end for

Output:
– Estimate of the optimal control $(\hat{a}_n)_{n=0}^{N-1}$;
– Estimate of the value function $(\hat{V}_n)_{n=0}^{N-1}$;

Comments on Hybrid-Now

✓ The training of the neural networks to approximate the optimal control and the value function at time n is uncoupled in Algorithm 1.6.

[18] and not Q-value, as proposed in the Actor-critic algorithms presented above.

✓ The training of the neural networks at time n is very fast since it only requires the simulation of the controlled process from time n to time $n+1$[19].

✗ The approximations of the value function, which are used to build the estimates of the optimal controls, implies that additional estimation and approximation errors appear, and need to be looked after, i.e. one should make sure that these errors do not make the algorithm diverge when running the backward procedure.

In Chapter 5, we study the consistence of the algorithms Hybrid-Now and Hybrid-Later, w.r.t. the size of the training set and the neural networks. We decided to only mention below the Theorem 1.2.2 page 158, where we derived the convergence of Hybrid-Now algorithm, .

Theorem 1.2.2 (Consistence of Hybrid-Now)**.** *Assume that there exist optimal feedback control, i.e.* $(a_k^{\text{opt}})_{k=n}^{N-1}$*, and denote by* V_n *the value function at time* $n = 0, \dots, N$*, and let* $X_n \sim \mu$*. Then, we have, as* $M \to +\infty$*:*

$$\begin{aligned}
&\mathbb{E}_M\Big[|\hat{V}_n^M(X_n) - V_n(X_n)|\Big] \\
&\qquad = \mathcal{O}_{\mathbb{P}}\Bigg(\left(\gamma_M^4 \frac{K_M \log(M)}{M}\right)^{\frac{1}{2(N-n)}} + \left(\gamma_M^4 \frac{\rho_M^2 \eta_M^2 \log(M)}{M}\right)^{\frac{1}{4(N-n)}} \\
&\qquad\quad + \sup_{n\le k\le N} \inf_{\Phi\in\mathcal{V}_M} \Big(\mathbb{E}_M\Big[|\Phi(X_k) - V_k(X_k)|^2\Big]\Big)^{\frac{1}{2(N-n)}} \\
&\qquad\quad + \sup_{n\le k\le N} \inf_{A\in\mathcal{A}_M} \Big(\mathbb{E}\Big[|A(X_k) - a_k^{\text{opt}}(X_k)|\Big]\Big)^{\frac{1}{2(N-n)}} \Bigg),
\end{aligned}$$

where $\mathbb{E}_M$ *is the expectation conditioned by the training set used to build the estimates* $(\hat{a}_k^M)_{n\le k\le N-1}$*.*

Theorem 1.2.2 states that the mean of the difference between the estimate of the value function at time n using HybridNow and the value function at time n cancels when the size of the training set goes to infinity and the space of the neural networks get larger and larger. Once again, an interesting part about the consistence result is that it takes into account the size of the training set.

Numerical Contributions (Chapter 6) Our algorithms are well-designed (or easily customizable) to solve several kind of control problems. This is what we tried to illustrate in Section 6.3 of Chapter 6. In this section, we studied the following different kinds of control problems:

- SemiLinearPDE: a control problem where the state space and the control space are of dimension 100.
- Option Hedging: a control problem where the controlled process is discrete.
- Valuation of Energy Storage: a control problem where the control space is finite, and the state space has continuous and discrete components.
- Smart Grid Management: a control problem where the state space is continuous, and the control admits a continuous and discrete component.

In particular Classif-Hybrid provided excellent results in the Smart Grid Management control problem, introduced in Section 6.3.4 page 204, which we decide to briefly present here. In the Smart Grid Management problem, we consider an agent who have a battery of charge C that can be charged using the received solar energy P, and who has to deliver the energy D to her clients. The agent can choose to activate a generator to deliver herself some energy to charge the battery or meet the demand of her clients at any time. She can also choose to deactivate the battery, however switching cost has to be paid every time when she changes the mode of the generator (switching cost forces her not to switch

[19] and not from time n to N, as it was the case in Algorithm NNcontPI.

too many times because it might deteriorate the generator). The agent is penalized when she does not meet the demand D of her clients, and needs to pay for the energy delivered by the generator. In Figure 1.2 below, which comes from Figures 6.11 page 212 and 6.10 page 211, we can see the optimal quantity of energy to get from the generator w.r.t. the residual demand $R = D - P$; estimated using Classif-Hybrid and Qknn[20] algorithms. In Figure 1.2, the case $m = 1$ refers to the situation where the generator is already activated right before time 1 and where deactivation would be penalized; the case where $m = 0$ refers to the situation where the generator is deactivated right before time 1, and where activation at time 1 would be penalized. We can see that the optimal decisions estimated by the two algorithms are very close and smooth. Moreover, the estimated optimal decisions look natural: when the residual demand is high and the battery is empty, i.e. R is large and is C small, it is optimal to activate the generator and buy some energy to meet the demand of the clients; on the other hand, if the residual demand is low and the battery is full: the agent does not need to activate the generator and can meet the demand by using the battery.

When comparing the value function, see Table 6.4 page 211, we can see that Classif-Hybrid does actually better than Qknn!

1.2.4 Perspectives

The quality of the estimates provided by our new algorithms highly depends on the quality of the training measures. The bootstrapping methods can help to build better and better estimates by relying on "adaptive" training measures, but they are not very efficient when the dimension of the problem is high.

It would be interesting not to rely on training measures, or at least to find adaptive methods to build training measures that work well in high-dimension. Notice that it is extremely difficult to have general method to get good training measures. In the community of the RL, this problem is referred to as *exploration* problem, and research in this field is very active.

1.3 Neural networks for PDEs and BSDEs

In the third part of this thesis, we propose and study a family of algorithms to solve PDEs, QVIs, BSDEs, and RBSDEs. We first present the framework, recall the usual and recent methods that are already available in the literature, and finally present our new algorithms. Consistency of our algorithms is proved, and the algorithms are tested on several numerical examples from dimension 1 to 40.

1.3.1 General framework

Let us fix $T > 0$. Let μ and σ be two functions such that $\mu : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$, and $\sigma : [0,T] \times \mathbb{R}^d \to \mathcal{M}_d(\mathbb{R})$, where $\mathcal{M}_d(\mathbb{R})$ is the space of the $d \times d$ real matrices. μ and σ are assumed to be smooth enough so that there is existence and uniqueness of a solution $\mathcal{X}$ to the following SDE:

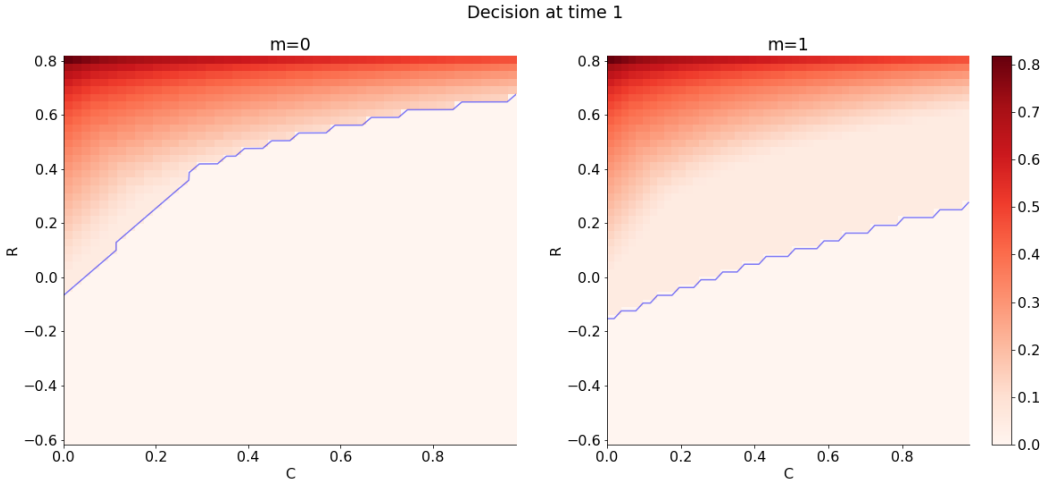
$$\mathcal{X}_t = x_0 + \int_0^t \mu(s, \mathcal{X}_s)\,\mathrm{d}s + \int_0^t \sigma(s, \mathcal{X}_s)\,\mathrm{d}W_s, \quad 0 \le t \le T,$$

where W is a d-dimensional Brownian motionon a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ equipped with a filtration $\mathbb{F} = (\mathcal{F}_t)_{0 \le t \le T}$ which statisfies the usual assumptions.

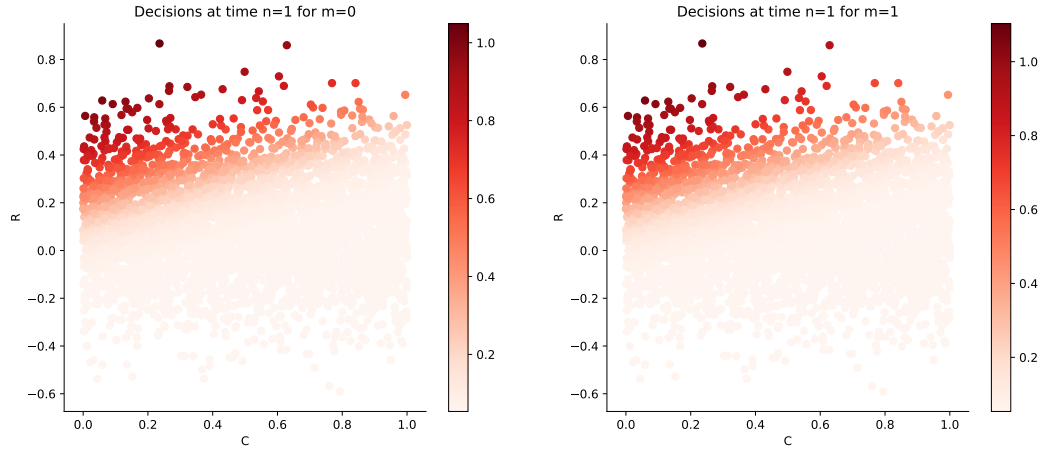
BSDEs and PDEs The first problem we tackle in the third part of the thesis is to find a couple of $\mathbb{F}$-adapted processes (Y, Z), which take values in $\mathbb{R} \times \mathbb{R}^d$ and such that the following holds:

$$Y_t = g(\mathcal{X}_T) + \int_t^T f(s, \mathcal{X}_s, Y_s, Z_s)\,\mathrm{d}s - \int_t^T Z_s^\intercal\,\mathrm{d}W_s, \quad 0 \le t \le T. \tag{1.3.1}$$

[20] Qknn was introduced in Section 1.1.1.3 page 5, and can be considered as an excellent benchmark.



(a) Estimate of the optimal quantity of energy to get from the generator at time $n = 1$ w.r.t. the couple of state variables (R, C), in the case where $m = 0$ or $m = 1$, using the Qknn algorithm. The region below the blue line is the one where it is optimal to turn off (or keep off) the generator. The region above the blue line is the one where it is optimal to deliver some energy from the generator.



(b) Estimate of the optimal quantity of energy to get from the generator at time $n = 1$ w.r.t. the couple of state variables (R, C), in the case where $m = 0$ or $m = 1$, using the Classif-Hybrid algorithm.

Figure 1.2 – Estimates of the optimal quantity of energy to get from the generator using Classif-Hybrid or Qknn.

Equation (1.3.1) is a Backward Stochastic Differential Equation (BSDE) since the terminal condition g is imposed, and the couple (Y, Z) is called solution to the BSDE. It is well-known that (1.3.1) admits a unique solution under some regularity conditions on f and g (see [PP90]) and that, moreover, we can write Y and Z as follows:

$$\begin{aligned} Y_t &= u(t, \mathcal{X}_t), \\ Z_t &= \sigma^\top D_x u(t, \mathcal{X}_t), \quad 0 \leq t \leq T, \end{aligned}$$

where u is solution to the following parabolic non-linear partial differential equation:

$$\begin{cases} \partial_t u + \mathcal{L}u + f(., ., u, \sigma^\intercal D_x u) = 0, & \text{on } [0, T) \times \mathbb{R}^d, \\ u(T, .) = g, & \text{on } \mathbb{R}^d, \end{cases} \tag{1.3.2}$$

and where $\mathcal{L}$ is the infinitesimal generator of $\mathcal{X}$, defined as follows

$$\mathcal{L}u := \frac{1}{2} \operatorname{Tr}\left(\sigma\sigma^\intercal D_x^2 u\right) + \mu . D_x u.$$

Hence, finding the solution to the BSDE (1.3.1) is connected to solving the PDE (1.3.2).

RBSDEs and QVIs The second problem that we tackle is to find the triplet (Y, Z, K) of $\mathbb{F}$-adapted processes solution to the following equation:

$$\begin{aligned} Y_t &= g(\mathcal{X}_T) + \int_t^T f(s, \mathcal{X}_s, Y_s, Z_s)\, \mathrm{d}s - \int_t^T Z_s^\intercal\, \mathrm{d}W_s + K_T - K_t, \\ Y_t &\geq g(X_t), \quad 0 \leq t \leq T, \end{aligned} \tag{1.3.3}$$

where K is an non-decreasing process such that

$$\int_0^T \big(Y_t - g(X_t)\big)\, \mathrm{d}K_t = 0.$$

(1.3.3) is a classical representation of optimal stopping time problems, and appears in particular in the problems of pricing of American options. The Y component of the solution to (1.3.3) reads $Y_t = u(t, X_t)$, where it is well-known that u is solution to the Quasi-Variational Inequality (QVI):

$$\begin{cases} \min\big[-\partial_t u - \mathcal{L}u - f(t, x, u, \sigma^\intercal D_x u), u - g\big] &= 0, & t \in [0, T),\ x \in \mathbb{R}^d, \\ u(T, x) &= g(x), & x \in \mathbb{R}^d, \end{cases} \tag{1.3.4}$$

as shown e.g. in [EK+97].
The second problem is equivalent to find the solution to the QVI (1.3.4).

1.3.2 Numerical methods available in the literature

When $\mathcal{X}$ cannot be simulated, the first step to estimate the solution of BSDEs or RBSDEs is to discretize the process on a time-grid: $\pi = \{t_0 = 0 < t_1 < \ldots < t_N = T\}$, with modulus $|\pi| = \max_{i=0,\ldots,N-1} \Delta t_i$, where $\Delta t_i := t_{i+1} - t_i$. The Euler discretization $X = X^\pi$ of $\mathcal{X}$ then reads:

$$X_{t_{i+1}} = X_{t_i} + \mu(t_i, X_{t_i})\Delta t_i + \sigma(t_i, X_{t_i})\Delta W_{t_i}, \quad i = 0, \ldots, N-1,\ X_0 = x_0,$$

where we denote $\Delta W_{t_i} := W_{t_{i+1}} - W_{t_i}$. To alleviate the notations, we denote by X the Euler discretization of $\mathcal{X}$, and omit the dependence of the latter to π. The approximation of (1.3.2) then follows from the Euler discretization of the forward representation (1.3.1):

$$u(t_{i+1}, X_{t_{i+1}}) \approx F(t_i, X_{t_i}, u(t_i, X_{t_i}), \sigma^\intercal(t_i, X_{t_i}) D_x u(t_i, X_{t_i}), \Delta t_i, \Delta W_{t_i}),$$

where we define:

$$F(t, x, y, z, h, \Delta) := y - f(t, x, y, z)h + z^\intercal \Delta. \tag{1.3.5}$$

1.3.2.1 Numerical methods for PDEs and BSDEs

We review in this section the some important algorithms already available in the literature to solve (1.3.1).

Finite-Difference Method The PDE (1.3.2) can be solved using finite difference method by discretizing the derivative operators on the grids (resp. sparse-grids) when dimension is low (resp. medium).
The convergence of finite difference methods is proved under some conditions on the time and space discretization size steps.

Comments on the finite difference methods:

- ✗ The finite difference-based algorithms impose conditions on the boundaries that are rarely provided in the statement of the control problems (1.3.2). In pratice, one should add some artificial boundary conditions to apply these methods. (see the Benchmark in Section 4.4.2.2 page 132 for an example of application of a finite difference based algorithm to solve the HJB of a control problem, which is a non-linear PDE)
- ✗ The complexity of these methods is $\mathcal{O}^{d+1}$, which explodes when dimension gets high! The method is actually adapted for low or medium dimensions.

Regression on basis functions As presented in [LGW06], an idea to estimate the solutions to BSDEs is to use Monte-Carlo regression on basis functions, with the following backward scheme:

$$\left\{\begin{array}{rcl} Y_{t_N} & = & g(X_{t_N}) \\ Z_{t_i} & = & \frac{1}{\Delta t_i}\mathbb{E}\left[Y_{t_{i+1}}\Delta W_{t_i}|\mathcal{F}_{t_i}\right], \\ Y_{t_i} & = & \mathbb{E}\left[Y_{t_{i+1}}|\mathcal{F}_{t_i}\right] + f(t_i, X_{t_i}, Y_{t_i}, Z_{t_i})\Delta t_i, \quad \text{for } i = N-1, \dots, 0. \end{array}\right.$$

At time t_i, Z_{t_i} is approximated by regressing $Y_{t_{i+1}}\Delta W_{t_i}$ at time n on basis functions. Then, Y_{t_i}, defined in (2.3.6) as as fixed-point of the contracting (for h small enough) operator $\mathbb{L}^2(\sigma(X_{t_i})) \to \mathbb{L}^2(\sigma(X_{t_i}))$: $Y_{t_i} \mapsto \mathbb{E}_i\left[Y_{t_{i+1}}\right] + hf(t_i, X_{t_i}, Y_{t_i}, Z_{t_i})$, is approximated using Picard iteration technics. The convergence of this algorithm is studied in [LGW06].

Comments on methods based on regression on basis functions:

- ✓ Methods based on regression on basis functions are very fast when dimension is no more than medium (≤ 10) since closed form formulas exist to compute the estimates of the conditional expectations.
- ✗ Methods based on regression on basis functions are very efficient to solve BSDEs with medium dimensions, but suffer from the curse of dimensionality: when dimension is high, overfitting or underfitting problems appear when regressing quantities on basis functions.

The two methods that we just presented suffer from the curse of dimensionality. Recent methods are available in the literature to solve BSDEs in high-dimension using deep learning.

DGM As introduced in [SS18], DGM[21] consists of representing the solution to (1.3.2), denoted $\mathcal{U}(t,x;\theta)$, by neural networks. Let us consider the following (training!) measures: ν_T whose support is $[0,T]$, ν_Ω whose support is Ω, $\nu_{\partial\Omega}$ whose support is $\partial\Omega$; and let us define the following objective function J:

$$J(\theta) = \|\partial_t\mathcal{U}(.,.;\theta) + \mathcal{L}\mathcal{U}(.,.;\theta) + f(t,x,\mathcal{U}(.,.;\theta), \sigma^\intercal D_x\mathcal{U}(.,.;\theta)\|_{[0,T]\times\Omega,\nu_\Omega} + \|\mathcal{U}(T,.;\theta) - g(.)\|_{\partial\Omega,\nu_{\partial\Omega}},$$

where $\|.\|_{[0,T]\times\Omega,\nu_\Omega}$ stands for the canonical norm of $\mathbb{L}^2\left([0,T]\times\Omega, \nu_T\times\nu_\Omega\right)$, and $\|.\|_{[0,T]\times\partial\Omega,\nu_{\partial\Omega}}$ stands for the canonical norm of $\mathbb{L}^2\left([0,T]\times\partial\Omega, \nu_T\times\nu_{\partial\Omega}\right)$.

The main idea in the DGM algorithm for the training of the neural networks is to find the optimal parameters for the neurla network to minimize J by SGD method. When the space of neural networks

[21] The acronym stands for Deep Galerkin Method.

goes large, the best neural network $\mathcal{U}(.,.;\theta^*)$ almost cancels the objective function, which implies that $\mathcal{U}(.,.;\theta^*)$ is solution to the PDE and meets the boundary conditions, hence is close to the solution u of (1.3.2).

Comments on DGM:

- ✓ We remind here that the computation of the derivatives of neural networks w.r.t. θ is not time-consuming when relying on reverse back-propapagation.
- ✓ The fact that the algorithm relies on training measures that are chosen by the agent can be positive when the agent really want to have precise estimates of the solution on some regions, and does not mind about being precise in other regions. Indeed, in this case: the agent can choose training measures whose densities are high at the interesting regions, and low in the other ones.

NNZ NNZ (Neural Network approximation of Z) is introduced in [EHJ17] and the consistency is studied in [HL18]. This algorithm is our main benchmark to solve high-dimensional BSDEs in Chapter 7. The main idea is to approximate $\mathcal{Z}_{t_i}$ for $t_i, i = 0, \dots, N-1$ using neural networks; and rely on the following formal equalities:

$$\begin{aligned} u(t_n, X_{t_n}) \approx u(t_{n-1}; X_{t_{n-1}}) - f(t_{n-1}, X_{t_{n-1}}, u(t_{n-1}, X_{t_{n-1}}), Z(t_{n-1}, X_{t_{n-1}}))(t_n - t_{n-1}) \\ + Z(t_n, X_{t_n})^T (W_{t_{n+1}} - W_{t_n}), \end{aligned}$$

to represent, by induction, for $n = 0, \dots, N$, the function $u(t_n, .)$ by a neural network $\mathcal{U}_n(., \theta)$ built from the composition of the neural networks representations of Z_{t_i}, with $i = 0, \dots, n-1$. The objective function for NNZ is the function J defined as follows:

$$J(\theta) := \mathbb{E}\left[\left(\mathcal{U}_N(X_N, \theta) - g(X_N) \right)^2 \right].$$

and the idea is that the optimal controls Z_{t_i}, for $i = 0, N-1$ will be learnt by training $\mathcal{U}_N(X_N, \theta)$ such that it cancels the objective function J.

Comments on NNZ:

- ✓ All the neural networks representing the Z_{t_i} are trained at the same time, which implies that the algorithm converges quickly.
- ✓✗ There is no approximation and estimation errors coming from the approximation of u. The price to pay for this feature is the time-consuming re-simulations of the process from time $k = 0, \dots, N$ to train the neural networks.
- ✗ Since the training of the neural networks is done all at the same time, NNZ can actually be seen as an algorithm that performs a training of large neural network with $n_Z \times N$ layers, where n_Z is the number of layers chosen to represent each Z_{t_i}. When N is large, the neural network to train is very deep, and the training of such a neural network is still very difficult today. Hence, the risk about NNZ is that the SGD actually converges to a poor local minima, or does not converge at all. The two situations were met in Chapter 7.

1.3.3 Contributions (Chapter 7)

Motivation Because of the curse of dimensionality, solving (1.3.1) and (1.3.3) in high-dimension has always been a difficult problem to tackle for practitioners. Likewise, it is tough to find solution to (1.3.2) and (1.3.4) in high-dimension. The usual methods based on regression on basis functions or finite difference can provide good estimates in low or medium dimension. However, either their complexities explode when dimension gets high, or they suffer from overfitting and underfitting.

Methods based on neural networks, such as NNZ and DGM algorithms, seem to be the only ones providing good estimates in high dimension. However, NNZ appears to provide poor estimates of the solution to the BSDE (or even does not converge at all in certain cases) when the number of time steps N is large or when the functions to learn are too complex (see Section 7.5.2 for examples on which NNZ provides poor estimates or diverges).

Question 3a:	*What stable and accurate scheme can we propose to solve BSDEs in high dimension?*
Question 3a(bis):	*What stable and accurate scheme can we propose to solve PDEs in high dimension?*
Question 3b:	*What stable and accurate scheme can we propose to solve RBSDEs in high dimension?*
Question 3b(bis):	*What stable and accurate scheme can we propose to solve QVIs in high dimension?*

As said above: Question 3a and 3a(bis) are related, so are Question 3b and 3b(bis).

We introduce in Chapter 7 two new algorithms, that we named DBDP1 and DBDP2, to answer Question 2a; and one new algorithm, named RDBDP to answer Question 2b. Consistence results are derived for all the algorithms, and they are tested and compared to NNZ on several examples. We briefly present the three algorithms in the next section, and highlight the important consistence results that have been derived.

Presentation of DBDP1 and consistency result

The main idea in DBDP1 algorithm is to approximate, for $i = N-1, \dots, 0$, the couple $(u(t_i,.), Z(t_i,.))$ by neural networks:

$$\begin{cases} u(t_i,.) &= \mathcal{U}_i(.;\theta) \\ Z(t_i,.) &= \mathcal{Z}_i(.;\theta), \end{cases}$$

where we denote by θ the parameters of the two neural networks $\mathcal{U}_i$ and $\mathcal{Z}_i$. We learn the optimal parameters θ_i^*, $i = 0, \dots, N-1$, which is the optimal one for $\mathcal{U}_i$ and $\mathcal{Z}_i$ to be close to $u(t_i,.)$ and Z_{t_i}, in a backward way, relying, as written in the pseudo-code in Algorithm 1.7, on the following equality:

$$\mathcal{U}_{i+1}(X_{t_{i+1}};\theta_{i+1}^*) \approx F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta_i^*), \mathcal{Z}_i(X_{t_i};\theta_i^*), \Delta t_i, \Delta W_{t_i}),$$

where θ_{i+1}^* is the optimal parameter of $\mathcal{U}_{i+1}$ and $\mathcal{Z}_{i+1}$ for the approximation of $u(t_{i+1},.)$ and Z_{t_i}, and is assumed to be already computed at time t_{i+1}, and where F was introduced in (1.3.5).

Algorithm 1.7 DBDP1 Algorithm

Input:

1: Initialize $\widehat{\mathcal{U}}_N^{(1)} = g$
2: **for** $i = N-1, \dots, 0$ **do**
3: Compute:

$$\theta_i^* \in \underset{\theta \in \mathbb{R}^{N_m}}{\operatorname{argmin}} \ \mathbb{E}\left| \widehat{\mathcal{U}}_{i+1}^{(1)}(X_{t_{i+1}}) - F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta), \mathcal{Z}_i(X_{t_i};\theta), \Delta t_i, \Delta W_{t_i}) \right|^2;$$

4: Assign $\widehat{\mathcal{U}}_i^{(1)} = \mathcal{U}_i(.;\theta_i^*)$;
5: Assign $\widehat{\mathcal{Z}}_i^{(1)} = \mathcal{Z}_i(.;\theta_i^*)$;
6: **end for**

Output:

– $(\widehat{\mathcal{U}}_i^{(1)})_{i=0}^N$: Estimates of Y_{t_i} for $i = 0, \dots, N$;
– $(\widehat{\mathcal{Z}}_i^{(1)})_{i=0}^N$: Estimates of Z_{t_i} for $i = 0, \dots, N$;

The convergence of DBDP1 is studied in Chapter 7 (see Theorem 7.4.1 page 225), and we present here the simplified version of the result:

Theorem 1.3.1. (Consistence of DBDP1) *Under classical regularity conditions on the coefficients of*

the BSDE, there exists a constant $C>0$, independent of the time discretization π, such that

$$\mathcal{E}\big[(\widehat{\mathcal{U}}^{(1)},\widehat{\mathcal{Z}}^{(1)}),(Y,Z)\big] \leq C\Big(\mathbb{E}\big|g(\mathcal{X}_T)-g(X_T)\big|^2+|\pi|+\varepsilon^Z(\pi) + \sum_{i=0}^{N-1}\big(N\varepsilon_i^{\mathcal{N},v}+\varepsilon_i^{\mathcal{N},z}\big)\Big),$$

where:

$$\mathcal{E}\big[(\widehat{\mathcal{U}}^{(1)},\widehat{\mathcal{Z}}^{(1)}),(Y,Z)\big] := \max_{i=0,\dots,N-1}\mathbb{E}\big|Y_{t_i}-\widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2+\mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}}\big|Z_t-\widehat{\mathcal{Z}}_i^{(1)}(X_{t_i})\big|^2dt\Big]$$

should be seen as a $\mathbb{L}^2$-error between the estimates $(\widehat{\mathcal{U}}_i^{(1)})_{i=0}^N$ and $(\widehat{\mathcal{Z}}_i^{(1)})_{i=0}^N$ provided by DBDP1 and the real solutions $u(t_i,.)$ and Z, for $i=0,\dots,N$,
and where:

- $$\varepsilon_i^{\mathcal{N},v} := \inf_{\xi}\mathbb{E}\big|\hat{v}_i(X_{t_i})-\mathcal{U}_i(X_{t_i};\xi)\big|^2, \qquad \varepsilon_i^{\mathcal{N},z} := \inf_{\eta}\mathbb{E}\big|\overline{\hat{z}_i}(X_{t_i})-\mathcal{Z}_i(X_{t_i};\eta)\big|^2$$

  should be seen as approximation errors coming from the approximation of functions using neural networks.

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

  is the $\mathbb{L}^2$-regularity de Z[22]

Theorem 1.3.1 states that the $\mathbb{L}^2$-error made by DBDP1 algorithm when estimating Y and Z is bounded by the sum of:

- a time-discretization of $[0,T]$ error,
- an Euler-discretization of X error,
- the $\mathbb{L}^2$-regularity of Z,
- the quantity $\sum_{i=0}^{N-1}\big(N\varepsilon_i^{\mathcal{N},v}+\varepsilon_i^{\mathcal{N},z}\big)$ that should be seen[23] as the sum of approximation errors that are made by approximating Y and Z in a backward way.

By standard arguments and under standard assumptions, we can show that these errors all vanish when the time-discretization step size goes to zero, and the space of considered neural networks gets larger.

Presentation of DBDP2, and consistency result

One may have noticed in the pseudo-code of DBDP1 that we ignored the well-known result: $Z=\sigma^\intercal(t_i,.)D_xu(t_i,.)$, by representing Z by the neural networks $\mathcal{Z}_i(.;\theta)$. Indeed, since $Z=\sigma^\intercal(t_i,.)D_xu(t_i,.)$, one can represent Z by the derivative of $\mathcal{U}_i(.;\theta)$. This is the main underlying idea of DBDP2.

Let us consider a neural network $\mathcal{U}$ to represent u. We then have the following representation for the couple (Y,Z):

$$\begin{cases} u(t_i,.) &= \mathcal{U}_i(.;\theta) \\ Z(t_i,.) &= \sigma^\intercal(t_i,.)D_x\mathcal{U}_i(.;\theta), \end{cases}$$

[22] We refer to [Zha04] for more details on the regularity of Z
[23] we refer to 1.3.4 for more details on this term

where θ is the parameter associated to $\mathcal{U}$. We learn the optimal parameter θ_i^*, such that $\mathcal{U}_N$ is close to $u(t_i,.)$, and $\sigma^\intercal(t_i,.)D_x\hat{\mathcal{U}}_i(.;\theta)$ close to $Z(t_i,.)$, in a backward way, relying, as we can see in the pseudo-code of DBDP2 written in Algorithm 1.8, on the following equality:

$$\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}};\theta_{i+1}^*) = F(t_i,X_{t_i},\mathcal{U}_i(X_{t_i};\theta_i^*),\sigma^\intercal(t_i,X_{t_i})D_x\mathcal{U}_i(X_{t_i};\theta_i^*),\Delta t_i,\Delta W_{t_i}).$$

Algorithm 1.8 DBDP2 Algorithm

Input:

1: Initialize $\widehat{\mathcal{U}}_N^{(1)} = g$
2: **for** $i = N-1,\dots,0$ **do**
3: Compute:

$$\theta_i^* \in \underset{\theta\in\mathbb{R}^{N_m}}{\operatorname{argmin}}\ \mathbb{E}\Bigg[\Big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{t_{i+1}}) - F(t_i,X_{t_i},\mathcal{U}_i(X_{t_i};\theta),\sigma^\intercal(t_i,X_{t_i})\hat{D}_x\mathcal{U}_i(X_{t_i};\theta),\Delta t_i,\Delta W_{t_i})\Big|^2\Bigg];$$

4: Assign $\widehat{\mathcal{U}}_i^{(1)} = \mathcal{U}_i(.;\theta_i^*)$;
5: Assign $\widehat{\mathcal{Z}}_i^{(1)} = \sigma^\intercal(t_i,.)D_x\mathcal{U}_i(.;\theta_i^*)$;
6: **end for**

Output:

– $(\widehat{\mathcal{U}}_i^{(1)})_{i=0}^N$: Estimates of Y_{t_i} for $i=0,\dots,N$;
– $(\widehat{\mathcal{Z}}_i^{(1)})_{i=0}^N$: Estimates of Z_{t_i} for $i=0,\dots,N$;

Remark 1.3.1. *The complexity to compute $\nabla\mathcal{U}_i(.;\theta)$ and $\mathcal{U}_i(.;\theta)$ is roughly the same, when using reverse back-propagation, as natively implemented in Tensorflow. The computation of $\nabla\mathcal{U}_i(.;\theta)$ when running DBDP2 is then not time-consuming.*

The convergence of DBDP2 is studied in Chapter 7 (see Theorem 7.4.2 page 230), and we present here the simplified version of the result:

Theorem 1.3.2. (Consistency of DBDP2) *Under usual regularity assumptions, there exists a constant $C>0$, independent of π such that*[24]

$$\mathcal{E}\big[(\widehat{\mathcal{U}}^{(2)},\widehat{\mathcal{Z}}^{(2)}),(Y,Z)\big] \le C\Big(\mathbb{E}\big|g(\mathcal{X}_T)-g(X_T)\big|^2 + \frac{\gamma_m^6}{N} + \varepsilon^Z(\pi) + N\sum_{i=0}^{N-1}\varepsilon_i^{\mathcal{N},m}\Big).$$

Theorem 1.3.1 states that the $\mathbb{L}^2$-norm of the error of estimation of u and Z by DBDP2 algorithm is bounded by the sum of

- a time-discretization of $[0,T]$ error,
- an Euler-discretization of X error,
- the quantity $\frac{\gamma_m^6}{N}$ which appears because we had to control the size of the space of neural network,
- the $\mathbb{L}^2$-regularity of Z,
- the quantity $\sum_{i=0}^{N-1}\left(N\varepsilon_i^{\mathcal{N},v}+\varepsilon_i^{\mathcal{N},z}\right)$ that should be seen[25] as the sum of approximation errors that are made by approximating Y and Z in a backward way.

All these errors vanish when the time-discretization step goes to zero, and the size of the neural network gets large.

[24] see Theorem 1.3.1 for the notations.
[25] we refer to 1.3.4 for more details on this term.

Presentation of RDBDP, and consistency result

The main idea of the RDBDP algorithm is to approximate the pair $(u(t_i,.), Z(t_i,.))$, for $i = N-1,\dots,0$, by neural networks in the following way:

$$\begin{cases} u(t_i,.) &= \mathcal{U}_i(.;\theta) \\ Z(t_i,.) &= \mathcal{Z}_i(.;\theta), \end{cases}$$

where we denote by θ the parameters of the two neural networks; and rely on the scheme described in Algorithm 1.9 to learn $u(t_i,.)$ et $Z(t_i,.)$, for $i = N-1,\dots,0$.

Algorithm 1.9 RDBDP Algorithm

Input:

1: Initialize $\widehat{\mathcal{U}}_N^{(1)} = g$
2: **for** $i = N-1,\dots,0$ **do**
3: Compute:

$$\theta_i^* \in \underset{\theta\in\mathbb{R}^{N_m}}{\operatorname{argmin}}\, \hat{L}_i(\theta),$$

where

$$\hat{L}_i(\theta) := \mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) - F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta), \mathcal{Z}_i(X_{t_i};\theta), \Delta t_i, \Delta W_{t_i})\big|^2$$

4: Assign $\widehat{\mathcal{U}}_i = \max\big[\mathcal{U}_i(.;\theta_i^*), g\big]$;
5: Assign $\widehat{\mathcal{Z}}_i = \mathcal{Z}_i(.;\theta_i^*)$;
6: **end for**

Output:

– $(\widehat{\mathcal{U}}_i^{(1)})_{i=0}^N$: Estimates of Y_{t_i} for $i = 0,\dots,N$;
– $(\widehat{\mathcal{Z}}_i^{(1)})_{i=0}^N$: Estimates of Z_{t_i} for $i = 0,\dots,N$;

The convergence of RDBDP is studied in Chapter 7 (see Theorem 7.4.4 page 235), and we present here the simplified version of the result:

Theorem 1.3.3. (Consistency of RDBDP) *Under usual assumptions on the regularity of the coefficients of the RBSDE, there exists a constant $C > 0$, independent of π, such that*

$$\mathcal{E}\big[(\widehat{\mathcal{U}}, \widehat{\mathcal{Z}}), (Y, Z)\big] \leq C\Big(\varepsilon(\pi) + \sum_{i=0}^{N-1}\big(N\varepsilon_i^{\mathcal{N},\tilde{v}} + \varepsilon_i^{\mathcal{N},\tilde{z}}\big)\Big),$$

where $\varepsilon(\pi) = O(|\pi|^{\frac{1}{2}})$ if g is smooth, and $\varepsilon(\pi) = O(|\pi|)$ if g and σ are smooth.

Numerical results

Numerical tests of our new algorithms DBDP1, DBDP2 and RDBDP are presented in Section 7.5 of Chapter 7. Our algorithms provided some very accurate results. In particular DBDP1 and DBDP2 appeared to be more stable than NNZ, which was taken as a benchmark in all the considered examples. Our algorithms converge as well as NNZ when the latter converges, and converge in some cases where NNZ either converges to poor estimates or diverges. We noticed that NNZ does not converge when the functions u and Z to learn are too complex, or N is too large. We refer to Section 7.5.1 page 237 for examples on which DBDP1, DBDP2 and NNZ work well; and Section 7.5.2 page 241 for examples where DBDP1 et DBDP2 work well, but NNZ diverges. Finally we did not observe any situations where NNZ converges whereas any of our algorithms fail to converge.

The RDBDP algorithm also appeared to work well in the considered application: we consider a problem of pricing an American option from dimension 1 to 40. Part of the results are available in Table 1.2, and the whole results are available in Table 7.11 page 245. For the payoff that we considered, the American put pricing problem can actually always be reduced to a problem of dimension 1, and

Dimension	nb steps	RDBDP	std	benchmark
1	80	0.0613818	0.00019	0.060903
5	80	0.107650	0.00016	0.10738
10	80	0.129923	0.00016	0.12996
20	80	0.15050	0.00010	0.1510
40	160	0.16758	0.00016	0.1680

Table 1.2 – Estimates of the price of an American option using RDBDP algorithm. We reported very accurate estimates of the price (computed by a tree-based algorithm after applying a trick to reduce the dimension of the problem to one) to the benchmark column. The mean and standard deviation are computed on 40 independent simulations. RDBDP is precise up to dimension 40!

can then be solved in a very accurate way using conventional methods such as the binomial tree model. The benchmark is then a binomial tree-based algorithm, which provides very accurate results. As we can see on Table 1.2, RDBDP appears to provide very accurate results up to dimension 40.

1.3.4 Perspectives

The results of convergence proved for our new algorithms are interesting, but it would be more natural to have the quantity $\sum_{i=0}^{N-1} \varepsilon_i^{\mathcal{N},\tilde{v}}$ instead of $\sum_{i=0}^{N-1} N\varepsilon_i^{\mathcal{N},\tilde{v}}$ in the bounds proposed in Theorems 1.3.1, 1.3.2, and 1.3.2 (Note that the N disappeared). Indeed, the quantity $\sum_{i=0}^{N-1} \varepsilon_i^{\mathcal{N},\tilde{v}}$ is exactly the error made by approximating some functions using neural networks at each time step, and the N that we had to add in our results is just an artefact. Adding the N to get consistency results was the best we could do.

Chapter 2

Introduction (in French)

La thèse se divise en trois parties que l'on peut lire indépendamment. Dans la première partie, nous proposons des algorithmes basés sur des méthodes de quantification et de régression pour résoudre numériquement les processus de décisions markoviennes (MDPs). Puis nous présentons deux applications : la première porte sur un problème de market-making, et la deuxième sur des problèmes de contrôle stochastique de type McKean-Vlasov[1]. Dans la seconde partie de cette thèse, nous présentons une famille de schémas numériques basée sur les réseaux de neurones pour résoudre les MDPs. Nous étudions d'abord la convergence théorique de nos nouveaux schémas en fonction de la taille du training set et de l'espace des réseaux de neurones considéré ; puis les testons sur des exemples variés issus de la littérature. Dans la troisième partie de cette thèse, nous présentons une classe de schémas numériques, à base de réseaux de neurones, pour résoudre numériquement les EDPs, IQVs, EDSRs et EDSRs réfléchies. Nous avons obtenu des résultats théoriques de convergence des schémas en fonction de l'erreur d'approximation des réseaux de neurones, et avons illustré leurs performances sur de nombreux tests numériques.

Contents

[1] i.e. des problèmes de contrôle où la dynamique du processus contrôlé dépend de la loi de ce dernier.

2.1 Méthodes de Quantification et de régression pour les MDPs. Application au market-making et aux problèmes de contrôle de type McKean-Vlasov avec information partielle

2.1.1 Méthodes de Quantification et de régression pour les problèmes de décisions Markoviennes (MDP)

Après avoir introduit les problèmes de décisions Markoviennes et rappelé quels sont les algorithmes qui existent aujourd'hui dans la littérature pour résoudre numériquement les MDPs, nous présentons un nouvel algorithme basé sur la regression locale et la quantification, qui est proposé et étudié dans la première partie de cette thèse.

2.1.1.1 Présentation des MDPs

Un problème de contrôle stochastique à temps discret et horizon fini est la donnée d'un horizon fini $N \in \mathbb{N} \setminus \{0\}$, et d'un processus (dit contrôlé) $X^\alpha = (X_n^\alpha)_{n=0}^N$, à valeurs dans $\mathcal{X} \subset \mathbb{R}^d$, avec $d \in \mathbb{N}$ la dimension du problème de contrôle, qui admet la dynamique suivante :

$$\left\{ \begin{array}{rcl} X_0^\alpha & = & x_0 \in \mathbb{R}^d, \\ X_{n+1}^\alpha & = & F_n(X_n^\alpha, \alpha_n, \varepsilon_{n+1}), \quad \text{pour } n = 0, \dots, N-1, \end{array} \right.$$

où :

- $(\varepsilon_n)_{n=1}^N$ est une suite de v.a. supposées i.i.d., à valeurs dans un espace de Borel $(E, \mathcal{B}(E))$, et définie sur un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$ muni de la tribu $\mathbb{F} = (\mathcal{F}_n)_{n=0}^N$ engendrée par le bruit $(\varepsilon_n)_{n=1}^N$ ($\mathcal{F}_0$ est la tribu triviale),
- le contrôle α est un processus $\mathbb{F}$-mesurable à valeur dans $\mathbb{A} \subset \mathbb{R}^q$,
- pour $n = 0, \dots, N-1$, les F_n sont des fonctions mesurables de $\mathbb{R}^d \times \mathbb{R}^q \times E$ dans $\mathbb{R}^d$.

Étant donné un facteur d'actualisation $\beta \geq 0$, une fonction de coût instantané $f : \mathbb{R}^d \times \mathbb{R}^q \to \mathbb{R}$, et une fonction de coût terminal $g : \mathbb{R}^d \to \mathbb{R}$, on définit $J(\alpha)$, la fonctionnelle de coût associée au processus de décision α, de la manière suivante :

$$J(\alpha) \quad = \quad \mathbb{E}\left[\sum_{n=0}^{N-1} \beta^n f(X_n^\alpha, \alpha_n) + g(X_N^\alpha) \right]. \tag{2.1.1}$$

On dit qu'un contrôle α est admissible si la fonction $J(\alpha)$ est bien définie, et l'on note dans la suite $\mathcal{C}$ l'ensemble des contrôles admissibles. Le problème de contrôle se formule de la manière suivante : d'abord trouver

$$V_0(x_0) := \inf_{\alpha \in \mathcal{C}} J(\alpha), \tag{2.1.2}$$

puis caractériser (au moins) une stratégie optimale $\alpha^* \in \mathcal{C}$, définie comme étant une stratégie qui minimise la fonctionnelle de coût, i.e. $V_0(x_0) =: J(\alpha^*)$.

Dans cette thèse on parlera de MDP à horizon fini, ou plus simplement de MDP, pour désigner les problèmes de contrôle à horizon fini et temps discret.

Observons qu'une première difficulté dans la résolution de (2.1.2) est que l'infimum est pris sur l'espace de dimension infinie des processus admissibles $\mathcal{C}$. En utilisant un principe de programmation dynamique, on peut réécrire (2.1.2) de manière plus simple. Pour se faire, introduisons d'abord les notations standards suivantes : on note par $\{P^a(x, dx'),\, a \in \mathbb{A},\, x \in \mathcal{X}\}$, la famille des probabilités de transition associée à la chaîne de Markov contrôlée (2.1.1.1), i.e.

$$P^a(x, dx') \quad = \quad \mathbb{P}\left[F(x, a, \varepsilon_1) \in dx'\right],$$

et de plus on note

$$P^a\varphi(x) = \int \varphi(x')P^a(x,dx') \quad = \quad \mathbb{E}\left[\varphi\big(F(x,a,\varepsilon_1)\big)\right],$$

pour toute fonction mesurable $\varphi : \mathcal{X} \to \mathbb{R}$.

Avec ces notations, pour toute fonction φ sur $\mathcal{X}$, et pour tout contrôle $\alpha \in \mathcal{C}$, on a :

$$\mathbb{E}[\varphi(X^\alpha_{n+1})|\mathcal{F}_n] \quad = \quad P^{\alpha_n}\varphi(X^\alpha_n), \quad \forall\, n \in \mathbb{N}.$$

Par un principe de programmation dynamique, la fonction valeur à l'instant n, pour $n = 0, \dots, N$, que l'on note V_n, est caractérisée comme solution du schéma rétrograde suivante[2] :

$$\left\{\begin{array}{rcl} V_N(x) & = & g(x), \quad \forall x \in \mathcal{X}, \\ Q_n(x,a) & = & f(x,a) + \beta P^a V_{n+1}(x), \quad \forall x \in \mathcal{X},\ a \in \mathbb{A}, \\ V_n(x) & = & \inf\limits_{a\in\mathbb{A}} Q_n(x,a), \qquad \text{pour } n = N-1, \dots, 0, \end{array}\right. \tag{2.1.3}$$

et de plus, le contrôle feedback optimal au temps n, noté α^*_n, est caractérisé de la manière suivante :

$$\alpha^*_n(x) \in \operatorname*{argmin}_{a\in\mathbb{A}} Q_n(x,a). \tag{2.1.4}$$

La fonction Q_n est communément appelé la "Q-value" ou encore "state-action function" dans la communauté de l'apprentissage par renforcement (RL), et V_n est la fonction valeur au temps n.

Remark 2.1.1. *Le schéma (2.1.3) est très général dans le sens où il peut approximer la solution de problème à horizon fini ou infini, à temps continu ou discret. Par exemple, considérons le problème à horizon infini et temps continu suivant :*

$$\mathscr{V}(x) := \sup_{\alpha\in\mathcal{C}} \mathbb{E}\left[\int_0^\infty e^{-\lambda t} f(X^\alpha_t)dt\right],$$

avec $\lambda > 0$, et où l'on suppose que le processus contrôlé X^α par un processus admissible α admet la dynamique suivante :

$$dX^\alpha_t = b(t, X_t, \alpha_t)dt + \sigma(t, X_t, \alpha_t)dW_t,$$

avec b et σ assez régulières. Faisons une discrétisation en temps : en notant $h \ll 1$ le pas de discrétisation en temps, on obtient l'approximation suivante :

$$\mathscr{V}(x) \approx \mathscr{V}_h(x) := \sup_{\alpha\in\mathcal{C}} \mathbb{E}\left[\sum_{n=0}^\infty h e^{-\lambda nh} f(X^\alpha_{nh})\right],$$

où la dynamique de la chaîne de Markov contrôlée $(X^\alpha_{nh})_{n\in\mathbb{N}}$ s'écrit :

$$\left\{\begin{array}{rcl} X^\alpha_0 & = & X_0 \\ X^\alpha_{h(n+1)} & = & b(nh, X^\alpha_{nh}, \alpha_{nh})h + \sigma(nh, X_{nh}, \alpha_{nh})\sqrt{h}\varepsilon_{n+1}, \end{array}\right.$$

avec $(\varepsilon_n)_{n\in\mathbb{N}}$ une suite i.i.d. de loi normale centrée réduite $\mathcal{N}(0,1)$.
On reconnaît un problème à temps discret et horizon infini. Sous des hypothèses standards[3], on montre qu'alors $\mathscr{V}_h$ est l'unique point fixe de l'opérateur

$$\begin{array}{rcl} \mathcal{T} : \mathcal{C}^0(\mathbb{R}) & \longrightarrow & \mathcal{C}^0(\mathbb{R}) \\ v & \longmapsto & \left(x \longmapsto \sup\limits_{a\in\mathbb{A}} \left\{r(x) + e^{-\lambda h}\mathbb{E}\left[v(X^a_h)|X_0 = x\right]\right\}\right), \end{array}$$

où l'on a posé $r(x) := hf(x)$. Le théorème du point fixe de Picard montre que l'on peut approximer le point fixe de $\mathcal{T}$ en itérant un nombre de fois N assez grand l'opérateur $\mathcal{T}$ en partant de la fonction

[2] Le schéma (2.1.3) est communément appelé équation de Bellman dans la littérature du contrôle stochastique.
[3] sous lesquelles, en particulier, $\mathcal{T}$ est contractant.

nulle. En utilisant cet argument, on montre que V_0, défini dans le schéma (2.1.5) ci-dessous, converge vers le point fixe de $\mathcal{T}$ quand N tend vers l'infini.

$$\begin{cases} V_N &= 0 \\ V_n(x) &= \sup_{a\in\mathbb{A}} \left\{ r_n(x) + e^{-\lambda h}\mathbb{E}\left[V_{n+1}(X_h^a)|X_0 = x\right]\right\}, \quad \textit{for all state } x \end{cases} \tag{2.1.5}$$

On reconnaît alors le schéma rétrograde (2.1.3), avec $\beta = e^{-\lambda h}$, $g = 0$, $f = r_n$.

Remark 2.1.2. *Dans le cas où la fonction valeur est point fixe d'un opérateur, il existe d'autres algorithmes issu de l'apprentissage par renforcement, comme Temporal Difference Learning ou le Deep Q-learning, et qui marchent beaucoup mieux que la simple utilisation d'un argument de Picard que nous venons de présenter, pour approximer cette dernière par le schéma (2.1.5). Nous référons à [SB98] pour une introduction exhaustive de ces derniers, et référons à [MKSR15] pour l'apprentissage du contrôle optimal pour battre les joueurs professionnels à des jeux vidéos comme ATARI, en utilisant un algorithme basé sur le Q-learning. Enfin, nous citons [Sil+16] et [Sil+17] où est expliqué comment jouer mieux que les joueurs professionnels au jeu de go en utilisant notamment des méthodes de Monte Carlo Tree Search pour explorer efficacement l'espace de contrôle.*

Le problème (2.1.3) est en théorie beaucoup plus simple à résoudre que (2.1.2) puisque l'infimum est maintenant pris sur $\mathbb{A}$ qui est un sous-ensemble $\mathbb{R}^q$, de dimension finie, et nous avons de plus la caractérisation pratique (2.1.4) du contrôle optimal qui accompagne (2.1.3). Pour autant, dans la plupart des cas, l'équation de Bellman (2.1.3) pose encore deux problèmes :

PB1 l'espérance conditionnelle P^a qui apparaît à chaque pas de temps dans l'expression de Q_n de (2.1.3) n'admet généralement pas d'expression explicite.

PB2 l'infimum dans l'expression de V_n de (2.1.3) n'admet généralement pas de formule fermée, et une approximation de cette dernière peut se révéler très coûteuse en temps de calculs (resp. impossible), dans le cas où le cardinal de l'espace d'états est très grand (resp. infini).

2.1.1.2 Algorithmes issus de la littérature pour les équations de Bellman (Chapitre 4)

Pour résoudre l'équation de Bellman, nous nous sommes intéressés dans la première partie de la thèse à des méthodes basées sur la régression et la quantification.

Regress-Now sur une base de fonctions La méthode regress-now sur une base de fonctions consiste à d'abord randomiser le contrôle, i.e. simuler le contrôle à partie d'un processus aléatoire exogène, puis à régresser la fonction valeur au temps $n+1$ sur la base de fonctions au temps n pour estimer la Q valeur au temps n. Il suffit alors d'optimiser la Q valeur au temps n pour obtenir une estimée de la fonction valeur à la date n. Nous référons à [KLP14], [CL10] et aux travaux plus récents dans [LM18] pour des méthodes basées sur la méthode regress-now pour résoudre numériquement des problèmes de contrôle.

L'une des principale difficulté lorsque l'on cherche à appliquer un algorithme du type regress-now est le choix de la base de fonctions : plus la dimension du problème de contrôle est grande, moins il est facile de trouver une bonne base de fonctions pour régresser la fonction valeur en évitant le sous ou sur-apprentissage. Pour pallier à ce problème, une idée développée dans [BSST18] consiste à adapter la base de fonctions $\mathcal{B}_n$ sur laquelle régresser V_{n+1}, la fonction valeur au temps $n+1$, au temps n, en ajoutant des fonctions du type V_{n+1} dans $\mathcal{B}_n$. En faisant ainsi, l'estimation de l'espérance conditionnelle est meilleure dans le cas courant où la MDP est issue d'un problème de contrôle en temps continu dans lequel on a la continuité de la fonction valeur par rapport au temps.

Regress-Later La méthode regress-later consiste à d'abord interpoler la fonction valeur au temps $n+1$ sur une base de fonctions, puis à régresser l'interpolation[4]. Nous référons à [BP17] pour une présentation rigoureuse théorique de la méthode regress-later appliquée aux problèmes de contrôle stochastique à horizon fini et temps discret. Cette méthode est particulièrement efficace dans le cas

[4] et non la fonction valeur au temps $n+1$!

où la régression des fonctions de base peut se faire de manière analytique, i.e. on possède des formules fermées pour le calcul des espérances conditionnelles, car alors la régression est quasi-instantanée, mais aussi ca évite d'avoir à surveiller une erreur supplémentaire due à la régression (et veiller à ne pas qu'elle explose). Nous référons à [Ala+19] pour une comparaison des algorithmes regress-later avec regress-now sur un exemple concret de management de micro-grille, dans lequel regress-later propose de meilleures estimées. Dans le cas où il n'existe pas de formule fermée pour la régression des fonctions de base choisies, on peut toujours approximer ces régressions par différentes méthodes comme la quantification.

Motivation

Il est surprenant de constater qu'aujourd'hui, beaucoup des algorithmes issus de la littérature sont basés sur des méthodes de régression Monte Carlo, i.e. opèrent une régression globale. En moyenne dimension, il est clair que c'est la seule méthode qui marche numériquement. Mais en petite dimension, d'autres méthodes de régression, de type locales, sont disponibles dans la littérature de statistique, et adaptables aux problèmes de contrôle comme détaillé par exemple dans [BKS10].
Aussi, les méthodes de quantification semblent adaptées pour résoudre les MDPs, comme montré dans [PPP04b].
Dans la première partie de cette thèse, nous voulons mixer les méthodes de régression locale avec les méthodes de quantification, dans le but de pouvoir résoudre des MDPs en moyenne dimension.

L'algorithme que nous proposons repose sur une méthode de régression locale, et semble adaptée au problème de market-making présenté à la Section 2.1.2.

2.1.1.3 Contributions

Nous proposons pour résoudre le problème de market-making, un algorithme baptisé Qknn, basé sur des techniques de quantification de variables aléatoires, pour approximer les espérances conditionnelles, et de régression locale (i.e. projection sur les plus proches voisins). La régression locale en utilisant les plus proches voisins a déjà été proposée pour résoudre des problèmes de contrôle, comme présenté par exemple dans [BKS10]. Comme on peut le voir dans le pseudo-code de Qknn, écrit dans Algorithme 2.1 : à chaque pas de temps n, l'agent choisit une famille finie de points appelé "training points" sur lesquels estimer la fonction valeur au temps n. Le calcul de l'espérance conditionnelle est fait en quantifiant le bruit exogène, i.e. en approximant la variable exogène de bruit ε_{n+1} par une variable aléatoire $\hat{\varepsilon}_{n+1}$ qui prend un nombre d'états fini $e_1, \dots, e_\ell$ avec probabilité $p_1, \dots, p_\ell$; choisie de telle manière que la norme $\mathbb{L}^2$ de la différence entre ε_{n+1} et $\hat{\varepsilon}_{n+1}$ soit aussi petite que possible[5].

Nous référons au Chapitre 3 pour une introduction plus développée de Qknn et de ses variantes. Dans ce même chapitre, nous présentons aussi le résultat de convergence obtenu pour Qknn (voir Théorème 3.4.1 page 78), dont nous proposons ci-dessous une version simplifiée :

Theorem 2.1.1. *En prenant $K = M^{2+d}$ points pour la quantification du bruit exogène ε_n, $n = 1, \dots, N$, et sous des hypothèses de régularité des coefficients, il existe une constante $C > 0$ telle que pour $n = 0, ..., N-1$,*

$$\|\hat{V}_n^Q(X_n) - V_n(X_n)\|_2 \;\leq\; C \sum_{k=n+1}^{N} \left(\varepsilon_k^{proj} + \varepsilon_k^Q\right) + \mathcal{O}\left(\frac{1}{M^{1/d}}\right),$$

quand $M \to +\infty$, où $\varepsilon_k^Q := \|\hat{\varepsilon}_k - \varepsilon_k\|_2$ est l'erreur de quantification et

$$\varepsilon_n^{proj} := \sup_{a \in A} \|\mathrm{Proj}_{n+1}\left(F(X_n, a, \hat{\varepsilon}_n)\right) - F(X_n, a, \hat{\varepsilon}_n)\|_2$$

est l'erreur de projection quand la décision a est prise au temps n.

Le théorème 2.1.1 stipule que la norme $\mathbb{L}^2$ de la différence entre l'estimation par Qknn de la fonction valeur et la fonction valeur est bornée par une quantité proportionnelle à la somme des erreurs de projection et des erreurs de quantification faites à chaque pas de temps. Ces deux types erreurs tendent vers 0 lorsque, respectivement, le nombre de point des grilles de projections et de quantification tendent vers l'infini.

[5] Nous référons par exemple à [GL00] pour l'existence d'une telle v.a.

Algorithm 2.1 Algorithme Qknn

Entrée :

- Γ_k, $k = 0, \dots, N$: grille de training points $\mathbb{R}^d$,
- $\Gamma = \{e_1, \dots, e_L\}$, $(p_\ell)_{1 \le \ell \le L}$: une L-grille optimale pour la quantification du bruit exogène ε, et ses poids associé.

1: Poser N : $\hat{V}^{\mathrm{Q}}_{\Delta t}(t_N, z) = g(z), \quad \forall z \in \Gamma_N$.
2: **for** $n = N-1, \dots, 0$ **do**
3: Estimer la fonction valeur à l'instant n :

$$\widehat{V}^{\mathrm{Q}}_{\Delta t}(t_n, z) = \max_{a \in A} \left[f(z,a)\,\Delta t + \sum_{\ell=1}^{L} p_\ell \widehat{V}^{\mathrm{Q}}_{\Delta t}\Big(t_{n+1}, \mathrm{Proj}_{\Gamma_{n+1}}\big(G_{\Delta t}(z,a,e_\ell)\big)\Big)\right], \quad \forall z \in \Gamma_n.$$

4: **end for**

Sortie :

– $\big(\hat{\alpha}(t_n, z)\big)_{z \in \Gamma_n, 0 \le n \le N-1}$: estimation de la stratégie optimale ;
– $\big(\hat{V}^{\mathrm{Q}}_{\Delta t}(0, z)\big)_{z \in \Gamma_0}$: estimation de la fonction valeur ;

2.1.2 Application des MDPs au market-making (Chapitre 3)

Nous nous intéressons dans cette section à un problème de market-making, que nous résolvons théoriquement en utilisant la théorie des MDPs, et numériquement en utilisant Qknn. Nous présentons d'abord le problème, puis nos contributions théoriques et numériques, disponibles dans le Chapitre 3.

2.1.2.1 Présentation du problème de market-making

On considère un titre financier qui s'échange sur un marché financier, ainsi qu'un carnet d'ordres, qui est un recueil de toutes les propositions de ventes et d'achats de ce titre. Les participants du marché observent le carnet d'ordres et peuvent à tout moment décider de :

- acheter ou vendre le titre au meilleur prix (on parle d'ordre au marché)
- émettre de nouvelles propositions de vente ou d'achat de l'actif à un prix donné (on parle d'ordres limites)
- annuler leurs ordres limites avant que ces derniers n'aient été exécutés (on parle d'ordre d'annulation)

Les prix auxquels les participants ont la possibilité d'émettre des ordres sont divisés en ticks. Un market-maker est une personne qui agit sur un carnet d'ordres en émettant des propositions de vente ou d'achat un titre à certains prix (i.e. il n'émet que des ordres limites et des ordres d'annulation). La fonction principale de ce dernier est de liquéfier le marché, par ses propositions de vente et d'achat à différent prix. Une stratégie suivie par le market-maker, appelée stratégie de market-making, consiste à émettre des ordres limites de vente et d'achat sur le marché pour le liquéfier, tout en maximisant une quantité d'intérêt propre au market-maker, comme sa richesse terminale par exemple.

Remark 2.1.3. *Il faut différencier un ordre limite qui est un ordre qu'un participant envoie pour proposer d'acheter ou de vendre le titre à un certain prix, d'un ordre au marché qui est un ordre envoyé par un participant pour acheter ou vendre le titre (au meilleur prix). Il n'y a pas de coût de transaction lorsque le participant envoie un ordre limite au marché (parce qu'essentiellement, le marché considère que ce dernier liquéfie le marché), alors que l'emission d'un ordre au marché s'accompagne de frais de transaction.*

Modèle et représentation mathématique du problème

Soit $K \geq 1$. On modélise un carnet d'ordres par K limites du côté ask (ask-side), et K limites du côté bid (bid-side), comme proposé dans [Abe+16]. On note pa_t le *best-ask* à l'instant t, qui est le prix le moins cher auquel le marché est prêt à vendre l'actif à l'instant t, et par pb_t le *best-bid* à l'instant t, qui est le prix le plus cher auquel un participant du marché est prêt à acheter l'actif à l'instant t. On représente le carnet d'ordres par le couple $\big(\underline{a}_t, \underline{b}_t\big) = \big(a_t^1, ..., a_t^K, b_t^1, \dots, b_t^K\big)$ où

- a_t^i est le nombre d'ordres limites de vente i ticks plus cher que pb_t.
- b_t^i est le nombre d'ordres limites d'achat i ticks moins cher que pa_t.

Le carnet d'ordres peut recevoir à tout moment trois types d'ordre émis par les participants et qui modifient le couple $(\underline{a}_t, \underline{b}_t)$: des ordres de vente ou d'achat au marché, des ordres limites de vente ou d'achat ; ou bien des ordres d'annulations sur l'ask-side ou le bid-side. On modélise les arrivées de ces ordres par des processus ponctuels.

On considère maintenant un market-maker qui observe le carnet d'ordres et peut choisir à tout moment envoyer des ordres limites au marché. On appelle $\mathbb{A}$ l'ensemble des stratégies admissibles de ce dernier, i.e. les processus $\mathbb{F}$-prévisibles, où $\mathbb{F}$ est la filtration engendré par les processus d'arrivée d'ordres, qui rendent compte des positions des ordres limites du market maker sur le carnet d'ordres. Dans le Chapitre 3, on note Z ce que l'on a appelé le "carnet d'ordres contrôlé", et qui est le processus contrôlé qui comprend l'ensemble les ordres limites de tous les participants ainsi que les ordres limites du market-maker. Z est un processus ponctuel qui vit dans un espace d'états discret que l'on appelle E lorsque l'on considère que les ordres limites ne sont pas divisibles.

Le but du market-maker est de suivre une stratégie de market-making, dite optimale, qui maximise la quantité suivante :

$$V(t,z) = \sup_{\alpha \in \mathbb{A}} \mathbb{E}^{\alpha}_{t,z}\left[\int_t^T f\big(\alpha_s, Z_s\big)\,\mathrm{d}s + g\big(Z_T\big)\right], \quad (t,z) \in [0,T] \times E,$$

où g est la récompense terminale (par exemple la richesse à la date terminale $T > 0$ du market-maker plus une pénalisation de l'inventaire terminal) ; et où f est la récompense instantanée (par exemple l'aversion au risque du market-maker).

2.1.2.2 Contributions

Contributions théoriques :

Dans la Section 3.3.2 du Chapitre 3, nous montrons[6] que le problème de market-making en temps continu (le processus contrôlé Z et la fonction valeur V) peut être réécrit comme un Processus de Décisions Markoviennes (MDP). Cette réécriture permet alors[7] de caractériser V, la fonction valeur du problème de market-making en temps continu, comme point fixe de l'opérateur $\mathcal{T}$ défini ainsi :

$$\begin{array}{rccc} \mathcal{T}: & \mathcal{C}^0([0,T]\times E) & \longrightarrow & \mathcal{C}^0([0,T]\times E) \\ & v & \longmapsto & (t,x) \mapsto \sup\limits_{a \in A_z} \Big\{ r(t,z,a) + \mathbb{E}\left[v(T_{n+1}, Z_{n+1}) | T_n = t, Z_n = z, a\right] \Big\}, \end{array}$$

où r est la récompense définie ci-dessous :

$$r(t,z,a) := -f\big(z,a\big)e^{-\lambda(z)(T-t)}(T-t)\mathbf{1}_{t>T} + f\big(z,a\big)\frac{1-e^{-\lambda(z)(T-t)}}{\lambda(z)} + e^{-\lambda(z)(T-t)}g(z)\mathbf{1}_{t\leq T},$$

avec λ l'intensité de saut du processus ponctuel des ordres limites. Plus précisément, nous montrons dans la Section 3.3 du Chapitre 3 le théorème suivant :

Theorem 2.1.2. *$\mathcal{T}$ admet un unique point fixe v qui coïncide avec la fonction valeur du problème de market-making.*

[6] voir Proposition 3.3.1 page 70
[7] voir Théorème 3.3.1 page 72

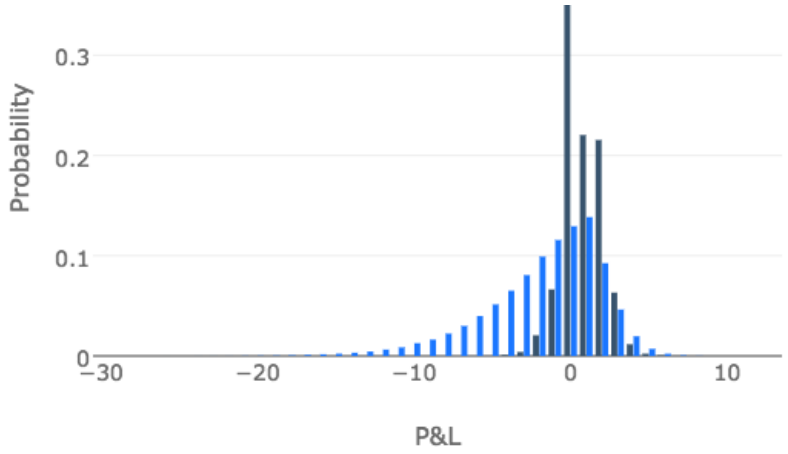


FIGURE 2.1 – Histogramme de la richesse terminale du market-maker lorsque celui-ci suit une bonne stratégie naïve (bleu clair), et la stratégie optimale estimée par Qknn (bleu foncé). On remarque que la richesse terminale est meilleure lorsque le market-maker suit les décisions estimées par Qknn, et ceci en limitant les grosses pertes.

Contributions numériques :

Remark 2.1.4. *L'espace d'états de la représentation MDP du problème de market-making est discret, sparse et grand ; et l'espace de contrôle est de cardinal fini et très petit. Les méthodes de type regress-now et regress-later, présentées ci-dessus ne semblent pas adaptées puisqu'il est difficile de choisir convenablement les fonctions de base à prendre lorsque l'espace d'états est discret, sparse et de dimension moyenne (≈10), pour éviter la sur-estimation et la sous-estimation.*

Dans la section 3.4.4 du Chapitre 3, nous présentons les résultats numérique de la résolution du problème de market-making en utilisant l'algorithme Qknn. Ce dernier semble adapté à ce problème, et en particulier, on observe qu'il fait mieux qu'une stratégie naïve de market-making, qui doit être vue comme un très bon benchmark. Nous illustrons dès maintenant le bon comportement de Qknn en proposant la figure 2.1, extraite du Chapitre 3, dans laquelle on a tracé l'histogramme de la richesse terminale du market-maker (i.e. son P&L), lorsque celui-ci suit soit la stratégie optimale estimée par Qknn, soit une bonne stratégie naïve. On observe que Qknn améliore la richesse clairement la richesse du market-maker, en minimisant les pertes.

2.1.3 Application des MDPs aux problèmes de contrôle de type McKean-Vlasov avec information partielle (Chapitre 4)

Nous proposons dans le Chapitre 4 une méthode basée sur le Markovian embedding pour réduire les problèmes de type McKean-Vlasov avec information partielle à des MDPs standards avec notamment un espace d'états de dimension finie. Des résultats numériques sont présentés pour divers problèmes de type McKean-Vlasov, où les méthodes regress-later, regress-now, Qknn et des méthodes des différence finie sont testées et comparées. Nous présentons d'abord les problèmes de contrôle de type McKean-Vlasov avec information partielle, qui est le cadre de cette section, puis nos contributions théoriques et numériques pour résoudre ces derniers.

2.1.3.1 Présentation des problèmes de contrôle de type McKean-Vlasov avec information partielle

On considère un problème de contrôle de type McKean-Vlasov (MKV) sous information partielle et bruit commun, qui est la donnée de deux mouvements Browniens indépendants B et W^0 définis sur

un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$, et d'un processus stochastique X dont la dynamique est contrôlée ainsi :

$$\mathrm{d}X_s = b(X_s, \mathbb{P}^{W^0}_{X_s}, \alpha_s)\,\mathrm{d}s + \sigma(X_s, \mathbb{P}^{W^0}_{X_s}, \alpha_s)\,\mathrm{d}B_s + \sigma_0(X_s, \mathbb{P}^{W^0}_{X_s}, \alpha_s)\,\mathrm{d}W^0_s, \quad X_0 = x_0 \in \mathbb{R}^n,$$

où $\mathbb{P}^{W^0}_{X_s}$ est la distribution de X_s conditionnée à $\mathcal{F}^0_s$ où $\mathbb{F}^0 = (\mathcal{F}^0_t)_t$ est la filtration générée par W^0, et où le contrôle α est un processus $\mathbb{F}^0$-progressif à valeur sur un espace polonais A. La condition de mesurabilité sur le contrôle signifie que l'agent n'observe que partiellement le processus, i.e. il observe le bruit commun.

Nous faisons les hypothèses Lipschitz standards sur les coefficients $b(x,\mu,a)$, $\sigma(x,\mu,a)$, $\sigma_0(x,\mu,a)$ par rapport à (x,μ) dans $\mathbb{R}^n \times \mathcal{P}_2(\mathbb{R}^n)$, uniformement dans $a \in A$, où $\mathcal{P}_2(\mathbb{R}^n)$ est l'espace des mesures de probabilité dans $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n))$ avec un second moment fini, et muni de la distance de Wasserstein $\mathcal{W}_2$. Cela implique que le MKV contrôlé défini dans (4.1.1) est bien défini. La fonctionnelle de coût, à horizon fini T associée au contrôle α est donnée par :

$$J(\alpha) = \mathbb{E}\Big[\int_0^T f(X_t, \mathbb{P}^{W^0}_{X_t}, \alpha_t)dt + g(X_T, \mathbb{P}^{W^0}_{X_T})\Big],$$

et l'objectif est de maximiser cette dernière sur un ensemble de contrôles admissibles $\mathcal{A}$:

$$V_0 = \sup_{\alpha \in \mathcal{A}} J(\alpha).$$

2.1.3.2 Contributions

Motivation Le Chapitre 4 a deux principaux objectifs : le premier est de montrer comment ramener des problèmes de contrôle de type McKean-Vlasov avec information partielle et bruit commun à des problèmes de contrôle fini-dimensionnel. Le deuxième objectif est de comparer différents algorithmes, et en particulier Qknn, sur de nombreux exemples de problèmes de type McKean-Vlasov.

Question 1 : Comment résoudre des problèmes de contrôle de dimension infini de type McKean-Vlasov avec information partielle et bruit commun ?

Réponse à la Question 1 :

Nous faisons deux types d'hypothèses sur les coefficients du modèle : la première est sur la dépendance en x, et l'autre sur la dépendance en μ.

Hypothèses sur la dépendance en x. On considère que le drift et les volatilités dans la dynamique de X sont linéaires par rapport à la variable d'état x, i.e.

$$\begin{cases} b(x,\mu,a) &= b_0(\mu,a) + b_1(\mu,a)x, \\ \sigma(x,\mu,a) &= \vartheta_0(\mu,a) + \vartheta_1(\mu,a)x, \\ \sigma_0(x,\mu,a) &= \gamma_0(\mu,a) + \gamma_1(\mu,a)x, \end{cases}$$

Nous considérons aussi que la récompense terminale et la récompense instantanée sont polynomiaux en x

$$\begin{aligned} f(x,\mu,a) &= f_0(\mu,a) + \sum_{k=1}^{p} f_k(\mu,a)x^k, \\ g(x,\mu) &= g_0(\mu) + \sum_{k=1}^{p} g_k(\mu)x^k, \end{aligned}$$

avec $p \geq 1$.

Hypothèses sur la dépendance en μ : Nous supposons que les coefficients du modèle dépendent de μ à travers les p premiers moments de cette dernière, i.e.,

$$\begin{cases} b_0(\mu,a) = \bar{b}_0(\bar{\mu}_1, \bar{\mu}_2, \ldots, \bar{\mu}_p, a), & b_1(\mu,a) = \bar{b}_1(\bar{\mu}_1, \bar{\mu}_2, \ldots, \bar{\mu}_p, a) \\ \vartheta_0(\mu,a) = \bar{\vartheta}_0(\bar{\mu}_1, \bar{\mu}_2, \ldots, \bar{\mu}_p, a), & \vartheta_1(\mu,a) = \bar{\vartheta}_1(\bar{\mu}_1, \bar{\mu}_2, \ldots, \bar{\mu}_p, a) \\ \gamma_0(\mu,a) = \bar{\gamma}_0(\bar{\mu}_1, \bar{\mu}_2, \ldots, \bar{\mu}_p, a), & \gamma_1(\mu,a) = \bar{\gamma}_1(\bar{\mu}_1, \bar{\mu}_2, \ldots, \bar{\mu}_p, a) \\ f_k(\mu,a) = \bar{f}_k(\bar{\mu}_1, \bar{\mu}_2, \ldots, \bar{\mu}_p, a), & g_k(\mu) = \bar{g}_k(\bar{\mu}_1, \bar{\mu}_2, \ldots, \bar{\mu}_p), \quad k = 0, \ldots, p, \end{cases}$$

où, pour $\mu \in \mathcal{P}_p(\mathbb{R})$, on note

$$\bar{\mu}_k \quad = \quad \int x^k \mu(dx), \quad k = 1, \dots, p.$$

Soit Y le processus suivant :

$$Y_t^{(k)} \quad = \quad \mathbb{E}[X_t^k | W^0], \quad k = 1, \dots, p.$$

En supposant pour simplifier que $n = 1$ et en utilisant la formule d'Itô et sous les hypothèses ci-dessus, on obtient les formules suivantes sur la dynamique de Y

$$\begin{aligned} dY_t^{(k)} &= B_k(Y_t^{(1)}, Y_t^{(2)}, \dots, Y_t^{(p)}, \alpha_t)dt + \Sigma_k(Y_t^{(1)}, Y_t^{(2)}, \dots, Y_t^{(p)}, \alpha_t)dW_t^0, \\ Y_0^{(k)} &= x_0^k, \quad k = 1, \dots, p, \end{aligned}$$

où l'on note $y_0 = 1$, $y_{-1} = 0$,

$$\begin{aligned} B_k(y_1, y_2, \dots, y_p, a) \quad = \quad & k\bar{b}_0(y_1, \dots, y_p, a)y_{k-1} + k\bar{b}_1(y_1, \dots, y_p, a)y_k \\ & + \frac{k(k-1)}{2}(\bar{\vartheta}_0(y_1, \dots, y_p, a))^2 y_{k-2} + \frac{k(k-1)}{2}(\bar{\vartheta}_1(y_1, \dots, y_p, a))^2 y_k \\ & + k(k-1)\bar{\vartheta}_0(y_1, \dots, y_p, a)\bar{\vartheta}_1(y_1, \dots, y_p, a)y_{k-1} \\ & + \frac{k(k-1)}{2}(\bar{\gamma}_0(y_1, \dots, y_p, a))^2 y_{k-2} + \frac{k(k-1)}{2}(\bar{\gamma}_1(y_1, \dots, y_p, a))^2 y_k \\ & + k(k-1)\bar{\gamma}_0(y_1, \dots, y_p, a)\bar{\gamma}_1(y_1, \dots, y_p, a)y_{k-1}, \quad k = 1, \dots, p, \\ \Sigma_k(y_1, y_2, \dots, y_p, a) \quad = \quad & k\left(\bar{\gamma}_0(y_1, \dots, y_p, a)y_{k-1} + \bar{\gamma}_1(y_1, \dots, y_p, a)y_k\right), \quad k = 1, \dots, p, \end{aligned}$$

et la fonctionnelle de coût associée à Y s'écrit :

$$J(\alpha) \quad = \quad \mathbb{E}\Big[\int_0^T \bar{f}(Y_t^{(1)}, Y_t^{(2)}, \dots, Y_t^{(p)}, \alpha_t)dt + \bar{g}(Y_T^{(1)}, Y_T^{(2)}, \dots, Y_T^{(p)})\Big].$$

Le problème de type McKean-Vlasov est ainsi réduit à un problème fini-dimensionnel où le processus contrôlé est maintenant $(Y^{(1)}, Y^{(2)}, \dots, Y^{(p)})$, ce qui répond à la Question 3.

Applications numériques à des problèmes de contrôle MKV avec informations partielles Nous référons à la Section 4.4 du Chapitre 4 pour des exemples de problèmes de contrôle de type McKean-Vlasov et d'informations partielles que nous avons résolus en utilisant différents algorithmes comme Qknn, regress-now, regress-later, ainsi que des méthodes de différence finie. Ce travail a permis notamment d'illustrer le haut potentiel de Qknn[8] pour résoudre des problèmes de contrôle en petite dimension.

Nous avons considéré un problème de liquidation de portefeuille, un problème de sélection de portefeuille (voir section 4.4.1.2) ; ainsi qu'un problème de risque systémique interbancaire (voir section 4.4.2). Les résultats numériques sont disponibles dans les tableaux 4.1, 4.2 et 4.3, pages 128, 131 et 134. Pour illustrer ce propos, nous présentons dès maintenant le tableau 2.1 qui est tiré de la section 4.4.2.2 (voir tableau 4.3 page 134), et où pour différent jeux de paramètres, nous testons différents algorithmes sur le problème de risque systémique. L'algorithme qui donne le résultat le plus petit est le plus précis, et comme on peut le voir, c'est Qknn qui est le plus précis.

2.2 Réseaux de neurones pour les MDPs

Le cadre dans lequel on se place dans la seconde partie de la thèse est le même que celui de la première partie. Nous référons à la Section 2.1.1.1 page 30 pour une présentation rigoureuse du cadre, et n'en

[8] et donc des méthodes de régression locale et de quantification

ρ	RLMC	CR	Q	Bench
0.2	8.73	8.98	8.69	8.77
0.4	8.02	8.25	7.91	8.06
0.6	6.93	6.97	6.68	6.79
0.8	4.86	4.82	4.62	4.67

$c = 100$ and $\eta = 10$.

c	RLMC	CR	Qknn	Bench
1	7.88	7.87	7.86	7.88
5	8.22	8.23	8.21	8.23
25	9.69	9.76	9.61	9.62
50	11.08	11.27	10.94	10.97

$\rho = 0.5$ and $\eta = 100$.

TABLE 2.1 – Estimations de la fonction valeur du problème de risque systémique pour différent jeux de paramètres ; en utilisant regress-later (RLMC), un algorithme de randomisation du contrôle (CR), Qknn ainsi qu'une méthode de discrétisation d'EDP. Plus la valeur est petite, plus l'algorithme est performant : ici Qknn n'est jamais battu !

rappelons ici que les grandes lignes : nous nous intéressons à un problème de contrôle stochastique à temps discret et horizon fini (MDP), i.e. nous cherchons

$$V_n(x) := \inf_{\alpha \in \mathcal{C}} \mathbb{E}\left[\sum_{k=n}^{N-1} f(X_k^\alpha, \alpha_k) + g(X_N^\alpha) \middle| X_n = x \right],$$

où l'horizon N est fixé, où le processus contrôlé $X^\alpha = (X_n^\alpha)_{n=0}^N$ admet la dynamique suivante

$$\begin{cases} X_0^\alpha &= x_0 \in \mathbb{R}^d, \\ X_{n+1}^\alpha &= F_n(X_n^\alpha, \alpha_n, \varepsilon_{n+1}), \quad \text{pour } n = 0, \dots, N-1, \end{cases}$$

avec $(\varepsilon_n)_{n=1}^N$ le bruit i.i.d. qui génère la filtration $\mathbb{F} = (\mathcal{F}_n)_{n=0}^N$; et où $\mathcal{C}$ est l'ensemble des contrôles admissibles.
Par un principe de programmation dynamique, la fonction valeur à l'instant n, pour $n = 0, \dots, N$, notée V_n, est caractérisée comme solution du schéma rétrograde suivant :

$$\begin{cases} V_N(x) &= g(x), \quad \forall x \in \mathcal{X}, \\ Q_n(x,a) &= f(x,a) + P^a V_{n+1}(x), \quad \forall x \in \mathcal{X},\ a \in \mathbb{A}, \\ V_n(x) &= \inf_{a \in \mathbb{A}} Q_n(x,a), \quad \text{pour } n = N-1, \dots, 0, \end{cases} \tag{2.2.1}$$

Nous proposons dans la seconde partie de la thèse de résoudre le problème de contrôle en utilisant des méthodes à base de réseaux de neurones. Les réseaux de neurones étant les meilleurs pour apprendre des fonctions en grande dimension, on s'attend à ce que ces méthodes soient très adaptées à ce cadre, i.e. à des dimensions $d \gg 1$. Dans la section qui suit, nous introduisons les réseaux de neurones considérés, ainsi que la méthode de descente de gradient, fondamentale pour entraîner ces derniers.

2.2.1 Représentation de fonctions par réseaux de neurones & apprentissage

Soit $L \geq 1$. On définit un perceptron à L couches comme la fonction paramétrée Φ :

$$x \in \mathcal{X} \longmapsto \Phi(z;\theta) \in \mathbb{R}^o,$$

qui est la composée de fonctions simples paramétrées[9], avec $o \in \mathbb{N}$, et où $\theta \in \Theta \subset \mathbb{R}^p$ est appelé le paramètre du perceptron Φ. Formellement, $\Phi = \Phi_L$, où Φ_L est définie par récurrence :

$$\begin{cases} \Phi_1(x) &= \sigma(A_1 x + b_1) \\ \Phi_{n+1} &= \sigma(A_{n+1}\Phi_n + b_n) \quad \text{for } 1 \leq n \leq L-1, \end{cases}$$

où σ une fonction monotone appelée *fonction d'activation* dans la littérature[10] ; où $A_1, \dots, A_L$ sont des matrices, et $b_1, \dots, b_L$ des vecteurs, dont les dimensions sont telles que le produits matricielles

[9] ces fonctions sont souvent appelées les "couches" (ou layer, en anglais) du réseau de neurones dans la littérature de l'apprentissage profond.

[10] Des exemples classiques de fonctions d'activation sont : la fonction ReLu : $x \mapsto \max(x, 0)$, la fonction ELU : $x \mapsto \begin{cases} \exp(x) - 1 & \text{si } x \leq 0 \\ x & \text{sinon} \end{cases}$, ou encore la fonction sigmoid $x \mapsto \frac{1}{1+\exp(-x)}$.

et somme de vecteurs soient bien définis ; et où pour tout vecteur y q-dimensionnel, on a utilisé la notation $\sigma(y) = (\sigma(y_1), \dots, \sigma(y_q))^\top$.

Les matrices $A_1, \dots, A_L$ sont appelés les poids du réseaux de neurones et les $b_1, \dots, b_L$ sont les biais. L'ensemble des poids et des biais forment les paramètres, que l'on note θ, du perceptron à L couches.

Les réseaux de neurones considérés dans cette thèse sont les perceptrons multicouches.

De manière générale : après avoir représenté la fonction d'intérêt par un réseau de neurones, nous devons apprendre les paramètres du réseau pour que ce dernier approxime au mieux la fonction d'intérêt. Dans la deuxième partie de cette thèse, l'approximation de la fonction d'intérêt passe par la minimisation d'une fonction objective définie sous forme d'une espérance. Nous introduisons dans la prochaine section les méthodes de descente de gradient stochastique (SGD) qui ont été créées pour ce type de tâches. Ces dernières sont très performantes et contribuent pour beaucoup au succès des réseaux de neurones en grande dimension.

Les méthodes d'optimisation d'espérance à paramètres

Considérons le problème qui consiste à minimiser par rapport à un paramètre θ l'espérance à paramètre suivante :

$$\mathscr{J}(\theta) \quad = \quad \mathbb{E}[G(X;\theta)], \qquad \text{avec } X \sim \mu, \tag{2.2.2}$$

où μ est une mesure de probabilité, et où G est une fonction différentiable et assez régulière pour que la dérivation sous le signe $\mathbb{E}$ de $\mathscr{J}$ soit licite. Ce type de problème est celui auquel nous faisons face lorsque l'on veut apprendre les paramètres optimaux pour représenter la fonction valeur ou le contrôle optimal dans les problèmes de contrôle.

Example 2.2.1. *Si l'on représente le contrôle au temps n par le réseau de neurones $A(.;\beta_n)$ et la fonction valeur au temps n par le réseau de neurones $\Phi(.;\theta_n)$; alors : en utilisant le schéma (2.2.1), le paramètre optimal β_n^* pour représenter le contrôle optimal au temps n vérifie :*

$$\beta_n^* = \operatorname*{argmin}_\beta \mathbb{E}\left[V_{n+1}(X_{n+1}^\beta) + f(X_n, A(X_n;\beta))\right],$$

et le paramètre optimal θ_n^* dont le réseau de neurone Φ représente le mieux V_n, la fonction valeur au temps n, vérifie :

$$\theta_n^* = \operatorname*{argmin}_\theta \mathbb{E}\left[\left(V_{n+1}(X_{n+1}^{\beta_n^*}) + f(X_n, A(X_n;\beta_n^*)) - \Phi(X_n;\theta)\right)^2\right].$$

La notation X_{n+1}^β dans les deux égalités ci-dessus signifie que le contrôle feedback $A(X_n;\beta$ est choisi au temps n. Ces deux problèmes sont bien des cas particuliers de (2.2.2).

Descente de gradient Une méthode standard et naturelle pour minimiser $\mathscr{J}$ est d'appliquer une méthode déterministe de descente de gradient dont l'algorithme 2.2 en est un pseudo-code rudimentaire.

Le calcul de $\hat{\partial}_\theta \mathscr{J}(\theta)$, qui donne la direction pour la descente de gradient, est très lourd en temps de calculs (notez que l'estimateur du gradient s'exprime comme somme de M termes !), ce qui rend l'algorithme 2.2 inutilisable. Pour alléger le temps de calculs, il est mis en place des algorithmes de descente de gradient encore plus stochastique, dont un pseudo-code est écrit dans Algorithme 2.3. Ce type d'algorithmes est beaucoup plus rapide, puisque l'estimateur stochastique $\hat{\partial}_\theta \mathscr{J}(\theta)$ qui donne une direction (stochastique) pour la descente de gradient est beaucoup très rapide à calculer (notez que l'estimateur du gradient s'exprime comme la somme d'un unique terme !).

Les algorithmes qui sont aujourd'hui le plus utilisés sont basés sur la descente de gradient stochastique. Citons les méthodes Adagrad, Adadelta, RMSprop et Adam, qui sont des améliorations (dans de nombreux cas pratique) bien connues dans la littérature du Machine-Learning, pour apprendre efficacement les paramètres optimaux, en jouant principalement sur un choix adapté du paramètre d'apprentissage η.

Algorithm 2.2 Descente de gradient

Input :
– paramètre d'apprentissage η ;
– un seuil $\varepsilon \ll 1$;
– Un training set $\{X_m, 1 \leq m \leq M\}$ avec $M \gg 1$;
– $\theta \in \mathbb{R}^q$;

1: **while** $|\hat{\partial}_\theta \mathscr{J}(\theta)| > \varepsilon$ **do**
2: Calculer

$$\hat{\partial}_\theta \mathscr{J}(\theta) := \hat{\mathbb{E}}[\partial_\theta V(X;\theta)],$$

où $\hat{\mathbb{E}}$ est l'espérance sous la mesure empirique $\frac{1}{M}\sum_{m=1}^{M} \delta_{X_m}$;
3: ▷ $\hat{\partial}_\theta \mathscr{J}(\theta)$ est un estimateur de la dérivée de $\mathscr{J}$ au point θ.
4: Re-assigner $\theta := \theta - \eta\, \hat{\partial}_\theta \mathscr{J}(\theta)$;
5: **end while**

Output : paramètre optimal θ ;

Algorithm 2.3 Descente de gradient stochastique

Input :
– paramètre d'apprentissage η ;
– un seuil $\varepsilon \ll 1$;
– un nombre d'itérations $M \gg 1$;
– un paramètre initial $\theta \in \mathbb{R}^q$;

1: **for** $m = 0, \ldots, M$ **do**
2: Tirer $X \sim \mu$;
3: Calculer

$$\hat{\partial}_\theta \mathscr{J}(\theta) := \partial_\theta V(X;\theta);$$

▷ $\hat{\partial}_\theta \mathscr{J}(\theta)$ est un estimateur de la dérivée de $\mathscr{J}$ au point θ.
4: Re-assigner $\theta := \theta - \eta\, \hat{\partial}_\theta \mathscr{J}(\theta)$;
5: **end for**

Output : paramètre optimal θ ;

Remark 2.2.1. *Dans cette thèse, nous nous sommes exclusivement servis de l'algorithme Adam, nativement implémenté dans Tensorflow, pour optimiser les espérances à paramètre.*

La descente de gradient stochastique est la méthode d'optimisation de l'espérance à paramètre $\mathscr{J}$ à qui les réseaux de neurones doivent certainement une grande partie de leur succès.

2.2.2 Revue de littérature sur les réseaux de neurones pour les MDPs

Nous passons en revue quelques méthodes numériques bien connues principalement dans la littérature de l'apprentissage par renforcement (RL) pour résoudre (2.1.2) en représentant les fonctions d'intérêt, comme les contrôles optimaux ou les fonctions valeur au temps $n = 0, \dots, N-1$, par des fonctions paramétrées.

Il existe deux grandes familles d'algorithmes pour résoudre (2.1.1) avec des réseaux de neurones. La première, que l'on appelle "méthodes directes", contient les algorithmes qui ne représentent que le contrôle optimal par un réseau de neurones, et reposent essentiellement sur les progrès récents qui ont été fait dans l'apprentissage des paramètres optimaux pour optimiser les "espérances à (très nombreux !) paramètres" (voir section 2.2.1). La deuxième famille, que l'on appelle "Acteur-critique", contient les algorithmes qui représentent le contrôle optimal ainsi que la Q-value par des réseaux de neurones, et qui reposent essentiellement sur la représentation (2.1.3) de la fonction valeur.

2.2.2.1 Méthodes directes

L'idée pour cette première famille d'algorithmes est de représenter le contrôle α_n au temps n par un réseau de neurones, pour tout $n = 0, \dots, N-1$; puis de voir que $J(\alpha)$ s'écrit alors comme une espérance paramétrique, qu'il suffit d'optimiser. Plus précisément, pour tout $n = 0, \dots, N-1$, représentons le contrôle feedback α_n par un réseau de neurones comme ci-dessous :

$$\alpha_n = A(X_n; \beta_n), \tag{2.2.3}$$

où $\beta_n \in \mathbb{R}^q$ est le paramètre, et réinjectons les représentations des contrôles dans (2.1.1). Le problème de contrôle (2.1.2) devient alors :

$$V_0(x_0) = \inf_{\beta_0, \dots, \beta_n \in \mathbb{R}^q} \mathbb{E}\left[\sum_{n=0}^{N-1} f(X_n, A_n(X_n, \beta_n)) + g(X_N) \right], \tag{2.2.4}$$

où la dynamique de (X_n) est contrôlée en utilisant les contrôles paramétriques (2.2.3). On reconnaît dans le membre de droite de (2.2.4) une intégrale à paramètre, où l'intégrale s'écrit sous forme d'une espérance, que l'on optimise efficacement en grande dimension par SGD, comme décrit dans la section 2.2.1.

Commentaires sur les méthodes directes

✓ pas d'approximation de la fonction valeur, donc pas de propagation d'erreur d'approximation de la fonction valeur à surveiller.

✗ Méthode très instable, surtout lorsque N est grand et que les contrôles à approximer sont des fonctions complexes. En effet, dans ce cas, il faut voir que l'optimisation de la représentation (2.2.4) revient à optimiser un réseau de neurones très profond (i.e. avec énormément de couches), ce qui est encore aujourd'hui un problème très challengeant.

2.2.2.2 Acteur-critique

Dans la deuxième famille d'algorithmes, on représente la Q-value ainsi que le contrôle par des réseaux de neurones, et l'on repose sur la représentation (2.1.3) de la fonction valeur. Plus précisement : pour tout $n = 0, \dots, N-1$, on pose

$$\alpha_n = A(X_n; \beta_n)$$

comme dans la section précédente ; et l'on pose aussi

$$Q(X_n, \alpha_n) = Q(X_n, A(X_n; \beta_n); \theta_n).$$

L'apprentissage des paramètres optimaux β_n^* et θ_n^*, pour $n = 0, \dots, N-1$ se fait de manière rétrograde comme décrit dans Algorithme 2.4. Il est à noter que l'on apprend les paramètres optimaux du contrôle optimal et de la Q-value en même temps dans ce type d'algorithme, i.e. on ne peut pas apprendre l'un puis l'autre ![11]

Algorithm 2.4 Acteur-critique

Input :
– paramètre d'apprentissage η ;
– un seuil $\varepsilon > 0$;
– initialiser au hasard $\theta_n, \beta_n \in \mathbb{R}^q$, pour $n = 0, \dots, N-1$;

Poser $V_N = g$;
for $n = N-1, \dots, 0$ **do**
  while $\max(\Delta\theta_n, \Delta\beta_n) > \varepsilon$ **do**
    Minimiser par rapport à θ_n la quantité suivante :

$$\mathbb{E}\left[(f(X_n, A(X_n; \beta_n)) + Q(X_{n+1}, A(X_{n+1}; \beta_{n+1}^*); \theta_{n+1}^*) - Q(X_n, A(X_n; \beta_n); \theta_n))^2\right],$$

    et réassigner θ_n comme son argmin ;
    Réassigner $\Delta\theta_n$ comme la différence de la valeur actualisée de θ_n et l'ancienne ;
    Minimiser par rapport à β_n la quantité suivante :

$$\mathbb{E}\left[f(X_n, A(X_n; \beta_n)) + Q(X_{n+1}, A(X_{n+1}; \beta_{n+1}^*); \theta_n)\right],$$

    et réassigner β_n comme son argmin ;
    Réassigner $\Delta\beta_n$ comme la différence de la valeur actualisée de β_n et l'ancienne ;
  end while
end for

Output :
- paramètres optimaux $(\theta_n^*)_{n=0}^{N-1}$ pour la représentation de la Q-value ;
- paramètres optimaux $(\beta_n^*)_{n=0}^{N-1}$ pour la représentation du contrôle optimal ;

Commentaires sur les méthodes acteur-critique

✓ L'apprentissage des paramètres optimaux des contrôles et fonctions valeurs aux temps $n = 0, \dots, N-1$ se fait un temps après l'autre et de manière rétrograde ; et de plus, on apprend les paramètres du contrôle et de la fonction valeur au temps n à partir de ceux des contrôle et fonction valeur au temps $n+1$.

✓ Si la MDP vient d'un problème de contrôle en temps continu dans lequel la fonction valeur et le contrôle optimal sont continus par rapport au temps, alors on peut énormément améliorer la qualité et la durée du training des réseaux de neurones en usant de la méthode de pré-training[12], i.e. initialiser les coefficients du contrôle et de la fonction valeur au temps n avec ceux du contrôle et de la fonction valeur au temps $n+1$, puis modifier ces derniers "légèrement"[13] en faisant de l'apprentissage.

✗ Une erreur d'approximation de la fonction valeur apparaît à chaque estimation de cette dernière par une fonction paramétrée. Cette erreur ce propage à cause du côté rétrograde de l'algorithme, et il faut donc veiller à ce qu'elle n'explose pas.

[11] Une caractérisation importante des nouveaux algorithmes que nous proposerons plus tard dans l'introduction, est que l'on a découplé l'apprentissage des deux paramètres optimaux.
[12] La méthode de pré-training porte aussi le nom de *transfer learning* dans la littérature du deep learning.
[13] en baissant le paramètre d'apprentissage η dans la SGD notamment.

2.2.3 Contributions (Chapitres 5 et 6)

Motivations : Comme nous venons de le voir, il existe déjà des algorithmes dans la littérature pour résoudre numériquement les problèmes de contrôle. Mais comme tout algorithme, ceux-ci présentent des défauts que nous cherchons à corriger, notamment le fait que l'apprentissage du contrôle optimal et de la fonction valeur soit couplé dans les méthodes acteur-critiques et que la programmation dynamique soit négligée dans les méthodes directes.

Question 2a : Comment améliorer l'apprentissage des contrôles optimaux, en reposant sur des arguments de programmation dynamique ?

Question 2b : Comment découpler l'apprentissage du contrôle optimal et de la fonction valeur ?

Nous proposons deux familles de nouveaux algorithmes dans cette thèse. La première famille regroupe ceux basés sur l'approximation du contrôle uniquement par réseaux de neurones, et reposent sur le principe de programmation dynamique pour alléger l'apprentissage des contrôles, répondant ainsi à la question 1a. La deuxième famille repose sur l'approximation du contrôle optimal et de la fonction valeur par réseaux de neurones et répondent à la question 1b.

Algorithmes qui répondent à la Question 2a Nous proposons les algorithmes NNcontPI et ClassifPI pour répondre à la question 2a. Nous présentons succinctement dès maintenant NNcontPI, et référons à la page 189 du Chapitre 6 pour la présentation du pseudo-code de ClassifPI.
L'idée dans NNcontPI, dont on propose un pseudo-code dans Algorithme 2.5, est de représenter le contrôle optimal à la date n par un réseau de neurones $A(.;\beta_n)$ et d'apprendre le paramètre optimal β_n^* de manière rétrograde, en reposant sur le principe de programmation dynamique et sur la notion de mesures de training $(\mu_n)_{n=0}^{N-1}$: supposons les paramètres optimaux β_k^* déjà appris pour $k = n+1, \dots, N-1$. Alors il vient :

$$\hat{\beta}_n \in \underset{\beta\in\mathbb{R}^q}{\operatorname{argmin}}\, \mathbb{E}\left[f\big(X_n, A(X_n;\beta)\big) + \sum_{k=n+1}^{N-1} f\big(X_k^\beta, \hat{a}_k\big(X_k^\beta\big)\big) + g\big(X_N^\beta\big)\right]$$

où X_n est tiré selon une mesure de "training" μ_n et où $\big(X_k^\beta\big)_{k=n+1}^N$ est définie par induction :

$$\begin{cases} X_n &\sim \mu_n, \\ X_{n+1}^\beta &= F\big(X_n, A\big(X_n;\beta\big), \varepsilon_{n+1}\big), \\ X_{k+1}^\beta &= F\big(X_k^\beta, \hat{a}_k\big(X_k^\beta;\beta\big), \varepsilon_{k+1}\big), \quad \text{for } k = n+1, \dots, N-1. \end{cases}$$

Commentaires sur NNcontPI

✓ Au temps n, seuls les paramètres de la représentation du contrôle au temps n sont appris, et ceci en utilisant les contrôles déjà appris aux temps $n+1, \dots, N-1$. Une excellente conséquence de cette propriété est que l'on peut améliorer énormément la qualité et le temps de training en usant de la méthode de pré-training.

✗ L'apprentissage du paramètre optimal β_n au temps n repose sur les mesures de training $\mu_k, k = n, \dots, N-1$, choisies par l'agent. Le choix des mesures de training peut s'avérer être une tâche difficile. Il est important[14] que la densité de la mesure de training μ_k choisie donne du poids dans les régions qui ont de fortes chances d'être visitées fréquemment par le processus contrôlé optimalement à l'instant k. Ceci implique que l'agent ait déjà une idée vague des régions où le processus optimal est susceptible d'aller[15].

✗ L'apprentissage des paramètres optimaux à la date n, en utilisant NNcontPI et ClassifPI, passe par la re-simulation de trajectoires de la date n à la date N, comme on peut le voir dans Algorithme 2.5 ; ce qui implique que ces méthodes sont très volatiles, surtout dans le cas où N est grand, ou que les récompenses f et g sont complexes.

Algorithm 2.5 NNContPI

Entrée : une suite de distributions d'entraînement $(\mu_n)_{n=0}^{N-1}$;

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

2: Compute

$$\hat{\beta}_n \in \underset{\beta \in \mathbb{R}^q}{\operatorname{argmin}}\, \mathbb{E}\left[f\big(X_n, A(X_n;\beta)\big) + \sum_{k=n+1}^{N-1} f\big(X_k^\beta, \hat{a}_k(X_k^\beta)\big) + g\big(X_N^\beta\big) \right]$$

où $X_n \sim \mu_n$ et où $\big(X_k^\beta\big)_{k=n+1}^{N}$ est définie par induction :

$$\begin{cases} X_{n+1}^\beta &= F\big(X_n, A\big(X_n;\beta\big), \varepsilon_{n+1}\big) \\ X_{k+1}^\beta &= F\big(X_k^\beta, \hat{a}_k\big(X_k^\beta;\beta\big), \varepsilon_{k+1}\big), \quad \text{for } k = n+1, \dots, N-1. \end{cases}$$

pour $m = 1, \dots, M$;

3: Pose $\hat{a}_n = A(.;\hat{\beta}_n)$; ▷ $\hat{a}_n$ est un estimateur du contrôle optimal à la date n

4: **end for**

Sortie : $(\hat{a}_n)_{n=0}^{N-1}$, estimateur des stratégies optimales aux instants n, pour $n = 0, \dots, N-1$;

Nous présentons dans le Chapitre 5 des résultats de convergence de l'algorithme NNcontPI en fonction de la taille de l'espace des réseaux de neurones ainsi que de la taille du training set.

Theorem 2.2.1 (Convergence de NNContPI)**.** *Supposons qu'il existe des contrôles optimaux feedbacks* $(a_k^{\mathrm{opt}})_{k=n}^{N-1}$ *et notons* V_n *les fonctions valeurs aux temps* $n = 0, \dots, N$*. Soit* $X_n \sim \mu$*. On a alors, lorsque* $M \to \infty$[16] *:*

$$\mathbb{E}\big[\hat{V}_n^M(X_n) - V_n(X_n)\big] = \mathcal{O}\Bigg(\frac{\rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}}{\sqrt{M}} + \sup_{n\le k\le N-1} \inf_{A\in\mathcal{A}_M} \mathbb{E}\left[\left| A(X_k) - a_k^{\mathrm{opt}}(X_k) \right| \right] \Bigg),$$

De plus, on a, quand $M \to \infty$[17] *:*

$$\mathbb{E}_M\big[\hat{V}_n^M(X_n) - V_n(X_n)\big] = \mathcal{O}_{\mathbb{P}}\Bigg(\rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}\sqrt{\frac{\log(M)}{M}} + \sup_{n\le k\le N-1} \inf_{A\in\mathcal{A}_M} \mathbb{E}\left[\left| A(X_k) - a_k^{\mathrm{opt}}(X_k) \right| \right] \Bigg),$$

où $\mathbb{E}_M$ *est l'espérance conditionnelle conditionnée au training set utilisé pour estimer les contrôles optimaux* $(\hat{a}_k^M)_{n\le k\le N-1}$*.*

Le théorème 2.2.1 stipule[18] que la fonction valeur au temps n, calculée à partir des estimations de la stratégie optimale du temps n au temps $N-1$, est proche de la vraie fonction valeur au temps n ; et la différence tend en moyenne vers 0 lorsque la taille du training set grandit et que l'espace des réseaux de neurones grossit. Ce résultat est intéressant notamment parce qu'il prend en compte la taille du training set utilisé dans la procédure de SGD, dans le résultat de convergence.

[14] surtout en grande dimension !

[15] Dans le cas où l'agent n'a aucune idée de quelle mesure de training choisir, il peut toujours utiliser des méthodes de bootstrapping. Nous référons à la section 4.3.3 page 118 pour plus de détails à ce sujet.

[16] La notation $x_M = \mathcal{O}(y_M)$ quand $M \to \infty$, signifie que le ratio $|x_M|/|y_M|$ est borné quand M tend vers l'infini.

[17] La notation $x_M = \mathcal{O}_{\mathbb{P}}(y_M)$ quand $M \to \infty$ signifie qu'il existe $c > 0$ telle que $\mathbb{P}\big(|x_M| > c|y_M|\big) \to 0$ quand M tend vers l'infini.

[18] remarquons que l'on a toujours $\hat{V}_n^M \ge V_n$, par définition.

Les algorithmes NNcontPI et ClassifPI sont testés numériquement sur des exemples dans le Chapitre 6 : ClassifPI obtient de très bons résultats dans l'exemple Valuation of Energy Storage présenté dans la section 6.3.3 page 200, même s'il se fait battre par Qknn[19] qui est certainement mieux adapté au problème (voir le tableau 6.2 page 203).

Algorithmes qui répondent à la question 2b Nous proposons les algorithmes Hybrid-Now, Hybrid-Later et Classif-Hybrid pour répondre à la question 2b. Nous référons aux Algorithmes 6.4 et 6.6 respectivement aux pages 192 et 209 pour un pseudo-code de Hybrid-Later et Classif-Hybrid, et choisissons de ne présenter ici que l'algorithme Hybrid-Now, dont le pseudo-code est disponible dans Algorithme 2.6.
Dans l'algorithme Hybrid-Now, nous représentons le contrôle et la fonction valeur au temps n par deux réseaux de neurones. Comme on peut le voir dans Algorithme 2.6, l'idée dans Hybrid-Now est d'abord d'apprendre le paramètre optimal du contrôle au temps n ; puis de reposer sur un principe martingale associé au processus contrôlé de manière optimale pour en déduire le paramètre optimal pour la représentation de la fonction valeur au temps n. Comme l'algorithme NNcontPI présenté précédemment, Hybrid-Now se base sur la donnée de mesures de training $(\mu_n)_{n=0}^{N-1}$ choisies par l'agent, où la densité de μ_n doit être prise grande dans les régions optimale où l'agent cherche à effectuer un apprentissage précis du contrôle optimal.

Algorithm 2.6 Hybrid-Now

Entrée : distributions de training $(\mu_n)_{n=0}^{N-1}$;

Set $\hat{V}_N = g$;
for $n = N-1, \dots, 0$ **do**
    Calculer :

$$\hat{\beta}_n \in \operatorname*{argmin}_{\beta \in \mathbb{R}^q} \mathbb{E}\Big[f\big(X_n, A(X_n;\beta)\big) + \hat{V}_{n+1}(X_{n+1}^{\beta}) \Big]$$

    où $X_n \sim \mu$, et $X_{n+1}^{\beta} = F\big(X_n, A(X_n;\beta)\big), \varepsilon_{n+1}\big)$;
    Assigner $\hat{a}_n = A(.;\hat{\beta}_n)$; ▷ $\hat{a}_n$ est un estimateur de la stratégie optimale au temps n ;
    Calculer

$$\hat{\theta}_n \in \operatorname*{argmin}_{\theta \in \mathbb{R}^p} \mathbb{E}\left[\Big(\big(f(X_n, \hat{a}_n(X_n)) + \hat{V}_{n+1}(X_{n+1}^{\hat{\beta}_n}) - \Phi(X_n;\theta)\big) \Big)^2 \right] ;$$

    Assigner $\hat{V}_n = \Phi(.;\hat{\theta}_n)$;
    ▷ $\hat{V}_n$ est un estimateur de la fonction valeur au temps n
end for

Sortie :
– estimateurs de la stratégie optimale : $(\hat{a}_n)_{n=0}^{N-1}$;
– estimateurs de la fonction valeur : $(\hat{V}_n)_{n=0}^{N-1}$;

Commentaires sur Hybrid-Now

✓ l'apprentissage des paramètres optimaux pour le contrôle et la fonction valeur au temps n est découplé dans Algorithme 2.6.

✓ L'apprentissage des paramètres optimaux pour la fonction valeur et le contrôle optimal au temps n est très rapide puisqu'il ne demande que la simulation du processus contrôlé du temps n au temps $n+1$[20].

✗ La représentation de la fonction valeur par un réseau de neurones implique l'apparition d'une erreur d'approximation et d'une erreur d'estimation supplémentaire, qu'il faut veiller à garder petite lorsque l'on exécute la boucle rétrograde de l'algorithme.

[19] L'algorithme Qknn a été présenté dans la Section 2.1.1.3
[20] et non de n à N comme dans l'algorithme NNcontPI.

Nous proposons, dans le Chapitre 5, une étude de la convergence de Hybrid-Now et Hybrid-Later, en fonction de la taille du training set et de l'espace de réseaux de neurones considéré. Nous ne mettons en avant ici seulement que le théorème 2.2.2 suivant obtenu pour la convergence de l'algorithme Hybrid-Now, que l'on retrouve page 158.

Theorem 2.2.2 (Convergence de Hybrid-Now)**.** *Supposons qu'il existe des contrôles optimaux qui soient feedback, i.e.* $(a_k^{\mathrm{opt}})_{k=n}^{N-1}$*, et notons par* V_n *les fonctions valeurs du problème au temps* $n = 0, \dots, N$*. Soit* $X_n \sim \mu$*. Alors, on a quand* $M \to +\infty$ *:*

$$\begin{aligned}
&\mathbb{E}_M\Big[|\hat{V}_n^M(X_n) - V_n(X_n)|\Big] \\
&\qquad = \mathcal{O}_{\mathbb{P}}\Bigg(\left(\gamma_M^4 \frac{K_M \log(M)}{M}\right)^{\frac{1}{2(N-n)}} + \left(\gamma_M^4 \frac{\rho_M^2 \eta_M^2 \log(M)}{M}\right)^{\frac{1}{4(N-n)}} \\
&\qquad\quad + \sup_{n \le k \le N} \inf_{\Phi \in \mathcal{V}_M} \left(\mathbb{E}_M\Big[|\Phi(X_k) - V_k(X_k)|^2\Big]\right)^{\frac{1}{2(N-n)}} \\
&\qquad\quad + \sup_{n \le k \le N} \inf_{A \in \mathcal{A}_M} \left(\mathbb{E}\Big[|A(X_k) - a_k^{\mathrm{opt}}(X_k)|\Big]\right)^{\frac{1}{2(N-n)}} \Bigg),
\end{aligned}$$

où $\mathbb{E}_M$ *est l'espérance conditionnée au training set qui a permis de construire les estimateurs* $\hat{a}_k^M$*, pour* $k = n, \dots, N-1$*.*

Le théorème 2.2.2 stipule que la différence entre la fonction valeur et l'approximation de cette dernière, calculée en utilisant les estimations du contrôle optimal par l'algorithme Hybrid-Now, tend en moyenne vers 0 lorsque la taille du training set et l'espace des réseaux de neurones tendent vers l'infini. Encore une fois, le résultat est intéressant parce qu'il prend en compte la taille du training set utilisé pour la SGD.

Nos algorithmes sont adaptés, ou facilement adaptables, à de nombreux types de problèmes de contrôle. C'est ce que nous avons tenté d'illustrer dans la section 6.3 du Chapitre 6, où nous avons étudié les exemples suivants :

- SemiLinearPDE, où l'espace d'états et de contrôle sont continus et de dimension 100.
- Option Hedging, où le processus contrôlé est discret.
- Valuation of Energy Storage, où l'espace de contrôle est fini, et l'espace d'états a une composante continue et une composante discrète.
- Smart Grid Management, où l'espace d'états est continu, et le contrôle admet une composante continue et une composante discrète.

En particulier Classif-Hybrid a donné d'excellents résultats dans l'exemple de Smart Grid Management, introduit dans la section 6.3.4 page 204, et que nous présentons succinctement ici. Le problème Smart Grid Management est celui que se pose un gestionnaire d'énergie (type EDF) qui a une batterie, de charge C, qu'il peut recharger avec de l'énergie solaire P qu'il reçoit, et qui doit subvenir à la demande D de ses consommateurs. Pour ce faire, ce dernier peut à tout temps choisir d'activer un générateur de courant et se délivrer la quantité d'énergie qu'il désire, ou bien de décharger sa batterie. L'agent peut recharger sa batterie avec l'énergie solaire ou bien avec le générateur de courant. Il peut aussi choisir de désactiver la batterie. Tout changement de mode de la batterie entraîne un coût de switching (ceci afin d'empêcher le gestionnaire d'activer et désactiver la batterie trop souvent pour ne pas détériorer la batterie). Le gestionnaire est pénalisé à chaque fois qu'il change le régime de la batterie (coût de switching), et lorsqu'il ne répond pas à la demande D du consommateur. Il paie de plus un coût lorsqu'il se fournit avec le générateur de courant, et celui-ci est proportionnel au courant délivré. Dans la figure 2.2 ci-dessous, extraite des figures 6.11 page 212 et 6.10 page 211, on peut voir la quantité optimale de courant à se faire délivrer par le générateur en fonction de la demande

résiduelle $R := D - P$; estimées soit par l'algorithme Classif-Hybrid, soit par un algorithme Qknn[21] qui fait figure de benchmark. Dans la figure 2.2, le cas $m = 1$ réfère à la situation où la batterie est déjà activée avant le temps 1 et où une désactivation au temps 1 entraînerait un coût de switching ; le cas où $m = 0$ réfère à la situation où la batterie est désactivée juste avant le temps 1 et où une activation au temps 1 entraînerait un coût de switching. Nous constatons que les décisions optimales estimées par les deux algorithmes sont très proches et très régulières. De plus elle font sens : quand la demande résiduelle est grande, i.e. R grand, et que la batterie est vide, i.e. C petit, il est optimal d'activer le générateur de courant et de générer de l'énergie pour répondre à la demande et éviter de se faire pénaliser ; en revanche si la demande est faible et que les batteries sont pleines : le gestionnaire n'a pas besoin d'activer le générateur, et peut se contenter d'utiliser sa batterie pour répondre à la demande.

En regardant les fonctions valeurs disponible dans le tableau 6.4 page 211, on s'aperçoit que Classif-Hybrid fait en fait mieux que le benchmark !

2.2.4 Perspectives

La performance de nos algorithmes dépend fortement de la qualité de nos mesure de training. Nous avons pensé aux méthodes de type bootstrapping pour construire de manière itérative de bonnes mesures de training, mais ces dernières ne marchent pas toujours en dimension élevée. L'idée principale à retenir est que, comme tel, si l'on ne connaît pas de bonne mesure de training en grande dimension, alors cela se ressentira dans la qualité des estimations des stratégies optimales par nos algorithmes. Il serait intéressant de trouver des méthodes qui marchent en grande dimension pour trouver de bonnes mesures de training.
Notons que cette recherche de bonne mesure de training fait en fait partie d'un domaine extrêmement actif dans la communauté de l'apprentissage par renforcement. Cette dernière parle de ce problème comme du problème d'exploration ; et de nombreux papiers sortent ces derniers temps à propos de la bonne manière de faire de l'exploration, i.e. trouver une bonne mesure de training.

2.3 Réseaux de neurones pour les EDPs et les EDSRs

Dans la troisième partie de cette thèse, nous proposons et étudions une famille de schémas numériques pour la résolution des EDPs, IQVs, EDSRs et des EDSRs réfléchies. Nous commençons par présenter le cadre, puis rappelons les méthodes numériques qui existent déjà pour résoudre ces problèmes, ainsi que leurs limites que l'on a voulu repousser en proposant nos nouveaux algorithmes. La convergence de nos nouveaux algorithmes est prouvée, et ces derniers sont testés numériquement avec succès sur différents exemples de dimension 1 à 40.

2.3.1 Cadre général

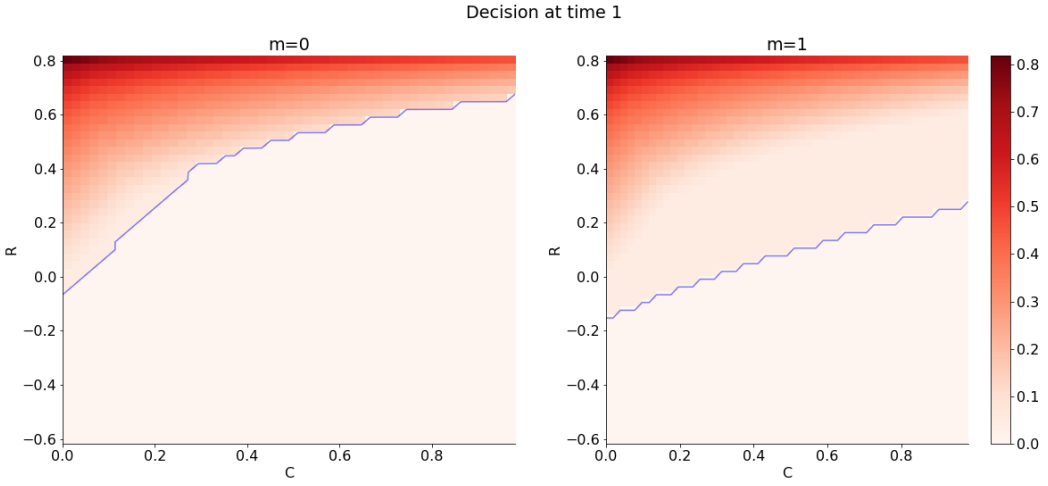
Fixons $T > 0$. Soient μ et σ deux fonctions telles que $\mu : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : [0,T] \times \mathbb{R}^d \to \mathcal{M}_d(\mathbb{R})$, où $\mathcal{M}_d(\mathbb{R})$ est l'espace des matrices réelles de taille $d \times d$. μ et σ sont supposées assez régulières pour qu'il y ait existence et unicité d'une solution $\mathcal{X}$ à l'équation différentielle stochastique suivante :

$$\mathcal{X}_t = x_0 + \int_0^t \mu(s, \mathcal{X}_s)\,\mathrm{d}s + \int_0^t \sigma(s, \mathcal{X}_s)\,\mathrm{d}W_s, \quad 0 \le t \le T,$$

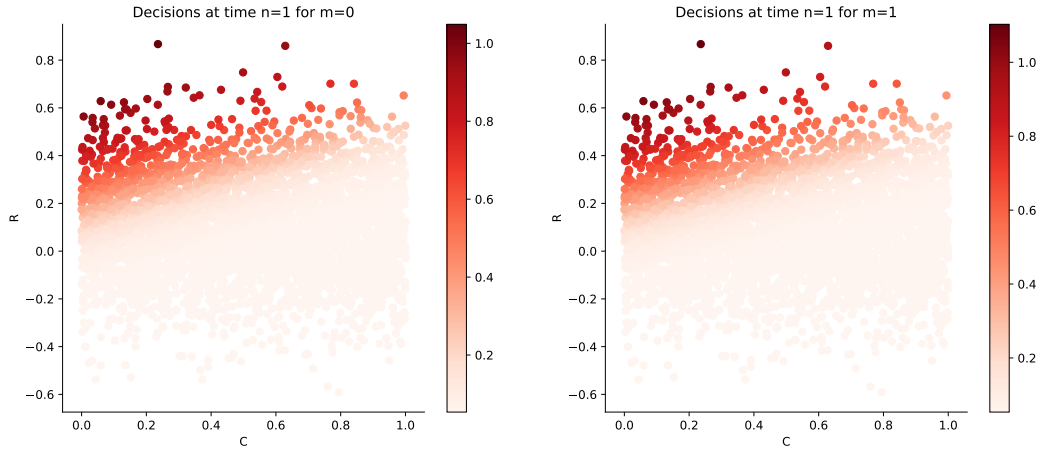
où W est un mouvement Brownien d-dimensionnel sur un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$ équipé de la filtration $\mathbb{F} = (\mathcal{F}_t)_{0 \le t \le T}$ qui satisfait les conditions usuelles.

Problème EDSR et EDP Le premier problème auquel on s'intéresse est celui qui consiste à trouver la paire (Y, Z) de processus $\mathbb{F}$-adaptés et à valeurs dans $\mathbb{R} \times \mathbb{R}^d$ qui soit solution de l'équation

[21] Qknn introduit dans la section 2.1.1.3 page 33.



(a) Estimation de la quantité optimale d'énergie à se faire fournir au temps $n = 1$ en fonction du couple de variables d'état (R, C), dans le cas où $m = 0$ ou $m = 1$, en utilisant l'algorithme Qknn basé sur de la régression locale. La zone au-dessous de la courbe bleue est celle où il est optimal d'éteindre (ou de laisser éteint) le générateur. La zone au-dessus de la courbe bleue est celle où il est optimal de se faire délivrer de l'énergie.



(b) Estimation de la quantité optimale d'énergie à se faire fournir au temps $n = 1$ en fonction du couple de variables d'état (R, C), dans le cas où $m = 0$ ou $m = 1$, en utilisant l'algorithme Classif-Hybrid.

FIGURE 2.2 – Quantité optimale d'énergie à se faire délivrer, estimée en utilisant Classif-Hybrid et Qknn.

suivante :

$$Y_t = g(\mathcal{X}_T) + \int_t^T f(s, \mathcal{X}_s, Y_s, Z_s)\,\mathrm{d}s - \int_t^T Z_s^\intercal\,\mathrm{d}W_s, \quad 0 \le t \le T, \tag{2.3.1}$$

L'équation (2.3.1) est une équation différentielle stochastiques rétrograde (BSDE) puisque la condition terminale est imposée, et le couple (Y, Z) est appelé la solution de l'EDSR. Il est bien connu que (2.3.1) admet des solutions sous certaines hypothèses de régularité de f et g (voir [PP90]) et que, de plus, Y and Z admettent les représentations suivantes :

$$\begin{aligned} Y_t &= u(t, \mathcal{X}_t), \\ Z_t &= \sigma^\top(t, \mathcal{X}_t) D_x u(t, \mathcal{X}_t) \quad 0 \le t \le T, \end{aligned}$$

où u est solution de l'équation différentielle parabolique aux dérivées partielles non linéaire suivante :

$$\begin{cases} \partial_t u + \mathcal{L}u + f(.,.,u,\sigma^\intercal D_x u) = 0, & \text{on } [0,T) \times \mathbb{R}^d, \\ u(T,.) = g, & \text{on } \mathbb{R}^d, \end{cases} \tag{2.3.2}$$

avec $\mathcal{L}$ le générateur infinitésimal de $\mathcal{X}$, défini ainsi

$$\mathcal{L}u := \frac{1}{2}\,\mathrm{Tr}\left(\sigma\sigma^\intercal D_x^2 u\right) + \mu.D_x u.$$

Résoudre l'EDSR (2.3.1) est donc équivalent à résoudre l'EDP (2.3.2).

Problème RBSDE Le deuxième problème auquel on s'intéresse dans la troisième partie de cette thèse est celui de trouver le triplet (Y, Z, K) solution de l'équation suivante :

$$\begin{aligned} Y_t =&\ g(\mathcal{X}_T) + \int_t^T f(s, \mathcal{X}_s, Y_s, Z_s)\,\mathrm{d}s - \int_t^T Z_s^\intercal\,\mathrm{d}W_s + K_T - K_t, \\ Y_t \ge&\ g(X_t), \quad 0 \le t \le T, \end{aligned} \tag{2.3.3}$$

où K est un processus croissant vérifiant :

$$\int_0^T \big(Y_t - g(X_t)\big) dK_t = 0.$$

(2.3.3) est communément appelée équation différentielle stochastique rétrograde réfléchie (EDSR réfléchie). C'est une représentation classique de problèmes de temps d'arrêt optimal, et apparaît en particulier dans les problèmes de pricing et de couverture d'options américaines. La solution à (2.3.3) s'écrit $Y_t = u(t, X_t)$, où il est bien connu que u est caractérisée comme solution de l'inégalité quasi-variationnelle (IQV) suivante :

$$\begin{cases} \min\big[-\partial_t u - \mathcal{L}u - f(t,x,u,\sigma^\intercal D_x u), u - g\big] &= 0, & t \in [0,T),\ x \in \mathbb{R}^d, \\ u(T,x) &= g(x), & x \in \mathbb{R}^d, \end{cases} \tag{2.3.4}$$

comme montré par exemple dans [EK+97].

La résolution de l'EDSR réfléchie (2.3.3) est donc équivalente à la résolution de l'IQV (2.3.4).

2.3.2 Méthodes numériques existantes dans la littérature

Quand $\mathcal{X}$ n'est pas simulable, la première étape pour obtenir des résultats numériques est de discrétiser le processus sur la grille de temps : $\pi = \{t_0 = 0 < t_1 < \ldots < t_N = T\}$, de module $|\pi| = \max_{i=0,\ldots,N-1} \Delta t_i$, $\Delta t_i := t_{i+1} - t_i$. La discrétisation d'Euler $X = X^\pi$ de $\mathcal{X}$ est alors :

$$X_{t_{i+1}} = X_{t_i} + \mu(t_i, X_{t_i})\Delta t_i + \sigma(t_i, X_{t_i})\Delta W_{t_i}, \quad i = 0, \ldots, N-1,\ X_0 = x_0,$$

où l'on note $\Delta W_{t_i} := W_{t_{i+1}} - W_{t_i}$. Pour simplifier la notation, nous notons X le processus discrétisé, et omettons ainsi la dépendance de ce dernier par rapport à la grille de discrétisation en temps π.. L'approximation de (2.3.2) est alors donnée par le schéma d'Euler associé dans la représentation forward (2.3.1) :

$$u(t_{i+1}, X_{t_{i+1}}) \approx F(t_i, X_{t_i}, u(t_i, X_{t_i}), \sigma^\intercal(t_i, X_{t_i}) D_x u(t_i, X_{t_i}), \Delta t_i, \Delta W_{t_i}),$$

où l'on a posé :

$$F(t, x, y, z, h, \Delta) := y - f(t, x, y, z)h + z^\intercal \Delta. \tag{2.3.5}$$

2.3.2.1 Méthodes numériques pour les EDSRs et les EDPs

Nous répertorions ici aux algorithmes qui existent déjà dans la littérature pour résoudre (2.3.1).

Différence finie L'EDP (2.3.2) peut se résoudre par méthode de différence finie en discrétisant les opérateurs de dérivation sur des grilles (resp. sparse-grids) en petite (resp. moyenne) dimension. La convergence de ce type d'algorithme est garantie sous certaines conditions entre les coefficients (notamment le pas de discrétisation en temps et celui en espace).
Commentaires sur les méthodes de différence finie :

- ✗ Les méthodes de différence finie imposent des conditions aux bords qui ne sont pas données dans (2.3.2). En pratique, il faut donc ajouter des conditions artificielles pour appliquer ce type de méthodes. (voir le Benchmark dans la Section 4.4.2.2 page 132 pour un exemple d'application d'une méthode de différence finie pour résoudre une EDP non-linéaire de type HJB)

- ✗ La complexité des méthodes basées sur les différences finies est en $\mathcal{O}^{d+1}$ et explose avec la dimension ! Cette méthode est en fait adaptée aux problèmes de petites dimensions (≤ 3), voir un peu plus si l'on utilise des méthodes sparses.

Régression sur une base de fonctions Une famille d'algorithmes est présentée dans [LGW06] pour résoudre les EDSRs. L'idée principale est d'estimer les solutions aux dates $t_i, i = 0, \dots, N$ de manière rétrograde, en utilisant le schéma suivant :

$$\begin{cases} Y_{t_N} &= g(X_{t_N}) \\ Z_{t_i} &= \frac{1}{\Delta t_i} \mathbb{E}\left[Y_{t_{i+1}} \Delta W_{t_i} | \mathcal{F}_{t_i}\right], \\ Y_{t_i} &= \mathbb{E}\left[Y_{t_{i+1}} | \mathcal{F}_{t_i}\right] + f(t_i, X_{t_i}, Y_{t_i}, Z_{t_i}) \Delta t_i, \quad \text{for } i = N-1, \dots, 0. \end{cases} \tag{2.3.6}$$

A la date t_i, Z_{t_i} est approximé en régressant l'espérance conditionnelle sur une base de fonction ; et Y_{t_i}, défini dans (2.3.6) comme le point fixe de l'opérateur contractant $\mathbb{L}^2(\sigma(X_{t_i})) \to \mathbb{L}^2(\sigma(X_{t_i}))$: $Y_{t_i} \mapsto \mathbb{E}_i\left[Y_{t_{i+1}}\right] + hf(t_i, X_{t_i}, Y_{t_i}, Z_{t_i})$, est approximé par itérations de Picard.La convergence des algorithmes est étudié dans ce même papier.
Commentaires sur les méthodes basées sur la regression sur une base de fonctions :

- ✓ Ces méthodes sont très rapides et efficaces en petite et moyenne dimension car elles reposent sur des formules exactes et fermées pour les étapes de régression.

- ✗ Ces méthodes sont inutilisables pour des problèmes en dimension supérieure à 10, car au-delà, il quasiment impossible de trouver combien et quelles fonctions de base prendre, pour assurer une approximation fidèle des fonctions inconnues tout en évitant le sur-apprentissage (overfitting) et/ou sous-apprentissage (underfitting).

Les deux méthodes que l'on vient de présenter souffrent de la dimension, et sont en particulier inapplicables en grande dimension. Certaines méthodes récentes, qui reposent sur les réseaux de neurones, sont plus adaptées à la grande dimension.

DGM La méthode DGM (pour Deep Galerkin Method) a été introduite dans [SS18], et consiste à approximer par réseaux de neurones $\mathcal{U}(t,x;\theta)$ la solution de (2.3.2). On considère ν_T une mesure à support sur $[0,T]$, ν_Ω une mesure à support sur Ω, $\nu_{\partial\Omega}$ une mesure à support sur $\partial\Omega$, et l'on définit la fonction objective J ainsi :

$$J(\theta) = \|\partial_t\mathcal{U}(.,.;\theta) + \mathcal{L}\mathcal{U}(.,.;\theta) + f(t,x,\mathcal{U}(.,.;\theta),\sigma^\intercal D_x\mathcal{U}(.,.;\theta)\|_{[0,T]\times\Omega,\nu_\Omega} + \|\mathcal{U}(T,.;\theta) - g(.)\|_{\partial\Omega,\nu_{\partial\Omega}},$$

où l'on note $\|.\|_{[0,T]\times\Omega,\nu_\Omega}$ la norme canonique sur $\mathbb{L}^2([0,T]\times\Omega,\nu_T\times\nu_\Omega)$, et $\|.\|_{[0,T]\times\partial\Omega,\nu_{\partial\Omega}}$ celle sur $\mathbb{L}^2([0,T]\times\partial\Omega,\nu_T\times\nu_{\partial\Omega})$.

L'algorithme DGM consiste à déterminer, par descente de gradient stochastique, le θ optimal, noté θ^*, qui minimise la fonction objective. L'algorithme converge car la fonction objective s'approche de zéro lorsque la classe de réseaux de neurones grandit ; ce qui implique que $\mathcal{U}(.,.;\theta^*)$ approxime de mieux en mieux la solution u de (2.3.2).

Commentaires sur DGM :

- ✓ Nous rappelons que le calcul de la dérivée du réseau de neurones n'est pas cher en temps de calculs quand elle est calculée en utilisant la rétro-propagation inverse.
- ✓ Le fait que DGM repose sur un choix des mesures de training peut être positif lorsque l'agent sait dans quelle région il recherche des estimées précises : il lui suffit de prendre une densité élevée dans les zones qu'il veut couvrir en priorité.

NNZ L'algorithme NNZ (Neural Network approximation for Z), introduit dans [EHJ17] et dont la convergence est étudiée dans [HL18], constitue notre principal benchmark pour résoudre les EDSRs en grande dimension. Il repose sur la représentation $\mathcal{Z}_{t_i}$ par réseau de neurones de Z_{t_i} à chaque pas de discrétisation en temps $t_i, i = 0,\dots,N-1$, ainsi que sur l'approximation suivante

$$\begin{aligned} u(t_n,X_{t_n}) \approx u(t_{n-1};X_{t_{n-1}}) &- f(t_{n-1},X_{t_{n-1}},u(t_{n-1},X_{t_{n-1}}),Z(t_{n-1},X_{t_{n-1}}))(t_n - t_{n-1}) \\ &+ Z(t_n,X_{t_n})^T(W_{t_{n+1}} - W_{t_n}), \end{aligned}$$

pour représenter par induction, pour tout $n = 0,\dots,N$, la fonction $u(t_n,.)$ par un réseau de neurones $\mathcal{U}_n(.,\theta)$ défini comme composé des réseaux de neurones qui représentent les Z_{t_i} avec $i = 0,\dots,n-1$. La fonction objectif de NNZ est la fonction J définie ainsi :

$$J(\theta) := \mathbb{E}\left[(\mathcal{U}_N(X_N,\theta) - g(X_N))^2\right].$$

et l'idée est d'entraîner $\mathcal{U}_N(X_N,\theta)$ pour qu'il annule J.

Commentaires sur l'algorithme NNZ

- ✓ Tous les réseaux de neurones qui représentent les Z_{t_i} sont entraînés en même temps, ce qui rend l'algorithme très rapide lorsque N est petit.
- ✓✗ Il n'y a pas d'erreur d'approximation des fonction u qui se diffuse de manière rétrograde. Mais le prix à payer est la resimulation des processus de $k = 0$ à N pour apprendre les paramètres optimaux des représentations des contrôles optimaux $\alpha_n^*, n = 0,\dots,N-1$.
- ✗ L'apprentissage de tous les réseaux de neurones qui représentent les Z_{t_i} étant global, NNZ requiert l'entraînement d'un réseau de neurones très profond, avec $n_Z \times N$ couches, où n_Z est le nombre de couches choisies pour apprendre chaque Z_{t_i}. L'entraînement de réseaux de neurones très profonds est encore un problème très difficile à résoudre, même en utilisant les derniers algorithmes de descente de gradient. En fait, il est très facile d'observer des cas où la SGD n'arrive pas à minimiser les fonctions objectives : l'algorithme peut converger par descente de gradient vers des minimums locaux qui sont loin d'être optimaux, ou ne pas converger pas du tout. Voir Chapitre 7 pour des exemples où il se produit l'une ou l'autre des deux situations.

2.3.3 Contributions (Chapitre 7)

Motivation A cause de la malédiction de la dimension, la résolution numérique de (2.3.1) et (2.3.3) a toujours été difficile en grande dimension. De même, il est difficile de résoudre les EDPs (2.3.2) et IQVs (2.3.4) en grande dimension.

Les méthodes usuelles basées sur la régression sur une base de fonctions, ou de différence finie, marchent très bien en petite et moyenne dimension uniquement. Les méthodes basées sur les réseaux de neurones comme NNZ et DGM sont des algorithmes bien adaptés à la grande dimension. Mais NNZ semble devoir se limiter à la résolution de BSDEs simples où le nombre de pas de discrétisation de la BSDE n'est pas trop élevé.

Question 3a :	*Quel schéma stable proposer pour résoudre les EDSRs en grande dimension ?*
Question 3a(bis) :	*Quel schéma stable proposer pour résoudre les EDPs en grande dimension ?*
Question 3b :	*Quel schéma stable proposer pour résoudre les EDSRs réfléchies en grande dimension ?*
Question 3b(bis) :	*Quel schéma stable proposer pour résoudre les EDPs avec frontière libre en grande dimension ?*

Remark 2.3.1. *Comme dit précédemment la Question 3a est équivalente à la Question 3a(bis). De même, la Question 3b est équivalente à la Question 3b(bis).*

Nous introduisons dans le Chapitre 7 de cette thèse deux algorithmes que l'on a baptisés DBDP1 et DBDP2, et qui répondent à la Question 2a ; ainsi qu'un algorithme, appelé RDBDP qui répond à la question 2b. Une étude théorique de la convergence de nos nouveaux algorithmes est menée, et de nombreux tests numériques sont disponibles pour illustrer les performances de nos algorithmes. Nous présentons ci-dessous succinctement les trois algorithmes et donnons les trois résultats de convergence importants établis dans la troisième partie de la thèse.

Présentation de DBDP1 et résultat de convergence associé

L'idée principale de l'algorithme DBDP1 est d'approximer, pour tout $i = N-1, \dots, 0$, la paire $(u(t_i,.), Z(t_i,.))$ par des réseaux de neurones :

$$\begin{cases} u(t_i,.) &= \mathcal{U}_i(.;\theta) \\ Z(t_i,.) &= \mathcal{Z}_i(.;\theta), \end{cases}$$

où l'on note par θ l'ensemble des paramètres des deux réseaux de neurones considérés. Nous apprenons θ_i^*, le paramètre optimal de $\mathcal{U}_i$ et $\mathcal{Z}_i$ pour l'approximation de $u(t_i,.)$ et Z_{t_i}, de manière rétrograde et en reposant, comme écrit dans Algorithme 2.7 ci-dessous, sur l'approximation suivante :

$$\mathcal{U}_{i+1}(X_{t_{i+1}};\theta_{i+1}^*) \approx F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta_i^*), \mathcal{Z}_i(X_{t_i};\theta_i^*), \Delta t_i, \Delta W_{t_i}),$$

où l'on a noté θ_{i+1}^* le paramètre optimal de $\mathcal{U}_{i+1}$ et $\mathcal{Z}_{i+1}$ pour l'approximation de $u(t_{i+1},.)$ et $Z_{t_{i+1}}$, et où la fonction F a été introduite dans (2.3.5).

La convergence de DBDP1 est étudiée dans le Chapitre 7 (voir théorème 7.4.1 page 225). Nous présentons ci-dessous une version simplifiée du résultat de convergence.

Theorem 2.3.1. (Consistance de DBDP1) *Sous des hypothèses classiques de régularité des coefficients, il existe une constante $C > 0$, indépendante de la grille de discrétisation en temps π, telle que*

$$\mathcal{E}\big[(\widehat{\mathcal{U}}^{(1)}, \widehat{\mathcal{Z}}^{(1)}), (Y,Z)\big] \leq C\Big(\mathbb{E}\big|g(\mathcal{X}_T) - g(X_T)\big|^2 + |\pi| + \varepsilon^Z(\pi) + \sum_{i=0}^{N-1}\big(N\varepsilon_i^{\mathcal{N},v} + \varepsilon_i^{\mathcal{N},z}\big)\Big),$$

Algorithm 2.7 Algorithme DBDP1

Entrée :

1: Initialiser $\widehat{\mathcal{U}}_N^{(1)} = g$
2: **for** $i = N-1, \dots, 0$ **do**
3: Calculer :

$$\theta_i^* \in \operatorname*{argmin}_{\theta} \mathbb{E}\left|\widehat{\mathcal{U}}_{i+1}^{(1)}(X_{t_{i+1}}) - F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta), \mathcal{Z}_i(X_{t_i};\theta), \Delta t_i, \Delta W_{t_i})\right|^2;$$

4: Poser $\widehat{\mathcal{U}}_i^{(1)} = \mathcal{U}_i(.;\theta_i^*)$;
5: Poser $\widehat{\mathcal{Z}}_i^{(1)} = \mathcal{Z}_i(.;\theta_i^*)$;
6: **end for**

Sortie :

– $(\widehat{\mathcal{U}}_i^{(1)})_{i=0}^N$: Estimations des Y_{t_i} pour $i = 0, \dots, N$;
– $(\widehat{\mathcal{Z}}_i^{(1)})_{i=0}^N$: Estimations des Z_{t_i} pour $i = 0, \dots, N$;

où

$$\mathcal{E}\big[(\widehat{\mathcal{U}}^{(1)}, \widehat{\mathcal{Z}}^{(1)}), (Y,Z)\big] := \max_{i=0,\dots,N-1} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \big|Z_t - \widehat{\mathcal{Z}}_i^{(1)}(X_{t_i})\big|^2 dt\Big]$$

est l'erreur $\mathbb{L}^2$ *entre l'estimée par DBDP1 de* (Y,Z) *et le couple* (Y,Z) *lui-même, où*

$$\varepsilon_i^{\mathcal{N},v} := \inf_{\xi} \mathbb{E}\big|\hat{v}_i(X_{t_i}) - \mathcal{U}_i(X_{t_i};\xi)\big|^2, \qquad \varepsilon_i^{\mathcal{N},z} := \inf_{\eta} \mathbb{E}\big|\overline{\hat{z}_i}(X_{t_i}) - \mathcal{Z}_i(X_{t_i};\eta)\big|^2$$

sont des erreurs d'approximation par réseaux de neurones qui s'annulent lorsque l'espace des réseaux de neurones considéré grandit ; et où

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

est la $\mathbb{L}^2$*-régularité de* Z*.*

Le théorème 2.3.1 stipule que l'erreur d'estimation en norme $\mathbb{L}^2$ de Y et Z par l'algorithme DBDP1 est bornée par la somme de :

- l'erreur de discrétisation en temps de $[0,T]$, i.e. $|\pi|$,
- l'erreur de discrétisation d'Euler du processus $\mathcal{X}$,
- la régularité $\mathbb{L}^2$ de $\mathcal{Z}$,
- la quantité $\sum_{i=0}^{N-1}\left(N\varepsilon_i^{\mathcal{N},v} + \varepsilon_i^{\mathcal{N},z}\right)$ qu'il faut voir comme la somme d'erreurs d'approximation par réseaux de neurones de certaines fonctions.

On montre par des arguments standards et sous des hypothèses standards, que toutes ces erreurs tendent vers 0 lorsque la discrétisation en temps s'affine et que la taille de l'espace des réseaux de neurones grandit.

Présentation de DBDP2, et résultat de convergence associé

Remarquons que dans DBDP1, nous ignorons le résultat bien connu suivant : $Z = \sigma^\intercal(t_i,.)D_x u(t_i,.)$, ce qui revient à ignorer que le réseau de neurones $\mathcal{Z}_i(.;\theta)$ s'exprime très simplement en fonction la dérivée de $\mathcal{U}_i(.;\theta)$. Nous présentons dans cette section l'algorithme DBDP2, qui est basé sur cette remarque.

Considérons donc ici qu'un unique réseau de neurones $\mathcal{U}$ pour apprendre u, et représentons (Y, Z) :

$$\begin{cases} Y(t_i,.) &= \mathcal{U}_i(.;\theta) \\ Z(t_i,.) &= \sigma^\intercal(t_i,.)D_x\mathcal{U}_i(.;\theta), \end{cases}$$

où cette fois-ci, θ est le paramètre de réseaux de neurones qui approxime u. Nous apprenons θ_i^*, le paramètre optimal pour l'approximation de $u(t_i,.)$ et $Z(t_i,.)$, de manière rétrograde, en reposant, comme écrit dans Algorithme 2.8 ci-dessous, sur l'approximation suivante :

$$\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}};\theta_{i+1}^*) = F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta_i^*), \sigma^\intercal(t_i, X_{t_i})D_x\mathcal{U}_i(X_{t_i};\theta_i^*), \Delta t_i, \Delta W_{t_i}).$$

Algorithm 2.8 Algorithme DBDP2

Entrée :

1: Initialiser $\widehat{\mathcal{U}}_N^{(1)} = g$
2: **for** $i = N-1, \dots, 0$ **do**
3: Calculer :

$$\theta_i^* \in \operatorname*{argmin}_\theta \mathbb{E}\Bigg[\Big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{t_{i+1}}) - F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta), \sigma^\intercal(t_i, X_{t_i})\hat{D}_x\mathcal{U}_i(X_{t_i};\theta), \Delta t_i, \Delta W_{t_i})\Big|^2\Bigg];$$

4: Poser $\widehat{\mathcal{U}}_i^{(1)} = \mathcal{U}_i(.;\theta_i^*)$;
5: Poser $\widehat{\mathcal{Z}}_i^{(1)} = \sigma^\intercal(t_i,.)D_x\hat{\mathcal{U}}_i(.;\theta_i^*)$;
6: **end for**

Sortie :

– $(\widehat{\mathcal{U}}_i^{(1)})_{i=0}^N$: Estimations des Y_{t_i} pour $i = 0, \dots, N$;
– $(\widehat{\mathcal{Z}}_i^{(1)})_{i=0}^N$: Estimations des Z_{t_i} pour $i = 0, \dots, N$;

Remark 2.3.2. *Calculer $\nabla\mathcal{U}_i(.;\theta)$ est du même ordre de complexité que calculer $\mathcal{U}_i(.;\theta)$, lorsque l'on utilise la différentiation automatique rétrograde, implémentée nativement dans Tensorflow ou PyTorch. Du coup, l'algorithme DBDP2 ne perd pas de temps à calculer $\nabla\mathcal{U}_i(.;\theta)$.*

La convergence de DBDP2 est étudiée dans le Chapitre 7 (voir Théorème 7.4.2 page 230), et nous présentons ici une version simplifiée du résultat :

Theorem 2.3.2. (Consistance de DBDP2) *Sous des hypothèses de régularité classiques, il existe une constante $C > 0$, indépendante de π telle que*[22]

$$\mathcal{E}\big[(\widehat{\mathcal{U}}^{(2)}, \widehat{\mathcal{Z}}^{(2)}), (Y, Z)\big] \leq C\Big(\mathbb{E}\big|g(\mathcal{X}_T) - g(X_T)\big|^2 + \frac{\gamma_m^6}{N} + \varepsilon^Z(\pi) + N\sum_{i=0}^{N-1}\varepsilon_i^{\mathcal{N},m}\Big).$$

Le théorème 2.3.1 stipule que l'erreur d'estimation en norme $\mathbb{L}^2$ de u et Z par l'algorithme DBDP2 est bornée par la somme de :

- l'erreur de discrétisation en temps de $[0, T]$, i.e. $|\pi|$,
- l'erreur de discrétisation d'Euler du processus $\mathcal{X}$,
- une erreur supplémentaire $\frac{\gamma_m^6}{N}$, due aux contraintes que l'on doit exercer sur la taille de l'espace de réseaux de neurones,

[22] Voir théorème 2.3.1 pour les notations utilisées.

- la régularité $\mathbb{L}^2$ de $\mathcal{Z}$,
- la quantité $\sum_{i=0}^{N-1}\left(N\varepsilon_i^{\mathcal{N},v}+\varepsilon_i^{\mathcal{N},z}\right)$ qu'il faut voir comme la somme d'erreurs d'approximation par réseaux de neurones de certaines fonctions.

Ces erreurs tendent vers 0 lorsque la discrétisation en temps s'affine et que la taille de l'espace des réseaux de neurones grandit, ce qui implique la convergence de DBDP2.

Présentation de RDBDP et résultat de consistance

L'idée principale de l'algorithme RDBDP est d'approximer, pour tout $i = N-1,\dots,0$, la paire $(u(t_i,.),Z(t_i,.))$ par des réseaux de neurones :

$$\begin{cases} u(t_i,.) &= \mathcal{U}_i(.;\theta) \\ Z(t_i,.) &= \mathcal{Z}_i(.;\theta), \end{cases}$$

où l'on note par θ l'ensemble des paramètres des deux réseaux de neurones considérés ; puis de reposer sur le schéma décrit dans Algorithme 2.9 ci-dessous, pour approximer les fonctions inconnues $u(t_i,.)$ et $Z(t_i,.)$.

Algorithm 2.9 Algorithme RDBDP

Entrée :

1: Initialiser $\widehat{\mathcal{U}}_N^{(1)} = g$
2: **for** $i = N-1,\dots,0$ **do**
3: Calculer :

$$\theta_i^* \in \operatorname*{argmin}_\theta \hat{L}_i(\theta),$$

où

$$\hat{L}_i(\theta) := \mathbb{E}\left|\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) - F(t_i,X_{t_i},\mathcal{U}_i(X_{t_i};\theta),\mathcal{Z}_i(X_{t_i};\theta),\Delta t_i,\Delta W_{t_i})\right|^2$$

4: Poser $\widehat{\mathcal{U}}_i = \max\left[\mathcal{U}_i(.;\theta_i^*),g\right]$;
5: Poser $\widehat{\mathcal{Z}}_i = \hat{\mathcal{Z}}_i(.;\theta_i^*)$;
6: **end for**

Sortie :

– $(\widehat{\mathcal{U}}_i^{(1)})_{i=0}^N$: Estimations des Y_{t_i} pour $i = 0,\dots,N$;
– $(\widehat{\mathcal{Z}}_i^{(1)})_{i=0}^N$: Estimations des Z_{t_i} pour $i = 0,\dots,N$;

La convergence de RDBDP est étudiée dans le Chapitre 7 (voir Théorème 7.4.4 page 235), et nous présentons ici une version simplifiée du résultat :

Theorem 2.3.3. (Consistance of RDBDP) *Sous des hypothèses classiques de régularité des coefficients, il existe une constante $C > 0$, indépendante de π, telle que*

$$\mathcal{E}\left[(\widehat{\mathcal{U}},\widehat{\mathcal{Z}}),(Y,Z)\right] \le C\Big(\varepsilon(\pi)+\sum_{i=0}^{N-1}\left(N\varepsilon_i^{\mathcal{N},\bar{v}}+\varepsilon_i^{\mathcal{N},\bar{z}}\right)\Big),$$

où $\varepsilon(\pi) = O(|\pi|^{\frac{1}{2}})$ sous des hypothèses de régularité de g, ou $\varepsilon(\pi) = O(|\pi|)$ sous des hypothèses de régularité de g et σ.

Le Théorème 2.3.3 stipule que l'erreur d'estimation de (Y,Z) par RDBDP tend vers 0 lorsque le pas de discrétisation en temps diminue et que l'espace des réseaux de neurones grandit.

Dimension	nb pas	valeur	std	référence
1	80	0.0613818	0.00019	0.060903
5	80	0.107650	0.00016	0.10738
10	80	0.129923	0.00016	0.12996
20	80	0.15050	0.00010	0.1510
40	160	0.16758	0.00016	0.1680

TABLE 2.2 – Estimation du prix d'une option américaine par l'algorithme RDBDP. Les prix de référence sont des excellentes approximations du prix de l'américaine, calculées en utilisant une astuce pour réduire la dimension à 1. La moyenne et l'écart-type sont calculés sur 40 lancements indépendants de RDBDP. RDBDP est très précis jusqu'à la dimension 40 !

Résultats numériques obtenus

Dans la section 7.5 du Chapitre 7 figurent de nombreux tests numériques qui mettent en avant les performances de nos algorithmes DBDP1, DBDP2 et RDBDP. Les résultats sont très satisfaisants ! En particulier nos algorithmes DBDP1 et DBDP2 sont beaucoup plus stables que NNZ, qui consistue notre principal Benchmark pour résoudre des EDSRs en grande dimension. Nos algorithmes convergent aussi bien que NNZ lorsque ce dernier converge, et convergent dans des exemples numériques dans lesquels NNZ soit diverge, soit converge vers une valeur qui n'est pas satisfaisante. En fait, NNZ a beaucoup de difficulté à converger lorsque le problème EDSR n'est pas simple, i.e. les fonctions u et Z à approximer sont complexes ou que N est grand. Nous référons à la section 7.5.1 page 237 pour des exemples simples sur lesquels DBDP1, DBDP2 et NNZ marchent bien, et la section 7.5.2 page 241 pour des exemples plus compliqués dans lesquels DBDP1 et DBDP2 marchent bien, mais NNZ diverge !

Enfin, nous n'avons constaté aucun exemple dans lequel NNZ fasse vraiment mieux que nos algorithmes, ce qui n'est pas étonnant lorsque l'on compare la structure des algorithmes.

RDBDP apparaît aussi comme un algorithme très performant dans l'application de pricing d'option américaine en grande dimension que l'on a considérée. Les résultats l'algorithme RDBDP sont aussi satisfaisants, et nous proposons dans Tableau 2.2 des résultats d'estimations par l'algorithme RDBDP de prix d'une option américaine pour différentes valeur de la dimension allant de 1 à 40. Pour le payoff considéré, il est possible de réduire de problème de pricing de dimension $d \geq 1$ à un problème de dimension 1. Les valeurs de référence sont calculées en réécrivant réduisant ainsi à 1 la dimension du problème, et en utilisant une méthode standard par arbres pour le calcul de prix d'option américaine. Ce tableau est extrait du tableau 7.11 page 245.

2.3.4 Perspectives

Le résultats de convergence proposés sont très satisfaisants mais l'on peut espérer faire plus naturel dans le terme associé à la somme des erreurs d'approximation par réseaux de neurones de la fonction u et Z. Par exemple, dans les théorèmes 2.3.1, 2.3.2 présentés ci-dessus, le terme d'erreurs d'approximation suivant apparaît : $\sum_{i=0}^{N-1} N\varepsilon_i^{\mathcal{N},\tilde{v}}$, ce que nous avons interprété comme la somme des erreurs d'approximation par réseaux de neurones de la fonction u qui sont faites à chaque pas de temps. Il aurait été plus naturel d'avoir la somme d'erreurs suivante $\sum_{i=0}^{N-1} \varepsilon_i^{\mathcal{N},\tilde{v}}$ dans les théorèmes, i.e. une vraie somme d'erreurs d'approximation par réseaux de neurones, où le N n'apparaît pas dans la somme. Le N est en fait un artefact de calcul, dont nous n'avons pas pu nous défaire.

Part I

Quantization and regression-type methods for MDPs. Application to market making and Mckean-Vlasov control problems.

Chapter 3

Algorithmic trading in a microstructural limit order book model[1]

Abstract We propose a microstructural modeling framework for studying optimal market making policies in a FIFO (first in first out) limit order book (order book). In this context, the limit orders, market orders, and cancel orders arrivals in the order book are modeled as Cox point processes with intensities that only depend on the state of the order book. These are high-dimensional models which are realistic from a micro-structure point of view and have been recently developed in the literature. In this context, we consider a market maker who stands ready to buy and sell stock on a regular and continuous basis at a publicly quoted price, and identifies the strategies that maximize her P&L penalized by her inventory.

We apply the theory of Markov Decision Processes and dynamic programming method to characterize analytically the solutions to our optimal market making problem. The second part of the paper deals with the numerical aspect of the high-dimensional trading problem. We use a control randomization method combined with quantization method to compute the optimal strategies. Several computational tests are performed on simulated data to illustrate the efficiency of the computed optimal strategy. In particular, we simulated an order book with constant/ symmetric/ asymmetrical/ state dependent intensities, and compared the computed optimal strategy with naive strategies.

Keywords: Limit Order Book; pure-jump controlled process; High-Frequency Trading; Queuing model; High-dimensional Stochastic Control; Markov Decision Process; Quantization; Local Regression

Contents

[1] This Chapter is based on a paper written in collaboration with Frédéric Abergel and Huyên Pham.

3.1 Introduction

Most of the markets use a limit order book (order book) mechanism to facilitate trade. Any market participant can interact with the order book by posting either market orders or limit orders. In such type of markets, the market makers play a fundamental role by providing liquidity to other market participants, typically to impatient agents who are willing to cross the bid-ask spread. The profit made by a market making strategy comes from the alternation of buy and sell orders.

From the mathematical modeling point of view, the market making problem corresponds to the choice of an optimal strategy for the placement of orders in the order book. Such a strategy should maximize the expected utility function of the wealth of the market maker up to a penalization of her inventory. In the recent litterature, several works focused on the problem of market making through stochastic control methods. The seminal paper by Avellaneda and Stoikov [AS07] inspired by the work of Ho and Stoll [HS79] proposes a framework for trading in an order driven market. They modeled a reference price for the stock as a Wiener process, and the arrival of a buy or sell liquidity-consuming order at a distance δ from the reference price is described by a point process with an intensity in an exponential form decreasing with δ. They characterized the optimal market making strategies that maximize an exponential utility function of terminal wealth. Since this paper, other authors have worked on related market making problems. Gueant, Lehalle, and Fernandez-Tapia [GLFT12] generalized the market making problem of [AS07] by dealing with the inventory risk. Cartea and Jaimungal [CJ13] also designed algorithms that manage inventory risk. Fodra and Pham [FP15b] and [FP15a] considered a model designed to be a good compromise between accuracy and tractability, where the stock price is driven by a Markov Renewal Process, and solved the market making problem. Guilbaud and Pham [GP13] also considered a model for the mid-price, modeled the spread as a discrete Markov chain that jumps according to a stochastic clock, and studied the performance of the market making strategy both theoretically and numerically. Cartea and Jaimungal [CJ10] employed a hidden Markov model to examine the intra-day changes of dynamics of the order book. Very recently, Cartea, Penalva, and Jaimungal [CPJ15] and Gueant [Gu16] published monographs in which they developped models for algorithmic trading in different contexts. Abergel and El Aoud [EAA15] extended the framework of Avellaneda and Stoikov to the options market making. A common feature of all these works is that a model for the price or/and the spread is considered, and the order book is then built from these quantities. This approach leads to models that predict well the long-term behavior of the order book. The reason for this choice is that it is generally easier to solve the market making problem when the controlled process is low-dimensional. Yet, some recent works have introduced accurate and sophisticated micro-structural order book models. These models reproduce accurately the short-term behavior of the market data. The focus is on conditional probabilities of events, given the state of the order book and the positions of the market maker. Abergel, Anane, Chakraborti, Jedidi, Muni Toke [Abe+16] proposed models of order book where the arrivals of orders in the order book are driven by Poisson processes or Hawkes processes. Stoikov, Talreja, and Cont [CST07] also modeled the orders arrivals with Poisson processes. Lehalle, Rosenbaum and Huang [HLR15] proposed a queue-reactive

model for the order book. In this model the arrivals of orders are driven by Cox point processes with intensities that only depend on the state of the order book (they are not time dependent). Other tractable dynamic models of order-driven market are available (see e.g. Stoikov, Talreja, and Cont [CST07], Rosu [Ros08], Cartea, Jaimungal, Ricci [CJR14]).

In this paper we adopt the micro-structural model of order book in [Abe+16], and solve the associated trading problem. The problem is formulated in the general framework of Piecewise Deterministic Markov Decision Process (PDMDP), see Bauerle and Rieder [BR11]. Given the model of order book, the PDMDP formulation is natural. Indeed, between two jumps, the order book remains constant, so one can see the modeled order book as a point process where the time becomes a component of the state space. As for the control, the market maker fixes her strategy as a deterministic function of the time right after each jump time. We prove that the value function of the market making problem is equal to the value function of an associated non-finite horizon Markov decision process (MDP). This provides a characterization of the value function in terms of a fixed point dynamic programming equation. Jacquier and Liu in [JL18] recently followed a similar idea to solve an optimal liquidation problem, while Baradel et al. [BBEM18] and Lehalle et al. [LOR18] also tackled this problem of reward functional maximization in a micro-structure model of order book framework.

The second part of the paper deals with the numerical simulation of the value functions. The computation is challenging because the micro-structural model used to model the order book leads to a high-dimensional pure jump controlled process, so evaluating the value function is computationally intensive. We rely on control randomization and Markovian quantization methods to compute the value functions. Markovian quantization has been proved to be very efficient for solving control problems associated with high-dimensional Markov processes. We first quantize the jump times and then quantize the state space of the order book. See Pages, Pham, Printemps [PPP04c] for a general description of quantization applied to controlled processes. The projections are time-consuming in the algorithm. Fast approximate nearest neighbors algorithms have been implemented to make it quicker (see [ML09]). We borrow the values of intensities of the arrivals of orders for the order book simulations from Huang et al. [HLR15] in order to test our optimal trading strategies.

The paper is organized as follows. The model setup is introduced in Section 3.2: we present the micro-structural model for the order book, and show how the market maker interacts with the market. In Section 3.3, we prove the existence and provide a characterization of the value function and optimal trading strategies. In Section 3.4, we introduce a quantization-based algorithm to numerically solve a general class of discrete-time control problem with finite horizon, and then apply it on our trading problem. We then present some results of numerical tests on simulated order book. Section 3.5 presents an extension of our model when order arrivals are driven by Hawkes processes, and finally the appendix collects some results used in the paper .

3.2 Model setup

3.2.1 Order book representation

We consider a model of the order book inspired by the one introduced in chapter 6 of [Abe+16].

Fix $K \geq 0$. An order book is supposed to be fully described by K limits on the bid side and K limits on the ask side. Denote by pa_t the *best ask* at time t, which is the cheapest price a participant in the market is willing to sell a stock at time t, and by pb_t the *best bid* at time t, which is the highest price a participant in the market is willing to buy a stock at time t. We use the pair of vectors $(\underline{a}_t, \underline{b}_t) = (a_t^1, ..., a_t^K, b_t^1, \dots, b_t^K)$

- a_t^i is the number of shares available i ticks away from pb_t,
- -b_t^i is the number of shares available i ticks away from pa_t,

to describe the order book. The vector $\underline{a}_t$ and $\underline{b}_t$ describe the ask and the bid sides at time t. The

quantities a_t^i, $1 \leq i \leq K$, live in the discrete space $q\mathbb{N}$ where $q \in \mathbb{R}^*$ is the minimum order size on each specific market (*lot size*). The quantities b_t^i, $1 \leq i \leq K$, live in the discrete space $-q\mathbb{N}$. By convention, the a^i are non-negative, and the b^i are non-positive for $0 \leq i \leq K$. The tick size ϵ represents the smallest intervall between price levels.
In the sequel we assume that the orders arrivals have the same size $q = 1$, and set the tick size to $\epsilon = 1$ for simplicity.

Constant boundary conditions are imposed outside the moving frame of size $2K$ in order to guarantee that both sides of the LOB are never empty: we assume that all the limits up to the K-th ones are equal to a_∞ in the ask side, and equal to b_∞ in the bid side.

We shall assume some conditions on the structure of the orders arrivals in the order book.

(Harrivals) The orders arrivals from general market participants (market orders, limit orders and cancel orders) occur according to Markov jump processes which intensities only depends on the state of the order book. Moreover, we assume that the all the intensities are at most linear w.r.t. the couple $(\underline{a}, \underline{b})$ and are constant between two events.

Under **(Harrivals)**, let us define

- λ^{M^+} the intensity of the buy-to-market orders flow M_t^+,
- λ^{M^-} the intensity of the sell-to-market orders flow M_t^-,
- $\lambda_i^{L^+}$, $i \in \{1, ...K\}$, the intensity of the sell orders flow L_i^+ at the i^{th} limit of the ask side,
- $\lambda_i^{L^-}$, $i \in \{1, ...K\}$, the intensity of the buy orders flow L_i^- at the i^{th} limit of the bid side,
- $\lambda_i^{C^+}$, $i \in \{1, ...K\}$, the intensity of the cancel orders flow C_i^+ at the i^{th} limit of the ask side,
- $\lambda_i^{C^-}$, $i \in \{1, ...K\}$, the intensity of the cancel orders flow C_i^- at the i^{th} limit of the bid side,

and let λ^L, λ^C, λ^M be such that

$$\begin{aligned}
\sum_{i=0}^{K} \lambda\big(L_i^{\pm}\big)(z) &\leq \lambda^L\big(|\underline{a}| + |\underline{b}|\big), \\
\sum_{i=0}^{K} \lambda\big(C_i^{\pm}\big)(z) &\leq \lambda^C\big(|\underline{a}| + |\underline{b}|\big), \\
\lambda\big(M^-\big)(z) + \lambda\big(M^+\big)(z) &\leq \lambda^M\big(|\underline{a}| + |\underline{b}|\big),
\end{aligned}$$

for all state $(\underline{a}, \underline{b})$ of the LOB. We remind that λ^L, λ^C, λ^M are well-defined under assumption **(Harrivals)**.

Remark 3.2.1. *The linear conditions on the intensities are required to prove that the control problem is well-posed.*

Remark 3.2.2. *We generalize the structure of the orders arrivals in section 3.5 by modeling them as Hawkes processes with exponential kernel.*

We provide in figure 3.1 a graphical representation of an LOB that may help to get more familiar with the introduced notations.

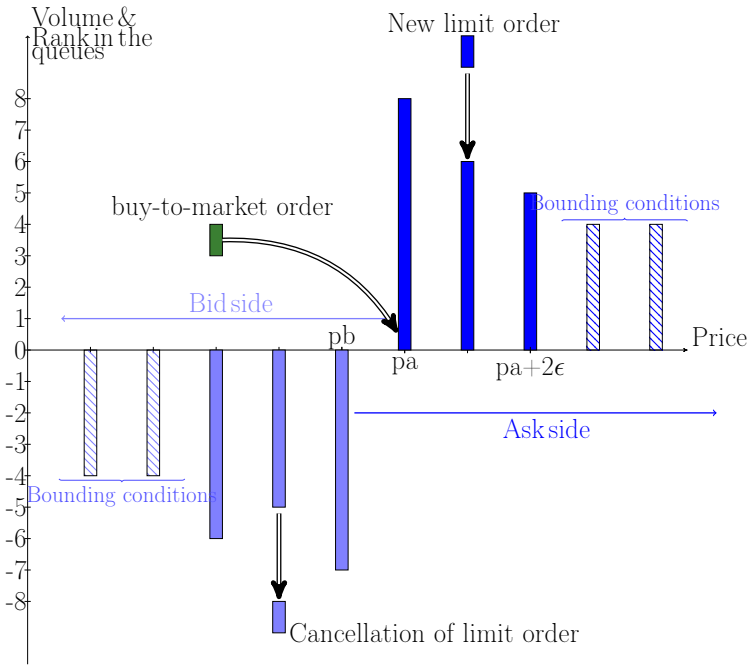


Figure 3.1 – Order book dynamics: in this example, $K = 3$, $q = 1$, $a_\infty = 4$, $b_\infty = -4$, $\underline{a} = (8, 6, 5)$, $\underline{b} = (-7, -5, -6)$. The spread is equal to 1. At any time, the order book can receive limit orders, market orders or cancel orders.

3.2.2 Market maker strategies

We assume that the market is governed by a *FIFO* (*First In First Out*) rule, which means that each limit of the order book is a queue where the first order in the queue is the first one to be executed. We consider a market maker who stands ready to send buy and sell limit orders on a regular and continuous basis at quoted prices. A usual assumption in stochastic control in order to characterize value function as solution of HJB equation is to constrain the control space to be compact. In this spirit, we shall make the following assumption on the market maker's decisions.

(Hcontrol) Assume that at any time, the total number of limit orders placed by the marker maker does not exceed a fixed (possibly large) integer $\bar{M}$.

3.2.2.1 Controls and strategies of the market maker

The market maker can choose at any time to keep, cancel or take positions in the order book (as long as she does not hold more than $\bar{M}$ positions in the order book). Her positions are fully described by the following $\bar{M}-$dimensional vectors $\underline{ra}_t$, $\underline{rb}_t$, $\underline{na}_t$, $\underline{nb}_t$ where $\underline{ra}$ (resp. $\underline{rb}$) records the limits in which the market maker's sell (resp. buy) orders are located; and $\underline{na}$ (resp. $\underline{nb}$) records the ranks in the queues of each market maker's sell (resp. buy) orders. In order to guarantee that the strategy of the market maker is predictable w.r.t. the natural filtration generated by the orders arrivals processes, we shall make the following assumption.

(Harrivals2) The intensities do not depend on the control. Moreover, the market maker does not cross the spread.

We discuss in Appendix how to control the intensity, by transferring the control on the probability measure, see Section 3.A

To simplify the theoritical analysis, we also make the following assumption: **(Harrivals3)** Assume that the market maker does not change her strategy between two orders arrivals of the order book.

In other words, the market maker makes a decision right after one of the order arrivals processes $L^{\pm}, C^{\pm}, M^{\pm}$ jumps, and keep it until the next the jump of an order arrival.

Note that assumption **(Harrivals3)** is mild if the order book jumps frequently, since the market maker can change her decisions frequently in such a case.

We provide in figure 3.2 a graphical representation of the controlled LOB. Notice that the market maker interacts with the order book by placing orders at some limits. The latter have ranks that evolve after each orders arrivals.

Denote by $(T_n)_{n\in\mathbb{N}}$ the sequence of jump times of the order book. We denote by $\mathbb{A}$ the set of the admissible strategies, defined as the predictable processes $\left(\underline{ra}_t, \underline{rb}_t\right)_{t\leq T}$ such that:

- for all $n \in \mathbb{N}$, $\left(\underline{ra}_t, \underline{rb}_t\right) \in \{0, ..., K\}^{\bar{M}} \times \{0, ..., K\}^{\bar{M}}$ are constant on $\left(T_n, T_{n+1}\right]$
- $ra_*, rb_* \geq a0$

where, for every vector $\underline{a}$: $a_* = \min_{0\leq i\leq K}\{a_i \ s.t. \ \underline{a}_i \neq -1\}$; and: $a0 = \operatorname{argmin}_{0\leq i\leq K}\left(\underline{a}_i \ s.t. \ a_i > 0\right)$. The control is the double vector of the positions of the $\bar{M}$ market maker's orders in the order book. By convention, we set: $ra_i(t) = -1$ if the ith market maker's order is not placed in the order book.

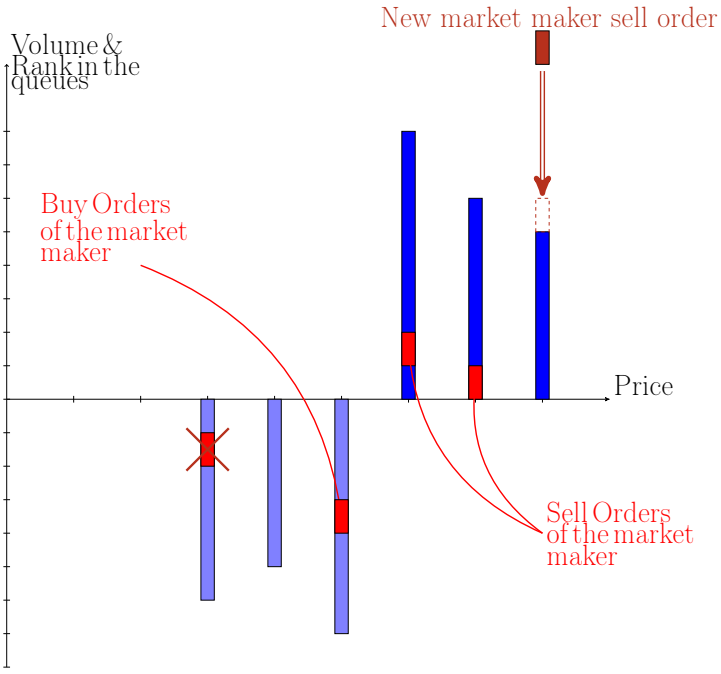


Figure 3.2 – Example of market maker's placements and decisions she might make. In this example: her positions are $\underline{ra} = (0, 1, -1, ...)$, $\underline{rb} = (0, 2, -1...)$. The ranks vectors associated are $\underline{na} = (2, 1, -1\dots)$ and $\underline{nb} = (4, 2, -1, ...)$. After each order arrival, she can send new limit orders, cancel some positions, or just keep the latter unchanged.

3.2.2.2 Controlled order book

We describe the controlled order book by the following state process Z:

$$Z_t := \left(X_t, Y_t, \underline{a}_t, \underline{b}_t, \underline{na}_t, \underline{nb}_t, pa_t, pb_t, \underline{ra}_t, \underline{rb}_t\right),$$

where, at time t:

- X_t is the cash held by the market maker on a zero interest account.

- Y_t is the inventory of the market maker, i.e. it is the (signed) number of shares held by the market maker.
- pa_t is the ask price, i.e. the cheapest price a general market participant is willing to sell stock.
- pb_t is the bid price, i.e. the highest price a general market participant is willing to buy stock.
- $\underline{a}_t = (a_1(t), \dots, a_K(t))$ (resp. $\underline{b}_t = (b_1(t), \dots, b_K(t))$) describes the ask (resp. bid) side: $i \in \{1, \dots, K\}$, $a_i(t)$ is the sum of all the general market participants' sell orders which are i ticks away from the bid (resp. ask) price.
- $\underline{ra}_t$ (resp. $\underline{ra}_t$) describes the market maker's orders in the ask (resp. bid) side: for $i \in \{1, ..., \bar{M}\}$, $\underline{ra}_t(i)$ is the number of ticks between the i-th market maker's sell (resp. bid) order and the bid (resp. ask) price. By convention, we set $\underline{ra}_t(i) = -1$ (resp. $\underline{rb}_t(i) = -1$) if the i-th sell (resp. buy) order of the market maker is not placed in the order book. As a result $\underline{ra}_t(i), \underline{rb}_t(i) \in \{0, \dots, K\} \cup \{-1\}$.
- $\underline{na}_t$ (resp. $\underline{nb}_t$) describes the ranks of the market maker's orders in the ask (resp. bid) side. For $i \in \{1, ..., \bar{M}\}$, $\underline{na}_t(i) \in \{-1, ..., |a| + \bar{M}\}$ (resp. $\underline{nb}_t(i) \in \{-1, ..., |b| + \bar{M}\}$) is the rank of the i-th sell (resp. buy) orders of the market maker in the queue. By convention, we assume that $\underline{na}_t(i) = -1$ (resp. $\underline{nb}_t(i) = -1$) if the i-th sell (resp. buy) order of the market maker is not placed in the order book.

The dynamics of (Z_t) has been computed in the case where the set of admissible strategies is restricted to those where the market maker only makes orders at the two best limits in the bid and ask sides. We present the computations in Section 3.B in the Appendix, for the case where the market maker can only send limit orders at the best-bid and best-ask. We only present the numerical results, in Section 3.4.4, in the case where the market maker can send limit orders at the two best limits in the ask and bid sides.

3.3 Existence and characterization of the optimal strategy

3.3.1 Definition and well-posedness of the value function

We denote by V the value function for the market-making problem, defined as follows:

$$V(t,z) = \sup_{\alpha \in \mathbb{A}} \mathbb{E}^{\alpha}_{t,z} \left[\int_t^T f\big(\alpha_s, Z_s\big)\, \mathrm{d}s + g\big(Z_T\big) \right], \quad (t,z) \in [0,T] \times E, \tag{3.3.1}$$

where:

- $\mathbb{A}$ is the set of the admissible strategies, defined in Section 3.2.2.1.
- f and g are respectively the running and terminal reward functions. A usual definition for g is the market maker's wealth function, possibly with an inventory penalization, i.e. $g : z \mapsto x + L(y) - \eta y^2$ where L[2] returns the amount earned from the immediate liquidation of the inventory; where η is the penalization parameter of the latter; and where we remind that y stands for the (signed) market maker's inventory.

[2] L is defined as follows:

$$L(z) = \begin{cases} \sum_{k=0}^{\jmath-1} \big[a_k(pa + k\epsilon)\big] + (y - a_0 - ... - a_{\jmath-1})(pa + \jmath\epsilon) & \text{if } y < 0 \\ -\sum_{k=0}^{\jmath-1} \big[b_k(pb - k\epsilon)\big] + (y + b_0 + ... + b_{\jmath-1})(pb - \jmath\epsilon) & \text{if } y > 0 \\ 0 & \text{if } y = 0, \end{cases}$$

for all state $z = \big(x, y, \underline{a}, \underline{b}, \underline{na}, \underline{nb}, pa, pb, \underline{ra}, \underline{rb}\big)$ of order book, where:

$$\jmath = \begin{cases} \min\big\{j \big| \sum_{i=0}^{j} a_i > -y\big\} & \text{if } y < 0 \\ \min\big\{j \big| \sum_{i=0}^{j} |b_i| > y\big\} & \text{if } y > 0. \end{cases}$$

- $\mathbb{E}^{\alpha}_{t,z}$ stands for the expectation conditioned by $Z_t = z$ and when strategy $\alpha = (\alpha_s)_{t\leq s<T}$ is followed on $[t, T]$.

We shall assume conditions on the rewards to insure the well-posedness of the market-making problem.

(Hrewards) The expected running reward is uniformly upper-bounded w.r.t. the strategies in $\mathbb{A}$, i.e.

$$\sup_{\alpha\in\mathbb{A}} \mathbb{E}^{\alpha}_{t,z}\Big[\int_t^T f^+(Z_s,\alpha_s)\,\mathrm{d}s\Big] < +\infty$$

holds. The terminal reward $g(Z_T)$ is a.s. no more than linear with respect to the number of events up to time T, denoted by N_T in the sequel, i.e. there exists a constant $c_1 > 0$ such as $g(Z_T) \leq c_1 N_T$, a.s..

Remark 3.3.1. *Under Assumption **(Hcontrols)**, Assumption **(Hrewards)** holds when g is defined as the wealth of the market maker plus an inventory penalization. In particular, we have $g(Z_T) \leq N_T\bar{M}$, where $\bar{M}$ is the maximal number of orders that can be sent by the market maker, which holds a.s. since the best profit the market maker can make is when her buy (resp. sell) limit orders are all executed, and then the price keeps going to the right (resp. left) direction. Hence the second condition of **(Hrewards)** holds with $c_1 = \bar{M}$.*

The following Lemma 3.3.1 tackles the well-posedness of the control problem.

Lemma 3.3.1. *Under **(Hrewards)** and **(Hcontrols)**, the value function is well-defined, i.e.*

$$\sup_{\alpha\in\mathbb{A}} \mathbb{E}^{\alpha}_{t,z}\left[g(Z_T) + \int_t^T r\big(\alpha_s, Z_s\big)\,\mathrm{d}s\right] < +\infty,$$

where, as defined previously, $\mathbb{E}^{\alpha}_{t,z}[.]$ stands for the expectation conditioned by the event $\{Z_t = z\}$, assuming that strategy $\alpha \in \mathbb{A}$ is followed in $[t, T]$.

Proof. Denote by $(N_t)_t$ the sum of all the arrivals of orders up to time t. Under **(Hrewards)**, we can bound $\mathbb{E}^{\alpha}_{t,z}\left[\int_t^T f\big(\alpha_s, Z_s\big)\,\mathrm{d}s + g(Z_T)\right]$, the reward functional at time t associated to a strategy $\alpha \in \mathbb{A}$, as follows:

$$\begin{aligned}\mathbb{E}^{\alpha}_{t,z}\left[\int_t^T f\left(\alpha_s, Z_s\right)\mathrm{d}s + g(Z_T)\right] &\leq \sup_{\alpha}\mathbb{E}^{\alpha}_{t,z}\left[g(Z_T)\right] + \sup_{\alpha}\mathbb{E}^{\alpha}\left[\int_t^T f^+(Z_s,\alpha_s)\,\mathrm{d}s\right]\\ &\leq c_1 \sup_{\alpha}\mathbb{E}^{\alpha}_{t,0}\left[N_T\right] + \sup_{\alpha}\mathbb{E}^{\alpha}_{t,z}\left[\int_t^T f^+(Z_s,\alpha_s)\,\mathrm{d}s\right], \quad (3.3.2)\end{aligned}$$

where once again, for all general process M and all $m \in E$, $\mathbb{E}^{\alpha}_{t,m}[M_T]$ stands for the expectation of M_T conditioned by $M_t = m$ and assuming that the market maker follows strategy $\alpha \in \mathbb{A}$ in $[t, T]$. Let us show that the first term in the r.h.s. of (3.3.2) is bounded. On one hand, we have:

$$\mathbb{E}^{\alpha}_{t,0}\left[N_T\right] \leq \|\lambda\|_{\infty}\int_0^T \mathbb{E}\left(|a|_t + |b|_t\right)\mathrm{d}t, \quad (3.3.3)$$

where $\|\lambda\|_{\infty} := \lambda^L + \lambda^C + \lambda^M$ is a bound on the intensity rate of N_t. On the other hand, there exists a constant $c_2 > 0$ such that $\mathrm{d}(|a| + |b|)_t \leq c_2\,\mathrm{d}L_t$ so that: $\mathbb{E}^{\alpha}_{t,|a|_0+|b|_0}\left[|a|_t + |b|_t\right] \leq |a|_0 + |b|_0 + c_3\int_0^t \mathbb{E}\left[|a|_s + |b|_s\right]\mathrm{d}s$. Applying Gronwall's inequality, we then get:

$$\mathbb{E}^{\alpha}_{t,|a|_0+|b|_0}\left[|a|_t + |b|_t\right] \leq \left(|a|_0 + |b|_0\right)e^{c_3 t}. \quad (3.3.4)$$

Plugging (3.3.4) into (3.3.3) finally leads to:

$$\mathbb{E}^{\alpha}_{t,0}\left[N_T\right] \leq c_4 e^{c_3 T}$$

wit c_3 and $c_4 > 0$ that do not depends on α, which proves that the first term in the r.h.s. of (3.3.2) is bounded. Also, its second term in the r.h.s. of (3.3.2) is bounded under **(Hrewards)**. Hence, the reward functional is bounded uniformly in α, which proves that the value function of the considered market-making problem is well-defined. □

3.3.2 Markov Decision Process formulation of the market-making problem

In this section, we aim first at reformulating the market-making problem as a Markov Decision Process (MDP), and secondly deriving a characterization of the value function as solution of a Bellman equation.

We consider the Markov Decision Process (MDP) characterized by the following information

$$\underbrace{[0,T]\times E}_{\text{state space}},\quad \underbrace{A_z}_{\text{market maker control}},\quad \underbrace{\lambda}_{\text{intensity of the jump}},\quad \underbrace{Q}_{\text{transitions kernel}},\quad \underbrace{r}_{\text{reward}}$$

such that:

- $E := \mathbb{R}\times\mathbb{N}\times\mathbb{N}^K\times\mathbb{N}^K\times\mathbb{N}^{\bar{M}}\times\mathbb{N}^{\bar{M}}\times\mathbb{N}^{\bar{M}}\times\mathbb{N}^{\bar{M}}\times\mathbb{R}\times\mathbb{R}$ is the state space of (Z_t). For $z\in E$, $z=\big(x,y,\underline{a},\underline{b},\underline{na},\underline{nb},\underline{ra},\underline{rb},pa,pb\big)$ where: x is the cash held by the market maker, y her inventory; $\underline{na}$ (resp. $\underline{nb}$) is the $\bar{M}$-dimensional vector of the ranks of the market maker's sell (resp. buy) orders in the queues ; $\underline{ra}$ (resp. $\underline{rb}$) is the $\bar{M}$-dimensional vector of the number of ticks the $\bar{M}$ market maker's sell (resp. buy) orders are from the bid (resp. ask) price; pa (resp. pb) is the ask-price (resp. bid-price).
- for every state $z\in E$, denote by A_z the space of the admissible controls which is the set of all the actions the market maker can take when the order book is at state z.

  $$A_z=\Big\{\underline{ra},\underline{rb}\in\{0,...,K\}^{\bar{M}}\times\{0,...,K\}^{\bar{M}}\Big|rb_*,ra_*\geq a0\Big\}$$

  , where we define $c_*=\min_i\{c_i|c_i\neq -1\}$ and $c0=\underset{0\leq i\leq K}{\operatorname{argmin}}\{\underline{c}_i>0\}$ for $\underline{c}\in\mathbb{N}^{\bar{M}}$. The control is the vectors of positions of the market maker's orders. The condition for the control to be admissible comes from the assumption that the market maker is not allowed to cross the spread.
- Given a market-making strategy α, the stochastic evolution is given by a marked point process (T_n,Z_n) where (T_n) is the increasing sequence of jump times of the controlled order book with intensity $\lambda(Z_{n-1})$. Just after the jump at time T_n, the process can jump again, due to the decision of the market maker. Then it remains constant on $]T_n,T_{n+1}[$ since the market maker does not change her strategy between two jumps.

  We denote by $\phi^a(z)\in E$ the state of the order book at time t such that $T_n<t<T_{n+1}$, given that $Z_{T_n}=z$ and given that the strategy a has been chosen by the market maker at time T_n.
- In the sequel, we denote $\big([0,T]\times E\big)^C:=\Big\{\big(t,z,a\big)\in E\times\{0,\dots,K\}^{2\bar{M}}\big|t\in[0,T],z\in E,a\in A_z\Big\}$, and $E^C:=\Big\{\big(z,a\big)\in E\times\{0,\dots,K\}^{2\bar{M}}\big|z\in E,a\in A_z\Big\}$. Q' is the stochastic kernel from E^C to E that describes the distribution of the jump goals, i.e., $Q'\big(B|z,u\big)$ is the probability that the order book jumps in the set B given that it was at state $z\in E$ right before the jump, and the control action $u\in A_z$ has been chosen right after the jump time.

  An admissible policy $\alpha=(\alpha_t)$ is entirely characterized by decision functions $f_n:[0,T]\times E\to A$ such that[3]

  $$\alpha_t=f_n(T_n,Z_n)\text{ for }t\in\big(T_n,T_{n+1}\big]$$

  By abuse of notation, we denote in the sequel by α the sequence of controls $(f_n)_{n=0}^{\infty}$. The intensity of the controlled process (Z_t) is:

  $$\lambda(z):=\lambda^{M^+}(z)+\lambda^{M^-}(z)+\sum_{1\leq j\leq K}\lambda^{L_j^+}(z)+\sum_{1\leq j\leq K}\lambda^{L_j^-}(z)+\sum_{1\leq j\leq K}\lambda^{C_j^+}(z)+\sum_{1\leq j\leq K}\lambda^{C_j^-}(z)$$

[3]Note that we restrict ourselves to the feedback controls here

It does not depend on the strategy α chosen by the market maker since we assumed that the general participants does not "see" the market maker's orders in the order book. The intensity of the order book process only depends on the vectors $\underline{a}$ and $\underline{b}$.

The transition kernel of the controlled order book, given a state z, is given by:

$$Q'\big(z'|z,u\big) = \begin{cases} \frac{\lambda^{M^+}(z)}{\lambda(z)} & \text{if } z' = e^{M^+}(\phi^u(z)) \\ \vdots & \\ \frac{\lambda^{C^+}(z)}{\lambda(z)} & \text{if } z' = e^{C_K^+}(\phi^u(z)), \end{cases}$$

where $\phi^u(z)$ is the new state of the controlled order book when decision u as been taken and when the order book was at state z before the decision; $e^{M^+}(z)$ is the new state of the order book right after it received a buy market order, given that it was at state z before the jump; and $e^{C_i^\pm}(z)$ is the new state of the order book right after it received a cancel order from a general market participant on its i^{th} ask/bid limit, given that it was at state z.

Let us fix an admissible policy $\alpha = (f_n)_{n=0}^\infty \in \mathbb{A}$ and take $t \in [0,T]$. Then, for all Borelian B in E, it holds:

$$\begin{aligned}\mathbb{P}\big(T_{n+1} - T_n \le t, Z_{n+1} \in B | T_0, Z_0, ...T_n, Z_n\big) &= \lambda(Z_n) \int_0^t e^{-\lambda(Z_n)s} Q'\big(B|Z_{T_n}, \alpha_{T_n}\big)\,\mathrm{d}s \\ &= \lambda(Z_n) \int_0^t e^{-\lambda(Z_n)s} Q'\big(B|Z_{T_n}, f_n(Z_n)\big)\,\mathrm{d}s,\end{aligned}$$

so that the stochastic kernel Q of the MDP is defined as follows:

$$Q\big(B\times C|t,z,\alpha\big) := \lambda(z)\int_0^{T-t} e^{-\lambda(z)s}\mathbf{1}_B(t+s)Q'\big(C|\phi^\alpha(z),\alpha\big)\,\mathrm{d}s + e^{-\lambda(z)(T-t)}\mathbf{1}_{T\in B, z\in C},$$

for all Borelian sets $B \subset \mathbb{R}_+$ and $C \subset E$, for all $(t,z) \in [0,T]\times E$, and for all $\alpha \in A$.

We denote by $(T_n, Z_n)_{n\in\mathbb{N}}$ the corresponding state of the controlled Markov chain. It remains to define the value function of this reformulated control problem.

Let r be the running reward function $r : [0,T]\times E^C \to \mathbb{R}$ defined as:

$$r(t,z,a) := -c\big(z,a\big)e^{-\lambda(z)(T-t)}(T-t)\mathbf{1}_{t>T} + c\big(z,a\big)\Big(\frac{1}{\lambda(z)} - \frac{e^{-\lambda(z)(T-t)}}{\lambda(z)}\Big) + e^{-\lambda(z)(T-t)}g(z)\mathbf{1}_{t\le T}, \tag{3.3.5}$$

and let us define the cumulated reward functional associated to the discrete-time Markov Decision Model for an admissible policy $(f_n)_{n=0}^\infty$ as:

$$V_{\infty,(f_n)}(t,z) = \mathbb{E}_{t,z}^{(f_n)}\left[\sum_{n=0}^\infty r\big(T_n, Z_n, f_n(T_n,Z_n)\big)\right].$$

The value function associated to $(T_n,Z_n)_{n\in\mathbb{N}}$ is then defined as the supremum of the cumulated reward functional over all the admissible controls in $\mathbb{A}$, i.e.

$$V_\infty(t,z) = \sup_{(f_n)_{n=0}^\infty \in \mathbb{A}} V_{\infty,\alpha}(t,z), \qquad (t,z)\in[0,T]\times E, \tag{3.3.6}$$

Notice that we used the same notation for admissible controls of the MDP and those of the continuous-time control problem.

Proposition 3.3.1. *The value function of the MDP defined by (3.3.6) coincides with (3.3.1), i.e. we have for all $(t,z) \in E'$:*

$$V_\infty(t,z) = V(t,z). \tag{3.3.7}$$

Proof. Step 1: We show that for all $\alpha = (f_n) \in \mathbb{A}$ and all $(t,z) \in E'$

$$V_\alpha(t,z) = V_\infty^{(f_n)}(t,z). \tag{3.3.8}$$

Let us first denote by $H_n := (T_0, Z_0, ..., T_n, Z_n)$. Notice then that for all admissible strategy α:

$$\begin{aligned} V_\alpha(t,z) &= \mathbb{E}_{t,z}^{\alpha}\Bigg[\sum_{n=0}^{\infty} \mathbf{1}_{T>T_{n+1}}\big(T_{n+1}-T_n\big)c\big(Z_n,\alpha_n\big) \\ &\qquad + \mathbf{1}_{[T_n\le T<T_{n+1})}\Big(g(Z_T) - \eta {Y_T}^2 + (T-T_n)c\big(Z_n,\alpha_n\big)\Big)\Bigg] \\ &= \sum_{n=0}^{\infty} \mathbb{E}_{t,z}^{(f_n)}\Big[r\big(T_n, Z_n, f_n(T_n,Z_n)\big)\Big], \end{aligned} \tag{3.3.9}$$

where we conditioned by H_n between the first and the second line. We recognize $V_\infty^{(f_n)}$ in the r.h.s. of (3.3.9), so that the proof of (3.3.8) is completed.

It remains to take the supremum over all the admissible strategies $\mathbb{A}$ in (3.3.8) to get (3.3.7). □

From Proposition 3.3.1, we deduce that the value function of the market-making problem is the same as the value function V_∞ of the discrete-time MDP. We now aim at solving the MDP control problem. To proceed, we first define the maximal reward mapping for the non finite horizon MDP:

$$\begin{aligned} (\mathcal{T}v)(t,z) &:= \sup_{a\in A_z}\Big\{r(t,z,a) + \int v(t',z')Q(t',z'|t,\phi^a(z),a)\Big\} \\ &= \sup_{a\in A_z}\Big\{r(t,z,a) + \lambda(z)\int_0^{T-t} e^{-\lambda(z)s}\int v(t+s,z')Q'\big(dz'|\phi^a(z),a\big)ds\Big\}, \end{aligned} \tag{3.3.10}$$

where we recall that:

- $\phi^\alpha(z)$ is the new state of the order book when the market maker follows the strategy α and the order book is at state z before the decision is taken.
- $\lambda(z)$ is the intensity of the order book process given that the order book is at state z.

We shall tighten assumption **(Hrewards)** in order to guarantee existence and uniqueness of a solution to (3.3.1), as well as characterizing the latter.

(HrewardsBis): The running and terminal rewards are at most quadratic w.r.t. the state variable, uniformly w.r.t. the control variable, i.e.

1. The running reward f is such that $|c|$ is uniformly bounded by a quadratic in z function, i.e. there exists $c_5 > 0$ such that:
$$\forall (z,a) \in E \times A, \quad |f(z,a)| \le c_5(1+|z|^2).$$
2. The terminal reward g has no more than a quadratic growth, i.e. there exists $c_6 > 0$ such that:
$$\forall z \in E, \quad |g(z)| \le c_6(1+|z|^2).$$

Remark 3.3.2. *Assumption **(HrewardsBis)** holds in the case where g is the terminal wealth of the market maker plus a penalization of her inventory, and where with no running reward, i.e. $f = 0$.*

The main result of this section is the following theorem that gives existence and uniqueness of a solution to (3.3.1), and moreover characterizes the latter as fixed point of the maximal reward operator defined in (3.3.10).

Theorem 3.3.1. *$\mathcal{T}$ admits a unique fixed point v which coincides with the value function of the MDP. Moreover we have:*

$$v = V_\infty = V.$$

Denote by f^* the maximizer of the operator $\mathcal{T}$. Then $(f^*, f^*, ...)$ is an optimal stationary (in the MDP sense) policy.

Remark 3.3.3. *Theorem 3.3.1 states that the optimal strategy is stationary in the MDP formulation of the problem, but of course, it is not stationary for the original time-continuous trading problem with finite horizon (3.3.1), since the time component is not a state variable anymore in the original formulation. Actually, given $n \in \mathbb{N}$ and the state of order book z at that time, the optimal decision to take at time T_n is given by $f^*(T_n, z)$.*

We devoted the next section to the proof of Theorem 3.3.1.

3.3.3 Proof of Theorem 3.3.1

Remind first that we defined in the previous section $E^C := \left\{(z,a) \in E \times \{0,\dots,K\}^{2\bar{M}} \middle| z \in E, a \in A_z\right\}$ and $\big([0,T]\times E\big)^C := \left\{(t,z,a) \in [0,T]\times E \times \{0,\dots,K\}^{2\bar{M}} \middle| t \in [0,T], z \in E, a \in A_z\right\}$.

Let us define the bounding functions:

Definition 3.3.1. *A measurable function $b : E \to \mathbb{R}_+$ is called a* bounding function *for the controlled process (Z_t) if there exists positive constants c_c, c_g, $c_{Q'}, c_\phi$ such that:*

1. *$|f(z,a)| \le c_c b(z)$ for all $(z,a) \in E^C$.*
2. *$|g(z)| \le c_g b(z)$ for all z in E.*
3. *$\int b(z')Q'(dz'|z,a) \le c_{Q'} b(z)$ for all $(z,a) \in E^C$.*
4. *$b(\phi_t^\alpha(z)) \le c_\phi b(z)$ for all $(t,z,\alpha) \in \big([0,T]\times E\big)^C$.*

Proposition 3.3.2. *Let b be such that :*

$$\forall z \in E, b(z) := 1 + |z|^2.$$

*Then, b is a bounding function for the controlled process (Z_t), under Assumption **(HrewardsBis)**.*

Proof. Let us check that b defined in 3.3.2 satisfies the four assertions of definition 3.3.1.

- Assertion 1 and 2 of definition 3.3.1 holds under **(HrewardsBis)**.
- First notice that $\underline{ra}, \underline{rb}$ are bounded by $\sqrt{\bar{M}}K$ (where we recall that K is the number of limits in each side of the order book, and $\bar{M}$ is the biggest number of limit orders that the market maker is allowed to send in the market). Secondly, $pa' \in B(pa, K), pb' \in B(pb, K)$, where $B(x,r)$ is the ball centered in x with radius $r > 0$, because of the limit conditions that we imposed in our LOB model. And last, we can see that $|\underline{a}'| \le |\underline{a}| + a_\infty K$. These three bounds are linear w.r.t. z so that assertion 3 holds.
- $\phi^\alpha(z) = z^\alpha$ only differs from z by its $\underline{na}$, $\underline{nb}$, and $\underline{ra}$, $\underline{rb}$ components.
  But $|\underline{na}| \le \sqrt{\bar{M}}\big(|\underline{a}| + \bar{M}\big)$ and $|\underline{nb}| \le \sqrt{\bar{M}}\big(|\underline{b}| + \bar{M}\big)$ are bounded by a linear function of $(\underline{a}, \underline{b})$, also $|\underline{ra}|$ and $|\underline{rb}|$ are bounded by the universal constant $\sqrt{\bar{M}}K$, so assertion 4 in Definition 3.3.1 holds.

□

Let us define

$$\Lambda := (4K+2)\sup\left\{\frac{\lambda^{M^\pm}}{|\underline{a}|+|\underline{b}|}, \frac{\lambda^{L^\pm}}{|\underline{a}|+|\underline{b}|}, \frac{\lambda^{C^\pm}}{|\underline{a}(z)|+|\underline{b}(z)|}\right\}.$$

Note that Λ is well-defined under **(Harrivals)**.

Proposition 3.3.3. *If b is a bounding function for (Z_t), then*

$$b(t,z) := b(z)e^{\gamma(z)(T-t)}, \text{ with } \gamma(z) = \gamma_0(4K+2)\Lambda\big(1+|\underline{a}|+|\underline{b}|\big) \text{ and } \gamma_0 > 0$$

is a bounding function for the MDP, i.e. for all $t \in [0,T], z \in E, a \in A_z$, we have:

$$|r(t,z,a)| \leq c_g b(t,z),$$
$$\int b(s,z')Q(ds,dz'|t,z,a) \leq c_\phi c_Q e^{C(T-t)}\frac{1}{1+\gamma_0}b(t,z),$$

with $C = \gamma_0\Lambda K(4K+2)\big(|a|_\infty + |b|_\infty\big)$.

Proof. Let $z' = \big(x', y', \underline{a}', \underline{b}', \underline{na}', \underline{nb}', \underline{ra}', \underline{rb}'\big)$ be the state of the order book after an exogenous jump occursn given that it was at state z before the jump. Since $|\underline{a}'| \leq |\underline{a}| + a_\infty K$ and $|\underline{b}'| \leq |\underline{b}| + b_\infty$, where a_∞ and b_∞ are defined as the border conditions of the order book, we have:

$$\gamma(z') \leq \gamma(z) + C, \tag{3.3.11}$$

with $C = \gamma_0\Lambda K(4K+2)(a_\infty + b_\infty)$. Then, we get:

$$\begin{aligned}
\int b(s,z')Q(ds,dz'|t,\phi^\alpha(z),\alpha) &= \lambda(z)\int_0^{T-t} e^{-\lambda(z)s}\int b(t+s,z')Q'\big(dz'|\phi_s^\alpha(z),\alpha\big)\,\mathrm{d}s \\
&= \lambda(z)\int_0^{T-t} e^{-\lambda(z)s}\int b(z')e^{\gamma(z')(T-(t+s))}Q'\big(dz'|\phi_s^\alpha(z),\alpha\big)\,\mathrm{d}s \\
&\leq \lambda(z)\int_0^{T-t} e^{-\lambda(z)s}\int b(z')e^{(\gamma(z)+C)(T-(t+s))}Q'\big(dz'|\phi_s^\alpha(z),\alpha\big)ds \\
&\leq \lambda(z)\int_0^{T-t} e^{-\lambda(z)s}e^{(\gamma(z)+C)(T-(t+s))}\int b(z')Q'\big(dz'|\phi_s^\alpha(z),\alpha\big)ds \\
&\leq \lambda(z)\int_0^{T-t} e^{-\lambda(z)s}e^{(\gamma(z)+C)(T-(t+s))}c_Q c_\phi b(z)ds \\
&\leq \frac{\lambda(z)c_Q c_\phi}{\lambda(z)+\gamma(z)+C}e^{(\gamma(z)+C)(T-t)}\Big(1-e^{-(T-t)(\lambda(z)+\gamma(z)+C)}\Big)b(z) \\
&\leq c_Q c_\phi\frac{\lambda(z)}{\lambda(z)+\gamma(z)+C}e^{C(T-t)}\Big(1-e^{-(T-t)(\lambda(z)+\gamma(z)+C)}\Big)b(t,z),
\end{aligned}$$

where we applied (3.3.11) at the thrid line. It remains to notice that

$$\frac{\lambda(z)}{\lambda(z)+\gamma(z)+C} = \frac{\lambda(z)}{\lambda(z)\big(1+\gamma_0\big)+\gamma_0\big[\underbrace{\Lambda(|a|+|b|)-\lambda(z)}_{\geq 0}\big]} \leq \frac{1}{1+\gamma_0},$$

we complete the proof of the Proposition. □

Denote by $\|.\|_b$ the *weighted supremum norm* such that for all measurable function $v : E' \to \mathbb{R}$,

$$\|v\|_b := \sup_{(t,z)\in E'}\frac{|v(t,z)|}{b(t,z)},$$

and define the set:

$$\mathbb{B}_b := \Big\{ v : E' \to \mathbb{R} | v \text{ is measurable and } \|v\|_b < \infty \Big\}.$$

Moreover let us define

$$\alpha_b := \sup_{(t,z,\alpha)\in E'\times\mathcal{R}} \frac{\int b(s,z')Q(ds,dz'|t,\phi^\alpha(z),\alpha)}{b(t,z)}.$$

From the preceding estimations we can bound α_b as follows:

$$\alpha_b \leq c_Q c_\phi \frac{1}{1+\gamma_0} e^{CT},$$

So that, by taking: $\gamma_0 = c_Q c_\phi e^{CT}$, we get: $\alpha_b < 1$. In the sequel, we then assume w.l.o.g. that $\alpha_b < 1$. Recall that the maximal reward mapping for the MDP has been defined as:

$$\mathcal{T}v : (t,z) \mapsto \sup_{a\in A_z} \left\{ r(t,z,a) + \lambda(z) \int_0^{T-t} e^{-\lambda(z)s} \int v(t+s,z') Q'\big(dz'|\phi^a(z),a\big)\, \mathrm{d}s \right\}$$

It is straightforward to see that:

$$\|\mathcal{T}v - \mathcal{T}w\|_b \leq \alpha_b \|v-w\|_b, \tag{3.3.12}$$

which implies that $\mathcal{T}$ is contracting, since $\alpha_b < 1$.

Let $\mathcal{M}$ be the set of all the continuous function in $\mathbb{B}_b$. Since b is continuous, $(\mathcal{M}, \|.\|_b)$ is a Banach space.

$\mathcal{T}$ sends $\mathcal{M}$ to $\mathcal{M}$. Indeed, for all continuous function v in $\mathbb{B}_b$, $(t,z,a) \mapsto r(t,z,a) + \lambda(z) \int_0^{T-t} e^{-\lambda(z)s} \int v(t+s,z') Q'\big(dz'|\phi^a(z),a\big) ds$ is continuous on $[0,T]\times E^C$. A_z is finite, so we get the continuity of the application:

$$\mathcal{T}v : (t,z) \mapsto \sup_{a\in A_z} \left\{ r(t,z,a) + \lambda(z) \int_0^{T-t} e^{-\lambda(z)s} \int v(t+s,z') Q'\big(dz'|\phi^a(z),a\big)\, \mathrm{d}s \right\}.$$

Proposition 3.3.4. *There exists a maximizer for $\mathcal{T}$, i.e. let $v \in \mathcal{M}$, then there exists a Borelian function $f : [0,T]\times E \to A$ such that for all $(t,z) \in E'$:*

$$\mathcal{T}v\Big(t,z,f\big(t,z\big)\Big) = \sup_{a\in A} \left\{ r(t,z,a) + \lambda(z) \int_0^{T-t} e^{-\lambda(z)s} \int v(t+s,z') Q'\big(dz'|\phi^a(z),a\big)\, \mathrm{d}s \right\}$$

Proof. $D^*(t,z) = \Big\{ a \in A \big| \mathcal{T}_a v(t,z) = \mathcal{T}v(t,z) \Big\}$ is finite, so it is compact. So $(t,z) \mapsto D^*(t,z)$ is a compact-valued mapping. Since the application $(t,z,a) \mapsto \mathcal{T}_a(t,z) - \mathcal{T}(t,z)$ is continuous, we get that $D^* = \Big\{ (t,z,a) \in E'^C \big| \mathcal{T}_a v(t,z) = \mathcal{T}v(t,z) \Big\}$ is borelian. Applying the measurable selection theorem yields to the existence of the maximizer. (see [BR11] p.352) □

Lemma 3.3.2. *The following holds:*

$$\sup_{\alpha\in\mathcal{A}} \mathbb{E}^\alpha_{t,z} \left[\sum_{k=n}^{\infty} |r(T_k,Z_k)| \right] \leq \frac{\alpha_b^n}{1-\alpha_b} b(t,z),$$

and in particular, we have:

$$\lim_{n\to\infty} \sup_{\alpha\in\mathcal{A}} \mathbb{E}^\alpha_{t,z} \left[\sum_{k=n}^{\infty} \big|r\big(t_k,Z_k\big)\big| \right] = 0.$$

Proof. By conditioning we get $\mathbb{E}^\alpha_{t,z}\Big[\big|r(T_k,Z_k)\big|\Big] \leq c_g {\alpha_b}^k b(t,z)$ for $k \in \mathbb{N}$, and for all $\alpha \in \mathbb{A}$. It remains to sum this inequality to complete the proof of Lemma 3.3.2. □

We can now prove Theorem 3.3.1.

Proof. We divided the proof of Theorem 3.3.1 into four steps.
Step 1: Inequality (3.3.12) and Proposition 3.3.3 imply that $\mathcal{T}$ is a stable and contracting operator defined on the Banach space $\mathcal{M}$. Banach's fixed point theorem states that $\mathcal{T}$ admits a fixed point, i.e. there exists a function $v \in \mathbb{M}$ such that $v = \mathcal{T}v$, and moreover we have $v = \lim_{n\to\infty} \mathcal{T}^n 0$. Notice that $\mathcal{T}^N 0$ coincides with v_0 defined recursively by the following Bellman equation:

$$\left\{ \begin{array}{rcl} v_N & = & 0 \\ v_n & = & \mathcal{T}v_{n+1} \text{ for } n = N-1, ..., 0. \end{array} \right. \tag{3.3.13}$$

The solution of the Bellman equation is always larger than the value function of the MDP associated (see e.g. Theorem 2.3.7 p.22 in [BR11]). Then we have: $\mathcal{T}^n 0 \geq \sup_{(f_k)} \mathbb{E}_n^{(f_k)}\left[\sum_{k=0}^{n-1} r(t_k, X_k)\right] =: J_n$, where J_n is the value function of the MDP with finite horizon n and terminal reward 0, associated to (3.3.13). Moreover, by Lemma 7.1.4 p.197 in [BR11], we know that $\left(J_n\right)_n$ converges as $n \to \infty$ to a limit that we denote by J. Passing at the limit in the previous inequality we get: $\lim_{n\to\infty} \mathcal{T}^n 0 \geq J$, i.e.

$$v \geq J. \tag{3.3.14}$$

Step 2: Let us fix a strategy $\alpha \in \mathbb{A}$, and take $n \in \mathbb{N}$. We denote $J_n(\alpha) := \mathbb{E}_0^{(\alpha_k)}\left[\sum_{k=0}^{n-1} r(t_k, X_k)\right]$, the reward functional associated to the control α on the discrete finite time horizon $\{0, \ldots, n\}$. By definition, we have $J_n(\alpha) \leq J_n$. We get by letting $n \to \infty$: $\lim_{n\to+\infty} J_n(\alpha) =: J_\infty(\alpha) \leq J$. Taking the supremum over all the admissible strategies α finally leads to:

$$V_\infty \leq J. \tag{3.3.15}$$

Step 3: Let us denote by f a maximizer of $\mathcal{T}$ associated to v, which exists, as stated in Proposition 3.3.4. v is the fixed point of $\mathcal{T}$ so that $v = \mathcal{T}_f^n(v)$, for $n \in \mathbb{N}$. Moreover $v \leq \delta$ where $\delta := \sup_{\alpha\in\mathcal{A}} \mathbb{E}\Big[\sum_{k=0}^{\infty} r^+(Z_k, \alpha_k)\Big]$, so that $\mathcal{T}_f^n(v) \leq \mathcal{T}_f^n 0 + \mathcal{T}_o^n \delta$, where $\mathcal{T}_o^n \delta = \sup_{\alpha} \mathbb{E}_n^{\alpha}\Big[\sum_{k=n}^{\infty} r^+(t_k, Z_k)\Big]$. Lemma 3.3.2 implies that $\mathcal{T}_o^n \delta \to 0$ as $n \to \infty$. Hence, we get:

$$v \leq J_f. \tag{3.3.16}$$

Step 4: Conclusion. Since it holds

$$J_f \leq V_\infty, \tag{3.3.17}$$

we get by combining (3.3.14), (3.3.15), (3.3.16) and (3.3.17):

$$V_\infty \leq J \leq v \leq J_f \leq V_\infty. \tag{3.3.18}$$

All the inequalities in (3.3.18) are then equalities, which completes the proof of Theorem 3.3.1. □

3.4 Numerical Algorithm

In this section, we first introduce an algorithm to numerically solve a general class of discrete-time control problem with finite horizon, and then apply it on the trading problem (3.3.1).

3.4.1 Framework

Let us consider a general discrete-time stochastic control problem over a finite horizon $N \in \mathbb{N} \setminus \{0\}$. The dynamics of the controlled state process $Z^\alpha = (Z_n^\alpha)_n$ valued in $\mathbb{R}^d$ is given by

$$Z_{n+1}^\alpha = F(Z_n^\alpha, \alpha_n, \varepsilon_{n+1}), \quad n = 0, \ldots, N-1, \; Z_0^\alpha = z \in \mathbb{R}^d,$$

with $(\varepsilon_n)_n$ is a sequence of i.i.d. random variables valued in some Borel space $(E, \mathcal{B}(E))$, and defined on some probability space $(\Omega, \mathbb{F}, \mathbb{P})$ equipped with the filtration $\mathbb{F} = (\mathcal{F}_n)_n$ generated by the noise $(\varepsilon_n)_n$ ($\mathcal{F}_0$ is the trivial σ-algebra), the control $\alpha = (\alpha_n)_n$ is an $\mathbb{F}$-adapted process valued in $A \subset \mathbb{R}^q$, and F is a measurable function from $\mathbb{R}^d \times \mathbb{R}^q \times E$ into $\mathbb{R}^d$.

Given a running cost function f defined on $\mathbb{R}^d \times \mathbb{R}^q$, a terminal cost function g defined on $\mathbb{R}^d$, the cost functional associated to a control process α is

$$J(\alpha) \quad = \quad \mathbb{E}\left[\sum_{n=0}^{N-1} f(Z_n^\alpha, \alpha_n) + g(Z_N^\alpha)\right].$$

The set $\mathbb{A}$ of admissible control is the set of control processes α satisfying some integrability conditions ensuring that the cost functional $J(\alpha)$ is well-defined and finite. The control problem, also called Markov decision process (MDP), is formulated as

$$V_0(x_0) \quad := \quad \sup_{\alpha \in \mathcal{A}} J(\alpha),$$

and the goal is to find an optimal control $\alpha^* \in \mathcal{A}$, i.e., attaining the optimal value: $V_0(z) = J(\alpha^*)$. Notice that problem (3.4.1)-(3.4.2) may also be viewed as the time discretization of a continuous time stochastic control problem, in which case, F is typically the Euler scheme for a controlled diffusion process.

Problem (3.4.2) is tackled by the dynamic programming approach. For $n = N, \ldots, 0$, the value function V_n at time n is characterized as solution of the following backward (Bellman) equation:

$$\left\{ \begin{array}{rcl} V_N(z) & = & g(z) \\ V_n(z) & = & \sup\limits_{a \in A} \left\{ f(z,a) + \mathbb{E}_{n,z}^a \left[V_{n+1}(Z_{n+1}) \right] \right\}, \quad z \in \mathbb{R}^d, \end{array} \right. \tag{3.4.3}$$

Moreover, when the supremum is attained in the DP formula at any time n by $a_n^*(z)$, we get an optimal control in feedback form given by: $\alpha^* = (a_n^*(Z_n^*))_n$ where $Z^* = Z^{\alpha^*}$ is the Markov process defined by

$$Z_{n+1}^* \quad = \quad F(Z_n^*, a_n^*(Z_n^*), \varepsilon_{n+1}), \quad n = 0, \ldots, N-1, \quad Z_0^* = z.$$

There are two usual ways that have been studied in the literature, to solve numerically (3.4.3): some methods make use of quantization to discretize to state space and approximate the conditional expectations by cubature methods; another way is to rely on MC regress-now or Later methods to regress the value functions V_{n+1} at time n for $n = 0, \ldots, N-1$ on basis functions or neural networks. See e.g. [KLP14] for the regress-now and [BP17] for the regress-Later methods for algorithms using basis functions, and e.g. [HPBL18] for regression on neural networks based on regress-now or regress-later techniques.

3.4.2 Presentation and rate of convergence of the Qknn algorithm

In this section, we present an algorithm based on k-nn estimates for local non-parametric regression of the value function, and optimal quantization to quantize the exogenous noise, in order to numerically solve (3.4.3).

Let us first introduce some ingredients of the quantization approximation:

- We denote by $\hat{\varepsilon}$ a K-quantizer of the E-valued random variable $\varepsilon_{n+1} \sim \varepsilon_1$, that is a discrete random variable on a grid $\Gamma = \{e_1, \ldots, e_K\} \subset E^K$ defined by

$$\hat{\varepsilon} = \mathrm{Proj}_\Gamma(\varepsilon_1) \quad := \quad \sum_{\ell=1}^{K} e_l 1_{\varepsilon_1 \in C_i(\Gamma)},$$

where $C_1(\Gamma)$, ..., $C_K(\Gamma)$ are Voronoi tesselations of Γ, i.e., Borel partitions of the Euclidian space $(E, |.|)$ satisfying

$$C_\ell(\Gamma) \subset \left\{ e \in E : |e - e_\ell| \;=\; \min_{j=1,\dots,K} |e - e_j| \right\}.$$

The discrete law of $\hat{\varepsilon}$ is then characterized by

$$\hat{p}_\ell := \mathbb{P}[\hat{\varepsilon} = e_\ell] \;=\; \mathbb{P}[\varepsilon_1 \in C_\ell(\Gamma)], \quad \ell = 1, \dots, K.$$

The grid points (e_ℓ) which minimize the L^2-quantization error $\|\varepsilon_1 - \hat{\varepsilon}\|_2$ lead to the so-called optimal L-quantizer, and can be obtained by a stochastic gradient descent method, known as Kohonen algorithm or competitive learning vector quantization (CLVQ) algorithm, which also provides as a byproduct an estimation of the associated weights $(\hat{p}_\ell)$. We refer to [PPP04c] for a description of the algorithm, and mention that for the normal distribution, the optimal grids and the weights of the Voronoi tesselations are precomputed on the website http://www.quantize.maths-fi.com

- Recalling the dynamics (3.4.1), the conditional expectation operator is equal to

$$P^{\hat{a}_n^M(z)} W(x) \;=\; \mathbb{E}\big[W(Z_{n+1}^{\hat{a}_n^M}) | Z_n = x\big] = \mathbb{E}\big[W(F(z, \hat{a}_n^M(z), \varepsilon_1))\big], \;\; z \in,$$

that we shall approximate analytically by quantization via:

$$\widehat{P}^{\hat{a}_n^M(z)} W(z) := \mathbb{E}\big[W(F(z, \hat{a}_n^M(z), \hat{\varepsilon}))\big] \;=\; \sum_{\ell=1}^{K} \hat{p}_\ell W(F(z, \hat{a}_n^M(z), e_\ell)).$$

Let us secondly introduce the notion of training distribution that will be used to build the estimators of value functions at time n, for $n = 0, \dots, N-1$. Let us consider a measure μ on the state space E. We refer to it in the sequel as the training measure. Let us take a large integer M, and for $n = 0, \dots, N$, introduce $\Gamma_n = \left\{ Z_1^{(1)}, \dots, Z_n^{(M)} \right\}$, where $\left(Z_n^{(m)} \right)_{m=1}^{M}$ is a i.i.d. sequence of r.v. following law μ. Γ_n should be seen as a training sampling to estimate the value function V_n at time n.

The proposed algorithm reads as:

$$\left\{ \begin{array}{rcll} \hat{V}_N^Q(z) & = & g(z), & \text{for } z \in \Gamma_N, \\ \hat{Q}_n(z,a) & = & \sum_{\ell=1}^K p_\ell \left[f(z,a) + \hat{V}_{n+1}^Q \left(\mathrm{Proj}_{n+1} \left(F\big(z, e_\ell, a\big) \right) \right) \right], & \\ \hat{V}_n^Q(z) & = & \sup\limits_{a \in A} \hat{Q}_n(z,a), & \text{for } z \in \Gamma_n, \; n = 0, \dots, N-1. \end{array} \right. \tag{3.4.4}$$

where, for $n = 0, \dots, N$, $\mathrm{Proj}_n(z)$ stands for the closest neighbor of $z \in E$ in the grid Γ_n, i.e. the operator $z \mapsto \mathrm{Proj}_n(z)$ is actually the euclidean projection on the grid Γ_n.

Remark 3.4.1. *We could have generalized the operator* Proj_n *by considering* $z \in E \mapsto \hat{z} = \frac{1}{k} \sum_{j=1}^k w_j Z_n^{(j)}$, *with the weight* w_j *such as*

$$w_j(z) = \frac{\left| z - Z_n^{(j)} \right|}{\sum_{i=1}^k \left| z - Z_n^{(i)} \right|},$$

and where $Z_n^{(j)}$ *stands for the* j^{th}*nearest neighbors of* z *in* Γ_n, *for* $j = 1, \dots, k$. *This generalization brings continuity to the estimates.*

In the sequel, we refer to (3.4.4) as the Qknn algorithm.

We shall make the following assumption on the transition probability of $(Z_n)_{0\leq n\leq N}$, to guarantee the convergence of the Qknn algorithm.

(Htrans) Assume that the transition probability $\mathbb{P}(Z_{n+1}\in A\big|Z_n=z,a)$ conditioned by $Z_n=z$ when control a is followed at time n admits a density r w.r.t. the training measure μ, which is uniformly bounded and lipschitz w.r.t. the state variable z, i.e. there exists $\|r\|_\infty>0$ such that for all $z\in E$ and control u taken at time n:

$$|r(y;n,x,a)|\leq\|r\|_\infty \quad \text{and} \quad |r(y;n,x,a)-r(y;n,x',a)|\leq[r]_L|x-x'|$$

and r is defined as follows:

$$\mathbb{P}(Z_{n+1}\in O\big|Z_n=z,u)=\int_O r(y;n,x,a)d\mu(y).$$

and where we denoted by $[r]_L$ the Lipschitz constant of r w.r.t. x.

Denote by $\mathrm{Supp}(\mu)$ the support of μ. We shall assume smoothness conditions on μ and F to provide a bound on the projection error.

(Hμ) We assume $\mathrm{Supp}(\mu)$ to be bounded, and denote by $\|\mu\|_\infty$ the smallest real such that $\mathrm{Supp}(\mu)\subset B\left(0,\|\mu\|_\infty\right)$. Moreover, we assume $x\in E\mapsto\mu\big(B(x,\eta)\big)$ to be Lipschitz, uniformly w.r.t. η, and we denote by $[\mu]_L$ its Lipschitz constant.

(HF) For $x\in E$ and $a\in A$, assume F to be $\mathbb{L}^1$-Lipschitz w.r.t. the noise component ε, i.e., there exists $[F]_L>0$ such that for all $x\in E$ and $a\in A$, for all r.v. ε and ε', we have:

$$\mathbb{E}\left[|F(x,a,\varepsilon)-F(x,a,\varepsilon')|\right]\leq[F]_L\mathbb{E}\left[|\varepsilon-\varepsilon'|\right]$$

We now state the main result of this section whose proof is postponed in Appendix 3.C.

Theorem 3.4.1. *Take $K=M^{2+d}$ points for the optimal quantization of the exogenous noise ε_n, $n=1,\dots,N$. There exist constants $[\hat{V}_n^Q]_L>0$, that only depends on the Lipschitz coefficients of f, g and F, such that, under **(Htrans)**, it holds for $n=0,...,N-1$, as $M\to+\infty$:*

$$\|\hat{V}_n^Q(X_n)-V_n(X_n)\|_2 \;\leq\; \sum_{k=n+1}^{N}\|r\|_\infty^{N-k}\left[\hat{V}_k^Q\right]_L\left(\varepsilon_k^{proj}+[F]_L\varepsilon_k^Q\right)+\mathcal{O}\left(\frac{1}{M^{1/d}}\right), \tag{3.4.5}$$

where $\varepsilon_k^Q:=\|\hat{\varepsilon}_k-\varepsilon_k\|_2$ stands for the quantization error, and

$$\varepsilon_n^{proj}:=\sup_{a\in A}\|\mathrm{Proj}_{n+1}\left(F(X_n,a,\hat{\varepsilon}_n)\right)-F(X_n,a,\hat{\varepsilon}_n)\|_2$$

stands for the projection error, when decision a is taken at time n.

Remark 3.4.2. *The constants $[\hat{V}_n^Q]_L>0$ are defined in (3.C.8).*

From Theorem 3.4.1, we can deduce consistency and provide a rate of convergence for the estimator $\hat{V}_n^Q$, $n=0,\dots,N-1$, under some rather tough yet usual compactness conditions on the state space.

Corollary 3.4.1. *Under **(Hμ)** and **(HF)**, the Qknn-estimator $\hat{V}_n^Q$ is consistent for $n=0,\dots,N-1$, when taking M^{d+1} points for the quantization; and moreover, we have for $n=0,...,N-1$, as $M\to+\infty$:*

$$\|\hat{V}_n^Q(X_n)-V_n(X_n)\|_2\leq\mathcal{O}\left(\frac{1}{M^{1/d}}\right).$$

Proof. We postpone the proof of Theorem 3.4.1 to Appendix 3.C. □

3.4.3 Qknn agorithm applied to the order book control problem (3.3.1)

We recall the expression of the controlled order book, as described in section 3:

$$Z_t = \big(X_t, Y_t, \underline{a}_t, \underline{b}_t, \underline{na}_t, \underline{nb}_t, pa_t, pb_t, \underline{ra}_t, \underline{rb}_t\big).$$

In section 3.3.3, we proved that the value function V is characterized as the unique solution of the Bellman equation (3.3.10). In this section, some implementation details on the Qknn algorithm are presented in order to numerically solve the market-making problem.

Training set design

Inspired by [FPS18], we use product-quantization method and randomization techniques to build the training set Γ_n on which we project (T_n, Z_n) that lies on $[0,T] \times E$, where T_n and Z_n stands for the n^{th} jump of Z and the state of Z at time t_n, i.e. $Z_n = Z_{T_n}$, for $n \geq 0$. This basic idea of Control Randomization consists in replacing in the dynamics of Z the endogenous control by an exogenous control $(I_{T_n})_{n\geq 0}$, as introduced in [KLP14]. In order to alleviate the notations, we denote by I_n the control taken at time T_n, for $n \geq 0$.

Initialization. Set: $\Gamma_0^E = \{z\}$ and $\Gamma_0^T = \{0\}$.

Randomize the control, using e.g. uniform distribution on A at each time step, and then simulate D randomized processes to generate $(T_n^k, Z_n^k)_{n=0,k=1}^{N,D}$.

For all $n = 1, \dots, N$, set $\Gamma_n^T = \{T_n^k, 1 \leq k \leq D\}$, which stands for the grid associated to the quantization of the n^{th} jump time T_n, and set $\Gamma_n^E = \{Z_n^k, 1 \leq k \leq D\}$ which stands for the grid associated to the quantization of the state Z_n of Z at time T_n.

Remark 3.4.3. *The way we chose our training sets is often referred to as an* exploration strategy *in the reinforcement learning literature. Of course, if one has ideas or good guess of where to optimally drive the controlled process, she shouldn't follow an exploration-type strategy to build the training set, but should rather use the guess to build it, which is referred to as the* exploitation strategy *in the reinforcement learning and the stochastic bandits literature. We refer to [Bal+19] for several other applications of the* exploration strategy *to build training sets.*

Let F and G be the Borelian functions such that $Z_n = F\big(Z_{n-1}, d_n, I_n\big)$ and $T_n = G\big(T_{n-1}, \epsilon_n, I_n\big)$, where $\epsilon_n \sim \mathcal{E}(1)$ stands for the temporal noise, and d_n is the state noise, for $n \geq 0$.

Let us fix $N \geq 1$ and consider $\big(\widehat{T}_n, \widehat{Z}_n\big)_{n=0}^N$, the dimension-wise projection of $\big(T_n, Z_n\big)_{n=0}^N$ on the grids $\Gamma_n^T \times \Gamma_n^E$, $n = 0, \dots, N$, i.e. $\widehat{T}_0 = 0$, $\widehat{Z}_0 = z$, and

$$\begin{cases} \widehat{T}_n = \text{Proj}\Big(G\big(\widehat{T}_{n-1}, \epsilon_n, I_n\big), \Gamma_n^T\Big), \\ \widehat{Z}_n = \text{Proj}\Big(F\big(\widehat{Z}_{n-1}, d_n, I_n\big), \Gamma_n^E\Big), \end{cases} \qquad \text{for } n = 1, \dots, N.$$

$\big(\widehat{T}_n, \widehat{Z}_n, I_n\big)_{n\in\{0,N\}}$ is a Markov chain, and its probability transition matrix at time $n = 1, ..., N$ reads:

$$\hat{p}_k^{ij}(a) = \mathbb{P}\Big[\hat{t}_k = t_k^j, \widehat{Z}_k = z_k^j \Big| \hat{t}_{k-1} = t_{k-1}^i, \widehat{Z}_{k-1} = z_{k-1}^i, I_k = a\Big] = \frac{\hat{\beta}_k^{ij}}{\hat{p}_{k-1}^i}, \quad i = 1, ..., N_{k-1}, j = 1, ..., N_k, a \in \mathcal{A}$$

where:

$$\hat{p}_{k-1}^i = \mathbb{P}\big[\hat{t}_{k-1} = t_{k-1}^i, \widehat{Z}_{k-1} = z_{k-1}'^i\big] = \begin{cases} \mathbb{P}\big[F\big(\hat{t}_{k-2}, \widehat{Z}_{k-2}, \epsilon_{k-1}, d_{k-1}\big) \in C_i(\Gamma_{k-1} \times \Gamma_{k-1}^E)\big] & \text{if} \quad k \geq 2 \\ 1 & \text{if} \quad k = 1 \end{cases}$$

$$\begin{aligned} \hat{\beta}_k^{ij} &= \mathbb{P}\Big[\hat{t}_{k-1} = t_{k-1}^i, \widehat{Z}_{k-1} = z_{k-1}^i, \hat{t}_k = t_k^j, \widehat{Z}_k = z_k'^j\Big] \\ &= \begin{cases} \mathbb{P}\Big[F_k\big(\hat{t}_{k-2}, \widehat{Z}_{k-2}, \epsilon_{k-1}, d_{k-1}\big) \in C_i(\Gamma_{k-1} \times \Gamma_{k-1}^E); F_k\big(\hat{t}_{k-1}, \widehat{Z}_{k-1}, \epsilon_k, d_k\big) \in C_i(\Gamma_k \times \Gamma_k^E)\Big] & \text{if } k \geq 2 \\ 1 & \text{if } k = 1 \end{cases} \end{aligned}$$

and where, for all $i, 0 \leq i \leq D$, for all $k \in \mathbb{N}$, we denoted by $C_i(\Gamma_k \times \Gamma_k^E)$ the Voronoï cell associated to the point (T_k^i, z_k^i).

Define then $\big(\widehat{T}_n^Q, \widehat{Z}_n^Q\big)_{n=0}^N$ as temporal noise-quantized version of $\big(\widehat{T}_n, \widehat{Z}_n, I_n\big)_{n=0}^N$. Note that we do not need to quantize the spacial noise since this noise already takes a finite number of states. Let $\hat{\varepsilon}_n$ be the quantized process associated to ϵ_n. The process $\big(\widehat{T}_n^Q, \widehat{Z}_n^Q\big)_{n=0}^N$ is then defined as follows: $\widehat{Z}_0^Q = z$, $\widehat{T}_0^Q = 0$ and $\forall 1 \leq n \leq N$:

$$\begin{cases} \widehat{T}_n^Q = \mathrm{Proj}\Big(G\big(\hat{t}_{n-1}, \hat{\varepsilon}_n, I_n\big), \Gamma_n^T\Big), \\ \widehat{Z}_n^Q = \mathrm{Proj}\Big(F\big(\widehat{Z}_{n-1}, d_n, I_n\big), \Gamma_n^E\Big). \end{cases}$$

Denote by $\left(\widehat{V}_n^{Q,(N,D)}\right)_{n=0}^N$ the solution of the Bellman equation associated to $\left(\widehat{T}_n^Q, \widehat{Z}_n^Q\right)_{n=0}^N$:

$$(\widehat{B}_{N,D}^Q): \begin{cases} \widehat{V}_N^{Q,(N,D)} & = & 0 \\ \widehat{V}_n^{Q,(N,D)}(t,z) & = & r(t,z,a) + \sup\limits_{a \in A} \left\{ \mathbb{E}_{t,z}^a \left[\widehat{V}_{n+1}^{Q,(N,D)}\big(\widehat{T}_{n+1}^Q, \widehat{Z}_{n+1}^Q\big) \right] \right\}, \text{ for } n = 0, \ldots, N, \end{cases}$$

where $\mathbb{E}_{t,z}^a[.]$ stands for the expectation conditioned by the events $\widehat{T}_n^Q = t, \widehat{Z}_n^Q = z$ and when decision $I_n = a$ is taken at time t.

We wrote the pseudo-code of the Qknn algorithm to compute $(\widehat{B}_{N,D}^Q)$ in Algorithm 3.1.

We discuss in Remark 3.4.4 the reasons why we can apply Theorem 3.4.1.

Remark 3.4.4. *When the number of jumps of the LOB $N \geq 1$ is fixed, the set of all the states that can take the controlled order book by jumping less than N times, denoted by $\mathcal{K}$ in the sequel, is finite. Hence, the reward function r, defined in (3.3.5), is bounded and Lipschitz on $\mathcal{K}$.*

The following proposition states that $\widehat{V}_n^{Q,(N,D)}$, built from the combination of time-discretization, k-nearest neighbors and optimal quantization methods, is a consistent estimator of the value function at time T_n, for $n = 0, \ldots, N-1$. It provides a rate of convergence for the Qknn-estimations of the value functions.

Proposition 3.4.1. *The estimators of the value functions provided by Qknn algorithm are consistent. Moreover, it holds as $M \to +\infty$:*

$$\left\| \widehat{V}_n^{Q,(N,M)}\left(\widehat{T}_n, \widehat{Z}_n\right) - V_n\big(T_n, Z_n\big) \right\|_{M,2} = \mathcal{O}\left(\alpha^N + \frac{1}{M^{2/d}}\right), \quad \text{for } n = 0, \ldots, N-1,$$

where we denote by $\|.\|_{M,2}$ the $\mathbb{L}^2(\mu)$ norm conditioned by the training sets that have been used to build the estimator $\widehat{V}_{n+1}^{Q,(N,M)}$.

Proof. Splitting the error of time cutting and quantization, we get:

$$\begin{aligned} \|V_n\big(T_n, Z_n\big) - \widehat{V}_n^{(N,M)}\big(\widehat{T}_n, \widehat{Z}_n\big)\|_{M,2} \leq \|V_n\big(T_n, Z_n\big) - V_n^{(N)}\big(T_n, Z_n\big)\|_{M,2} \\ + \|V_n^{(N)}\big(T_n, Z_n\big) - \widehat{V}_n^{(N,M)}\big(\widehat{T}_n, \widehat{Z}_n\big)\|_{M,2}. \end{aligned} \tag{3.4.8}$$

Step 1: Applying Lemma 3.3.2, we get the following bound on the first term in the r.h.s. of (3.4.8):

$$\|V_n\big(T_n, Z_n\big) - V_n^{(N)}\big(T_n, Z_n\big)\|_{M,2} \leq \frac{\alpha^N}{1-\alpha} \|b\|_\infty, \tag{3.4.9}$$

Algorithm 3.1 Generic Qknn Algorithm

Inputs:
– N: number of time steps
– z: state in E at time $T_0 = 0$
– $\Gamma^\varepsilon = \{e_1, \dots, e_L\}$ and $(p_\ell)_{\ell=1}^L$: the grid and the weights for the optimal quantization of $(\varepsilon_n)_{n=1}^N$.
– Γ_n and Γ_n^E the grids for the projection of respectively the time and the state components at time n, for $n = 0, \dots, N$.

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

2: Compute the approximated $Qknn$-value at time n:

$$\begin{aligned}\hat{Q}_n(z,a) &= r\left(T_n, z, a\right) \\ &\quad + \sum_{\ell=1}^{L} p_\ell \widehat{V}_{n+1}^Q \left(\text{Proj}\left(G(z, e_\ell, a), \Gamma_{n+1}^T\right), \text{Proj}\left(F(z, e_\ell, a), \Gamma_{n+1}^E\right)\right), \\ &\qquad\qquad \text{for } (z,a) \in \Gamma_n \times A_z;\end{aligned}$$

3: Compute the optimal control at time n

$$\hat{A}_n(z) \in \underset{a \in A_z}{\text{argmin}}\ \hat{Q}_n(z,a), \quad \text{for } z \in \Gamma_n,$$

where the argmin is easy to compute since A_z is finite for all $z \in E$;

4: Estimate analytically by quantization the value function:

$$\widehat{V}_n^Q(z) = \hat{Q}_n\left(z, \hat{A}_n(z)\right), \qquad \forall z \in \Gamma_n;$$

5: **end for**

Output:
– $(\widehat{V}_0^Q)$: Estimate of $V(0, z)$;

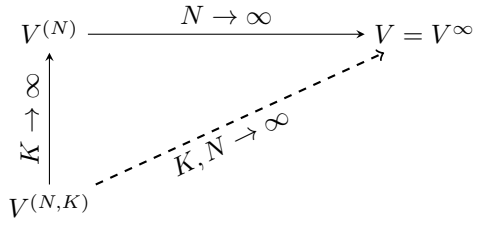


Figure 3.3 – Numerical resolution of the algorithmic control problem. We first bound the number of events and then quantize the state space.

where $\|b\|_{\infty}$ stands for the supremum of b over $[0,T] \times E$.

Step 2: Note that the assumptions of Theorem 3.4.1 are met as noticed in Remark 3.4.4, so that the latter provides the following bound for the second term in the r.h.s. of (3.4.8):

$$\left\| V_n^{(N)}(T_n, Z_n) - \widehat{V}_n^{Q,(N,M)}(\widehat{T}_n, \widehat{Z}_n) \right\|_{M,2} \underset{M\to\infty}{=} \mathcal{O}\left(\frac{1}{M^{2/d}}\right). \tag{3.4.10}$$

It remains to plug (3.4.9) and (3.4.10) into (3.4.8) to complete the proof of Proposition 3.4.1. □

We provide a diagram in figure 3.3 to summarize the two main steps in the estimation of the value function of the market-making problem defined in (3.3.1).

3.4.4 Numerical results

In this section, we propose several settings to test the efficiency of Qknn on simulated order books. We take no running reward, i.e. $f = 0$, and take the wealth of the market maker as terminal reward, i.e. $g(z) = x$. The intensities are taken constant in some tests, and state dependent on other tests. The values of the intensities are similar to the ones in [HLR15]. Although the intensities are assumed uncontrolled in section 3.3 for predictability reasons, the latter are controlled processes in this section, i.e. the intensities of the order arrivals depends on the orders in the order book from all the participant plus the ones of the market maker. The optimal trading strategies have been computed among two different classes of strategies: in section 3.4.4.1, we tested the algorithm to approximate the optimal strategy among those where the market maker is only allowed to place orders only at the best bid and the best ask. The dynamics of the controlled order book for such a class of controls are available in Section 3.B in the Appendix. In Section 3.4.4.2, we computed the optimal trading strategy among the class of the strategies where the market maker allows herself to place orders on the two best limits on each side of the order book. Note that the second class of controls is more general than the first one.

The search of the k nearest neighbors, that arise when estimating the conditional expectations using the Qknn algorithm, is very time-consuming; especially in the considered market-making problem which is of dimension more than 10. The efficiency of Qknn then highly depends on the algorithm used to find the k nearest neighbors in high-dimension. Qknn algorithm has been implemented using the Fast Library for Approximate Nearest Neighbors algorithm (FLANN), introduced in [ML09] and already available in libraries in C++, Python, Julia and many other languages. This algorithm is

based on tree methods. Note that recent algorithms based on graph also proved to perform well, and can also be used.

3.4.4.1 Case 1: The market maker only place orders at the best ask and best bid.

Denote by A1lim the class of controls where the placements of orders in allowed on the best ask and best bid exclusively. We implement the Qknn algorithm to compute the optimal strategy among those in A1lim. We then compared the optimal strategy with a naive strategy which consists in always placing one order at the best bid and one order at the best ask. The naive strategy is called 11 in the plots, and can be seen as a benchmark. The naive strategy is a good benchmark when the model for the intensities of order arrivals is symmetrical, i.e. the intensities for the bid and the ask sides are the same. Indeed, in this case, the market maker can expect to earn the spread in average.

Numerical results:

In Figure 3.4, we take constant intensities to model the limit and market orders arrivals, and linear intensity to model the cancel orders. In this setting, as we can see in the figure, the strategy computed using Qknn algorithm performs as well as the naive strategy. Note that, obviously, the market maker has to take enough points for the state quantization in order for Qknn algorithm to perform well. In figure 3.5, we plotted the P&L of the market maker when the latter compute the optimal strategy using only 6000 points for the state space discretization, and for such a low number of points for the grid, Qknn algorithm performs poorly.

In Figure 3.6, we plotted the empirical histogram of the P&L of the market maker using the Qknn-estimated optimal strategy, computed with grids of size $N = 1000, 10000, 100000, 1000000$ for the state space discretization; and the empirical histogram of the P&L of the market maker using the naive strategy. One can see that the larger the size of the grids are, the better the Qknn-estimation of the optimal strategy is.

We plot in Figure 3.7 the results of simulations run taking a short terminal time T=1, and intensities that depend on the size of the queues. In this setting, notice that the naive strategy does not perform well anymore, but the Qknn algorithm still does well, when the market maker takes enough points for state space discretization.

In figure 3.7, we plot the P&L of the market maker following the Qknn strategy and the naive strategy, and we took the same parameters as in figure 3.6 to run the simulations expect from the terminal time that we set as T=10. As expected[4], the expected wealth of the market maker is larger when terminal time is larger and when the latter follows the Qknn-estimated optimal strategy. Note that the expectation of the latter remains the same when she follows the naive strategy.

3.4.4.2 Case 2: the market maker place orders on the first two limits of the Orders Book

We extend the class of admissible controls to the ones where the market maker places order on the first two limits on the bid and ask sides of the order book. Denote by $A2lim$ the latter. We run simulations to test the Qknn algorithm on A2lim. In figure 3.8 and figure 3.9, we plot the empirical distributions of the P&L when the market maker follows the three different strategies:

- Qknn-estimated optimal strategy among those in A2lim (PLOpt2lim).
- Qknn-estimated optimal strategy among those in A1lim (PLOpt1lim).
- naive strategy, i.e. always place orders on the best bid and best ask queues (PL11).

[4]The value function for the market-making problem is by definition a non-decreasing function w.r.t. the time component

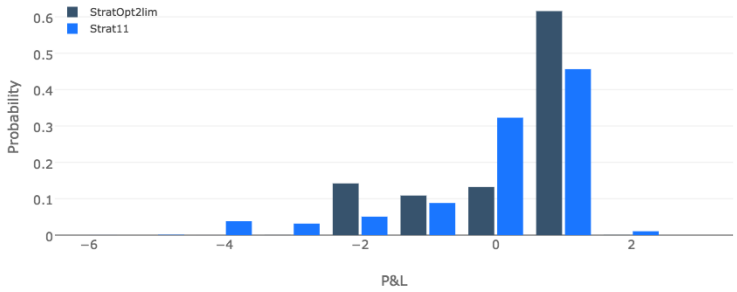


Figure 3.4 – Symmetrical intensities, Size of the grids: 100000. Short terminal time: T=1. Notice that the Qknn strategy reduces the variance of the P&L, but the expected wealth when following Qknn strategy (StratOpt2lim) is the same as the one following the naive strategy (Strat11).

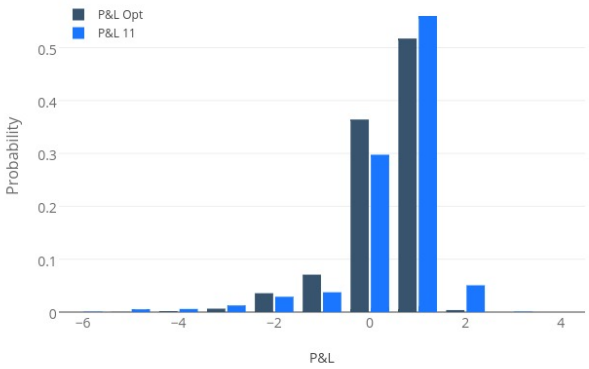


Figure 3.5 – Symmetrical and constant intensities. Size of the grids: 6000. The computed optimal strategy is less efficient than the naive strategy.

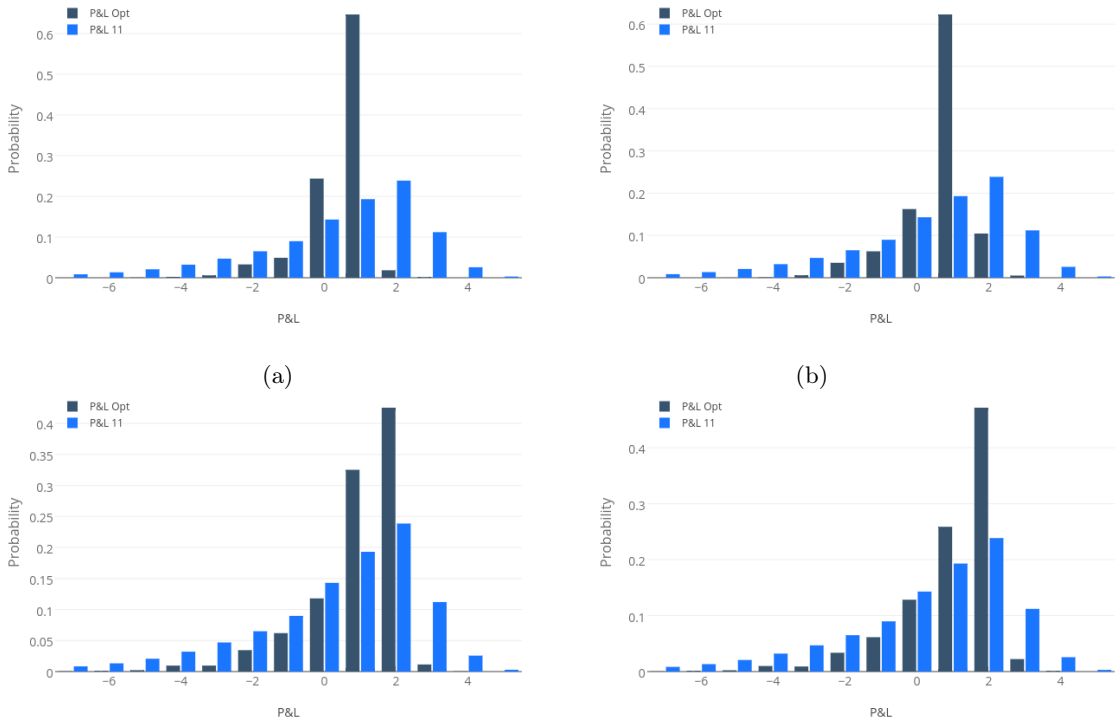


Figure 3.6 – P&L when the intensities λ^M , λ_i^L and λ_i^C depend on the state of the order book. Figure 3.6a shows the P&L of the market maker when following the Qknn-estimated optimal strategy computed with 1000 points for the state space discretization. Figure 3.6b shows the P&L when following the Qknn-estimated optimal strategy computed with 9000 points for the state space discretization. Figure 3.6c shows the P&L when following the Qknn-estimated optimal strategy computed with 100000 points for the state space discretization. Figure 3.6d shows the P&L when following the Qknn-estimated optimal strategy computed with 1000000 points for the state space discretization. The reader can see that the market maker increases her expected terminal wealth by taking more and more points for the state space discretization. Also, the naive strategy is beaten when the intensities are state dependent.

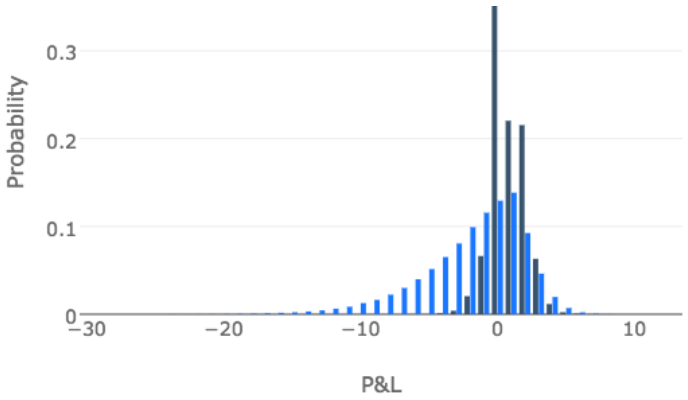


Figure 3.7 – P&L of the market maker following the optimal strategy and following the naive strategy 11. Symmetrical state dependent intensities. Long Terminal Time: T=10. Notice that the Qknn strategy does better than the naive strategy when the intensities are state dependent.

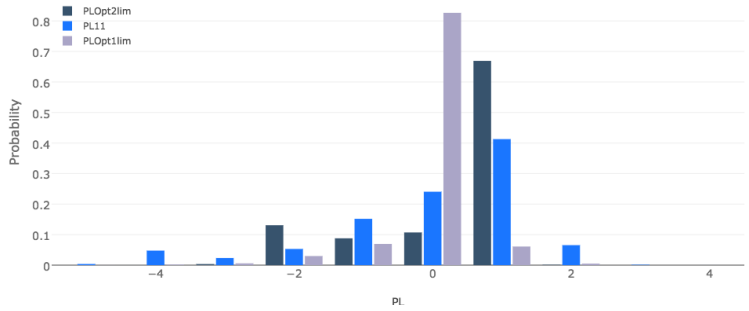


Figure 3.8 – P&L of the market maker who follows optimal strategies and the naive strategy (PL11). Short Terminal Time. asymmetrical intensities for the market order arrivals: the intensity for the buying market order process is taken higher than the one for the selling market order process. The wealth of the market maker is greater when she places orders on the two first limits of each sides of the order book, rather than when she places orders only on the best limits at the bid and ask sides.

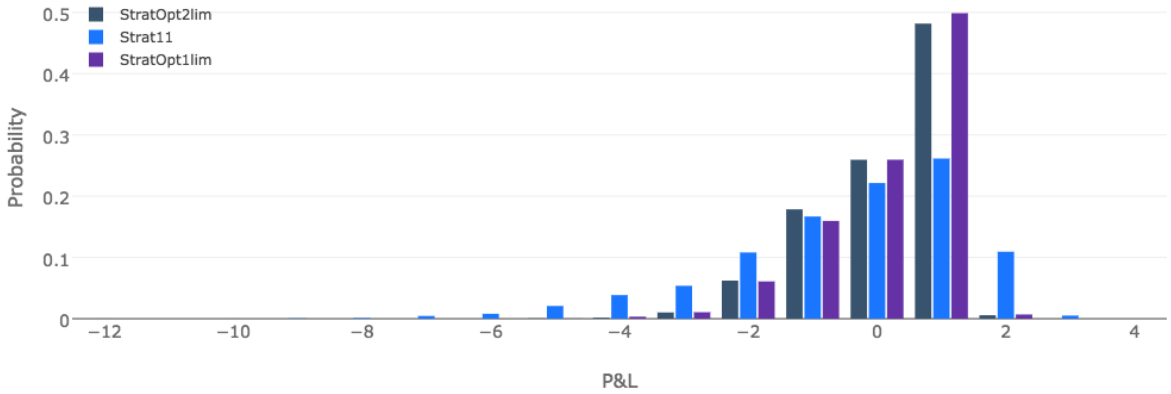


Figure 3.9 – P&L when following the optimal strategy or the naive strategy (PL11). Long Terminal Time. Symmetrical intensities for the arrival of market orders. 400000 points for the quantization. Notice that the Qknn strategy computed on the extended class of controls, i.e. order placements on the two first limits (StratOpt2lim), performs as well as the one computed on the original class of controls, i.e. order placements on the best-bid and best-ask (StratOpt1lim).

Note that the P&L of the market maker is always better when the class of admissible controls is extended, see figure 3.8, but in some models of order books, the extended set of controls doest not improve the P&L, i.e. $\sup_{\alpha \in A2lim} V^\alpha = \sup_{\alpha \in A1lim} V^\alpha$.

3.5 Model extension to Hawkes Processes

We consider in this section a market maker who aims at maximizing a function of her terminal wealth, penalizing her inventory at terminal time T in the case where the orders arrivals are driven by Hawkes processes.

Let us first present the model with Hawkes processes for the LOB.

Model for the LOB:

We assume that the order book receives limit, cancel, and market orders. We denote by L^+ (resp. L^-) the limit order arrivals process the ask (resp. bid) side; by C^+ (resp. C^-) the cancel order on the ask (resp. bid) side; and by M^+ (resp. M^-) the buy (resp. sell) market order arrivals

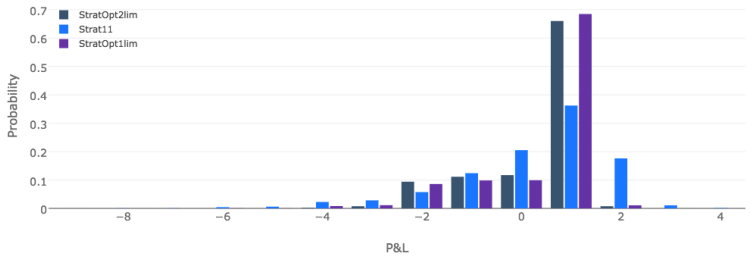


Figure 3.10 – P&L of the market maker who follows the optimal strategy and following the naive strategy 11. Long Terminal Time. Constant and symmetrical intensities for the arrivals of orders. Notice that the strategies computed by Qknn algorithm when taking A2lim performs as well as the one computed on the two best limits of the order book exclusively. Then, in this setting, placing orders only at the best-ask and best-bid seems to be the the optimal strategy.

processes. In this section, the limit orders arrivals are assumed to follow Hawkes processes dynamics, and moreover we assume the kernel to be exponential. The order arrivals are then modeled by a (4K+2)-variate Hawkes process (N_t) with a vector of exogenous intensities λ_0 and exponential kernel ϕ, i.e. $\phi^{ij}(t) = \alpha^{ij}\beta_{ij}e^{\beta_{ij}t}\mathbf{1}_{t\geq 0}$. Note that in the presented model, the following holds:

(Hλ) λ is assumed to be independent of the control.

Denoting by $D = 4K + 2$ the dimension of (N_t), the m^{th} component of the intensity λ of N_t writes, under **(Hλ)**:

$$\lambda_t^m = \lambda_0^m + \sum_{j=1}^{D} \alpha_{mj} \int_0^t e^{-\beta_{mj}(t-s)}\, \mathrm{d}N_s^j, \qquad \text{for } m = 1, \dots, D,$$

or equivalently:

$$\mathrm{d}\lambda_t^m = \sum_{j=1}^{D} \alpha_{mj} \Big[-\beta_{mj}\big(\lambda_t^m - \lambda_0^m\big)dt + \alpha_{mj}\, \mathrm{d}N_t^j \Big], \qquad \text{for } m = 1, \dots, D,$$

with given initial conditions: $\lambda_0^m \in \mathbb{R}_+^*$ for $m = 1, \dots, D$. It is well-known that for this choice of intensity, the couple becomes $(N_t, \lambda_t)_{t\geq 0}$ Markovian. See e.g. Lemma 6 in [Mas98] for a proof of this result.

We can now rewrite the control problem (3.3.1) in the particular case where the order book is driven by Hawkes processes, there is no running reward, i.e. $f = 0$, and where the terminal reward G stands for the terminal wealth of the market maker penalized by her inventory. We then consider the following problem in this section:

$$V(t, \lambda, z) := \sup_{\alpha \in \mathbb{A}} \mathbb{E}_{t,z,\lambda}^{\alpha}\big[G\big(Z_T\big)\big], \tag{3.5.1}$$

where $G(z)$ denotes the wealth of the market maker when the controlled order book is at state z, plus a term of penalization of her inventory; and where $\mathbb{A}$ is the set of the admissible controls, i.e. the predictable decisions taken by the market maker until a terminal time $T > 0$.

We now present the main result of this section.

Theorem 3.5.1. *V is characterized as the unique solution of the following HJB equation:*

$$\begin{cases} f(T,z,\lambda) &= G(z), \text{ for } z \in E \\ 0 &= \dfrac{\partial f}{\partial t}(t,z,\lambda) - \displaystyle\sum_{m=1}^{D}\left[\sum_{j=1}^{D}\beta_{mj}\big(\lambda^m-\lambda_0^m\big)\frac{\partial f}{\partial \lambda^m}(t,z,\lambda)\right. \\ & \qquad \left. +\lambda^m \sup_{a\in A_z}\big[f\big(t,e_m^a(z),\lambda+\alpha_m\big)-f\big(t,z,\lambda\big)\big]\right], \\ & \qquad\qquad \text{for } 0\le t<T, \quad \text{and} \quad (t,z,\lambda)\in \mathbb{R}_+\times E\times \mathbb{R}_+^*. \end{cases} \tag{3.5.2}$$

Moreover, V admits the following representation

$$V(t,z,\lambda) = \sup_{\alpha\in\mathbb{A}}\sum_{n=0}^{\infty}\mathbb{E}^{\alpha}_{t,z,\lambda}\left[1_{T_n\le T}G\left(Z^{\alpha}_{T_n}\right)\exp\left\{-|\lambda_0|(T-T_n)\right.\right. \\ \left.\left. + \sum_{m=1}^{D}\frac{\lambda^m_{T_n}-\lambda^m_0}{\sum_{j=1}^{D}\beta_{mj}}\left(e^{-\sum_{j=1}^{D}\beta_{mj}(T-T_n)}-1\right)\right\}\right], \tag{3.5.3}$$

where, for $n\ge 0$, T_n stands for the n^{th} jump time of Z after time t, and $(Z^{\alpha}_{T_n})_{n=0}^{\infty}$ is seen as a MDP controlled by $\alpha\in\mathbb{A}$; and where $\mathbb{E}^{\alpha}_{t,z,\lambda}[.]$ stands for the expectation conditioned by $Z_t=z,\lambda_t=\lambda$ when the control α is followed.

Remark 3.5.1. *V is characterized in (3.5.3) as the value function associated to an MDP with infinite horizon, for which the instantaneous reward writes:*

$$r(t,z,\lambda) \quad = \quad \mathbb{1}_{t\le T}\,G\left(z\right)\exp\left\{-|\lambda_0|_1(T-t)+\sum_{m=1}^{D}\frac{\lambda^m-\lambda_0^m}{\sum_{j=1}^{D}\beta_{mj}}\left(e^{-\sum_{j=1}^{D}\beta_{mj}(T-t)}-1\right)\right\},$$

where $|.|_1$ denotes the $\mathbb{L}^1\big(\mathbb{R}^D\big)$ norm.

Proof: (of Theorem 3.5.1)

Step 1: Let us check that (3.5.3) holds, where V is defined as solution of (3.5.1).

We want to show that (3.5.10) is the expression of the maximal reward operator associated to the PDMDP (3.3.9) that we will define later. First notice that $(\lambda_t,Z_t)_t$ is a PDMDP, since $(\lambda_t,Z_t)_t$ is deterministic between two jumping times. We then aim at rewriting the expression of the value function defined in (3.5.1) as the value function associated to a infinite horizon control problem of the PDMDP $(\lambda_t,Z_t)_t$. To do so, we first notice that by conditioning on the time jumps we get:

$$\begin{aligned} V(t,z,\lambda) &= \sup_{\alpha\in\mathbb{A}}\mathbb{E}^{\alpha}_{t,z,\lambda}\Big[G\big(Z^{\alpha}_T\big)\Big] \\ &= \sup_{\alpha\in\mathbb{A}}\mathbb{E}^{\alpha}_{t,z,\lambda}\left[\sum_{n=0}^{\infty}1_{T_n\le T<T_{n+1}}G\big(Z^{\alpha}_{T_n}\big)\right] \\ &= \sup_{\alpha\in\mathbb{A}}\sum_{n=0}^{\infty}\mathbb{E}^{\alpha}_{t,z,\lambda}\Big[1_{T_n\le T}G\big(Z^{\alpha}_{T_n}\big)\mathbb{P}\Big(T-T_n\le T_{n+1}-T_n\Big|T_n\Big)\Big], \end{aligned} \tag{3.5.4}$$

where $\big(T_n\big)_n$ is the sequence of jump times of N. This process is a jump process with intensity $\mu_s=\sum_{m=1}^{D}\lambda^m_s$. Since it holds, conditioned to $\mathcal{F}_{T_n}$:

$$\mu_s = \sum_{m=1}^{D}\big(\lambda^m_{T_n}-\lambda^m_0\big)e^{-\sum_{j=1}^{D}\beta_{mj}(s-t)}, \qquad \text{for } s\in[T_n,T_{n+1}),$$

then, we have:

$$\mathbb{P}\Big(T_{n+1}-T_n \geq T-T_n \big| T_n\Big) = \int_{T-T_n}^{\infty} \mu_s e^{-\int_0^s \mu_u du}\,\mathrm{d}s$$
$$= \exp\left\{ -|\lambda_0|(T-T_n) + \sum_{m=1}^{D} \frac{\lambda^m_{T_n}-\lambda^m_0}{\sum_{j=1}^D \beta_{mj}}\Big(e^{-\sum_{j=1}^D \beta_{mj}(T-T_n)}-1\Big)\right\}. \tag{3.5.5}$$

Plugging (3.5.5) into (3.5.4), the value function rewrites:

$$V(t,z,\lambda) = \sup_{\alpha\in\mathbb{A}} \sum_{n=0}^{\infty} \mathbb{E}^{\alpha}_{t,z,\lambda}\left[1_{T_n\leq T} G\left(Z^{\alpha}_{T_n}\right) \exp\left\{ -|\lambda_0|(T-T_n) \right.\right.$$
$$\left.\left. + \sum_{m=1}^{D} \frac{\lambda^m_{T_n}-\lambda^m_0}{\sum_{j=1}^D \beta_{mj}}\Big(e^{-\sum_{j=1}^D \beta_{mj}(T-T_n)}-1\Big)\right\}\right], \tag{3.5.6}$$

which completes the step 1. The r.h.s of (3.5.6) can be seen as the value function of an infinite horizon control problem associated to the PDMDP.

Step 2: Let us show that V is the unique solution to (3.5.2).

Notice first that the solutions to the following HJB equation

$$\begin{cases} G(z) &= f(T,z,\lambda) \\ 0 &= \frac{\partial f}{\partial t} - \sum_{m=1}^D \sum_{j=1}^D \beta_{mj}\big(\lambda^m-\lambda^m_0\big)\frac{\partial f}{\partial \lambda^m} + \lambda^m \sup\limits_{a\in A_z} \big[f\big(t,e^a_m(z),\lambda+\alpha_m\big)-f\big(t,z,\lambda\big)\big], \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } 0\leq t<T. \end{cases}$$

are the fixed points of the operator $\mathcal{T}=\mathcal{T}_1\circ\mathcal{T}_2$ where $\mathcal{T}_1$ and $\mathcal{T}_2$ are defined as follows:

$$\mathcal{T}_1 : F \mapsto f \text{ solution of } \begin{cases} \frac{\partial f}{\partial t} - \sum_{m=1}^D \sum_{j=1}^D \beta_{mj}\big(\lambda^m-\lambda^m_0\big)\frac{\partial f}{\partial \lambda^m} = F(t,z,\lambda) \\ f(T,z,\lambda)=G(z), \end{cases}$$

and:

$$\mathcal{T}_2 : f \mapsto -\sum_{m=1}^{D} \lambda^m \sup_{a\in A_z} \big[f\big(t,e^a_m(z),\lambda+\alpha_m\big)-f\big(t,z,\lambda\big)\big].$$

We now use the characteristic method to rewrite the image of $\mathcal{T}_1$.
Let us take function F, and define $f=\mathcal{T}_1(F)$. Let us fix $t\in[0,T]$ and $\lambda\in(\mathbb{R}_+)^D$, and denote by g the function $g(s,z)=f(s,z,\lambda^1_s,...,\lambda^D_s)$ where, for $m=1,\dots,D$, $s\mapsto\lambda^m_s$ is a differentiable function defined on $[t,T]$ as solution to the following ODE:

$$\begin{cases} \frac{d\lambda^m_s}{ds} &= -\sum_{j=1}^D \beta_{mj}\big(\lambda^m_s-\lambda^m_0\big), \qquad \text{for all } t<s\leq T, \\ \lambda^m_t &= \lambda^m. \end{cases} \tag{3.5.7}$$

For $m=1,\dots,D$, basic theory on ODE provides existence and uniqueness of a solution to (3.5.7), which is given by:

$$\lambda^m_s = \lambda^m_0 + \big(\lambda^m-\lambda^m_0\big)e^{-\sum_{j=1}^D \beta_{mj}(s-t)}, \quad \text{for } s\in[t,T], \text{ and } m=1,\dots,D.$$

Since $\frac{\partial g}{\partial s} = \frac{\partial f}{\partial s} + \sum_{m=1}^D \frac{d\lambda^m_s}{ds}\frac{\partial f}{\partial \lambda^m}$, then $g(t,z)=G(z)-\int_t^T F(s,z,\lambda_s)\,\mathrm{d}s$, which finally leads to the following expression of $\mathcal{T}_1(F)$:

$$\mathcal{T}_1(F) = f(t,z,\lambda) = G(z) - \int_t^T F\big(s,z,\lambda_s\big)\,\mathrm{d}s. \tag{3.5.8}$$

Replacing F by $\mathcal{T}_2(f)$ in (3.5.8), we get that f is fixed point of $\mathcal{T}_1\circ\mathcal{T}_2$ if and only if:

$$f(t,\lambda,z) + \sum_{m=1}^{D}\int_t^T \lambda^m_s f\big(s,z,\lambda_s\big)\,\mathrm{d}s = G(z) - \sum_{m=1}^{D}\int_t^T \lambda^m_s \sup_{a\in A_z} f\big(s,e^a_m(z),\lambda_s+\alpha_m\big)\,\mathrm{d}s.$$

Notice

$$\frac{\partial f(s,\lambda_s,z)e^{-\sum_{j=1}^{D}\int_t^s \lambda_u^j du}}{\partial s} = -\sum_{m=1}^{D} \lambda_s^m e^{-\sum_{j=1}^{D}\int_t^s \lambda_u^j du} \sup_{a\in A_z} f\big(s, e_m^a(z), \lambda_s + \alpha_m\big),$$

so that:

$$\begin{aligned} f(t,\lambda,z) &= G(z)e^{-\sum_{m=1}^{D}\int_t^T \lambda_s^m \,\mathrm{d}s} + \sum_{m=1}^{N}\int_t^T \lambda_s^m e^{-\int_t^s \lambda_u du} \sup_{a\in A_z} f\big(s, e_m^a(z), \lambda_s + \alpha_m\big)\,\mathrm{d}s \\ &= G(z)e^{-\sum_{m=1}^{D}\int_t^T \lambda_s^m \,\mathrm{d}s} + \sup_{a\in A_z} \mathbb{E}^a_{t,\lambda,z}\left[f\big(T_1, Z_1, \lambda_{T_1} + \alpha_m\big)\right], \end{aligned} \tag{3.5.9}$$

where T_1 is the first jump time of N larger than t, we denote $Z_1 = Z_{T_1}$. Equation (3.5.9) shows that the fixed point of $\mathcal{T}_1 \circ \mathcal{T}_2$ is characterized as the fixed point of the operator $\mathcal{T}$ defined for any smooth enough function f by:

$$\mathcal{T}(f) = G(z)e^{-\sum_{m=1}^{D}\int_t^T \lambda_s^m \,\mathrm{d}s} + \sup_{a\in A_z} \mathbb{E}^a_{t,\lambda,z}\Big[f\big(T_1, Z_1, \lambda_{T_1} + \alpha_m\big)\Big], \tag{3.5.10}$$

where $\mathbb{E}^a_{t,\lambda,z}[.]$ stands for the expectation conditioned by the events $\lambda_t = \lambda$ and $Z_t = z$, when decision a is taken at time t. We recognize here the maximal reward operator of the value function defined in (3.5.6). Basic theory on PDMDP shows that the maximal reward operator $\mathcal{T}$ admits V as unique fixed point, which completes step 2. □

3.A From uncontrolled to controlled intensity

Remind that the results state in Section 3.3 hold when assuming that the intensities of the orders arrivals are uncontrolled. In particular, we assumed in this section that the market maker has no influence on the next exogenous event that will occur. This can be seen as a weak assumption if the market maker is a small player, but never holds in the case where the latter is a large player.
In this section, we show how to alleviate Assumption **(Harrivals2)** by rewriting the initial control problems (3.3.1) with controlled intensities as a control problems with uncontrolled intensities under a new (controlled) probability measure. The results and proofs in this section are inspired from [Bré81].

Consider a LOB which can receive at any time limit, cancel, and market orders. Denote by L^+ (res. L^-) the limit sell (resp. buy) order arrival process, received on the ask (resp. buy) side. Denote by C^+ (resp. C^-) the cancel order on the ask (resp. bid) side. Denote by M^+ (resp. M^-) the buy (resp. sell) market order process. The orders arrivals process is then a $(4K+2)$ dimensional process. Recall that E is the state space of the order book. The order book is modeled by a jump process $Z : [0,T] \to E$ such that the order arrivals processes have uncontrolled stochastic intensities $\lambda^i(\underline{a}, \underline{b})$, for $i = 1, \ldots, 4K+2$, that only depend on the bid and ask sides, i.e. $(\underline{a}, \underline{b})$ of the order book under $\mathbb{P}$. We underline that, by assumption, the intensities are uncontrolled under $\mathbb{P}$.
Let us fix $(\alpha_t)_{0\le t\le T} \in \mathbb{A}$ an admissible control, i.e. a predictive process w.r.t. the natural filtration $\big(\mathcal{F}_t\big)_{t>0}$ generated by the uncontrolled orders arrivals processes under $\mathbb{P}$.

(HarrivalsL): We assume in this section that the intensities are Lipschitz and bounded, i.e. there exist $[\lambda]_L > 0$ and $\|\lambda\|_\infty > 0$ such that

$$\left|\lambda^i(\underline{a}, \underline{b}) - \lambda^i(\underline{a}', \underline{b}')\right| \;\le\; [\lambda]_L\left(|\underline{a} - \underline{a}'| + |\underline{b} - \underline{b}'|\right),$$

and

$$\lambda^i(\underline{a}, \underline{b}) \;\le\; \|\lambda\|_\infty, \qquad \text{for } i = 1, \ldots, 4K+2,$$

for $\underline{a}, \underline{a}' \in \mathbb{N}^K$ and $\underline{b}, \underline{b}' \in (-\mathbb{N})^K$.

We want to define the probability $\mathbb{P}_\alpha$ as the absolutely continuous probability w.r.t. $\mathbb{P}$, which Radon-Nikodym derivative writes:

$$L_t^\alpha = \left.\frac{d\mathbb{P}_\alpha}{d\mathbb{P}}\right|_{\mathcal{F}_t} = \prod_{i=1}^{4K+2}\prod_{n=1}^{\infty}\mu^i_{\alpha,T_n^i}\mathbf{1}_{T_n^i\le t}\exp\left\{\int_0^t\left(1-\mu_s^i\right)\lambda_s^i\,\mathrm{d}s\right\}, \quad \text{for } 0\le t\le T, \tag{3.A.1}$$

where for $i=1,\dots,4K+2$, we denote by μ^i_{α,T_n^i} the quotient of the controlled intensity at time T_n^i of the n^{th} jump of the i^{th} process and the uncontrolled intensity, i.e. denoting by $\underline{a}^\alpha$ and $\underline{b}^\alpha$ the ask and bid where the market order's orders are counted, we define:

$$\mu_\alpha^i = \frac{\lambda(\underline{a}^\alpha,\underline{b}^\alpha)}{\lambda(\underline{a},\underline{b})}.$$

Remark 3.A.1. *Under **(HarrivalsL)**, it holds:*

$$|\lambda(\underline{a}^\alpha,\underline{b}^\alpha)-\lambda(\underline{a},\underline{b})|\le[\lambda]_L M,\ \textit{for } \underline{a},\underline{a}'\in\mathbb{N}^K,\ \textit{ and }\ \underline{b},\underline{b}'\in(-\mathbb{N})^K, \tag{3.A.2}$$

where we remind that M stands for the limit number of orders that can be hold by the market maker at the same time in the LOB.

Remark 3.A.2. *From Remark 3.A.1, it is straightforward to see that μ_α^i is bounded under **(HarrivalsL)**, and moreover:*

$$\mu_\alpha^i \le 1+\frac{[\lambda]_L M}{\lambda_{\min}},\quad \textit{for } i=1,\dots,4K+2,$$

where we denote $\lambda_{\min} = \inf_{i=1,\dots,4K+2}\inf_{z\in E}\lambda^i(z)$, and assume the latter to be strictly positive. Note that the bound is uniform w.r.t. the control and the state variables.

Proposition 3.A.1. *For every $\alpha\in\mathbb{A}$, it holds under **(HarrivalsL)**:*

$$\mathbb{E}\left[L_T^\alpha\right] = 1, \tag{3.A.3}$$

which implies in particular that $\mathbb{P}_\alpha$ is well-defined.
Moreover, the orders arrivals admit the controlled intensities $\lambda(\underline{a}^\alpha,\underline{b}^\alpha)$, for $i=1,\dots,4K+2$, under $\mathbb{P}_\alpha$, where we remind that $\underline{a}^\alpha$ and $\underline{b}^\alpha$ stand for the vector of orders on the ask and the bid sides, where the market maker's orders are counted.

Proof. We divided the proof of Proposition 3.A.1 into two steps.

Step 1: We show (3.A.3).
Let us fix $\alpha\in\mathbb{A}$ and write the integral representation of $(L_t^\alpha)_{0\le t\le T}$:
for $t\in[0,T]$,

$$L_t^\alpha = 1+\sum_{i=1}^{4K+2}\int_0^t L_{s-}^\alpha\left(\mu_{\alpha,s}^i-1\right)\mathrm{d}\tilde{M}_s^i, \quad \text{for } i=1,\dots,4K+2, \tag{3.A.4}$$

where $\tilde{M}$ stands for the local martingale which dynamic writes: $\mathrm{d}\tilde{M}_s^i = \mathrm{d}N_s^i-\lambda^i\left(\underline{a}_s,\underline{b}_s\right)\mathrm{d}s$. It is then sufficient to show that

$$\mathbb{E}\left[\int_0^T L_{s-}^\alpha\left(\mu_{\alpha,s}^i-1\right)\lambda_s^i\,\mathrm{d}s\right] < +\infty, \quad \text{for } i=1,\dots,4K+2, \tag{3.A.5}$$

to get that the $\left(\int_0^t L_{s-}^\alpha\left(\mu_{\alpha,s}^i-1\right)\mathrm{d}\tilde{M}_s^i\right)_{0\le t\le T}$ are martingales for $i=1,\dots,4K+2$ (as proved e.g. in [Bré81]), and complete the proof of Step 1, using (3.A.4).

Plugging (3.A.2) into (3.A.1), we get:

$$L_s \quad \leq \quad \|\mu\|_\infty^{A_s} e^{[\lambda]_L MT}, \qquad \text{for } 0 \leq s \leq T, \tag{3.A.6}$$

where we denote $\|\mu\|_\infty := 1 + \frac{[\lambda]_L M}{\lambda_{\min}}$, and where $(A_t)_{t\in[0,T]}$ stands for the sum of all the order arrivals process up to time t, for $t \in [0,T]$.
Moreover, as stated in Remark 3.A.1, we have for all $i = 1, \ldots, 4K+2$:

$$\begin{aligned} |\big(\mu_s^i - 1\big)\lambda_s^i(z)| &= |\lambda(\underline{a}^\alpha, \underline{b}^\alpha) - \lambda(\underline{a}, \underline{b})| \\ &\leq [\lambda]_L M. \end{aligned} \tag{3.A.7}$$

Plugging (3.A.7) and (3.A.6) into the l.h.s. of (3.A.5), we get:

$$\mathbb{E}\left[\int_0^T L_{s-}^\alpha \big(\mu_{\alpha,s}^i - 1\big)\lambda_s^i \, \mathrm{d}s\right] \quad \leq \quad \int_0^T \mathbb{E}\left[\|\mu\|_\infty^{A_s}\right] e^{[\lambda]_L MT} [\lambda]_L M \, \mathrm{d}s \tag{3.A.8}$$

Notice that the intensity of A is bounded by $\|\lambda\|_\infty$, under **(HarrivalsL)**, so that:

$$\begin{aligned} \mathbb{E}\left[\|\mu\|_\infty^{A_s}\right] &\leq e^{-\|\lambda\|_\infty s} \sum_{n=0}^{+\infty} \frac{\|\mu\|_\infty^n \left(\|\lambda\|_\infty s\right)^n}{n!} \\ &\leq \exp\left\{\|\lambda\|_\infty T \left(\|\mu\|_\infty - 1\right)\right\}, \qquad \text{for } s \in [0,T], \end{aligned} \tag{3.A.9}$$

Combining (3.A.8) and (3.A.9), we can prove that (3.A.5) holds, which completes the proof of Step 1.

Step 2: We refer to the T3 Theorem in Chapter VI of [Brémaud81] for a proof of the second assertion in Proposition 3.A.1. □

3.B Dynamics of the controlled order book (simplified version)

In this section, we give the expressions for the dynamics of the controlled order book process (Z_t). The market maker control has been simplified to a couple (la_t, lb_t), where $la = 1$ (resp. 0) if the market maker holds (does not hold) a sell order at the best ask limit, and $lb = 1$ (resp. 0) if the market maker holds (does not hold) a buying order at the best bid limit. So to speak, the market maker considers to place orders at the best ask limit or at the best bid limit exclusively. In the numerical simulations that we run, we also had to calculate the dynamics of (Z_t) for the set of generalized controls in which the market maker is allowed to post orders on the two first limits at the bid and at the ask side. The expression of the dynamics for the generalized controls are very similar to the ones for the simplified controls.

To understand the dynamics of the rank of the orders of the market maker, we need a model for the cancellation of orders. Suppose for example that the market maker holds an order whose rank is na in the queue, with $na < a_{A^{-1}(0)}$. Suppose that the cancel process $L^{C^+}_{A^{-1}(0)}$ jumps. Then two scenarios can occur:

- If the rank of the canceled order is greater than the one of the market maker, then na_t stays constant.
- If the rank of the canceled order is smaller than the one of the market maker, then $na_t = na_{t^-} + 1$.

Model:
We consider a Bernoulli variable X^a with parameter: $\underbrace{\frac{na-1}{a_{A^{-1}(0)}}}_{\alpha} \delta_1 + \underbrace{\frac{a_{A^{-1}(0)} + 1 - na}{a_{A^{-1}(0)}}}_{\beta} \delta_0$.
We assume that the canceled order is in front of the market maker's order in the queue if $X^a = 1$,

and behind it if $X^a = 0$.

We proceed for the bid side as we just did for the ask side. We consider a random variable X^b following a Bernoulli law with parameter: $\underbrace{\frac{nb-1}{|b_{B^{-1}(0)}|}}_{\alpha}\delta_1 + \underbrace{\frac{|b_{B^{-1}(0)}|+1-nb}{|b_{B^{-1}(0)}|}}_{\beta}\delta_0$.

3.B.1 Dynamics of X_t et Y_t

The dynamic of the amount hold by the market maker on a no-interest-bearing account $(X_t)_{t\in\mathbb{R}_+}$ is as follows:

$$dX_t = la_t pa_{t-}\mathbf{1}_{\{na_{t-}=1\}}dM_t^+ - lb_t pb_{t-}\mathbf{1}_{\{nb_{t-}=1\}}dM_t^-$$

The market maker's inventory (Y_t) follows the dynamic:

$$dY_t = -la_t\mathbf{1}_{\{na_{t-}=1\}}dM_t^+ + \mathbf{1}_{\{nb_{t-}=1\}}lb_t dM_t^-$$

where:

- $\hat{a} = sup\{a_i : \sum_{j=1}^{i-1} a_j = 0\}$ et $\hat{b} = sup\{b_i : \sum_{j=1}^{i-1} b_j = 0\}\}$
- $M_t^{\pm}$ are Cox processes with intensities $\lambda^{M^{\pm}}$

3.B.2 Dynamics of the a_t et b_t

We remind that a_i is the number of orders located i ticks away from the best buy order.
We denote by $\mathcal{J}$ the $shift$ operator that re-index a side of the book when an event occurred on the opposite side.

$\mathcal{J}^{L_i^-}, i \in \{1,\dots,B^{-1}(0)\}$ is the shift operator that shifts the bid side due to the jump of a L_i^+ for $i \in \{0, K\}$. We get:

$$\mathcal{J}^{L_i^-}(\underline{a}) = \left(a_{i+1},...,a_K,\underbrace{a_\infty,\dots,a_\infty}_{i \text{ times}}\right)$$

Dynamics of a_i:

$$
\begin{aligned}
\mathrm{d}a_i = {} & (1-lb_t)\,\mathrm{d}L_i^+ + lb_t\,\mathrm{d}L^+_{i-(A^{-1}(0)-rb_{t-})} + \Big[\big(1-lb_t\big) + lb_t \mathbb{1}_{\{nb_{t-}>1\}}\Big]\Big(\mathcal{J}^{M^-}\big(a_i\big) - a_i\Big)\,\mathrm{d}M^-(t) \\
& - (1-lb_t) dC_i^+ - lb_t\,\mathrm{d}C^+_{i-(A^{-1}(0)-rb_{t-})} \\
& + (1-la_t)\Bigg[- \mathbb{1}_{\{i=A^{-1}(0)\}}\,\mathrm{d}M_t^+ + \big(\mathcal{J}^{C^-}(a_i) - a_i\big)\,\mathrm{d}C^-_{A^{-1}(0)} \\
& \qquad + (1-lb_t) \sum_{j=1}^{A^{-1}(0)-1} (\mathcal{J}_{0,0}^{L_j^-}(a_i) - a_i)\,\mathrm{d}L_j^-(t) \\
& \qquad + lb_t \sum_{j=1}^{A^{-1}(0)-1} (\mathcal{J}_{0,1}^{L_j^-}(a_i) - a_i)\,\mathrm{d}L_j^-(t)\Bigg] \\
& + la_t\Bigg[- \mathbb{1}_{\{na_{t-}>1\}}\mathbb{1}_{\{i=A^{-1}(0)\}}\,\mathrm{d}M_t^+ + \big(\mathcal{J}^{C^-}(a_i) - a_i\big)\,\mathrm{d}C^-_{ra_{t-}} \\
& \qquad + lb_t \sum_{j=1}^{ra_{t-}-1} (\mathcal{J}_{1,1}^{L_j^-}(a_i) - a_i)\,\mathrm{d}L_j^-(t) \\
& \qquad + (1-lb_t) \sum_{j=1}^{ra_{t-}-1} (\mathcal{J}_{1,0}^{L_j^-}(a_i) - a_i)\,\mathrm{d}L_j^-(t)\Bigg]
\end{aligned}
$$

with $\mathcal{J}$ such that:

$$
\mathcal{J}^{C^-}(a_i) = \begin{cases} a_\infty & \text{si } i > B^{-1}(1) - B^{-1}(0) + K \\ a_{i-\left(B^{-1}(1)-B^{-1}(0)\right)} & \text{si } i > \left(B^{-1}(1) - B^{-1}(0)\right. \\ 0 & \text{si } i \le B^{-1}(1) - B^{-1}(0) \end{cases}
$$

$$
\mathcal{J}^{M^-}(a_i) = \begin{cases} a_{i-\left(B^{-1}(1)-B^{-1}(0)\right)} & \text{si } i > \left(B^{-1}(1) - B^{-1}(0)\right. \\ 0 & \text{si } i \le B^{-1}(1) - B^{-1}(0) \end{cases}
$$

$$
\mathcal{J}_{0,0}^{L_j^-}(a_i) = \begin{cases} a_{i+j} & \text{si } i+j \le K \\ 0 & \text{si } i+j < K \end{cases}
$$

$$
\mathcal{J}_{0,1}^{L_j^-}(a_i) = \begin{cases} a_{i+rb_{t-}-j} & \text{si } i+rb_{t-}-j \le K \\ a_\infty & \text{si } i+rb_{t-}-j > K \end{cases}
$$

$$
\mathcal{J}_{1,0}^{L_j^-}(a_i) = \begin{cases} a_{i+ra_{t-}-j} & \text{si } i+ra_{t-}-j \le K \\ a_\infty & \text{si } i+ra_{t-}-j > K \end{cases}
$$

$$
\mathcal{J}_{1,1}^{L_j^-}(a_i) = \begin{cases} 0 & \text{si } i+rb_{t-}-j < 0 \\ a_{i+rb_{t-}-j} & \text{si } i+rb_{t-}-j \le K \\ a_\infty & \text{si } i+rb_{t-}-j > K \end{cases}
$$

We remind that b_i is the number of buy order located i ticks away from the best sell order.

Dynamics of b_i:

$$
\begin{aligned}
db_i = &-(1-la_t)dL_i^- - la_t\,\mathrm{d}L^-_{i-(A^{-1}(0)-ra_{t-})} + \Big[(1-la_t) + la_t\mathbb{1}_{\{na_{t-}>1\}}\Big]\Big(\mathcal{J}^{M^+}(b_i) - b_i\Big)\,\mathrm{d}M^+(t)\\
&+ (1-la_t)\,\mathrm{d}C_i^- + la_t\,\mathrm{d}C^-_{i-(A^{-1}(0)-ra_{t-})}\\
&+ (1-lb_t)\Bigg[\mathbb{1}_{\{i=A^{-1}(0)\}}\,\mathrm{d}M_t^- + \big(\mathcal{J}^{C^+}(b_i) - b_i\big)\,\mathrm{d}C^+_{A^{-1}(0)}\\
&\qquad + (1-la_t)\sum_{j=1}^{B^{-1}(0)-1}\big(\mathcal{J}_{0,0}^{L_j^+}(b_i) - b_i\big)\,\mathrm{d}L_j^+(t)\\
&\qquad + la_t\sum_{j=1}^{B^{-1}(0)-1}\big(\mathcal{J}_{1,0}^{L_j^+}(b_i) - b_i\big)\,\mathrm{d}L_j^+(t)\Bigg]\\
&+ lb_t\Bigg[\mathbb{1}_{\{nb_{t-}>1\}}\mathbb{1}_{\{i=A^{-1}(0)\}}\,\mathrm{d}M_t^- + \big(\mathcal{J}^{C^+}(b_i) - b_i\big)\,\mathrm{d}C^+_{rb_{t-}}\\
&\qquad + la_t\sum_{j=1}^{rb_{t-}-1}\big(\mathcal{J}_{1,1}^{L_j^+}(b_i) - b_i\big)\,\mathrm{d}L_j^+(t)\\
&\qquad + (1-la_t)\sum_{j=1}^{rb_{t-}-1}\big(\mathcal{J}_{0,1}^{L_j^+}(b_i) - b_i\big)\,\mathrm{d}L_j^+(t)\Bigg]
\end{aligned}
$$

with $\mathcal{J}$ the shift operators:

$$
\mathcal{J}^{C^+}(b_i) = \begin{cases} b_\infty & \text{si } i + A^{-1}(1) - A^{-1}(0) > K \\ b_{i-\left(A^{-1}(1)-A^{-1}(0)\right)} & \text{si } i > \left(A^{-1}(1) - A^{-1}(0)\right. \\ 0 & \text{si } i \le A^{-1}(1) - A^{-1}(0) \end{cases}
$$

$$
\mathcal{J}^{M^+}(b_i) = \begin{cases} b_\infty & \text{si } i + A^{-1}(1) - A^{-1}(0) > K \\ b_{i-\left(A^{-1}(1)-A^{-1}(0)\right)} & \text{si } i > \left(A^{-1}(1) - A^{-1}(0)\right. \\ 0 & \text{si } i \le A^{-1}(1) - A^{-1}(0) \end{cases}
$$

$$
\mathcal{J}_{0,0}^{L_j^+}(b_i) = \begin{cases} b_{i+j} & \text{si } i + j \le K \\ 0 & \text{si } i + j > K \end{cases}
$$

$$
\mathcal{J}_{1,0}^{L_j^+}(b_i) = \begin{cases} b_{i-j+A^{-1}(0)} & \text{si } i - j + A^{-1}(0) \le K \\ b_\infty & \text{si } i - j + A^{-1}(0) > K \end{cases}
$$

$$
\mathcal{J}_{0,1}^{L_j^+}(b_i) = \begin{cases} b_{i+rb_{t-}-j} & \text{si } i + rb_{t-} - j \le K \\ b_\infty & \text{si } i + rb_{t-} - j > K \end{cases}
$$

$$
\mathcal{J}_{1,1}^{L_j^+}(b_i) = \begin{cases} 0 & \text{si } i + rb_{t-} - j < 0 \\ b_{i+rb_{t-}-j} & \text{si } i + rb_{t-} - j \le K \\ b_\infty & \text{si } i + rb_{t-} - j > K \end{cases}
$$

3.B.3 Dynamics of na_t and nb_t

Dynamics of na_t:

X^a has been introduced in part 3.B. It models whether the canceled order is behind or in front of the market maker's order in the queue.

We get:

$$
\begin{aligned}
dna_t = la_t \Bigg[& - X^a \Big((1-lb_t)\, \mathrm{d}C^+_{A^{-1}(0)}(t) + lb_t\, \mathrm{d}C^+_{rb_{t-}} \Big) + \Big(- \mathbb{1}_{\{na_{t-}>1\}} + \big(a_{A^{-1}(0)} + 1 - na_{t-}\big)\mathbb{1}_{\{na_{t-}=1\}} \Big)\, \mathrm{d}M^+_t \\
& + (2-na_{t-}) \Big\{ (1-lb_t) \sum_{i=1}^{ra_{t-}(t)-1} \mathrm{d}L^+_i + lb_t \sum_{i=1}^{ra_{t-}(t)-(A^{-1}(0)-rb_{t-})-1} \mathrm{d}L^+_i \Big\} \Bigg] \\
& + (1-la_t) \Bigg[\Big(a_{A^{-1}(0)} \mathbb{1}_{\{a_{A^{-1}(0)}>1\}} + (a_{A^{-1}(1)}+1)\mathbb{1}_{\{a_{A^{-1}(0)}=1\}} - na_{t-} \Big)\, \mathrm{d}M^+_t \\
& \quad + lb_t \Bigg[(2-na_{t-}) \sum_{j=1}^{rb_{t-}-1} dL^+_j + (a_{A^{-1}(0)} + 2 - na_{t-})\, \mathrm{d}L^+_{rb_{t-}} \\
& \qquad + (a_{A^{-1}(0)} + 1 - na_{t-}) \sum_{j=rb_{t-}+1}^{K} \big(\mathrm{d}L^+_j + dC^+_j \big) \\
& \qquad + \Big(a_{A^{-1}(0)} \mathbb{1}_{\{a_{A^{-1}(0)}>1\}} + (a_{A^{-1}(1)}+1)\mathbb{1}_{\{a_{A^{-1}(0)}=1\}} - na_{t-} \Big)\, \mathrm{d}C^+_{rb_{t-}} \Bigg] \\
& \quad + (1-lb_t) \Bigg[(2-na_{t-}) \sum_{j=1}^{B^{-1}(0)-1} \mathrm{d}L^+_j + (a_{A^{-1}(0)} + 2 - na_{t-})\, \mathrm{d}L^+_{A^{-1}(0)} \\
& \qquad + (a_{A^{-1}(0)} + 1 - na_{t-}) \sum_{j=B^{-1}(0)+1}^{K} \Big(\mathrm{d}L^+_j + \mathrm{d}C^+_j \Big) \\
& \qquad + \Big(a_{A^{-1}(0)} \mathbb{1}_{\{a_{A^{-1}(0)}>1\}} + (a_{A^{-1}(1)}+1)\mathbb{1}_{\{a_{A^{-1}(0)}=1\}} - na_{t-} \Big)\, \mathrm{d}C^+_{A^{-1}(0)} \Bigg] \\
& \quad + \big(a_{A^{-1}(0)} + 1 - na_{t-} \big) \Big[\mathrm{d}M^-_t + \sum_{j=1}^{K} \Big(\mathrm{d}L^-_j + \mathrm{d}C^-_j \Big) \Big] \Bigg]
\end{aligned}
$$

$$
\begin{aligned}
dna_t = (la_t == 0)(-1-na_{t-}) \Bigg[& \mathrm{d}M^+_t + \mathrm{d}M^-_t \\
& + (lb_t != 1) \sum_{i=1}^{K} \big(\mathrm{d}L^+_i + \mathrm{d}L^-_i + \mathrm{d}C^+_i + \mathrm{d}C^-_i \big) \\
& + (lb_t == 1) \Big[\sum_{i=0}^{K-(A^{-1}(0)-rb_{t-})} \big(dL^+_i + dC^+_i \big) + \sum_{i=1}^{K} \big(dL^-_i + \mathrm{d}C^-_i \big) \Big] \Bigg] \\
+ \mathbf{1}_{la_t=1} \Bigg\{ & \Big(\mathbf{1}_{na_{t-}=-1} + \mathbf{1}_{na_{t-}!=-1} \mathbf{1}_{ra_{t-}>A^{-1}(0)} \Big) \Big[\big(a_{A^{-1}(0)} - na_{t-} \big) \Big[\mathrm{d}M^+_t + \mathbf{1}_{lb_t=1}\, \mathrm{d}C^+_{rb_{t-}} + \mathbf{1}_{lb_t!=1}\, \mathrm{d}C^+_{A^{-1}(0)} \Big] \\
& + \big(a_{A^{-1}(0)} + 1 - na_{t-} \big) \Big[\mathbf{1}_{lb_t!=1} \sum_{i=1}^{K} \big(dL^+_i + dC^+_i \big) + \mathbf{1}_{lb_t=1} \sum_{i=1}^{K-(A^{-1}(0)-rb_{t-})} \big(\mathrm{d}L^+_i + dC^+_i \big) \Big] \Big] \Bigg\}
\end{aligned}
$$

Dynamics of nb_t:

$$
\begin{aligned}
dnb_t = lb_t\Bigg[& - X^b\Big((1-la_t)\,\mathrm{d}C^-_{B^{-1}(0)}(t) + la_t\,\mathrm{d}C^-_{ra_{t-}}\Big) + \Big(- \mathbb{1}_{\{nb_{t-}>1\}} + \big(|b_{B^{-1}(0)}| + 1 - nb_{t-}\big)\mathbb{1}_{\{nb_{t-}=1\}}\Big)\,\mathrm{d}M^-_t \\
& + (2-nb_{t-})\Big\{(1-la_t)\sum_{i=1}^{rb_{t-}(t)-1}\mathrm{d}L^-_i + la_t\sum_{i=1}^{rb_{t-}(t)-(B^{-1}(0)-ra_{t-})-1}\mathrm{d}L^-_i\Big\}\Bigg] \\
& + (1-lb_t)\Bigg[\Big(|b_{A^{-1}(0)}|\mathbb{1}_{\{|b_{B^{-1}(0)}|>1\}} + (|b_{B^{-1}(1)}|+1)\mathbb{1}_{\{|b_{B^{-1}(0)}|=1\}} - nb_{t-}\Big)\,\mathrm{d}M^-_t \\
& \quad + la_t\Bigg[(2-nb_{t-})\sum_{j=1}^{ra_{t-}-1} dL^-_j + (|b_{A^{-1}(0)}|+2-nb_{t-})\,\mathrm{d}L^-_{ra_{t-}} \\
& \qquad + (|b_{A^{-1}(0)}|+1-nb_{t-})\sum_{j=ra_{t-}+1}^{K}\Big(\mathrm{d}L^-_j + \mathrm{d}C^-_j\Big) \\
& \qquad + \Big(|b_{A^{-1}(0)}|\mathbb{1}_{\{|b_{B^{-1}(0)}|>1\}} + (|b_{B^{-1}(1)}|+1)\mathbb{1}_{\{|b_{B^{-1}(0)}|=1\}} - nb_{t-}\Big)\,\mathrm{d}C^-_{ra_{t-}}\Bigg] \\
& \quad + (1-la_t)\Bigg[(2-nb_{t-})\sum_{j=1}^{B^{-1}(0)-1}\mathrm{d}L^-_j + (|b_{B^{-1}(0)}|+2-nb_{t-})\,\mathrm{d}L^-_{A^{-1}(0)} \\
& \qquad + (|b_{B^{-1}(0)}|+1-nb_{t-})\sum_{j=B^{-1}(0)+1}^{K}\Big(\mathrm{d}L^-_j + \mathrm{d}C^-_j\Big) \\
& \qquad + \Big(|b_{A^{-1}(0)}|\mathbb{1}_{\{|b_{B^{-1}(0)}|>1\}} + (|b_{B^{-1}(1)}|+1)\mathbb{1}_{\{|b_{B^{-1}(0)}|=1\}} - nb_{t-}\Big)\,\mathrm{d}C^-_{A^{-1}(0)}\Bigg] \\
& \quad + \big(|b_{A^{-1}(0)}|+1-nb_{t-}\big)\Big[\mathrm{d}M^+_t + \sum_{j=1}^{K}\Big(\mathrm{d}L^+_j + \mathrm{d}C^+_j\Big)\Big]
\end{aligned}
$$

3.B.4 Dynamics of pa and pb

Dynamics of $(pa_t)_t$:
Denoting by δ the tick, we have:

$$
\begin{aligned}
\mathrm{d}P_t^A = \delta(1-la_t)\Bigg[&\Big(\big(A^{-1}(1)-ra_{t-}\big)\,\mathrm{d}M^+(t)\\
&+lb_t\Bigg[-\sum_{i=1}^{rb_{t-}-1}\Big[rb_{t-}-(A^{-1}(0)-ra_{t-})-j\Big]\,\mathrm{d}L_i^+(t)+\big(A^{-1}(0)-ra_{t-}\big)\sum_{j=rb_{t-}}^{K}dL_i^+\\
&\qquad+\big(A^{-1}(1)-ra_{t-}\big)\,\mathrm{d}C^+_{rb_{t-}}+\sum_{j=rb_{t-}+1}^{K}\big(A^{-1}(0)-ra_{t-}\big)\,\mathrm{d}C_j^+\Bigg]\\
&+\Big(1-lb_t\Big)\Bigg[-\sum_{i=1}^{A^{-1}(0)-1}\big(ra_{t-}-j\big)\,\mathrm{d}L_i^+(t)+\big(A^{-1}(0)-ra_{t-}\big)\sum_{j=A^{-1}(0)}^{K}dL_i^+\\
&\qquad+\big(A^{-1}(1)-ra_{t-}\big)\,\mathrm{d}C^+_{A^{-1}(0)}+\sum_{j=A^{-1}(0)+1}^{K}\big(A^{-1}(0)-ra_{t-}\big)\,\mathrm{d}C_j^+\Bigg]\\
&+\mathbb{1}_{\{ra_{t-}\neq A^{-1}(0)\}}\big(A^{-1}(0)-ra_{t-}\big)\Big(\mathrm{d}M_t^-+\sum_{j=1}^{K}\mathrm{d}L_t^-+\sum_{j=1}^{K}\mathrm{d}C_t^-\Big)\Bigg]\\
+\delta la_t\Bigg[&\big(A^{-1}(0)-r_t^A)\Big)\,\mathrm{d}M^+(t)\\
&-lb_t\sum_{i=1}^{rb_{t-}-(A^{-1}(0)-ra_{t-})-1}\Big(ra_{t-}-\big(j+A^{-1}(0)-rb_{t-}\big)\Big)\,\mathrm{d}L_i^+(t)\\
&-\Big(1-lb_t\Big)\sum_{i=1}^{ra_{t-}-1}\big(ra_{t-}-j\big)\,\mathrm{d}L_i^+(t)\Bigg]
\end{aligned}
$$

Dynamics of (pb_t):

$$
\begin{aligned}
\mathrm{d}P_t^B = -\delta(1-lb_t)\Bigg[&\Big(\big(B^{-1}(1)-rb_{t-}\big)\,\mathrm{d}M^-(t)\\
&+la_t\Bigg[-\sum_{i=1}^{ra_{t-}-1}\Big[ra_{t-}-(B^{-1}(0)-rb_{t-})-j\Big]\,\mathrm{d}L_i^-(t)+\big(B^{-1}(0)-rb_{t-}\big)\sum_{j=ra_{t-}}^{K}dL_i^-\\
&\qquad+\big(B^{-1}(1)-rb_{t-}\big)\,\mathrm{d}C^-_{rb_{t-}}+\sum_{j=ra_{t-}+1}^{K}\big(B^{-1}(0)-rb_{t-}\big)\,\mathrm{d}C_j^-\Bigg]\\
&+\Big(1-la_t\Big)\Bigg[-\sum_{i=1}^{B^{-1}(0)-1}\big(rb_{t-}-j\big)\,\mathrm{d}L_i^-(t)+\big(B^{-1}(0)-rb_{t-}\big)\sum_{j=B^{-1}(0)}^{K}dL_i^-\\
&\qquad+\big(B^{-1}(1)-rb_{t-}\big)\,\mathrm{d}C^-_{A^{-1}(0)}+\sum_{j=A^{-1}(0)+1}^{K}\big(B^{-1}(0)-rb_{t-}\big)\,\mathrm{d}C_j^-\Bigg]\\
&+\mathbb{1}_{\{rb_{t-}\neq A^{-1}(0)\}}\big(A^{-1}(0)-rb_{t-}\big)\Big(\mathrm{d}M_t^++\sum_{j=1}^{K}dL_j^++\sum_{j=1}^{K}dC_j^+\Big)\Bigg]\\
-\delta lb_t\Bigg[&\Big(B^{-1}(0)-r_t^B)\Big)\,\mathrm{d}M^-(t)\\
&-la_t\sum_{i=1}^{ra_{t-}-(B^{-1}(0)-rb_{t-})-1}\Big(rb_{t-}-\big(j+B^{-1}(0)-ra_{t-}\big)\Big)\,\mathrm{d}L_i^-(t)\\
&-\Big(1-la_t\Big)\sum_{i=1}^{rb_{t-}-1}\big(rb_{t-}-j\big)\,\mathrm{d}L_i^-(t)\Bigg]
\end{aligned}
$$

3.B.5 Dynamics of ra and rb

We remind that ra_t denotes the number of ticks between the market maker's order and the best buy order in the order book. We assumed in this simplified control problem that the market maker is allowed to place no more than one order on the best ask and best bid limits. So ra and rb are vectors of size 1 here.

Dynamics of ra:

$$
\begin{aligned}
\mathrm{d}\, ra_t = la_t \Bigg[& \mathbf{1}_{\{na=1\}} \Big(A^{-1}(0) - r_t^A \Big)\, \mathrm{d}M_t^+ \\
& + (1 - lb_t) \sum_{i=1}^{ra_{t-}-1} \Big(i - ra_{t-} \Big)\, \mathrm{d}L_i^+ + lb_t \sum_{i=1}^{rb_{t-}-(B^{-1}(0)-ra_{t-})-1} \Big(i + B^{-1}(0) - rb_{t-} - ra_{t-} \Big)\, \mathrm{d}L_i^+ \\
& + \sum_{i=1}^{ra_{t-}-1} \big(i - ra_{t-} \big)\, \mathrm{d}L_i^- + \big(B^{-1}(1) - B^{-1}(0) \big)\, \mathrm{d}C^-_{ra_{t-}} \\
& \Big(\Big((1 - lb_t) + lb_t \mathbb{1}_{\{nb_{t-}>1\}} \Big) \Big[B^{-1}(1) - B^{-1}(0) \Big] \Big)\, \mathrm{d}M_t^- \Bigg] \\
+ (l - la_t) \Bigg[& lb_t \Bigg[\sum_{j=1}^{rb_{t-}-1} \Big(j + B^{-1}(0) - rb_{t-} - ra_{t-} \Big)\, \mathrm{d}L_j^+ + \big(A^{-1}(0) - ra_{t-} \big) \sum_{j=rb_{t-}}^{K} \mathrm{d}L_j^+ \\
& \qquad \big(A^{-1}(1) - ra_{t-} \big)\, \mathrm{d}C^+_{rb_{t-}} + \big(A^{-1}(0) - ra_{t-} \big) \sum_{j=rb_{t-}+1}^{K} \mathrm{d}C_j^+ \Bigg] \\
& + (1 - lb_t) \Bigg[\sum_{j=1}^{B^{-1}(0)-1} \Big(j - ra_{t-} \Big)\, \mathrm{d}L_j^+ + \big(A^{-1}(0) - ra_{t-} \big) \sum_{j=B^{-1}(0)}^{K} \big(\mathrm{d}C_j^+ + \mathrm{d}C_j^- \big) \\
& \qquad + \big(A^{-1}(1) - ra_{t-} \big)\, \mathrm{d}C^+_{A^{-1}(0)} + \sum_{j=A^{-1}(0)+1}^{K} \big(A^{-1}(0) - ra_{t-} \big)\, \mathrm{d}C_j^+ \\
& + \sum_{j=1}^{A^{-1}(0)-1} (j - ra_{t-})\, \mathrm{d}L_j^- + \sum_{j=A^{-1}(0)}^{K} \big(A^{-1}(0) - ra_{t-} \big)\, \mathrm{d}L_j^- \\
& + \big(B^{-1}(1) - ra_t \big)\, \mathrm{d}C^-_{B^{-1}(0)} + \big(A^{-1}(0) - ra_{t-} \big) \sum_{j=A^{-1}(0)+1}^{K} \mathrm{d}C_j^- \\
& + \Big[\Big((1 - lb_t) + lb_t \mathbb{1}_{\{nb_{t-}>1\}} \Big) \Big(B^{-1}(1) - ra_{t-} \Big) + lb_t \mathbb{1}_{\{nb_{t-}=1\}} \big(B^{-1}(0) - ra_{t-} \big) \Big]\, \mathrm{d}M_t^- \\
& + \big(A^{-1}(1) - ra_{t-} \big)\, \mathrm{d}M_t^+
\end{aligned}
$$

We remind that rb_t is the number of ticks between the market maker's order and the best sell order in the order book.

Dynamics of rb:

$$
\begin{aligned}
\mathrm{d}\, rb_t = lb_t \Bigg[& \mathbf{1}_{\{nb=1\}} \Big(B^{-1}(0) - rb \Big)\, \mathrm{d}M_t^- \\
& + (1 - la_t) \sum_{i=1}^{rb_{t-}-1} \Big(i - rb_{t-} \Big)\, \mathrm{d}L_i^- + la_t \sum_{i=1}^{ra_{t-}-(A^{-1}(0)-rb_{t-})-1} \Big(i + A^{-1}(0) - ra_{t-} - rb_{t-} \Big)\, \mathrm{d}L_i^- \\
& + \sum_{i=1}^{rb_{t-}-1} \big(i - rb_{t-} \big)\, \mathrm{d}L_i^+ + \big(A^{-1}(1) - A^{-1}(0) \big)\, \mathrm{d}C^+_{rb_{t-}} \\
& \Big(\Big((1 - la_t) + la_t \mathbb{1}_{\{na_{t-}>1\}} \Big) \Big[A^{-1}(1) - A^{-1}(0) \Big] \Big)\, \mathrm{d}M_t^+ \Bigg] \\
+ (l - lb_t) \Bigg[& la_t \Bigg[\sum_{j=1}^{ra_{t-}-1} \Big(j + A^{-1}(0) - ra_{t-} - rb_{t-} \Big)\, \mathrm{d}L_j^- + \sum_{j=ra_{t-}}^{K} \big(B^{-1}(0) - rb_{t-} \big)\, \mathrm{d}L_j^- \\
& \big(B^{-1}(1) - rb_{t-} \big)\, \mathrm{d}C^-_{ra_{t-}} + \sum_{j=ra_{t-}+1}^{K} \big(B^{-1}(0) - rb_{t-} \big)\, \mathrm{d}C_j^- \Bigg] \\
& + (1 - la_t) \Bigg[\sum_{j=1}^{A^{-1}(0)-1} \Big(j - rb_{t-} \Big)\, \mathrm{d}L_j^- + \sum_{j=A^{-1}(0)}^{K} \big(B^{-1}(0) - rb_{t-} \big)\, \mathrm{d}L_j^- \\
& + \big(B^{-1}(1) - rb_{t-} \big)\, \mathrm{d}C^-_{B^{-1}(0)} + \sum_{j=B^{-1}(0)+1}^{K} \big(B^{-1}(0) - rb_{t-} \big)\, \mathrm{d}C_j^- \\
& + \sum_{j=1}^{B^{-1}(0)-1} (j - rb_{t-})\, \mathrm{d}L_j^+ + \sum_{j=B^{-1}(0)}^{K} \big(B^{-1}(0) - rb_{t-} \big)\, \mathrm{d}L_j^+ \\
& + \big(A^{-1}(1) - rb_t \big)\, \mathrm{d}C^+_{A^{-1}(0)} + \big(B^{-1}(0) - rb_{t-} \big) \sum_{j=B^{-1}(0)+1}^{K} \mathrm{d}C_j^+ \\
& + \Bigg[\Big((1 - la_t) + la_t \mathbb{1}_{\{na_{t-}>1\}} \Big) \big(A^{-1}(1) - rb_{t-} \Big) + la_t \mathbb{1}_{\{na_{t-}=1\}} \big(A^{-1}(0) - rb_{t-} \big) \Bigg]\, \mathrm{d}M_t^+ \\
& + \big(B^{-1}(1) - rb_{t-} \big)\, \mathrm{d}M_t^-
\end{aligned}
$$

3.C Proof of Theorem 3.4.1 and Corollary 3.4.1

We divided the proofs of Theorem 3.4.1 and Corollary 3.4.1 into several Lemmas that we state and prove now.

Lemma 3.C.1 aims to bound the projection error. It relies on [GKKW02], see p.93, as well as Zador's theorem, stated in section 3.D for sake of completeness.

Lemma 3.C.1. *Assume $d \geq 3$, and take $K = M^{d+2}$ points for the optimal quantization of ε_n, then it holds under **(Hμ)** and **(HF)**, as $M \to +\infty$,*

$$
\varepsilon_n^{proj} \quad = \quad \mathcal{O}\left(\frac{1}{M^{1/d}} \right), \tag{3.C.1}
$$

where we remind that $\varepsilon_n^{proj} := \sup_{a \in A} \| \mathrm{Proj}_{n+1} \left(F(X_n, a, \hat{\varepsilon}_n) \right) - F(X_n, a, \hat{\varepsilon}_n) \|_2$ stands for the average projection error.

Proof. Let us take $\eta > 0$, and observe that

$$\mathbb{P}\left(\left|\mathrm{Proj}_{n+1}\left[F(X_n,a,\hat{\varepsilon}_{n+1})\right]-F(X_n,a,\hat{\varepsilon}_{n+1})\right|^2>\eta\right)=\mathbb{E}\left[\prod_{m=1}^{M}\mathbb{E}\left[\mathbf{1}_{\left|X_{n+1}^{t,(m)}-F(X_n,a,\hat{\varepsilon}_{n+1})\right|>\sqrt{\eta}}\middle|X_n,\hat{\varepsilon}_{n+1}\right]\right]$$

$$=\mathbb{E}\left[\left(1-\mu\left[B\left(F(X_n,a,\hat{\varepsilon}_{n+1}),\sqrt{\eta}\right)\right]\right)^M\right],$$

where for all $x \in E$ and $\eta > 0$, $B(x,\eta)$ denote the ball of center x and radius η. Since $x \mapsto (1-x)^M$ is M-Lipschitz, we get by application of Zador's theorem:

$$\begin{aligned}&\mathbb{P}\left(\left|\mathrm{Proj}_{n+1}\left[F(X_n,a,\hat{\varepsilon}_{n+1})\right]-F(X_n,a,\hat{\varepsilon}_{n+1})\right|^2>\eta\right)\\&\qquad\le M[F]_L[\mu]_L\left\|\hat{\varepsilon}_{n+1}-\varepsilon_{n+1}\right\|_2+\mathbb{E}\left[\left(1-\mu\left(B\left(F(X_n,a,\varepsilon_{n+1}),\sqrt{\eta}\right)\right)\right)^M\right]\\&\qquad=\frac{M[F]_L[\mu]_L}{K^{1/d}}+\mathbb{E}\left[\left(1-\mu\left(B\left(F(X_n,a,\varepsilon_{n+1}),\sqrt{\eta}\right)\right)\right)^M\right]+\mathcal{O}\left(\frac{M}{K^{1/d}}\right),\end{aligned}$$

as the number of points for the quantization of the exogenous noise K goes to $+\infty$, and where M stands for the size of the grids Γ_n.

Let us introduce $A_1,...,A_{N(\eta)}$, a cubic partition of $\mathrm{Supp}(\mu)$, which is bounded under **(Hμ)**, such that for all $j = 1,\dots,N(\eta)$, A_j has diameter η. Also, Notice that there exists $c > 0$, which only depends on $\mathrm{Supp}(\mu)$, such as

$$N(\eta)\le\frac{c}{\eta^d}. \tag{3.C.2}$$

If $x \in A_j$, then $A_j \subset B(x,\eta)$, therefore:

$$\begin{aligned}\mathbb{E}\left[\left(1-\mu\left(B(X_n,\eta)\right)\right)^M\right]&=\sum_{j=1}^{N(\eta)}\int_{A_j}\left(1-\mu(B(x,\eta))\right)^M\mu(dx)\\&\le\sum_{j=1}^{N(\eta)}\int_{A_j}\left(1-\mu(A_j)\right)^M\mu(dx).\end{aligned} \tag{3.C.3}$$

Also notice that:

$$\sum_{j=1}^{N(\eta)}\mu(A_j)\left(1-\mu(A_j)\right)^M\le\sum_{j=1}^{N(\eta)}\max_z z(1-z)^M\le\frac{e^{-1}N(\eta)}{M}. \tag{3.C.4}$$

Combining (3.C.3) and (3.C.4) leads to

$$\mathbb{E}\left[\left(1-\mu\left(B(X_n,\eta)\right)\right)^M\right]\le\frac{e^{-1}N(\eta)}{M}. \tag{3.C.5}$$

Let $L = 2\|\mu\|_\infty$ stands for the diameter of the support of μ. We then get, as $M \to +\infty$,

$$\begin{aligned}
&\mathbb{E}\Big[\,\big|\mathrm{Proj}_{n+1}\big[F(X_n,a,\hat{\varepsilon}_{n+1})\big] - F(X_n,a,\hat{\varepsilon}_{n+1})\big|^2\,\Big] \\
&\qquad = \int_0^\infty \mathbb{P}\Big(\,\big|\mathrm{Proj}_{n+1}\big[F(X_n,a,\hat{\varepsilon}_{n+1})\big] - F(X_n,a,\hat{\varepsilon}_{n+1})\big|^2 > \eta\Big)\,\mathrm{d}\eta \\
&\qquad \le \int_0^{L^2} \frac{M[F]_L[\mu]_L}{K^{2/d}} + \mathbb{P}\Big(\big|\mathrm{Proj}_{n+1}\big[F(X_n,a,\hat{\varepsilon}_{n+1})\big] - F(X_n,a,\epsilon_{n+1})\big| > \sqrt{\eta}\Big)\,\mathrm{d}\eta \\
&\qquad = \int_0^{L^2} \min\left(1, \frac{e^{-1}N(\sqrt{\eta})}{M}\right)\mathrm{d}\eta + \mathcal{O}\left(\frac{M}{K^{1/d}}\right) \\
&\qquad = \int_0^{L^2} \min\left(1, \frac{c\eta^{-d/2}}{eM}\right)\mathrm{d}\eta + \mathcal{O}\left(\frac{M}{K^{1/d}}\right) \\
&\qquad = \int_0^{(c/(eM))^{(2/d)}} 1 d\eta + \int_{(c/(eM))^{(2/d)}}^{L^2} \frac{c\eta^{-d/2}}{eM}\,\mathrm{d}\eta + \mathcal{O}\left(\frac{M}{K^{1/d}}\right) \\
&\qquad = \frac{\tilde{c}^2}{M^{2/d}} + \mathcal{O}\left(\frac{M}{K^{1/d}}\right), \qquad (3.C.6)
\end{aligned}$$

where $\tilde{c}$ is defined as $\tilde{c} := \sqrt{\frac{d}{d-2}}\left(\frac{c}{e}\right)^{1/d}$, and where we used (3.C.5) and (3.C.2) to go from the second to the third line. It remains to take $K = M^{d+1}$ points for the optimal quantization of the exogenous noise, and then take square root of equality (3.C.6), in order to derive (3.C.1). □

Lemma 3.C.2. *Assume $d \ge 3$, take $K = M^{d+2}$ points for the optimal quantization of ε_n, and let $x \in E$. Then it holds under **(Hμ)** and **(HF)**, as $M \to +\infty$:*

$$\varepsilon_n^{proj}(x) \quad = \quad \mathcal{O}\left(\frac{1}{M^{1/d}}\right),$$

where $\varepsilon_n^{proj}(x)$, defined as $\varepsilon_n^{proj}(x) := \sup_{a\in A} \|\mathrm{Proj}_{n+1}\left(F(x,a,\hat{\varepsilon}_n)\right) - F(x,a,\hat{\varepsilon}_n)\|_2$, stands for the later-projection error at state x.

Proof. Following the same steps as those used to prove Lemma 3.C.1, we show that:

$$\begin{aligned}
&\mathbb{P}\left(\big|\mathrm{Proj}_{n+1}\big[F(x,a,\hat{\varepsilon}_{n+1})\big] - F(x,a,\hat{\varepsilon}_{n+1})\big|^2 > \eta\right) \\
&\qquad = \frac{M[F]_L[\mu]_L}{K^{1/d}} + \mathbb{E}\left[\Big(1 - \mu\big(B\big(F(x,a,\varepsilon_{n+1}), \sqrt{\eta}\big)\big)\Big)^M\right] + \mathcal{O}\left(\frac{M}{K^{1/d}}\right),
\end{aligned}$$

as $K \to +\infty$, and moreover,

$$\mathbb{E}\left[\Big(1 - \mu\big(B\big(F(x,a,\varepsilon_{n+1}), \sqrt{\eta}\big)\big)\Big)^M\right] \le \frac{e^{-1}N(\eta)}{M},$$

holds, which is enough to complete the proof of Lemma 3.C.2. □

Lemma 3.C.3. *Under **(HF)**, for $n = 0,\dots,N$ there exists constant $\left[\hat{V}_n^Q\right]_L > 0$ such that for $x, x' \in E$, it holds as $M \to \infty$:*

$$\left|\hat{V}_n^Q(x) - \hat{V}_n^Q(x')\right| \;\le\; \left[\hat{V}_n^Q\right]_L |x - x'| + \mathcal{O}\left(\frac{1}{M^{1/d}}\right). \qquad (3.C.7)$$

Moreover, following bounds holds on $\left[\hat{V}_n^Q\right]_L$, *for* $n = 0, \dots, N$:

$$\begin{cases} \left[\hat{V}_N^Q\right]_L & \leq [g]_L \\ \left[\hat{V}_n^Q\right]_L & \leq [f]_L + [F]_L \left[\hat{V}_{n+1}^Q\right]_L, \qquad \text{for } n = 0, ..., N-1. \end{cases} \tag{3.C.8}$$

Proof. Let us show that by induction that $\hat{V}_N^Q$ is Lipschitz. First, notice that (3.C.7) holds at terminal time $n = N$, if one define $\left[\hat{V}_N^Q\right]_L$ as $\left[\hat{V}_N^Q\right]_L = [g]_L$. Let us take $x, x' \in E$. Assume $\left|\hat{V}_{n+1}^Q(x) - \hat{V}_{n+1}^Q(x')\right| \leq \left[\hat{V}_{n+1}^Q\right]_L |x - x'| + \mathcal{O}\left(\frac{1}{M^{1/d}}\right)$ holds for some $n = 0, \dots, N-1$. Let us show that

$$\left|\hat{V}_n^Q(x) - \hat{V}_n^Q(x')\right| \leq \left[\hat{V}_n^Q\right]_L |x - x'| + \mathcal{O}\left(\frac{1}{M^{1/d}}\right),$$

where $\left[\hat{V}_n^Q\right]_L$ is defined in (3.C.8). Notice that, by the dynamic programming principle and the triangular inequality, it holds:

$$\begin{aligned} |\hat{V}_n^Q(x) - \hat{V}_n^Q(x')| &\leq [f]_L |x - x'| \\ &\quad + \sup_a \mathbb{E}_n^a \left[\left|\hat{V}_{n+1}^Q\big(\text{Proj}_{n+1}\left(F(x, a, \hat{\varepsilon}_{n+1})\right)\big) - \hat{V}_{n+1}^Q\big(\text{Proj}_{n+1}\left(F(x', a, \hat{\varepsilon}_{n+1})\right)\big)\right|\right] \\ &\leq [f]_L |x - x'| + \left[\hat{V}_{n+1}^Q\right]_L \sup_a \mathbb{E}\left[\left|\text{Proj}_{n+1}\left(F(x, a, \hat{\varepsilon}_{n+1})\right) - F(x, a, \hat{\varepsilon}_{n+1})\right|\right] \\ &\quad + \mathcal{O}\left(\frac{1}{M^{1/d}}\right) \\ &\leq \left([f]_L + \left[\hat{V}_{n+1}^Q\right]_L [F]_L\right) |x - x'| + \mathcal{O}\left(\frac{1}{M^{1/d}}\right) \\ &\leq \left[\hat{V}_n^Q\right]_L |x - x'| + \mathcal{O}\left(\frac{1}{M^{1/d}}\right), \end{aligned}$$

which completes the proof of (3.C.7). □

We now proceed to the proof of Theorem 3.4.1.

Proof. (of Theorem 3.4.1) Combining inequality $|u_1 + u_2 + u_3|^2 \leq 3\left(|u_1|^2 + |u_2|^2 + |u_3|^2\right)$ that holds for all $u_1, u_2, u_3 \in \mathbb{R}$ with inequality $\left|\sup_{i \in I} a_i - \sup_{i \in I} b_i\right| \leq \sup_{i \in I} |a_i - b_i|$ that holds for all families $(a_i)_{i \in I}$ and $(a_i)_{i \in I}$ of reals, and all subset I of $\mathbb{R}$, we have:

$$\begin{aligned} \|\hat{V}_n^Q(X_n) - V_n(X_n)\|_2^2 \leq 3\, \mathbb{E}\Bigg[&\sup_{a \in A} \mathbb{E}_{n, X_n} \left|\hat{V}_{n+1}^Q\left(\text{Proj}_{n+1}\left(F(X_n, a, \hat{\varepsilon}_{n+1})\right)\right) - \hat{V}_{n+1}^Q(F(X_n, a, \hat{\varepsilon}_{n+1}))\right|^2 \\ &+ \sup_{a \in A} \mathbb{E}_{n, X_n} \left|\hat{V}_{n+1}^Q(F(X_n, a, \hat{\varepsilon}_{n+1})) - \hat{V}_{n+1}^Q(F(X_n, a, \varepsilon_{n+1}))\right|^2 \\ &+ \sup_{a \in A} \mathbb{E}_{n, X_n} \left|\hat{V}_{n+1}^Q(F(X_n, a, \varepsilon_{n+1})) - V_{n+1}(F(X_n, a, \varepsilon_{n+1}))\right|^2 \Bigg] \end{aligned}$$

where $\mathbb{E}_{n, X_n}$ stands for the expectation conditioned by the state X_n at time n. It holds as $M \to +\infty$,

using Lemma 3.C.3:

$$\|\hat{V}_n^Q(X_n) - V_n(X_n)\|_2^2 \leq 3\ \left[\hat{V}_n^Q\right]_L \mathbb{E}\Bigg[\sup_a \mathbb{E}_{n,X_n}\left[|\mathrm{Proj}_{n+1}\left(F(X_n,a,\hat{\varepsilon}_{n+1})\right) - F(X_n,a,\hat{\varepsilon}_{n+1})|^2\right] + \sup_a \mathbb{E}_{n,X_n}\left[|F(X_n,a,\hat{\varepsilon}_{n+1}) - F(X_n,a,\varepsilon_{n+1})|^2\right]\Bigg] + 3\ \|r\|_\infty \mathbb{E}\Big[|\hat{V}_{n+1}^Q(X_{n+1})) - V_{n+1}(X_{n+1}))|^2\Big] + \left(\frac{1}{M^{1/d}}\right) \tag{3.C.9}$$

Under **(HF)**, (3.C.9) can then be rewritten as:

$$\begin{aligned}\|\hat{V}_n^Q(X_n) - V_n(X_n)\|_2^2 \ &\leq\ 3\left[\hat{V}_n^Q\right]_L \left([F]_L^2(\epsilon_n^Q)^2 + (\epsilon_n^{proj})^2\right) \\ &\quad + 3\|r\|_\infty \|\hat{V}_{n+1}^Q(X_{n+1}) - V_{n+1}(X_{n+1})\|_2^2 + \left(\frac{1}{M^{1/d}}\right).\end{aligned}$$

(3.4.5) then follows by induction, which completes the proof of Theorem 3.4.1. □

Proof. (of Corollary 3.4.1)
Corollary 3.4.1 is straightforward by plugging the bound for the projection error provided by Lemma 3.C.1 and the one of the quantization error provided by the Zador's Theorem into (3.4.5). □

3.D Zador's Theorem

Theorem 3.D.1 (Zador's theorem)**.** *Let us take $n = 0, \dots, N$, and denote by K the number of points for the quantization of the exogenous noise ε_n.*
Assume that $\mathbb{E}\left[|\varepsilon_n|^{2+\eta}\right] < +\infty$ for some $\eta > 0$. Then, there exists a universal constant $C > 0$ such that:

$$\lim_{M \to +\infty} \left(M^{\frac{1}{d}} \|\hat{\varepsilon}_n - \varepsilon_n\|_2\right) = C$$

Proof. We refer to [GL00] for a proof of Theorem 3.D.1. □

Chapter 4

A class of finite-dimensional numerically solvable McKean-Vlasov control problems[1]

Abstract We address a class of McKean-Vlasov (MKV) control problems with common noise, called polynomial conditional MKV, and extending the known class of linear quadratic stochastic MKV control problems. We show how this polynomial class can be reduced by suitable Markov embedding to finite-dimensional stochastic control problems, and provide a discussion and comparison of three probabilistic numerical methods for solving the reduced control problem: quantization, regression by control randomization, and regress-later methods. Our numerical results are illustrated on various examples from portfolio selection and liquidation under drift uncertainty, and a model of interbank systemic risk with partial observation.

Keywords: McKean-Vlasov control, polynomial class, quantization, regress-later, control randomization.

Contents

[1]This Chapter is based on a paper written in collaboration with Alessandro Balata, Mathieu Laurière, Huyên Pham, and Isaque Pimentel. This paper is published in ESAIM Proceedings and surveys, 65, (2019)

4.1 Introduction

The optimal control of McKean-Vlasov (also called mean-field) dynamics is a rather new topic in the area of stochastic control and applied probability, which has been knowing a surge of interest with the emergence of the mean-field game theory. It is motivated on the one hand by the asymptotic formulation of cooperative equilibrium for a large population of particles (players) in mean-field interaction, and on the other hand from control problems with cost functional involving nonlinear functional of the law of the state process (e.g., the mean-variance portfolio selection problem or risk measure in finance).

In this paper, we are interested in McKean-Vlasov (MKV) control problems under partial observation and common noise, whose formulation is described as follows. On a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ equipped with two independent Brownian motions B and W^0, let us consider the controlled stochastic MKV dynamics in $\mathbb{R}^n$:

$$dX_s = b(X_s, \mathbb{P}^{W^0}_{X_s}, \alpha_s)ds + \sigma(X_s, \mathbb{P}^{W^0}_{X_s}, \alpha_s)dB_s + \sigma_0(X_s, \mathbb{P}^{W^0}_{X_s}, \alpha_s)dW^0_s, \quad X_0 = x_0 \in \mathbb{R}^n, \tag{4.1.1}$$

where $\mathbb{P}^{W^0}_{X_s}$ denotes the conditional distribution of X_s given W^0 (or equivalently given $\mathcal{F}^0_s$ where $\mathbb{F}^0 = (\mathcal{F}^0_t)_t$ is the natural filtration generated by W^0), and the control α is $\mathbb{F}^0$-progressive valued in some Polish space A. This measurability condition on the control means that the controller has a partial observation of the state, in the sense that she can only observe the common noise. We make the standard Lipschitz condition on the coefficients $b(x, \mu, a)$, $\sigma(x, \mu, a)$, $\sigma_0(x, \mu, a)$ with respect to (x, μ) in $\mathbb{R}^n \times \mathcal{P}_2(\mathbb{R}^n)$, uniformly in $a \in A$, where $\mathcal{P}_2(\mathbb{R}^n)$ is the set of all probability measures on $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n))$ with a finite second-order moment, endowed with the 2-Wasserstein metric $\mathcal{W}_2$. This ensures the well-posedness of the controlled MKV stochastic differential equation (SDE) (4.1.1). The cost functional over a finite horizon T associated to the stochastic MKV equation (4.1.1) (sometimes called conditional MKV equation) for a control process α, is

$$J(\alpha) = \mathbb{E}\Big[\int_0^T f(X_t, \mathbb{P}^{W^0}_{X_t}, \alpha_t)dt + g(X_T, \mathbb{P}^{W^0}_{X_T})\Big],$$

and the objective is to maximize over an admissible set $\mathcal{A}$ of control processes the cost functional:

$$V_0 = \sup_{\alpha \in \mathcal{A}} J(\alpha). \tag{4.1.2}$$

The set $\mathcal{A}$ of admissible controls usually requires some integrability conditions depending on the growth conditions on f, g, in order to ensure that $J(\alpha)$ is well-defined for $\alpha \in \mathcal{A}$ (more details will be given in the examples, see Section 4.4). Notice that classical partial observation control problem (without MKV dependence on the coefficients) arises as a particular case of (4.1.1)-(4.1.2). We refer to the introduction in [PW17] for the details.

Let us recall from [PW17] the dynamic programming equation associated to the conditional MKV control problem (4.1.2). We start by defining a suitable dynamic version of this problem. Let us consider $\mathcal{F}_0$ a sub σ-algebra of $\mathcal{F}$ independent of B, W^0. It is assumed w.l.o.g. that $\mathcal{F}_0$ is rich enough in the sense that $\mathcal{P}_2(\mathbb{R}^n) = \{\mathcal{L}(\xi) : \xi \in L^2(\mathcal{F}_0; \mathbb{R}^n)\}$, where $\mathcal{L}(\xi)$ denotes the law of ξ. Given a control $\alpha \in \mathcal{A}$, we consider the dynamic version of (4.1.1) starting from $\xi \in L^2(\mathcal{F}_0; \mathbb{R}^n)$ at time $t \in [0, T]$, and written as:

$$\begin{aligned} X^{t,\xi,\alpha}_s = \xi &+ \int_t^s b(X^{t,\xi,\alpha}_u, \mathbb{P}^{W^0}_{X^{t,\xi,\alpha}_u}, \alpha_u)du + \int_t^s \sigma(X^{t,\xi,\alpha}_u, \mathbb{P}^{W^0}_{X^{t,\xi,\alpha}_u}, \alpha_u)dB_u \\ &+ \int_t^s \sigma_0(X^{t,\xi,\alpha}_u, \mathbb{P}^{W^0}_{X^{t,\xi,\alpha}_u}, \alpha_u)dW^0_u\ ,\ t \le s \le T. \end{aligned}$$

Let us then define the dynamic cost functional:

$$J(t, \xi, \alpha) = \mathbb{E}\Big[\int_t^T f(X^{t,\xi,\alpha}_s, \mathbb{P}^{W^0}_{X^{t,\xi,\alpha}_s}, \alpha_s)ds + g(X^{t,\xi,\alpha}_T, \mathbb{P}^{W^0}_{X^{t,\xi,\alpha}_T})\Big],$$

for $(t,\xi) \in [0,T] \times L^2(\mathcal{F}_0;\mathbb{R}^n), \alpha \in \mathcal{A}$, and notice by the law of conditional expectations, and as α is $\mathbb{F}^0$-progressive that

$$J(t,\xi,\alpha) = \mathbb{E}\Big[\int_t^T \hat{f}(\mathbb{P}^{W^0}_{X_s^{t,\xi,\alpha}}, \alpha_s)ds + \hat{g}(\mathbb{P}^{W^0}_{X_s^{t,\xi,\alpha}})\Big],$$

where $\hat{f} : \mathcal{P}_2(\mathbb{R}^n) \times A \to \mathbb{R}$, $\hat{g} : \mathcal{P}_2(\mathbb{R}^n) \to \mathbb{R}$ are defined by

$$\begin{aligned}\hat{f}(\mu,a) &= \mu(f(\cdot,\mu,a)) = \int_{\mathbb{R}^n} f(x,\mu,a)\mu(dx),\\ \hat{g}(\mu) &= \mu(g(\cdot,\mu)) = \int_{\mathbb{R}^n} g(x,\mu)\mu(dx).\end{aligned}$$

Moreover, notice that the conditional law of $X_s^{t,\xi,\alpha}$ given W^0 depends on ξ only through its law $\mathcal{L}(\xi)$, and we can then define for $\alpha \in \mathcal{A}$:

$$\rho_s^{t,\mu,\alpha} = \mathbb{P}^{W^0}_{X_s^{t,\xi,\alpha}}, \quad \text{for } t \le s,\ \mu = \mathcal{L}(\xi) \in \mathcal{P}_2(\mathbb{R}^n).$$

Therefore, the dynamic cost functional $J(t,\xi,\alpha)$ depends on $\xi \in L^2(\mathcal{F}_0;\mathbb{R}^n)$ only through its law $\mathcal{L}(\xi)$, and by an abuse of notation, we write $J(t,\mu,\alpha) = J(t,\xi,\alpha)$ when $\mu = \mathcal{L}(\xi)$. We then consider the value function for the conditional MKV control problem (4.1.2), defined on $[0,T] \times \mathcal{P}_2(\mathbb{R}^n)$ by

$$v(t,\mu) = \sup_{\alpha\in\mathcal{A}} J(t,\mu,\alpha) = \sup_{\alpha\in\mathcal{A}} \mathbb{E}\Big[\int_t^T \hat{f}(\rho_s^{t,\mu,\alpha},\alpha_s)ds + \hat{g}(\rho_T^{t,\mu,\alpha})\Big],$$

and notice that at time $t = 0$, when $\xi = x_0$ is a constant, then $V_0 = v(0,\delta_{x_0})$.

It is shown in [PW17] that dynamic programming principle (DPP) for the conditional MKV control problem (4.1) holds: for $(t,\mu) \in [0,T] \times \mathcal{P}_2(\mathbb{R}^n)$,

$$v(t,\mu) = \sup_{\alpha\in\mathcal{A}} \mathbb{E}\Big[\int_t^\theta \hat{f}(\rho_s^{t,\mu,\alpha},\alpha_s)ds + v(\theta,\rho_\theta^{t,\mu,\alpha})\Big],$$

for any $\mathbb{F}^0$-stopping time θ valued in $[t,T]$. Next, by relying on the notion of differentiability with respect to probability measures introduced by P. L. Lions [Lio12] (see also the lecture notes [Car10]) and the chain rule (Itô's formula) along flow of probability measures (see [BLPR17], [CCD14]), we derive the HJB equation for v:

$$\begin{cases} \partial_t v + \sup_{a\in A}\Big[\hat{f}(\mu,a) + \mu\big({}^a v(t,\mu)\big) + \mu\otimes\mu\big(\mathbb{M}^a v(t,\mu)\big)\Big] = 0, \quad (t,\mu) \in [0,T)\times\mathcal{P}_2(\mathbb{R}^n),\\ v(T,\mu) = \hat{g}(\mu), \quad \mu \in \mathcal{P}_2(\mathbb{R}^n),\end{cases}$$

where for $\phi \in \mathcal{C}_b^2(\mathcal{P}_2(\mathbb{R}^n))$, $a \in A$, and $\mu \in \mathcal{P}_2(\mathbb{R}^n)$, ${}^a\phi(\mu)$ is the function $\mathbb{R}^n \to \mathbb{R}$ defined by

$${}^a\phi(\mu)(x) = \partial_\mu\phi(\mu)(x).b(x,\mu,a) + \frac{1}{2}\mathrm{tr}\big(\partial_x\partial_\mu\phi(\mu)(x)(\sigma\sigma^\intercal + \sigma_0\sigma_0^\intercal)(x,\mu,a)\big),$$

and $\mathbb{M}^a\phi(\mu)$ is the function $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ defined by

$$\mathbb{M}^a\phi(\mu)(x,x') = \frac{1}{2}\mathrm{tr}\big(\partial_\mu^2\phi(\mu)(x,x')\sigma_0(x,\mu,a)\sigma_0^\intercal(x',\mu,a)\big).$$

The HJB equation (4.1) is a fully nonlinear partial differential equation (PDE) in the infinite-dimensional Wasserstein space. In general, this PDE does not have an explicit solution except in the notable important class of linear-quadratic MKV control problem. Numerical resolution for MKV control problem or equivalently for the associated HJB equation is a challenging problem due to the nonlinearity of the optimization problem and the infinite-dimensional feature of the Wasserstein space. In this work, our purpose is to investigate a class of MKV control problems which can be reduced to finite-dimensional problems in view of numerical resolution.

4.2 Polynomial McKean-Vlasov control problem

4.2.1 Main assumptions

We make two kinds of assumptions on the coefficients of the model: one on the dependence on x and the other on the dependence on μ.

Assumptions: dependence on x: we consider a class of models where the coefficients of the MKV equation are linear w.r.t. the state variable X, i.e., they are in the form

$$\left\{\begin{array}{rcl} b(x,\mu,a) & = & b_0(\mu,a)+b_1(\mu,a)x, \\ \sigma(x,\mu,a) & = & \vartheta_0(\mu,a)+\vartheta_1(\mu,a)x, \\ \sigma_0(x,\mu,a) & = & \gamma_0(\mu,a)+\gamma_1(\mu,a)x, \end{array}\right.$$

while the running and terminal cost functions are polynomial in the state variable in the sense that they are in the form (for simplicity we present here the one-dimensional case $n=1$)

$$\begin{array}{rcl} f(x,\mu,a) & = & f_0(\mu,a)+\sum_{k=1}^{p} f_k(\mu,a)x^k, \\ g(x,\mu) & = & g_0(\mu)+\sum_{k=1}^{p} g_k(\mu)x^k, \end{array}$$

for some integer $p \geq 1$.

Assumptions: dependence on μ: we assume that all the coefficients depend on μ through its first p moments, i.e., they are in the form

$$\left\{\begin{array}{rclrcl} b_0(\mu,a) & = & \bar{b}_0(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a), & b_1(\mu,a) & = & \bar{b}_1(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a) \\ \vartheta_0(\mu,a) & = & \bar{\vartheta}_0(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a), & \vartheta_1(\mu,a) & = & \bar{\vartheta}_1(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a) \\ \gamma_0(\mu,a) & = & \bar{\gamma}_0(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a), & \gamma_1(\mu,a) & = & \bar{\gamma}_1(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a) \\ f_k(\mu,a) & = & \bar{f}_k(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a), & g_k(\mu) & = & \bar{g}_k(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p), \quad k=0,\ldots,p, \end{array}\right.$$

where, given $\mu \in \mathcal{P}_p(\mathbb{R})$, we denote by

$$\bar{\mu}_k \quad = \quad \int x^k \mu(dx), \quad k=1,\ldots,p.$$

We assume that the coefficients $\bar{b}_0, \bar{b}_1, \bar{\vartheta}_0, \bar{\vartheta}_1, \bar{\gamma}_0, \bar{\gamma}_1$ are Lipschitz w.r.t. the p first arguments uniformly w.r.t. the control argument $a \in A$. This condition will ensure existence and uniqueness of a solution to the finite-dimensional MKV SDE defined later in (4.2.1).

Notice that in this case, the functions $\hat{f}$ and $\hat{g}$ defined in (4.1)-(4.1) are given by

$$\begin{array}{rcl} \hat{f}(\mu,a) & = & \bar{f}_0(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a)+\sum_{k=1}^{p}\bar{f}_k(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a)\bar{\mu}_k \\ & =: & \bar{f}(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p,a), \\ \hat{g}(\mu) & = & \bar{g}_0(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p)+\sum_{k=1}^{p}\bar{g}_k(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p)\bar{\mu}_k \\ & =: & \bar{g}(\bar{\mu}_1,\bar{\mu}_2,\ldots,\bar{\mu}_p). \end{array}$$

Remark 4.2.1. *In the multidimensional case, we should consider a class of multi-polynomial functions f and g of degree p in the form*

$$f(x,\mu,a) \;=\; \sum_{|\mathbf{k}|=0}^{p} f_{\mathbf{k}}\left((\mu^{\mathbf{k}'})_{|\mathbf{k}'|\leq p},a\right)x^{\mathbf{k}}, \qquad g(x,\mu) \;=\; \sum_{|\mathbf{k}|=0}^{p} g_{\mathbf{k}}\left((\mu^{\mathbf{k}'})_{|\mathbf{k}'|\leq p}\right)x^{\mathbf{k}},$$

where we use multi-index notations $\mathbf{k} = (k_1, \ldots, k_n) \in \mathbb{N}^n$, $|\mathbf{k}| = k_1 + \ldots + k_n$, $x^{\mathbf{k}} = x_1^{k_1} \ldots x_n^{k_n}$ *for* $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ *and*

$$\mu^{\mathbf{k}} \quad = \quad \int_{\mathbb{R}^n} x^{\mathbf{k}} \mu(dx).$$

4.2.2 Markovian embedding

Given the controlled process $X = X^\alpha$ solution to the stochastic MKV dynamics (4.1.1), denote by

$$Y_t^{(k)} \quad = \quad \mathbb{E}[X_t^k | W^0], \quad k = 1, \ldots, p.$$

To alleviate the notations, we assume that $n = 1$ (otherwise multi-indices should be used). From the linear/polynomial assumptions (4.2.1)-(4.2.1), by Itô's formula and taking conditional expectations, we can derive the dynamics of $(Y^{(1)}, Y^{(2)}, \ldots, Y^{(p)})$ as

$$\begin{aligned} dY_t^{(k)} &= B_k(Y_t^{(1)}, Y_t^{(2)}, \ldots, Y_t^{(p)}, \alpha_t)dt + \Sigma_k(Y_t^{(1)}, Y_t^{(2)}, \ldots, Y_t^{(p)}, \alpha_t)dW_t^0, \\ Y_0^{(k)} &= x_0^k, \quad k = 1, \ldots, p, \end{aligned} \tag{4.2.1}$$

where, by convention $y_0 = 1$, $y_{-1} = 0$,

$$\begin{aligned} B_k(y_1, y_2, \ldots, y_p, a) &= k\bar{b}_0(y_1, \ldots, y_p, a)y_{k-1} + k\bar{b}_1(y_1, \ldots, y_p, a)y_k \\ &\quad + \frac{k(k-1)}{2}(\bar{\vartheta}_0(y_1, \ldots, y_p, a))^2 y_{k-2} + \frac{k(k-1)}{2}(\bar{\vartheta}_1(y_1, \ldots, y_p, a))^2 y_k \\ &\quad + k(k-1)\bar{\vartheta}_0(y_1, \ldots, y_p, a)\bar{\vartheta}_1(y_1, \ldots, y_p, a)y_{k-1} \\ &\quad + \frac{k(k-1)}{2}(\bar{\gamma}_0(y_1, \ldots, y_p, a))^2 y_{k-2} + \frac{k(k-1)}{2}(\bar{\gamma}_1(y_1, \ldots, y_p, a))^2 y_k \\ &\quad + k(k-1)\bar{\gamma}_0(y_1, \ldots, y_p, a)\bar{\gamma}_1(y_1, \ldots, y_p, a)y_{k-1}, \quad k = 1, \ldots, p, \\ \Sigma_k(y_1, y_2, \ldots, y_p, a) &= k\left(\bar{\gamma}_0(y_1, \ldots, y_p, a)y_{k-1} + \bar{\gamma}_1(y_1, \ldots, y_p, a)y_k\right), \quad k = 1, \ldots, p, \end{aligned}$$

while the cost functional is written as

$$J(\alpha) \quad = \quad \mathbb{E}\Big[\int_0^T \bar{f}(Y_t^{(1)}, Y_t^{(2)}, \ldots, Y_t^{(p)}, \alpha_t)dt + \bar{g}(Y_T^{(1)}, Y_T^{(2)}, \ldots, Y_T^{(p)})\Big].$$

The MKV control problem is then reduced in this polynomial framework into a finite-dimensional control problem with $\mathbb{F}^0$-adapted controlled variables $(Y^{(1)}, Y^{(2)}, \ldots, Y^{(p)})$. In the next section, we describe three probabilistic numerical methods for solving finite-dimensional stochastic control problems and will apply in section 4.4 each of these methods to three examples arising from polynomial MKV control problems under partial observation and common noise.

4.3 Numerical methods

In this section, we introduce our numerical methods for the resolution of the reduced problem (4.2.1)-(4.2.2).

Let us introduce the process Z^α, valued in $\mathbb{R}^d$, controlled by an adapted process α taking values in A, solution to

$$dZ_t^\alpha = b(Z_t^\alpha, \alpha_t)dt + \sigma_0(Z_t^\alpha, \alpha_t)dW_t^0, \quad Z_0^\alpha = z_0 \in \mathbb{R}^d,$$

and the performance measure

$$J(t, z, \alpha) = \mathbb{E}\left[\int_t^T f(Z_t^\alpha, \alpha_t)dt + g(Z_T^\alpha)\Big| Z_t^\alpha = z\right],$$

which assesses the average performance of the control.

Introduce now a time discretization $t_n = n\Delta t$, $n = 0, \dots, N$, $\Delta t = T/N$, and denote by $\mathcal{A}_{\Delta t}$ the space of discrete processes $(\alpha_{t_n})_{n=0}^{N-1}$ such that for all n, $n = 0, \dots, N-1$, α_{t_n} is $\mathcal{F}_{t_n}^0$-measurable.

We can write the Euler approximation of the SDE governing the process $Z = Z^\alpha$, with $\alpha \in \mathcal{A}_{\Delta t}$ (to alleviate notations, we sometimes omit the dependence on α when there is no ambiguity, and keep the same notation Z for the discrete and continuous process)

$$Z_{t_{n+1}} = Z_{t_n} + b(Z_{t_n}, \alpha_{t_n})\Delta t + \sigma_0(Z_{t_n}, \alpha_{t_n})\Delta W_{t_n}^0, \tag{4.3.1}$$

where $\Delta W_{t_n}^0 \sim \mathcal{N}(0, \Delta t)$ is an increment of W^0.

The discrete time approximation of $J(t_n, z, \alpha)$ is defined as:

$$J_{\Delta t}(t_n, z, \alpha) = \mathbb{E}\left[\sum_{k=n}^{N-1} f(Z_{t_k}, \alpha_{t_k})\Delta t + g(Z_{t_N}) \Big| Z_{t_n} = z\right],$$

where $\alpha \in \mathcal{A}_{\Delta t}$.

4.3.1 Value and Performance iteration

For $n = 0, \dots, N$, consider $V_{\Delta t}(t_n, z) = \sup_{\alpha \in \mathcal{A}_{\Delta t}} J_{\Delta t}(t_n, z, \alpha)$, the discrete time approximation of the value function at time t_n: $V\big(t_n, z\big) = \sup_{\alpha \in \mathcal{A}} J(t_n, z, \alpha)$. The dynamic programming principle states that $\big(V_{\Delta t}(t_n, \cdot)\big)_{0 \le n \le N}$ is solution to the Bellman equation:

$$\begin{cases} V_{\Delta t}(t_N, z) = g(z) \\ V_{\Delta t}(t_n, z) = \sup_{a \in A} \Big\{ f(z, a)\Delta t + \mathbb{E}_{n,z}^a \big[V_{\Delta t}(t_{n+1}, Z_{t_{n+1}})\big] \Big\}, \quad n = N-1, \dots, 0, \end{cases} \tag{4.3.2}$$

where $\mathbb{E}_{n,z}^a[\cdot]$ denotes the expectation conditioned on the event $\{Z_{t_n} = z\}$ and when using the control $\alpha_{t_n} = a$ at time t_n. Observe that for $n = 0, \dots, N-1$, the equation (4.3.2) provides a backward procedure to recursively compute the $V_{\Delta t}(t_n, \cdot)$ if we know how to analytically compute the conditional expectations $\mathbb{E}_{n,z}^a[V_{\Delta t}(t_{n+1}, Z_{t_{n+1}})]$ for all $z \in \mathbb{R}^d$ and all control $a \in A$. We refer to the procedure in (4.3.2) as value iteration.

An alternative approach to compute $V_{\Delta t}(t_n, \cdot)$, for $n = 0, \dots, N-1$, is to notice that once again by the dynamic programming principle, it holds that $\big(V_{\Delta t}(t_n, \cdot)\big)_{0 \le n \le N}$ is solution to the backward equation

$$\begin{cases} V_{\Delta t}(t_N, z) & = g(z) \\ V_{\Delta t}(t_n, z) & = \sup_{a \in A} \left\{ f(z, a)\Delta t + \mathbb{E}_{n,z}^a \left[\sum_{k=n+1}^{N-1} f\big(Z_{t_k}, \alpha_{t_k}^*(Z_{t_k})\big)\Delta t + g(Z_{t_N}) \right] \right\}, \\ & \qquad\qquad\qquad\qquad\qquad\qquad \text{for } n = N-1, \dots, 0, \end{cases} \tag{4.3.3}$$

where for $k = n+1, \dots, N-1$, the control $\alpha_{t_k}^*$ is the optimal control at time t_k defined as follows:

$$\alpha_{t_k}^*(z) = \operatorname*{argmax}_{a \in A} \left\{ f(z, a)\Delta t + \mathbb{E}_{k,z}^a \left[\sum_{\ell=k+1}^{N-1} f\big(Z_{t_\ell}^*, \alpha_{t_\ell}^*(Z_{t_\ell}^*)\big)\Delta t + g(Z_{t_N}^*) \right] \right\},$$

and where $\big(Z_{t_k}^*\big)_{n \le k \le N}$ is the process Z controlled by the following control α from time t_n to t_N:

$$\begin{cases} \alpha_{t_n} = a, \\ \alpha_{t_k} = \alpha_{t_k}^* \text{ for } n+1 \le k \le N-1. \end{cases}$$

For $n = 0, \dots, N-1$, the scheme (4.3.3) provides once again a backward procedure to compute $V_{\Delta t}(t_n, \cdot)$, assuming that we know how to analytically compute the conditional expectations

$$\mathbb{E}^a_{n,z}\left[\sum_{k=n+1}^{N-1} f\big(Z_{t_k}, \alpha^*_{t_k}(Z_{t_k})\big)\Delta t + g(Z_{t_N})\right],$$

for all $z \in \mathbb{R}^d$ and all control $a \in A$. We refer to the procedure in (4.3.3) as the performance iteration[2].

Except for trivial cases, closed-form formulas for the conditional expectations appearing in the value and policy iteration procedures are not available, and they have to be approximated, which is the main difficulty when implementing both approaches to compute the value functions. In the next section, we discuss different ways to approximate conditional expectations and derive the corresponding estimations of the value functions $V_{\Delta t}(t_n, \cdot)$ for $n = 0, \dots, N-1$.

4.3.2 Approximation of conditional expectations

In this subsection, we present three numerical methods that we apply later to conditional MKV problems. Two of these methods belong to the class of Regression Monte Carlo techniques, a family of algorithms whose effectiveness highly relies on the choice of the basis functions used to approximate conditional expectations; the third algorithm, Quantization, approximate the controlled process $Z^\alpha_{t_n}$ with a particular finite state Markov chain for which expectations can be approximated quickly.

4.3.2.1 Regression Monte Carlo

In the simpler uncontrolled case, the family of Regression Monte Carlo algorithms is based on the idea of approximating the conditional expectation $\mathbb{E}\big[V_{\Delta t}(t_{n+1}, Z_{t_{n+1}})\big|Z_{t_n}\big]$, for $n = 0, \dots, N-1$, by the orthogonal projection of $V_{\Delta t}(t_{n+1}, Z_{t_{n+1}})$ onto the space generated by a finite family of $\big\{\phi_k(Z_{t_n})\big\}_{k\geq 1}$ where $(\phi_k)_{k\geq 1}$ is a family of *basis functions*, i.e., a family of measurable real-valued functions defined on $\mathbb{R}^d$ such that $\big(\phi_k(Z_{t_n})\big)_{k\geq 1}$ is total in $L^2(\sigma(Z_{t_n}))$[3] and such that for all scalars β_k and all $K \geq 1$, if $\sum_{k=1}^K \beta_k \phi_k(Z_{t_n}) = 0$ a.s. then $\beta_k = 0$, for $k = 1, \dots, K$.

The expectation $\mathbb{E}\big[V_{\Delta t}(t_{n+1}, Z_{t_{n+1}})\big|Z_{t_n}\big]$ should then be approximated as follows:

$$\mathbb{E}\big[V_{\Delta t}(t_{n+1}, Z_{t_{n+1}})\big|Z_{t_n}\big] \approx \sum_{k=1}^K \beta^n_k \phi_k(Z_{t_n}),$$

where $K \geq 1$ is fixed and $\beta^n = (\beta^n_1, \dots, \beta^n_K)^\top$ is defined as:

$$\beta^n = \underset{\beta\in\mathbb{R}^K}{\operatorname{argmin}}\left\{\mathbb{E}\left[\left|V_{\Delta t}(t_{n+1}, Z_{t_{n+1}}) - \sum_{k=1}^K \beta_k \phi_k(Z_{t_n})\right|^2\right]\right\}. \tag{4.3.4}$$

Notice that β^n is defined in (4.3.4) as the minimizer of a quadratic function, and can then be rewritten by straightforward calculations as:

$$\beta^n = \mathbb{E}\left[\phi(Z_{t_n})\phi(Z_{t_n})^\top\right]^{-1}\mathbb{E}\left[V_{\Delta t}(t_{n+1}, Z_{t_{n+1}})\phi(Z_{t_n})\right], \tag{4.3.5}$$

where we use the notation $\phi = (\phi_1, \dots, \phi_K)^\top$, and where we assumed that $\mathbb{E}\left[\phi(Z_{t_n})\phi(Z_{t_n})^\top\right]$ is invertible[4].

[2] This procedure is also referred to as the "policy iteration" in the literature.

[3] $L^2(\sigma(Z_{t_n}))$ is the space of the square-integrable $\sigma(Z_{t_n})$-measurable r.v.

[4] If the assumption does not hold, the least squares problem can still be solved via SVD approach, which is consistent with regression techniques. See e.g. chapter 8 in [Gob16].

In order to estimate a solution to (4.3.5) we rely on Monte Carlo simulations to approximate expectations with finite sums. Consider the training set $\{(Z^m_{t_n}, Z^m_{t_{n+1}})\}_{m=1}^M$ at time t_n obtained by running $M \geq 1$ forward simulations of the process Z from time $t_0 = 0$ to t_{n+1}. β^n defined in (4.3.5) can then be estimated by

$$\hat{\beta}^n = \left(\hat{\mathcal{A}}^M_n\right)^{-1} \frac{1}{M} \sum_{m=1}^M V_{\Delta t}(t_{n+1}, Z^m_{t_{n+1}})\phi(Z^m_{t_n}),$$

where we denote by $\hat{\mathcal{A}}^M_n$ the estimator $\frac{1}{M}\sum_{m=1}^M \phi(Z^m_{t_n})\phi(Z^m_{t_n})^\top$ of the covariance matrix

$$\mathcal{A}_n = \mathbb{E}\left[\phi(Z_{t_n})\phi(Z_{t_n})^\top\right].$$

The procedure presented above offers a convenient mean to approximate conditional expectations when the dynamics of the process Z are uncontrolled. When controlled, however, one has to account for the effect of the control on the conditional expectations either explicitly, via Control Randomization, or implicitly, via Regress-Later.

Remark 4.3.1. *As recently detailed in [BRS18], there is another way of building estimates of β^n, for $n = 0, \dots, N-1$, which is more accurate, less costly to compute, and appears to be more efficient in pratice. The idea is the following: assuming that the basis of functions is orthonormal w.r.t. μ, we have:*

$$\beta^n_k := (V_{\Delta t}(t_{n+1}, .), \phi_k)_{\mathbb{L}^2(\mu)} = \mathbb{E}\left[\mathbb{E}[V_{\Delta t}(t_{n+1}, Z_{t_{n+1}})|Z_{t_n}]\phi_k(Z_{t_n})\right] = \mathbb{E}\left[V_{\Delta t}(t_{n+1}, Z_{t_{n+1}})\phi_k(Z_{t_n})\right]$$

Hence, one can build $\bar{\beta}^n$, defined below, as a better estimate of β^n than $\hat{\beta}^n$.

$$\bar{\beta}^n_k = \frac{1}{M} \sum_{m=1}^M V_{\Delta t}(t_{n+1}, Z^m_{t_{n+1}})\phi_k(Z^m_{t_n}), \quad \textit{for} k = 0, \dots, K.$$

The general case, where the basis functions are note orthonormal, requires multiplication with the Gram matrix formed by $(\phi_k, \phi_l)_{1\leq k,l\leq K}$, but the idea remains the same.

Control Randomization

In order to explicitly account for the effect of the control, one could directly introduce dependence on the control in the basis function. This basic idea of Control Randomization consists in replacing in the dynamics of Z the endogenous control by an exogenous control $(I_{t_n})_{0\leq n\leq N}$, as introduced in [KLP14]. Trajectories of $(Z_{t_n}, I_{t_n})_{0\leq n\leq N}$ can then be simulated from time t_0 to time t_N. Consider the training set $\{Z^m_{t_n}, I^m_{t_n}\}^{N,M}_{n=0,m=1}$, with $M \geq 1$, where $I^m_{t_n}$ are i.i.d. samples from a "training distribution" μ_n with support in A. The training set will be used to estimate the optimal β^n coefficients for $n = 0, \dots, N-1$. In the case of value iteration, $\{V_{\Delta t}(t_{n+1}, Z^m_{t_{n+1}})\}_{m=1}^M$ is regressed against basis functions (which are, in this context, functions of the state and the control) evaluated at the points $\{Z^m_{t_n}, I^m_{t_n}\}_{m=1}^M$, as follows:

$$\mathbb{E}^a_{n,z}\left[V_{\Delta t}(t_{n+1}, Z_{t_{n+1}})\right] \approx \sum_{k=1}^K \hat{\beta}^n_k \phi_k(z, a),$$

where $\hat{\beta}^n$ is an estimator of

$$\beta^n := \underset{\beta\in\mathbb{R}^K}{\operatorname{argmin}} \left\{\mathbb{E}\left[\left(V_{\Delta t}\left(t_{n+1}, Z_{t_{n+1}}\right) - \sum_{k=1}^K \beta_k \phi_k\left(Z_{t_n}, I_{t_n}\right)\right)^2\right]\right\},$$

defined as

$$\hat{\beta}^n = \left(\hat{\mathcal{A}}^M_n\right)^{-1} \frac{1}{M} \sum_{m=1}^M \left[V_{\Delta t}\left(t_{n+1}, Z^m_{t_{n+1}}\right)\phi(Z^m_{t_n}, I^m_{t_n})\right],$$

where $\phi = (\phi_1, \dots, \phi_K)^\top$ and

$$\hat{\mathcal{A}}_n^M = \frac{1}{M} \sum_{m=1}^{M} \phi(Z_{t_n}^m, I_{t_n}^m) \phi(Z_{t_n}^m, I_{t_n}^m)^\top \tag{4.3.6}$$

estimates the covariance matrix $\mathcal{A}_n = \mathbb{E}\left[\phi(Z_{t_n}, I_{t_n}) \phi(Z_{t_n}, I_{t_n})^\top\right]$.

Notice that the basis functions take state and action variables as input in the case of Control Randomization-based method, i.e., their domain is $\mathbb{R}^d \times A$. Also, observe that the estimated conditional expectation highly depends on the choice of the randomization for the control[5].

An optimal feedback control at time t_n given $Z_{t_n} = z$ is approximated by the expression (see Subsection 4.3.4 for more practical details on the computation of the argmax):

$$\hat{\alpha}_{t_n}(z) = \underset{a \in A}{\operatorname{argmax}} \left\{ f(z, a)\Delta t + \sum_{k=1}^{K} \hat{\beta}_k^n \phi_k(z, a) \right\}. \tag{4.3.7}$$

The value function at time t_n is then estimated using Control Randomization method and value iteration procedure as

$$\widehat{V}_{\Delta t}^{\mathrm{CR}}(t_n, z) = f\big(z, \hat{\alpha}_{t_n}(z)\big)\Delta t + \sum_{k=1}^{K} \hat{\beta}_k^n \phi_k\big(z, \hat{\alpha}_{t_n}(z)\big), \quad z \in \mathbb{R}^d.$$

Notice that Control Randomization can be easily employed in a performance iteration procedure by computing controls (4.3.7), keeping in mind that at each time t_n we need to re-simulate new trajectories $\{\tilde{Z}_{t_k}^m\}_{k=n,m=1}^{N,M}$ iteratively from the initial condition $\tilde{Z}_{t_n}^m = z$, using the estimated optimal strategies $(\hat{\alpha}_{t_k})_{k=n+1}^{N-1}$ to compute the quantities $\sum_{k=n}^{N-1} f\big(t_k, \tilde{Z}_{t_k}^m, \hat{\alpha}_{t_k}(\tilde{Z}_{t_k}^m)\big) + g(\tilde{Z}_{t_N}^m)$, for $1 \le m \le M$.

Regress-Later

We present now a regress-later idea in which conditional expectation with respect to Z_{t_n} is computed in two stages. First, a conditional expectation with respect to $Z_{t_{n+1}}$ is approximated in a regression step by a linear combination of basis functions of $Z_{t_{n+1}}$. Then, analytical formulas are applied to condition this linear combination of functions of future values on the present value Z_{t_n}. For further details, see [GY04], [BS12], [NMS17] or [BP18]. With this approach, the effect of the control is factored in implicitly, through its effect on the (conditional) distribution of $Z_{t_{n+1}}$ conditioned on Z_{t_n}.

Unlike the traditional Regress-Now method for approximating conditional expectations (which we discussed so far in the uncontrolled case and in Control Randomization), the Regress-Later approach, as studied in [BP18], imposes conditions on basis functions:

Assumption 4.3.1. *For each basis function ϕ_k, $k = 1, \dots, K$, the conditional expectation*

$$\hat{\phi}_k^n(z, a) = \mathbb{E}_{n,z}^a[\phi_k(Z_{t_{n+1}})]$$

can be computed analytically.

Using the Regress-Later approximation of the conditional expectation and recalling Assumption 4.3.1 we obtain the optimal control $\alpha_{t_n}^m$ corresponding to the point $Z_{t_n}^m$, sampled independently from a "training distribution" μ_n (see Subsection 4.3.3 for further details):

$$\alpha_{t_n}^m = \underset{a \in A}{\operatorname{argmax}} \Big\{ f(Z_{t_n}^m, a)\Delta t + \sum_{k=1}^{K} \hat{\beta}_k^{n+1} \hat{\phi}_k^n\big(Z_{t_n}^m, a\big) \Big\}.$$

[5] Basically, different randomized controls may drive the process Z to very different locations, and the estimations will suffer from inaccuracy on the states that have been rarely visited.

Notice that we are able to exploit the linearity of conditional expectations because

$$\hat{\beta}^{n+1} = \underset{\beta\in\mathbb{R}^K}{\operatorname{argmin}} \left\{ \sum_{m=1}^{M} \Big[V_{\Delta t}(t_{n+1}, Z^m_{t_{n+1}}) - \sum_{k=1}^{K} \beta_k \phi_k(Z^m_{t_{n+1}}) \Big]^2 \right\} \tag{4.3.8}$$

is a constant once the training sets at times t_k, $k = n+1, \dots, N$, are fixed.

The value function at time t_n, is then estimated using Regress-Later method and value iteration procedure as

$$\widehat{V}^{\mathrm{RL}}_{\Delta t}(t_n, Z^m_{t_n}) = f(Z^m_{t_n}, \alpha^m_{t_n})\Delta t + \sum_{k=1}^{K} \hat{\beta}^{n+1}_k \hat{\phi}^n_k(Z^m_{t_n}, \alpha^m_{t_n}).$$

Notice that contrary to Control Randomization, Regress-Later does not require the training points to be distributed as $Z_{t_{n+1}}$ conditioned on Z_{t_n} because the projection (4.3.8) is an approximation to an expectation conditional to the measure μ_n which can be chosen freely to optimize the precision of the sample estimation. On the other hand Regress-Later, similarly to Control Randomization, can be easily employed in a performance iteration procedure by generating forward trajectories at each time step.

Remark 4.3.1. *Recall that the Regress-Later method uses training points that are i.i.d at each time step and independent across time steps. Contrary to other Regression Monte Carlo approaches, Regress-Later does not require to use the information about the conditional distribution during the regression step as that is accounted for in the second step of the method, when conditional expectations are computed analytically.*

4.3.2.2 Quantization

We propose in this section a quantization-based algorithm to numerically solve control problems. We may also refer to the latter as the Q-algorithm or Q in all the numerical examples considered in Section 4.4, where Q stands for Quantization. Let us first introduce some ingredients of Quantization, and then propose different ways of using them to approximate conditional expectations.

Let $(E, |.|)$ be a Euclidean space. We denote by $\hat{\varepsilon}$ a L-quantizer of an E-valued random variable ε, that is a discrete random variable on a grid $\Gamma = \{e_1, \dots, e_L\} \subset E^L$ defined by

$$\hat{\varepsilon} \;=\; \mathrm{Proj}_\Gamma(\varepsilon) \;=\; \sum_{\ell=1}^{L} e_\ell 1_{\varepsilon\in C_\ell(\Gamma)},$$

where $C_1(\Gamma)$, ..., $C_L(\Gamma)$ are the Voronoi cells corresponding to Γ, i.e., they form a Borel partition of E satisfying

$$C_\ell(\Gamma) \;\subset\; \Big\{ e \in E : |e - e_\ell| \;=\; \min_{j=1,\dots,L} |e - e_j| \Big\},$$

and where Proj_Γ stands for the Euclidean projection on Γ.
The discrete law of $\hat{\varepsilon}$ is then characterized by

$$p_\ell \;=\; \mathbb{P}[\hat{\varepsilon} = e_\ell] \;=\; \mathbb{P}[\varepsilon \in C_\ell(\Gamma)], \quad \ell = 1, \dots, L.$$

The grid of points $(e_\ell)_{\ell=1}^L$ which minimizes the L^2-quantization error $\|\varepsilon - \hat{\varepsilon}\|_2$ leads to the so-called optimal L-quantizer, and can be obtained by a stochastic gradient descent method, known as Kohonen algorithm or competitive learning vector quantization (CLVQ) algorithm, which also provides as a byproduct an estimation of the discrete law $(p_\ell)_{\ell=1}^L$. We refer to [PPP04c] for a description of the algorithm, and mention that for the normal distribution, the optimal grids and the weights of the Voronoi tesselations are precomputed for dimension up to 10 and are available on the website http://www.quantize.maths-fi.com.

In practice, optimal grids of the Gaussian random variable $\mathcal{N}_1(0,1)$ of dimension 1 with 25 to 50 points, have been used to solve the control problems considered in Section 4.4.

We now propose two ways to approximate conditional expectations. The first approximation belongs to the family of the constant piecewise approximation, and the other one is an improvement on the first one, where the continuity of the approximation w.r.t. the control variable is preserved.

In the sequel, assume that for $n = 0, \dots, N-1$ we have a set Γ_n of points in $\mathbb{R}^d$ that should be thought of as a training set used for estimating $V(t_n, \cdot)$. See Subsection 4.3.3 for more details on how to build Γ_n.

Piecewise constant interpolation

We assume here that we already have an estimate of $V_{\Delta t}(t_{n+1}, \cdot)$, the value function at time t_{n+1}, for $n \in \{0, \dots, N-1\}$, and we denote by $\widehat{V}^Q_{\Delta t}(t_{n+1}, \cdot)$ the estimate.
The conditional expectation is then approximated as

$$\mathbb{E}^a_{n,z}\big[\widehat{V}^Q_{\Delta t}\big(t_{n+1}, Z_{t_{n+1}}\big)\big] \approx \sum_{\ell=1}^{L} p_\ell \widehat{V}^Q_{\Delta t}\Big(t_{n+1}, \mathrm{Proj}_{\Gamma_{n+1}}\big(G_{\Delta t}(z,a,e_\ell)\big)\Big), \qquad \text{for } z \in \Gamma_n,$$

where:

- $G_{\Delta t}$ is defined, using the notations introduced in (4.3.1), as
$$G_{\Delta t}(z,a,\varepsilon) = z + b(z,a)\Delta t + \sigma_0(z,a)\sqrt{\Delta t}\,\varepsilon. \tag{4.3.9}$$
- $\mathrm{Proj}_{\Gamma_n}(.)$ stands for the Euclidean projection on Γ_n.
- $\Gamma = \{e_1, \dots, e_L\}$ and $\{p_\ell\}_{1 \le \ell \le L}$ are respectively the optimal L-quantizer and its associated sequence of weights of the exogenous noise ε. See above for more details.

An optimal feedback control at time t_n and point $z \in \Gamma_n$ is approximated by the expression (see Subsection 4.3.4 for more practical details on the computation of the argmax):

$$\hat{\alpha}^Q_{t_n}(z) = \underset{a \in A}{\mathrm{argmax}} \left\{ f(z,a)\Delta t + \sum_{\ell=1}^{L} p_\ell \widehat{V}^Q_{\Delta t}\Big(t_{n+1}, \mathrm{Proj}_{\Gamma_{n+1}}\big(G_{\Delta t}(z,a,e_\ell)\big)\Big) \right\}.$$

The value function at time t_n, is then estimated using the piecewise constant approximation and value iteration procedure as

$$\widehat{V}^{\mathrm{Q}}_{\Delta t}(t_n, z) = f\big(Z^m_{t_n}, \hat{\alpha}^Q_{t_n}(z)\big)\Delta t + \sum_{\ell=1}^{L} p_\ell \widehat{V}^Q_{\Delta t}\left(t_{n+1}, \mathrm{Proj}_{\Gamma_{n+1}}\Big(G_{\Delta t}\big(z, \hat{\alpha}^Q_{t_n}(z), e_\ell\big)\Big)\right).$$

Remark 4.3.2. *Clearly, the constant piecewise approximation can be easily extended to control problems of all dimensions $d \ge 1$. However the latter is, in most cases, not continuous w.r.t. the control variable since it remains constant on each Voronoi cells (see, e.g., Figure 4.1 p.128). As a result, the optimization process over the control space suffers from high instability and inaccuracy, which implies a poor estimation of the value function $V(t_n, \cdot)$.*

Semi-linear interpolation Once again, we assume here that we already have $\widehat{V}^{\mathrm{Q}}_{\Delta t}(t_{n+1}, \cdot)$, an estimate of the value function at time t_{n+1}, with $n \in \{0, \dots, N-1\}$, and wish to provide an estimation of the conditional expectation in the particular case where the controlled process lies in dimension d=1. Consider the following piecewise linear approximation of the conditional expectation, which is continuous w.r.t. the control variable a:

$$\mathbb{E}^a_{n,z}\big[\widehat{V}^{\mathrm{Q}}_{\Delta t}\big(t_{n+1}, Z_{t_{n+1}}\big)\big] \approx \sum_{\ell=1}^{L} p_\ell \Big[\lambda_a^{e_\ell, z} \widehat{V}^{\mathrm{Q}}_{\Delta t}\big(t_{n+1}, z_+\big) + (1 - \lambda_a^{e_\ell, z})\widehat{V}^{\mathrm{Q}}_{\Delta t}\big(t_{n+1}, z_-\big)\Big], \qquad \text{for } z \in \Gamma_n, \tag{4.3.10}$$

where for all $\ell = 1, \dots, L$, z_- and z_+ are defined as follows:

- z_- and z_+ are the two closest states in Γ_{n+1} from $G_{\Delta t}(z,a,e_\ell)$, such that $z_- < G_{\Delta t}(z,a,e_\ell) < z_+$, if such states exist; and, in this case, we define $\lambda_a^{e_\ell,z} = \frac{G_{\Delta t}(z,a,e_\ell)-z_-}{z_+-z_-}$.
- Otherwise, z_- and z_+ are equal and defined as the closest state in Γ_{n+1}^Z from $G_{\Delta t}(z,a,e_\ell)$ and we define $\lambda_a^{e_\ell,z} = 1$.

Remark 4.3.3. *This second approximation is continuous w.r.t. the control variable, which brings stability and accuracy to the optimal control task (see Subsection 4.3.4), and also ensures an accurate estimate of the value function at time t_n. We will mainly use this approximation in the numerical tests (see Section 4.4).*

Remark 4.3.4. *Although the dimension $d = 1$ plays a central role to define clearly the states z_- and z_+ in (4.3.10), the semi-linear approximation can actually be generalized to a certain class of control problems of dimension greater than 1, using multi-dimensional Quantization (see, e.g., the comments on the Q-algorithm designed to solve the Portfolio Optimization example, in Subsection 4.4.1.2). However, it is not well-suited to solve numerically general control problems in dimension greater than 1. For these cases, other interpolating methods such as the use of Gaussian processes are more appropriated (see, e.g., [LM18] for an introduction on the use of Gaussian processes in Regression Monte Carlo).*

The optimal feedback control at time t_n and point $z \in \Gamma_n$ is approximated as (see Subsection 4.3.4 for more practical details on the computation of the argmax):

$$\hat{\alpha}_{t_n}^{\mathrm{Q}}(z) = \underset{a\in A}{\operatorname{argmax}} \left\{ f(z,a)\Delta t + \sum_{\ell=1}^{L} p_\ell \Big[\lambda_a^{e_\ell,z} \widehat{V}_{\Delta t}^{\mathrm{Q}}\big(t_{n+1}, z_+\big) + (1-\lambda_a^{e_\ell,z}) \widehat{V}_{\Delta t}^{\mathrm{Q}}\big(t_{n+1}, z_-\big) \Big] \right\}.$$

The value function at time t_n is then estimated using the semi-linear approximation and value iteration procedure as

$$\widehat{V}_{\Delta t}^{\mathrm{Q}}(t_n,z) = f\big(z, \hat{\alpha}_{t_n}^{\mathrm{Q}}(z)\big)\Delta t + \sum_{\ell=1}^{L} p_\ell \Big[\lambda_{\hat{\alpha}_{t_n}^{\mathrm{Q}}(z)}^{e_\ell,z} \widehat{V}_{\Delta t}^{\mathrm{Q}}\big(t_{n+1}, z_+\big) + (1-\lambda_{\hat{\alpha}_{t_n}^{\mathrm{Q}}(z)}^{e_\ell,z}) \widehat{V}_{\Delta t}^{\mathrm{Q}}\big(t_{n+1}, z_-\big) \Big],$$

where z_+ and z_- are defined using the control $\hat{\alpha}_{t_n}^{\mathrm{Q}}(z)$. See (4.3.10) for their definitions.

4.3.3 Training points design

We discuss here the choice of the training measure μ and the sets $(\Gamma_n)_{n=0,\dots,N-1}$ used to compute the numerical approximations in Regression Monte Carlo and Quantization. Two cases are considered in this section. The first one is a knowledge-based selection, relevant when the controller knows with a certain degree of confidence where the process has to be driven in order to optimize her reward functional. The second case, on the other hand, is when the controller has no idea where or how to drive the process to optimize the reward functional.

4.3.3.1 Exploitation only strategy

In the knowledge-based setting there is no need for exhaustive and expensive (in time mainly) exploration of the state space, and the controller can directly choose training sets Γ constructed from distributions μ that assign more points to the parts of the state space where the optimal process is likely to be driven.

In practice, at time t_n, assuming we know that the optimal process is likely to stay in the ball centered around the point m_n and with radius r_n, we chose a training measure μ_n centered around m_n as, for example $\mathcal{N}(m_n, r_n^2)$, and build the training set as sample of the latter. In the Regress-Later setting this can be done straightforwardly, while Control Randomization requires one to select

a measure for the random control such that the controlled process Z is driven in such area of the state space.

Taking samples according to μ to build grids makes them random. Another choice, which we used in the Quantization-based algorithm, is to use the (deterministic) optimal grid of $\mathcal{N}(m_n, \sigma_n^2)$ with reduced size (typically take 50 points for a problem in dimension 1, 250 for one of dimension 2 when $\sigma_n^2 = 1,\dots$), which can be found at www.quantize.maths-fi.com, to reduce the size of the training set and alleviate the complexity of the algorithms.

Remark 4.3.5. *As the reader will see, we chose the training sets based on the "exploitation only strategy" procedure, i.e. by guessing where to drive optimally the process, when solving the Liquidation Problem introduced in Subsection 4.4.1.*

4.3.3.2 Explore first, exploit later

Explore first: If the agent has no idea of where to drive the process to receive large rewards, she can always proceed to an exploration step to discover favorable subsets of the state space. To do so, the Γ_n, for $n = 0, \dots, N-1$, can be built as uniform grids that cover a large part of the state space, or μ can be chosen uniform on such domain. It is essential to explore far enough to have a well understanding of where to drive and where not to drive the process.

Exploit later: The estimates for the optimal controls at time t_n, $n = 0, \dots, N-1$, that come up from the *Explore first* step, are relatively good in the way that they manage to avoid the wrong areas of state space when driving the process. However, the training sets that have been used to compute the estimated optimal control are too sparse to ensure accuracy on the estimation. In order to improve the accuracy, the natural idea is to build new training sets by simulating M times the process using the estimates on the optimal strategy computed from the *Explore first* step, and then proceed to another estimation of the optimal strategies using the new training sets. This trick can be seen as a two steps algorithm that improves the estimate of the optimal control.

Remark 4.3.6. *In Control Randomization, multiple runs of the method are often needed to obtain precise estimates, because the initial choice of the dummy control could drive the training points far from where the optimal control would have driven them. In practice, after having computed an approximate policy backward in time, such policy is used to drive M simulations of the process forward in time, which in turn produce control paths that can be fed as a random controls in a new backward procedure, leading to more accurate results.*

Remark 4.3.7. *We applied the "explore first, exploit later" idea to solve the Portfolio Optimization problem introduced in Subsection 4.4.1.*

4.3.4 Optimal control searching

Assume in this section that we already have the estimates $\widehat{V}_{\Delta t}(t_k, \cdot)$ for the value function at time t_k, for $k = n+1, \dots, N$, and want to estimate $V(t_n, \cdot)$ the value function at time t_n.

The optimal control searching task consists in optimizing the function[6]

$$\widehat{Q}_n : (z, \cdot) \mapsto f(z, a)\Delta t + \hat{\mathbb{E}}^a_{n,z}\left[\widehat{V}_{\Delta t}(t_{n+1}, Z_{t_{n+1}})\right]$$

over the control space A, for each $z \in \Gamma_n$, and where we denote by $\hat{\mathbb{E}}^a_{n,z}\left[\widehat{V}_{\Delta t}(t_{n+1}, Z_{t_{n+1}})\right]$ an approximation of $\mathbb{E}^a_{n,z}\left[\widehat{V}_{\Delta t}(t_{n+1}, Z_{t_{n+1}})\right]$ using Regress-Later, or Control Randomization or Quantization-based methods (see Subsection 4.3.2). Once again, we remind that importance of this task is motivated

[6] often referred to as the Q-function, or action-value function, in the reinforcement learning literature. Be aware that Q stands here for the "Quality" of an action taken in a given state, and in particular does not refer to Quantization.

by the dynamic programming principle stating that for all $n = 0, \dots, N-1$, we can approximate the value function at time n as follows

$$\widehat{V}_{\Delta t}(t_n, z) = \sup_{a \in A} \widehat{Q}_n(z, a),$$

where $\widehat{V}_{\Delta t}(t_n, \cdot)$ is our desired estimate of the value function at time n.

4.3.4.1 Low cardinality control set

In the case where the control space A is discrete (with a relatively small cardinality), one can solve the optimization problem by an exhaustive search over all the available controls without compromising the computational speed.

Remark 4.3.8. *Note that in the case where the control space is continuous, one can always discretize the latter in order to rely on the effectiveness of extensive search to solve the optimal control problem. However, the control space discretization brings an error. So the control might have to include a high number of points in the discretization in order to reduce the error thereby causing a considerable slow down of the computations.*

4.3.4.2 High cardinality/continuous control space

If we assume differentiability almost everywhere, as follows from the semi-linear approximation in Quantization, and most choices of basis functions in Regression Monte Carlo, we can carry on the optimization step by using some gradient-based algorithm for optimization of differentiable functions. Actually, many optimizing algorithms (Brent, Golden-section Search, Newton gradient-descent,...) are already implemented in standard libraries of most programming languages like Python (see, e.g., package `scipy.optimize`), Julia (see, e.g., package `Optim.jl`), C and C++ (see, e.g., package `NLopt`).

Remark 4.3.9. *When the control is of dimension 1, polynomials of order smaller than 5 are employed as basis functions in Regression Monte Carlo as well as for the running reward f. The optimal control can then be computed analytically as a function of the regression coefficients, since every polynomial equation of order smaller than 4 can be solved by radicals.*

Concretely, in all the examples considered in Section 4.4, we used the Golden-section Search or the Brent methods when testing Quantization-based algorithm to find the optimal controls at each point of the grids. These algorithms were very accurate to find the optimal controls, and we made use of Remark 4.3.9 to find the optimal controls using the Regress-Later-based algorithm.

4.3.5 Upper and lower bounds

After completing the backward procedure, we can compute an unbiased estimation of the value of the control policy by using Monte Carlo simulations and sample average. Assume already computed (or simply available) the matrix of regression coefficients, in the case of Regression Monte Carlo, and discrete probability law $\hat{p}$ for Quantization, we can use this information to implicitly compute the control and simulate forward many trajectories of the controlled process starting from a common initial condition. We can then evaluate the average performance measure by computing the sample average of the rewards collected on each trajectory. Denoting such approximation by $\hat{\widehat{V}}_{\Delta t}(0, z)$, and recalling that by definition $J_{\Delta t}(0, z, \alpha) \leq J_{\Delta t}(0, z, \alpha^*)$, for all $\alpha \in \mathcal{A}_{\Delta t}$ and where α^* represents the optimal control process; it holds $\hat{\widehat{V}}_{\Delta t}(0, z) \leq V_{\Delta t}(0, z)$, for $z \in \mathbb{R}^d$.

The argument above implies that, neglecting the time-discretization error, we obtain a lower bound for $V_{\Delta t}(0, \cdot)$ by evaluating the estimated policy. To get an upper bound of the value function via duality, see [HSZD16], [HL17] based on [Rog02], and [BGS18].

4.3.6 Pseudo-codes

In this section, we present the pseudo-code for the three approaches presented in the previous sections. For simplicity, we will only show the algorithms designed using value iteration procedure. However, the performance iteration update rule can be substituted in the codes below provided that forward simulations are run to obtain a pathwise realization of the controlled process and associated rewards.

4.3.6.1 Pseudo-code for a Regress-Later-based algorithm

We present in Algorithm 4.1 a pseudo-code to estimate $V_{\Delta t}(t_n, \cdot)$, for $n = 0, \dots, N-1$, using Value Iteration and based on Regress-Later method. For $n = 0, \dots, N-1$, we denote by $\hat{V}^{\mathrm{RL}}_{\Delta t}(t_n, \cdot)$ the derived estimation of $V_{\Delta t}(t_n, \cdot)$, and will refer to it as the RLMC algorithm in the numerical tests presented in Section 4.4.

Note that we use the same training measure μ at each time step so that there is only one covariance matrix to estimate (since $\mathcal{A}_{t_n}$ is the same for all $n = 0, \dots, N-1$). Denote by $\hat{\mathcal{A}}^M$ the estimator, as defined in (4.3.6).

Algorithm 4.1 Regress-Later Monte Carlo algorithm (RLMC) - Value iteration

Inputs:

- M: number of training points,
- μ: distribution of training points,
- K: number of basis functions,
- $\{\phi_k\}_{k=1}^K$: family of basis functions.

1: Estimate the covariance matrix $\hat{\mathcal{A}}^M$.
2: Generate i.i.d. training points $\{Z^m_{t_N}\}_{m=1}^M$ accordingly to the distribution μ.
3: Initialize the value function $\hat{V}^{\mathrm{RL}}_{\Delta t}(t_N, Z^m_{t_N}) = g(Z^m_{t_N})$, $\forall m = 1, \dots, M$.
4: **for** $n = N-1$ to 0 **do**
5: $\quad \hat{\beta}^n = \hat{\mathcal{A}}_M^{-1} \frac{1}{M} \sum_{m=1}^M \left[\hat{V}^{\mathrm{RL}}_{\Delta t}(t_{n+1}, Z^m_{t_{n+1}})\phi(Z^m_{t_{n+1}})\right]$.
6: $\quad$ Generate a new layer of i.i.d. training points $\{Z^m_{t_n}\}_{m=1}^M$ accordingly to the distribution μ.
7: $\quad$ For all $m = 1, \dots, M$ do

$$\hat{V}^{\mathrm{RL}}_{\Delta t}(t_n, Z^m_{t_n}) = \sup_{a \in A} \Big\{ f(Z^m_{t_n}, a)\Delta t + \sum_{k=1}^K \hat{\beta}^n_k \hat{\phi}^n_k(Z^m_{t_n}, a) \Big\}.$$

8: **end for**
9: **Evaluate the policy to obtain** $\hat{\hat{V}}^{\mathrm{RL}}_{\Delta t}$.

Outputs: $\{\hat{\beta}^n_k\}_{n,k=1}^{N,K}$, $\hat{\hat{V}}^{\mathrm{RL}}_{\Delta t}(0, z)$ for $z \in \mathbb{R}^d$.

4.3.6.2 Pseudo-code for a Control Randomization-based algorithm

We present in Algorithm 4.2 a pseudo-code to estimate $V_{\Delta t}(t_n, \cdot, \cdot)$, for $n = 0, \dots, N-1$, using Value Iteration and based on Control Randomization method. For $n = 0, \dots, N-1$, we denote by $\hat{V}^{\mathrm{CR}}_{\Delta t}(t_n, \cdot)$ the derived estimation of $V_{\Delta t}(t_n, \cdot)$, and will refer to it as the CR algorithm in the numerical tests presented in Section 4.4.

Algorithm 4.2 Control Randomization algorithm (CR) - Value iteration

Inputs:

- M: number of training points,
- μ: initial distribution of dummy control,
- K: number of basis functions,
- $\{\phi_k\}_{k=1}^K$: family of basis functions.

1: Estimate the covariance matrix $\hat{\mathcal{A}}^M$.
2: Generate m trajectories, $\{Z_{t_n}^m, I_{t_n}^m\}_{n=0,m=1}^{N,M}$, where $Z_{t_n}^m$ is driven by $I_{t_n}^m$, and the $I_{t_n}^m$ are i.i.d with distribution μ.
3: Initialize the value function $\hat{V}_{\Delta t}^{\mathrm{CR}}(t_N, Z_{t_N}^m) = g(Z_{t_N}^m),\ m = 1, \dots, M$.
4: **for** $n = N-1$ to 0 **do**
5: $\quad \hat{\beta}^n = (\hat{\mathcal{A}}^M)^{-1} \frac{1}{M} \sum_{m=1}^M \left[\hat{V}_{\Delta t}^{\mathrm{CR}}(t_{n+1}, Z_{t_{n+1}}^m)\phi(Z_{t_n}^m, I_{t_n}^m)\right]$.
6: $\quad$ For all $m = 1, \dots, M$ do

$$\hat{V}_{\Delta t}^{\mathrm{CR}}(t_n, Z_{t_n}^m) = \sup_{a \in A} \Big\{ f(Z_{t_n}^m, a)\Delta t + \sum_{k=1}^K \hat{\beta}_k^n \phi_k(Z_{t_n}^m, a) \Big\}.$$

7: **end for**
8: **Evaluate the policy to obtain** $\hat{\hat{V}}_{\Delta t}^{\mathrm{CR}}$.

Outputs: $\{\hat{\beta}_k^n\}_{n=0,k=1}^{N,K}$, $\hat{\hat{V}}_{\Delta t}^{\mathrm{CR}}(0, z)$ for $z \in \mathbb{R}^d$.

4.3.6.3 Pseudo-code for a Quantization-based algorithm

We present in Algorithm 4.3 a pseudo-code to estimate $V_{\Delta t}(t_n, \cdot, \cdot)$, for $n = 0, \dots, N-1$, using value iteration procedure and based on Quantization method. For $n = 0, \dots, N-1$, we denote by $\hat{V}_{\Delta t}^{\mathrm{Q}}(t_n, \cdot)$ the derived estimation of $V_{\Delta t}(t_n, \cdot)$, and will refer to it as the Q-algorithm in the numerical tests presented in Section 4.4.

Note that we made use of a piecewise constant approximation of conditional expectations to approximate $\hat{V}_{\Delta t}^{\mathrm{Q}}(t_n, \cdot)$ in order to keep the algorithm simple. Also, note that, as said previously, in most of the numerical tests run in Section 4.4, we will use optimal grids available at www.quantize.maths-fi.com and will take $L = 25$ to 50 points for the size of the optimal grid of the Gaussian noise ε.

4.4 Applications and numerical results

4.4.1 Portfolio Optimization under drift uncertainty

4.4.1.1 The model

We consider a financial market model with one risk-free asset, assumed to be equal to one, and d risky assets of price process $S = (S^1, \dots, S^d)$ governed

$$dS_t = \mathrm{diag}(S_t)(\beta_t dt + \sigma dB_t^0), \quad S_0 = s_0 \in \mathbb{R}^d,$$

where B^0 is a d-dimensional Brownian motion on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P}^0)$, σ is the $d \times d$ invertible matrix volatility coefficient, assumed to be known and constant. However, the drift (β_t) of the asset (which is typically a diffusion process governed by another independent Brownian

Algorithm 4.3 Quantization algorithm (Q) - Value iteration

Inputs:

- Γ_k, $k = 0, \ldots, N$: grids of training points in $\mathbb{R}^d$,
- $\Gamma = \{e_1, \ldots, e_L\}$, $(p_\ell)_{1 \le \ell \le L}$: the L-optimal grid of the exogenous noise ε, and its associated weights,

1: Initialize the estimated value function at time N: $\hat{V}^{\mathrm{Q}}_{\Delta t}(t_N, z) = g(z), \quad \forall z \in \Gamma_N$.
2: **for** $n = N-1$ to 0 **do**
3: Estimate the value function at time t_n as follows:

$$\widehat{V}^{\mathrm{Q}}_{\Delta t}(t_n, z) = \max_{a \in A} \left[f(z,a)\,\Delta t + \sum_{\ell=1}^{L} p_\ell \widehat{V}^{\mathrm{Q}}_{\Delta t}\Big(t_{n+1}, \mathrm{Proj}_{\Gamma_{n+1}}\big(G_{\Delta t}(z,a,e_\ell)\big)\Big)\right], \quad \forall z \in \Gamma_n.$$

4: **end for**
5: **Evaluate the policy to obtain** $\hat{V}^{\mathrm{Q}}_{\Delta t}$.

Outputs: $\big(\hat{\alpha}(t_n, z)\big)_{z \in \Gamma_n, 0 \le n \le N-1}$, $\big(\hat{V}^{\mathrm{Q}}_{\Delta t}(0, z)\big)_{z \in \Gamma_0}$.

motion B) is unknown and unobservable like the Brownian motion B^0. The agent can actually only observe the stock prices S, and we denote by $\mathbb{F}^S$ the filtration generated by the price process S, which should be view as the available information.

In this context, we shall consider two important classes of optimization problems in finance:

(1) *Portfolio Liquidation.* We consider the problem of an agent (trader) who has to liquidate a large number y_0 of shares in some asset (we consider one stock, $d = 1$) within a finite time T, and faces execution costs and market price impact. In contrast with frictionless Merton problem, we do not consider mark-to-market value of the portfolio and instead consider separately the amount on the cash account and the inventory Y, i.e., the position or number of shares held at any time. The strategy of the agent is then described by a real-valued $\mathbb{F}^S$-adapted process α, representing the velocity at which she buys ($\alpha_t > 0$) or sells ($\alpha_t < 0$) the asset, and the inventory is thus given by

$$Y_t \;=\; y_0 + \int_0^t \alpha_u du, \quad 0 \le t \le T.$$

The objective of the trader is to minimize over α the total liquidation cost

$$J_1(\alpha) \;=\; \mathbb{E}^0\big[\int_0^T \alpha_t\big(S_t + f(\alpha_t)\big)dt + \ell(Y_T)\big]$$

where $f(.)$ is an increasing function with $f(0) = 0$, representing a temporary price impact, and $\ell(.)$ is a loss function, i.e., a convex function with $\ell(0) = 0$, penalizing the trader when she does not succeed to liquidate all her shares.

(2) *Portfolio Selection.* The set $\mathcal{A}$ of portfolio strategies, representing the amount invested in the assets, consists in all $\mathbb{F}^S$-adapted processes α valued in some set A of $\mathbb{R}^d$, and satisfying $\int_0^T |\alpha_t|^2 dt < \infty$. The dynamics of wealth process $X = X^\alpha$ associated to a portfolio strategy α is then governed by

$$\begin{aligned} dX_t &= \alpha_t S_t^{-1} dS_t \\ &= \alpha_t.\beta_t dt + \alpha_t^\intercal \sigma dB_t^0, \quad X_0 \;=\; x_0 \in \mathbb{R}, \end{aligned}$$

and as in Merton Portfolio Selection problem, the objective of the agent is to maximize over portfolio strategies the utility of terminal wealth

$$J_2(\alpha) \;=\; \mathbb{E}^0[U(X_T)],$$

where U is a utility function on $\mathbb{R}$, e.g., CARA function $U(x) = -\exp(-px)$, $p > 0$.

Let us show how one can reformulate the above problems into a McKean-Vlasov type problem under partial observation and common noise as described in Section 4.1. We first introduce the so-called probability reference $\mathbb{P}$, which makes the observation price process a martingale. Let us then define the process

$$Z_t \;=\; \exp\big(-\int_0^t \sigma^{-1}\beta_u dB_u^0 - \frac{1}{2}\int_0^t |\sigma^{-1}\beta_u|^2 du\big), \quad 0 \le t \le T,$$

which is a $(\mathbb{P}^0, \mathbb{F})$-martingale (under suitable integrability conditions on β), and defines a probability measure $\mathbb{P} \sim \mathbb{P}^0$ through its density: $\frac{d\mathbb{P}}{d\mathbb{P}^0}\Big|_{\mathcal{F}_t} = Z_t$, and under which the process

$$W_t^0 \;=\; B_t^0 + \int_0^t \sigma^{-1}\beta_u du, \quad 0 \le t \le T,$$

is a $(\mathbb{P}, \mathbb{F})$-Brownian motion by Girsanov's theorem, and the dynamics of S is

$$dS_t \;=\; \mathrm{diag}(S_t)\sigma dW_t^0.$$

Notice that $\mathbb{F}^S = \mathbb{F}^0$ the filtration generated by W^0. We also denote by $L_t = 1/Z_t$ the $(\mathbb{P}, \mathbb{F})$-martingale governed by

$$dL_t \;=\; L_t\sigma^{-1}\beta_t.dW_t^0.$$

Next, we use Bayes formula and rewrite the gain (resp. cost) functionals of our two portfolio optimization problems as

$$\begin{aligned}
J_1(\alpha) &= \mathbb{E}\big[\int_0^T L_t\alpha_t(S_t + f(\alpha_t))dt + L_T\ell(Y_T)\big] \\
&= \mathbb{E}\big[\int_0^T \bar{L}_t^0\alpha_t(S_t + f(\alpha_t))dt + \bar{L}_T^0\ell(Y_T)\big] \\
&= \mathbb{E}\big[\int_0^T \bar{L}_t^0\alpha_t(\bar{S}_t^0 + f(\alpha_t))dt + \bar{L}_T^0\ell(\bar{Y}_T^0)\big], \\
J_2(\alpha) &= \mathbb{E}\big[L_T U(X_T)\big] \;=\; \mathbb{E}\big[\bar{L}_T^0 U(X_T)\big] \;=\; \mathbb{E}\big[\bar{L}_T^0 U(\bar{X}_T^0)\big]
\end{aligned}$$

where $\bar{L}_t^0 = \mathbb{E}[L_t|W^0] = \int \ell \mathbb{P}_{L_t}^{W^0}(d\ell)$, $\bar{X}_t^0 = \mathbb{E}[X_t|W^0] = \int x\mathbb{P}_{X_t}^{W^0}(dx) = X_t$, $\bar{Y}_t^0 = \mathbb{E}[Y_t|W^0] = \int y\mathbb{P}_{Y_t}^{W^0}(dy) = Y_t$, $\bar{S}_t^0 = \mathbb{E}[S_t|W^0] = \int s\mathbb{P}_{S_t}^{W^0}(ds) = S_t$, and we used the law of conditional expectations and the fact that S, X and Y are $\mathbb{F}^0$-adapted. This formulation of the functional J_1 (resp. J_2) fits into the MKV framework of Section 4.1 with state variables (X, L, β) (resp. (Y, S, L, β))

We now consider the particular case when β is an $\mathcal{F}_0$-measurable random variable distributed according to some probability distribution $\nu(db)$: this corresponds to a Bayesian point of view when the agent's belief about the drift is modeled by a prior distribution. In this case, let us show how our partial observation problem can be embedded into a finite-dimensional full observation Markov control problem. Indeed, by noting that β is independent of the Brownian motion W^0 under $\mathbb{P}$, we have

$$\bar{L}_t^0 \;=\; \mathbb{E}\big[\exp\big(\sigma^{-1}\beta.W_t^0 - \frac{1}{2}|\sigma^{-1}\beta|^2 t\big)\big|W^0\big] \;=\; F(t, W_t^0),$$

where

$$F(t,w) \;=\; \int \exp\big(\sigma^{-1}b.w - \frac{1}{2}|\sigma^{-1}b|^2 t\big)\nu(db).$$

Hence, the functionals J_1 and J_2 can be written as

$$\begin{aligned} J_1(\alpha) &= \mathbb{E}\big[\int_0^T F(t,W_t^0)\alpha_t(S_t+f(\alpha_t))dt + F(T,W_T^0)\ell(Y_T)\big], \\ J_2(\alpha) &= \mathbb{E}\big[F(T,W_T^0)U(X_T)\big]. \end{aligned}$$

We are then reduced to a $(\mathbb{P},\mathbb{F}^0)$-control problem with state variables (W^0,X) for problem (1) and (W^0,S,Y) for problem (2), with the following dynamics under $\mathbb{P}$:

$$\begin{aligned} dS_t &= \mathrm{diag}(S_t)\sigma dW_t^0, \quad S_0 = s_0 \in (\mathbb{R}_+)^d, \\ dX_t &= \alpha_t^\intercal \sigma dW_t^0, \quad X_0 = 0, \\ dY_t &= \alpha_t dt, \quad Y_0 = y_0 \in \mathbb{R}_+ . \end{aligned}$$

Remark 4.4.1. The case where the drift β is modeled by a linear Gaussian process is another example of partial observation. This would lead to the well-known Kalman-Bucy filter, hence to a finite-dimensional control problem. However, for general unobserved drift process β, we fall into an infinite dimensional control problem involving the filter process.

4.4.1.2 Numerical results

Let us now illustrate numerically the impact of uncertain Bayesian drift on the Portfolio Liquidation problem and the Portfolio Selection problem in dimension $d = 1$, by considering a Gaussian prior distribution $\beta \sim \nu = \mathcal{N}(b_0,\gamma_0^2)$, with $b_0 \in \mathbb{R}$ and $\gamma_0 > 0$. In this case, F is explicitly given by:

$$F(t,w) = \frac{\sigma}{\sqrt{\sigma^2+\gamma_0^2 t}} \exp\Big(\frac{1}{2(\sigma^2+\gamma_0^2 t)}(-b_0^2 t + 2b_0\sigma w + \gamma_0^2 w^2)\Big).$$

1. Portfolio Liquidation. Let us first consider the Portfolio Liquidation problem (1) with a linear price impact function $f(a) = \gamma a$, $\gamma > 0$, and a quadratic loss function $\ell(y) = \eta y^2$, $\eta > 0$. The optimal trading rate is given by (see [Pha16])

$$\alpha_t^* = -\frac{Y_t^*}{T-t+\gamma/\eta} + \frac{1}{2\gamma}\Big(\frac{1}{T-t+\gamma/\eta}\int_t^T \mathbb{E}^0[S_u|\mathcal{F}_t^S]du - S_t\Big)$$

where Y^* is the associated inventory with feedback control α^*: $dY_t^* = \alpha_t^* dt$, $Y_0^* = y_0$. Since we consider a Gaussian prior $\mathcal{N}(b_0,\gamma_0^2)$ for β, the optimal trading rate is explicitly given by

$$\alpha_t^* = -\frac{1}{T-t+\gamma/\eta}\left\{Y_t^* + \frac{1}{2\gamma}\left[-\frac{1}{\gamma_0}\sqrt{\frac{\pi}{2}}e^{-\frac{b_0^2}{2\gamma_0^2}}\left(\mathrm{erfi}\left(\frac{b_0+\gamma_0^2(T-t)}{\sqrt{2}\gamma_0}\right) - \mathrm{erfi}\left(\frac{b_0}{\sqrt{2}\gamma_0}\right)\right) + (T-t+\frac{\gamma}{\eta})\right]S_t\right\},$$

where `erfi` is the imaginary error function, defined as:

$$\mathrm{erfi}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{t^2}dt.$$

Remark 4.4.2. *In the particular case where the price process is a martingale, i.e., $b_0 = 0$, and in the limiting case when the penalty parameter η goes to infinity, corresponding to the final constraint $Y_T = 0$, we see that α_t^* converges to $-Y_t^*/(T-t)$, hence it becomes independent of the price process, and this leads to an explicit optimal inventory: $Y_t^* = y_0\frac{T-t}{T}$ with constant trading rate $\alpha_t^* = -y_0/T$. We retrieve the well-known VWAP strategy obtained in [AC01].*

We solve the problem numerically, taking $N = 100$ for the time discretization, and fixing the other parameters as follows: γ=5, S_0=6, Y_0=1, η=100 and σ=0.4. We run two sets of forward Monte Carlo simulations for $b_0 = 0.1$, $T = 1$ and $b_0 = -0.1$, $T = 0.5$ changing the value of γ_0. We tested the Regress-Later Monte Carlo (RLMC), the Control Randomization (CR) and the Quantization

(Q) algorithms. In particular, we wanted to compare the performance of these algorithms with $(\alpha^*_{t_n})_{n=0}^{N-1}$, where α^*, defined above, is the optimal strategy associated to the continuous-time Portfolio Liquidation problem. We refer to this discrete-time strategy as α^* (i.e., re-using the same notation), and we use Opt, or *continuous-time* optimal strategy when we want to stress the fact that this strategy is optimal for the *continuous-time* control problem, and not for the discrete time one. We also tested a benchmark strategy (Bench) which consists in liquidating the inventory at a constant rate $-y_0/T$. The test consisted in computing the estimates $\hat{\hat{V}}_{\Delta t}(t_0 = 0, S_0 = 6, Y_0 = 1)$ associated to the different algorithms.

We display the results obtained by the different algorithms in Table 4.1 and plot them in Figure 4.2. One can observe in Figure 4.2 that for $\Delta t = \frac{1}{100}$ the estimations $\hat{\hat{V}}_{\Delta t}(t_0 = 0, S_0 = 6, Y_0 = 1)$ of the value function $V_{\Delta t}(t_0 = 0, S_0 = 6, Y_0 = 1)$, provided by RLMC, CR or Q-based methods, are sometimes such that

$$\hat{\hat{V}}_{\Delta t}(t_0 = 0, S_0 = 6, Y_0 = 1) \leq \hat{J}_{\Delta t}(t_0 = 0, S_0 = 6, Y_0 = 1, \alpha^*),$$

where $\hat{J}_{\Delta t}(\cdot, \cdot, \cdot, \alpha^*)$ is a Monte Carlo estimate of $J_{\Delta t}(\cdot, \cdot, \cdot, \alpha^*)$ applying strategy $(\alpha^*_{t_n})_{n=0}^{N-1}$ (see in Figure 4.2 the curve Opt). It means that RLMC, CR, or Q-based methods sometimes provide better estimations of the optimal strategy than α^* for the discrete time control problem. However, since under suitable conditions (see, e.g., [KLP14]), the optimal strategy for the discrete time control problem $\alpha^*_{\Delta t}$ converges toward α^*, i.e. we have $\alpha^*_{\Delta t} \xrightarrow[\Delta t \to 0]{} \alpha^*$, then it holds:

$$\hat{J}_{\Delta t}(t_0 = 0, S_0 = 6, Y_0 = 1, \alpha^*) \xrightarrow[\Delta t \to 0]{} V(t_0 = 0, S_0 = 6, Y_0 = 1).$$

Figure 4.3 shows a sample of the inventory $(Y_t)_{t \in [0,T]}$ when the agent follows α^* and the Quantization algorithm. One can notice that given the chosen penalization parameters, it is optimal to short some stocks at terminal time. Finally, notice that the concavity of the curves comes from the fact that the running cost does not penalize the inventory. If so, we expect the curves of the inventory w.r.t. time to be convex, see, e.g., [GS13].

Details on the RL and CR algorithms implementation

The implementation of Regression Monte Carlo algorithms has required intense tuning and the use of the performance iteration technique introduced in Subsection 4.3.3.2 in order to obtain satisfactory results. Paramount is, in addition, the distribution chosen for the training points in Regress-Later and for the initial control in Control Randomization. The problem of finding the best set of data to provide to the backward procedure is similar in the two Regression Monte Carlo algorithms. However little study is available in the literature; for more details on this problem in the Regress-Later setting see [NMS17] and [BP18]. In the case of RL algorithm a training measure μ_n has been chosen in order to sufficiently explore the state space in the Y dimension, in particular we considered $\mu_n = \mathcal{U}[-0.5, 0.5 + \frac{T - t_n}{t_n}]$. Similarly for CR we seek a distribution of the random control such that the controlled process Y results in having a distribution similar to μ_n. In order to achieve such goal we follow the "explore first, exploit later" approach presented in Subsection 4.3.3.2 and use a perturbed version of the empirical distribution of the control (to avoid concentration of the training points) obtained at previous iteration of the method to determine the random control at next iteration of the method.

In order to choose the basis functions, we used the fact that we expect the value function to be convex in the Y dimension with minimum around the optimal inventory level and monotone in the S dimension. For RL algorithm we choose therefore the following set of basis functions: $\{s, y, y^2, sy, sy^2\}$, where we take the square function y^2 as a general approximator for convex functions around their minima (where we expect the measure μ_n to be concentrated). On the other hand, CR requires that we guess what the functional form of the conditional expectation of the value function is with respect to the control process. Considering our argument on square function approximating general convex functions we choose to add the set $\{\alpha, \alpha^2, \alpha y, \alpha s\}$ to the set of basis functions used by RL.

Note that there is no need for time-consuming optimal control searching with such a choice of basis functions, as explained in Remark 4.3.9.

Finally note that we observed very high volatility in the quality of the policy estimated by control randomization. For this reason we estimated the policy 50 times, and report in Table 4.1 the results provided by the best performing one; increasing the number of training points further affects the variability only marginally.

Details on the Q algorithm implementation

To numerically solve this example, we used the optimal grid of the Gaussian random variable with $L = 50$ points, denoted by Γ_L^ε, to define the grid[7] $\Gamma_n^W = t_n\Gamma_L^\varepsilon$ that discretizes W_{t_n}, the Brownian motion at time t_n, and the grid $\Gamma_n^Y = Y_0 - \frac{t_n}{T} + t_n\Gamma_L^\varepsilon$ that discretizes Y_{t_n}, the inventory at time t_n, for $n = 0, ..., N$. Note that Γ_n^Y, for $n = 0, ..., N$, is centered at point $Y_0 - \frac{t_n}{T}$ because we guessed that the optimal liquidation rate was close to $\frac{Y_0}{T}$ (see Figure 4.3 to check that our guess is correct).
We then considered the grid $\Gamma_n = \Gamma_n^W \times \Gamma_n^Y$ to discretize $Z_{t_n} = (W_{t_n}, Y_{t_n})$, $n = 0, ..., N$.

We first tried to design a quantization algorithm using the following expression for the conditional expectation approximations:

$$\mathbb{E}^a_{n,(w,y)}\Big[\widehat{V}^{\mathrm{Q}}_{\Delta t}\Big(t_{n+1}, \mathrm{Proj}_{\Gamma^W_{n+1}}\big(W_{t_{n+1}}\big), \mathrm{Proj}_{\Gamma^Y_{n+1}}\big(Y_{t_{n+1}}\big)\Big)\Big] \tag{4.4.6}$$
$$\approx \sum_{\ell=1}^{L} p_\ell \widehat{V}^{\mathrm{Q}}_{\Delta t}\Big(t_{n+1}, \mathrm{Proj}_{\Gamma^W_{n+1}}\big(G_{\Delta t}((w,y),a,e_\ell)\big), \mathrm{Proj}_{\Gamma^Y_{n+1}}\big(G_{\Delta t}((w,y),a,e_\ell)\big)\Big),$$
$$\text{for } (w,y,a) \in \Gamma_n^W \times \Gamma_n^Y \times A,$$

where the first and second components of the process $Z = (W, Y)$ are projected respectively on the grids Γ_n^W and Γ_n^Y; and $\mathrm{Proj}_{\Gamma_n^W}$ (resp. $\mathrm{Proj}_{\Gamma_n^Y}$) stands for the Euclidean projection of the first (resp. second) component of $Z = (W, Y)$ on Γ_n^W (resp. Γ_n^Y).
This approximation belongs to the family of constant piecewise approximations, and is in the spirit of multidimensional component-wise-quantization methods already studied in the literature (see, e.g., [FPS18]).
Unfortunately, as it can be seen in Figure 4.1, approximation (4.4.6) is discontinuous w.r.t. the control variable a in such a way that the optimal control searching task suffered from instability and inaccuracy, which implied bad value function estimations at time $n = 0, ..., N-1$. We thus had to use a better conditional expectation approximation.

We then decided to smooth the previous approximation of the conditional expectations w.r.t. the control variable by considering the following

$$\mathbb{E}^a_{n,(w,y)}\Big[\widehat{V}^{\mathrm{Q}}_{\Delta t}\Big(t_{n+1}, \mathrm{Proj}_{\Gamma^W_{n+1}}\big(W_{t_{n+1}}\big), \mathrm{Proj}_{\Gamma^Y_{n+1}}\big(Y_{t_{n+1}}\big)\Big)\Big]$$
$$\approx \sum_{\ell=1}^{L} p_\ell \Big[\lambda_a^{e_\ell,(w,y)} \widehat{V}^{\mathrm{Q}}_{\Delta t}\big(t_{n+1}, \mathrm{Proj}_{\Gamma^W_{n+1}}\big[G^w_{\Delta t}((w,y),a,e_\ell)\big], y_+\big)$$
$$+ (1 - \lambda_a^{e_\ell,(w,y)}) \widehat{V}^{\mathrm{Q}}_{\Delta t}\Big(t_{n+1}, \mathrm{Proj}_{\Gamma^W_{n+1}}\big[G^w_{\Delta t}((w,y),a,e_\ell)\big], y_-\Big)\Big],$$

where, in the spirit of the semi-linear approximation presented in Subsection 4.3.2, we have for all $\ell = 1, ..., L$:

- $G^w_{\Delta t}((w,y),a,e_\ell)$ and $G^y_{\Delta t}((w,y),a,e_\ell)$ respectively stand for the first and the second component of $G_{\Delta t}((w,y),a,e_\ell)$, i.e., $G_{\Delta t}((w,y),a,e_\ell) = \big(G^w_{\Delta t}((w,y),a,e_\ell), G^y_{\Delta t}((w,y),a,e_\ell)\big)$. See (4.3.9) for the definition of $G_{\Delta t}$.

[7] We use the notation $tB = \{tb, b \in B\}$, where $t \in \mathbb{R}$ and B is a set.

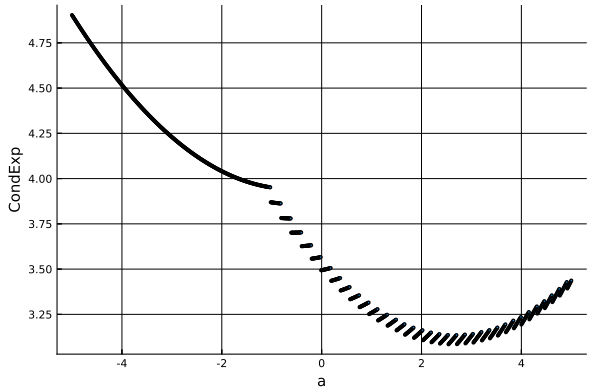


Figure 4.1 – Plot of the quantized-based piecewise-constant approximation of the conditional expectation CondExp:$a \mapsto \sum_{e\in\Gamma^\varepsilon} \mathbb{P}(\widehat{\varepsilon} = e)\widehat{V}^{\mathrm{Q}}_{\Delta t}\Big(t_{n+1}, \mathrm{Proj}_{\Gamma^W_{n+1}}\big(G_{\Delta t}((w,y),a,e)\big), \mathrm{Proj}_{\Gamma^Y_{n+1}}\big(G_{\Delta t}((w,y),a,e)\big)\Big)$. We took $n = N-1$, $w = 0$, and $y = -0.18$ to plot the curve. Observe that the approximation is discontinuous w.r.t. the control variable a in such a way that it makes the search of the minimizer of this function very difficult by usual (gradient descent-based) algorithms. Also, observe that the minimum of the function, which is actually equal to the estimation of the value function at time $N-1$ at point $(w = 0, y = -0.18)$, suffers from inaccuracy.

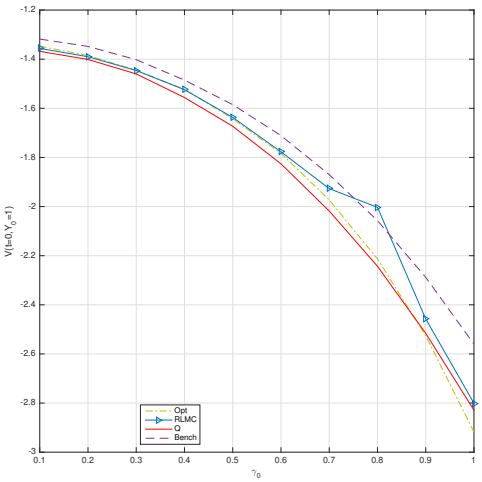
Table 4.1 – Portfolio Liquidation results. Estimations of the value functions at point $(s_0 = 6, y_0 = 1)$ and time 0 provided by different algorithms.

	$b_0 = 0.1$, $T = 1$					$b_0 = -0.1$, $T = 1/2$				
γ_0	Opt	RLMC	CR	Q	Bench	Opt	RLMC	CR	Q	Bench
0.1	-1.347	-1.356	-1.278	-1.368	-1.318	3.689	3.687	3.995	3.686	4.144
0.2	-1.385	-1.390	-1.283	-1.401	-1.348	3.682	3.682	3.847	3.679	4.138
0.3	-1.445	-1.446	-1.314	-1.460	-1.402	3.670	3.674	4.034	3.667	4.126
0.4	-1.523	-1.524	-1.323	-1.556	-1.485	3.655	3.674	4.128	3.650	4.108
0.5	-1.642	-1.637	-1.348	-1.673	-1.585	3.636	3.664	4.243	3.630	4.088
0.6	-1.783	-1.777	-1.425	-1.826	-1.711	3.611	3.640	4.386	3.607	4.064
0.7	-1.973	-1.927	-1.513	-2.018	-1.870	3.581	3.613	4.783	3.572	4.029
0.8	-2.213	-2.003	-1.637	-2.243	-2.057	3.545	3.575	5.142	3.537	3.992
0.9	-2.526	-2.457	-1.819	-2.516	-2.288	3.500	3.530	5.345	3.498	3.952
1	-2.918	-2.801	-1.806	-2.829	-2.560	3.453	3.513	6.765	3.452	3.903

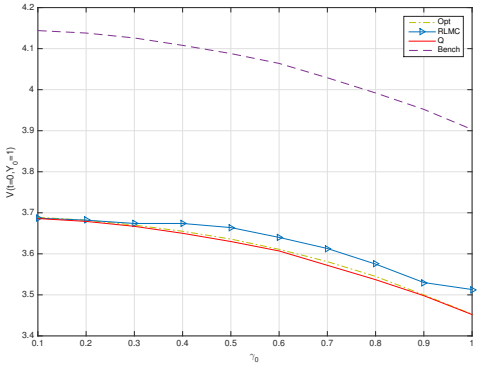
- y_- and y_+ are the two closest states in Γ^Y_{n+1} from $G^y_{\Delta t}((w,y),a,e_\ell)$, such that $y_- < G^y_{\Delta t}((w,y),a,e_\ell) < y_+$ if such point exists; y_- and y_+ are equal to the closest state in Γ^Y_{n+1} from $G^y_{\Delta t}((w,y),a,e_\ell)$ otherwise.
- $\lambda_a^{e_\ell,(w,y)} = \frac{G^y_{\Delta t}((w,y),a,e_\ell)-y_-}{y_+-y_-}$ in the first case of the definition of y_- and y_+ above; $\lambda_a^{e_\ell,(w,y)} = 1$ otherwise.

This approximation is a slight generalization (to dimension d=2) of the semi-linear approximation developed in (4.3.10). Its main interest lies in the continuity of the approximation w.r.t. the control variable a, which provides stability and accuracy to the usual (gradient descent-based) algorithms for the optimal controls searching, as can be seen on the numerical results (see, e.g., Table 4.1).

2. Portfolio Selection. Consider the Portfolio Selection problem with one risky asset. We choose a CARA utility function $U(x) = -\exp(-px)$, with $p > 0$. It has been shown in [BGP16, Corollary



$b_0 = 0.1$ and $T = 1$.



$b_0 = -0.1$ and $T = 0.5$.

Figure 4.2 – Results for the Portfolio Liquidation problem. Estimation of the value function at point $(s_0 = 6, y_0 = 1)$ at time 0 provided by different strategies w.r.t. γ_0. We took γ=5, S_0=6, Y_0=1, η=100 and σ=0.4.

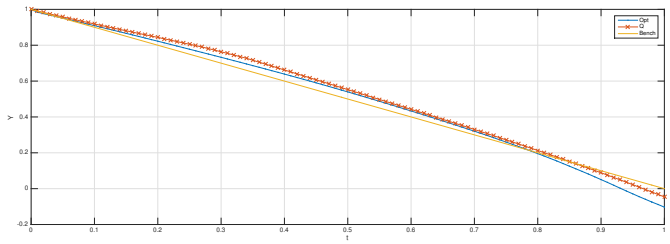


Figure 4.3 – Simulation of $(Y_t)_{t\in[0,T]}$ using the (continuous-time) optimal strategy (Opt), the (Q) estimated one, and the Benchmark strategy (Bench) to solve the Portfolio Liquidation problem. We took $T = 1$, $\sigma = 0.4$, $\gamma_0 = 1$, $b_0 = 0.1$, $S_0 = 6$, $Y_0 = 1$, $N = 100$, $\gamma = 5$, $\eta = 100$.

1] that the optimal portfolio strategy is explicitly given by

$$\alpha_t^* \quad = \quad \frac{\sigma^2 + \gamma_0^2 t}{\sigma^2 + \gamma_0^2 T} \frac{\hat{\beta}_t}{p\sigma^2}$$

where

$$\hat{\beta}_t \quad = \quad \mathbb{E}^0[\beta | \mathcal{F}_t^S] \quad = \quad \frac{\sigma^2}{\sigma^2 + \gamma_0^2 t} b_0 + \frac{\gamma_0^2}{\sigma^2 + \gamma_0^2 t} \Big(\ln \frac{S_t}{S_0} + \frac{1}{2}\sigma^2 t \Big),$$

is the posterior mean of the drift (Bayesian learning on the drift), and the optimal performance by

$$J_2(\alpha^*) \quad = \quad -\exp\Big[-p\Big(x_0 + \frac{1}{2p} \big(\ln \big(\frac{\sigma^2 + \gamma_0^2 T}{\sigma^2} \big) - \frac{\gamma_0^2 T}{\sigma^2 + \gamma_0^2 T} \big) + \frac{b_0^2}{2p\sigma^2} \frac{\sigma^2 T}{\sigma^2 + \gamma_0^2 T} \Big) \Big].$$

The Portfolio Selection problem, even though in many aspects similar to the Portfolio Liquidation problem, is interesting in its own right because the control acts only on the variance of the controlled wealth process. We tested the Regress-Later Monte Carlo (RLMC), the Control Randomization (CR) and the Quantization (Q) algorithm on the Portfolio Selection problem. Similarly to what has been done for Portfolio Liquidation problem, we discretized time choosing $N = 100$ and solved the discrete time problem associated. We considered two set of experiments, $b_0 = 0.1$, $T = 1$ and $b_0 = -0.1$, $T = 0.5$, for different values of $\gamma_0 \in [0, 1]$, $p = 1$, $\sigma = 0.4$. Given all these different parameters, we compared the performance of these algorithms with the one of the optimal strategy for the continuous-time problem α^* (Opt). The general test consists in computing a forward Monte Carlo with 500000 samples, following optimal strategy estimated using different strategies, to provide estimates of $V(t_0 = 0, X_0 = 0, W_0 = 0)$ the value function at time 0.

We present the results of our numerical experiments in Table 4.2. One can see that the Quantization algorithm performs similarly to the theoretical optimal strategy (Opt) for the continuous time problem, which can be interpreted as stability and accuracy of the Q algorithm, and also shows that the time discretization error is almost zero here.

We also present in Figure 4.4 a sample of the wealth of the agent following the optimal strategy and the (Q) estimated one. One can see that the strategies slightly differ when the drift is high, and remain the same when the drift is low. The small difference can be explained by the fact that the optimal strategy (Opt) is not optimal for the discrete time version of the problem.

Details on the Q algorithm implementation
We designed the same Quantization algorithm as the one built to solve the Portfolio Liquidation problem. We nevertheless had to take a larger number of points in the grids to minimize the back-propagation of errors from the borders of the girds; and had to use the "explore first, exploit later" idea (see Subsection 4.3.3.2) to improve the results.

Details on the RL and CR algorithms implementation
When implementing Regression Monte Carlo algorithms, and choosing basis functions, the control on variance implies that low order polynomial can not be used alone, as they can easily cause the control to be bang-bang between the boundaries of its domain. Similarly, piecewise approximations are not very effective, as the dependence on the control is very weak, requiring a high number of local supports and making the computational complexity overwhelming. We tested both value and performance iteration and tried to employ different kinds of basis functions and training points. Unfortunately, both Regress-Later and Control Randomization do not cope well with controlling the dynamics of a process through the variance only. A tailor-made implementation of Regression Monte Carlo to deal with this kind of problems is outside the scope of this paper and further investigation will follow in future work. For now, we chose not to provide results based on RL and CR methods.

Table 4.2 – Portfolio Selection results. Estimations of the value function at point $(x_0 = 0, S_0 = 6)$ time 0 using the *continuous-time* optimal strategy (Opt) and (Q) estimated optimal strategy.

	$b_0 = 0.1$, $T = 1$		$b_0 = -0.1$, $T = 0.5$	
γ_0	Opt	Q	Opt	Q
0.1	-0.985	-0.985	-0.992	-0.992
0.2	-0.982	-0.982	-0.991	-0.991
0.3	-0.973	-0.973	-0.988	-0.988
0.4	-0.954	-0.953	-0.981	-0.981
0.5	-0.927	-0.927	-0.969	-0.969
0.6	-0.896	-0.896	-0.952	-0.952
0.7	-0.863	-0.863	-0.932	-0.932
0.8	-0.830	-0.830	-0.910	-0.910
0.9	-0.797	-0.797	-0.886	-0.886
1	-0.767	-0.766	-0.863	-0.863

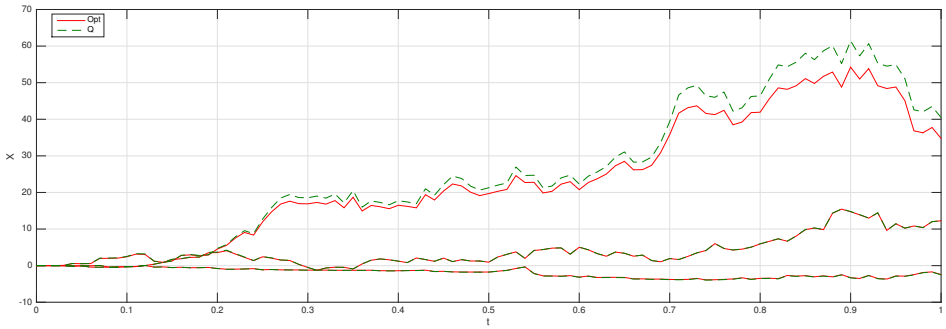


Figure 4.4 – 3 simulations of the agent's wealth $(X_t)_{t\in[0,T]}$ when the latter follows the continuous-time optimal strategy (Opt) and the (Q) estimated optimal strategy to solve the Portfolio Selection problem. We took σ=0.4, T=1, P=0.1, γ_0=5, b_0=0,1. One can see that the two strategies are the same when the drift is low; but Q performs slightly better than Opt when the drift is high, which is a time-discretization effect.

4.4.2 A model of interbank systemic risk with partial observation

4.4.2.1 The model

We consider the following model of systemic risk inspired by the model in [CFS15]. The log-monetary reserves of N banks lending to and borrowing from each other are governed by the system

$$dX_t^i = \frac{\kappa}{N}\sum_{j=1}^{N}(X_t^j - X_t^i)dt + \sigma X_t^i(\sqrt{1-\rho^2}dW_t^i + \rho dW_t^0), \quad i = 1,\dots,N$$

where W^i, $i = 1,\dots,N$, are independent Brownian motions, representing the idiosyncratic risk of each bank, W^0 is a common noise independent of W^i, $\sigma > 0$ is given real parameter, $\rho \in [-1,1]$, and where X_0^i, $i = 1,...,N$ are i.i.d.. The mean-reversion coefficient $\kappa > 0$ models the strength of interaction between the banks where bank i can lend to and borrow from bank j with an amount proportional to the difference between their reserves. In the asymptotic regime when $N \to \infty$, the theory of propagation of chaos implies that the reserve state X^i of individual banks become independent and identically distributed conditionally on the common noise W^0, with a state governed by

$$dX_t = \kappa(\mathbb{E}[X_t|W^0] - X_t)dt + \sigma X_t(\sqrt{1-\rho^2}dB_t + \rho dW_t^0)$$

for some Brownian motion B independent of W^0.

Let us now consider a central bank, viewed as a social planner, who only observes the common noise and not the reserves of each bank, and can influence the strength of the interaction between the individual banks, through an $\mathbb{F}^0$-adapted control process α_t. The reserve of the representative bank in the asymptotic regime is then driven by

$$dX_t = (\kappa+\alpha_t)(\mathbb{E}[X_t|W^0] - X_t)dt + \sigma X_t(\sqrt{1-\rho^2}dB_t + \rho dW_t^0), \quad X_0 \sim X_0^1,$$

and we consider that the objective of the central bank is to minimize

$$J(\alpha) = \mathbb{E}\Big[\int_0^T \Big(\frac{1}{2}\alpha_t^2 + \frac{\eta}{2}(X_t - \mathbb{E}[X_t|W^0])^2\Big)dt + \frac{c}{2}(X_T - \mathbb{E}[X_T|W^0])^2\Big],$$

where $\eta > 0$ and $c > 0$ penalize the departure of the reserve from the average. This is a MKV control problem under partial observation, but notice that it does not belong to the class of linear quadratic (LQ) MKV problems due to the control α which appears in a multiplicative form with the state. However, it fits into our class of polynomial MKV problem, and can be embedded into standard control problem as follows: We set $\bar{X}_t = \mathbb{E}[X_t|W^0]$ and $Y_t = \mathbb{E}[(X_t - \bar{X}_t)^2|W^0]$. The cost functional is then written as

$$J(\alpha) = \mathbb{E}\Big[\int_0^T \Big(\frac{1}{2}\alpha_t^2 + \frac{\eta}{2}Y_t\Big)dt + \frac{c}{2}Y_T\Big]$$

where the dynamics of $\bar{X}$ and Y are governed by

$$\begin{aligned} d\bar{X}_t &= \sigma\rho\bar{X}_t dW_t^0, \quad \bar{X}_0 = x_0 = \mathbb{E}[X_0], \\ dY_t &= \big[\big(\sigma^2 - 2(\kappa+\alpha_t)\big)Y_t + \sigma^2(1-\rho^2)\bar{X}_t^2\big]dt + 2\rho\sigma Y_t dW_t^0, \quad Y_0 = \mathrm{Var}(X_0). \end{aligned}$$

We have then reduced the problem to a $(\mathbb{P},\mathbb{F}^0)$-control problem, with state variables $(\bar{X},Y)$ of dimension two, which is not LQ but can be solved numerically.

4.4.2.2 Numerical results

For this problem, in the absence of analytical solution, we decided to compare the estimations of the value function at time 0 provided by our algorithms with a numerical approximation based on

finite difference scheme provided by Mathematica, of the solution to the 2-dimensional HJB equation associated to the systemic risk problem:

$$\begin{cases} \partial_t V + \frac{\eta}{2} y + \Big(\big(\sigma^2 - 2\kappa\big) y + \sigma^2 (1-\rho^2) x^2 \Big) \partial_y V + \sup_{a \in A} \left[\frac{1}{2} a^2 - 2ay \partial_y V \right] \\ \qquad\qquad + \frac{\sigma^2 \rho^2 x^2}{2} \partial^2_{xx} V + 2\sigma^2 \rho^2 xy \partial_{xy} V + 2\sigma^2 \rho^2 y^2 \partial^2_{yy} V = 0, \qquad \text{for } (t,x,y) \in [0,T) \times \mathbb{R} \times \mathbb{R}_+, \\ V(T,x,y) = \frac{c}{2} y, \quad \forall (x,y) \in \mathbb{R} \times \mathbb{R}_+ \,. \end{cases} \tag{4.4.7}$$

We refer to the solution of this partial differential equation (obtained using Mathematica using finite differences as explained below) as the Benchmark (or simply Bench) in the sequel.

We computed $\hat{\hat{V}}_{\Delta t}(t_0 = 0, x_0 = 10, y_0 = 0)$ using RL, CR and Q methods by considering a sample of size 500 000, and using the following parameters $T = 1$, $\sigma = 0.1$, $\kappa = 0.5$ and $X_0 = 10$. We recall that $\hat{\hat{V}}_{\Delta t}(t_0 = 0, x_0, y_0)$ is an estimation of $V(0, x_0 = 10, y_0 = 0)$, the value function at (x_0, y_0) and time 0 (see its definition on the last step of each pseudo-code presented in Subsection 4.3.6).

In Table 4.3 we display the numerical results of experiments run for two situations: we took $\eta = 10$, $c = 100$ and $\eta = 100$, $\rho = 0.5$ and vary the value of ρ in the first case, and vary the value of c in the second one. Plots of the two tables are also available in Figure 4.5. One can observe that the algorithms performs well. Mainly, Bench and Q provide slightly better results than the Regression Monte Carlo-based algorithms (the curves of Bench and Q are below those of the other two).

Figure 4.6 shows two examples of paths $(X_t)_{t\in[0,T]}$ controlled by RLMC (curve "RLMC"), $(X_t)_{t\in[0,T]}$ naively controlled by $\alpha = 0$ (curve "uncontrolled"), and the conditional expectation of X $(\bar{X}_t)_{t\in[0,T]}$ (curve "$E(X|W)$"). One can see in these two examples that the (RLMC estimated) optimal control is as follows:

- do nothing when the terminal time is far, i.e., take $\alpha = 0$, not to pay any running cost.
- catch $\bar{X}$ when the terminal time is getting close, to minimize the terminal cost.

We finally present a sample of paths $(Y_t)_{t\in[0,T]}$ controlled by the decisions given by Q in Figure 4.7. One can see that the (Q estimated) optimal strategy minimizes the running cost first by letting Y grow; and deals with the terminal cost later by making Y small when the terminal time is approaching.

Details on the RL and CR algorithms implementation

For the implementation of the RL algorithm we decided to use polynomial basis functions up to degree 2. This choice allows us to compute the optimal control analytically as a function of the regression coefficients (see Remark 4.3.9). Compared to other optimization techniques, explicit expression allows for much faster and error-free computations (see Remark 4.3.9). For CR, we used basis functions up to degree 3 in all dimensions to obtain more stable results.

Regarding the choice of the training measure in RL, we employed marginal normal distributions on each dimension. As we know that the inventory dimension Y represents the conditional variance of the original process X, we centered the training distribution μ_n at zero but considered only training points $Y_n^m \geq 0$. In CR, on the other hand, we need to carefully choose the distribution of the random control so that the process Y does not become negative. Notice in fact that the Euler approximation, contrary to the original SDE describing Y, does not remain positive and we would therefore need to carefully choose a control to avoid driving Y negative. In order to achieve such goal, without having to worry too much about the control, we modified the Euler approximation of (4.4.7) to feature a reflexive boundary at zero. Such features allow to train the estimated control policy to not overshoot when trying to drive the process Y to zero, without having Y to become negative.

Details on the Q algorithm implementation

ρ	RLMC	CR	Q	Bench
0.1	8.88	9.12	8.76	8.94
0.2	8.73	8.98	8.69	8.77
0.3	8.42	8.69	8.32	8.48
0.4	8.02	8.25	7.91	8.06
0.5	7.61	7.73	7.37	7.51
0.6	6.93	6.97	6.68	6.79
0.7	5.94	6.07	5.78	5.87
0.8	4.86	4.82	4.62	4.67
0.9	3.32	3.10	3.02	2.97

$c = 100$ and $\eta = 10$.

c	RLMC	CR	Q	Bench
0	7.79	7.78	7.77	7.79
1	7.88	7.87	7.86	7.88
5	8.22	8.23	8.21	8.23
10	8.63	8.64	8.61	8.62
25	9.69	9.76	9.61	9.62
50	11.08	11.27	10.94	10.97

$\rho = 0.5$ and $\eta = 100$.

Table 4.3 – Results for the systemic risk problem. Estimations of the value function at point ($x_0 = 10$) at time 0 provided by different strategies. We took $T = 1$, $N = 100$, $\sigma = 0.1$, $\kappa = 0.5$, $X_0 = 10$.

As stated above, it is straightforward that $Y > 0$ on $(0, T]$. However, the Euler scheme used to approximate the dynamics of Y does not prevent the associated process $(Y_{t_i})_{0<i\leq N}$ to be non-positive. When implementing the Q algorithm for the systemic risk problem, we forced $\left(\text{Proj}_{\Gamma_i^Y}\left(Y_{t_i}\right)\right)_{0<i\leq N}$ to remain positive by simply choosing positive points for the grids Γ_i^Y that quantize the states of Y_{t_i}, at time t_i for $i = 0, ..., N$.

Also, given the expression of the instantaneous and terminal reward, one can expect Y to stay close to 0, but we do not have any idea of how small Y should stay for the strategy to be optimal (cf. Figure 4.7 to see a posteriori where Y lies). To deal with this situation, we decided to adopt the "explore first, exploit later" procedure. First, we chose some random grids with a lot of points near 0 and computed the optimal strategy on these grids. Then, we ran forward Monte Carlo simulations and generated an empirical distribution of the quantized Y. Second, we build new grids of Quantization for Y by generating new points according to the empirical distribution that we got from in the previous step. Finally, we computed new (hopefully better) estimations of the optimal strategy by running the Q algorithm using the new grids. The Q strategy performed better after applying this step, but not significantly since our first naive guess for the grids (i.e., before bootstrapping) was already good enough.

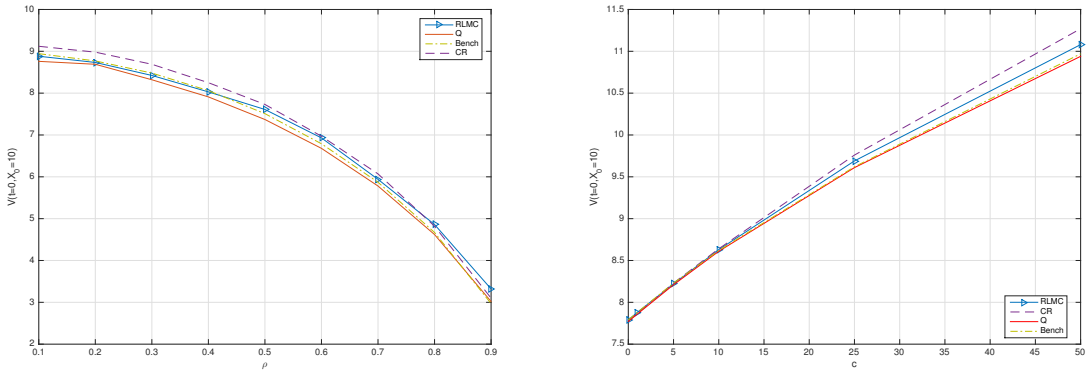
Details on the implementation of the deterministic algorithm for the resolution of the HJB

We use the `NDSolve` function in Mathematica based on finite difference method to solve (4.4.7). Note that usually terminal and boundary conditions are required to get numerical results. The final condition: $V(T, x, y) = \frac{c}{2}y$ is already given by (4.4.7). However, the boundary conditions on $V(t, 0, y)$ and $V(t, x, 0)$ are missing, except the trivial condition consisting of $V(t, 0, 0) = 0$. We then provided the HJB without boundary conditions to the Mathematica function `NDSolve`, and let the latter add artificial boundary conditions by itself to output results.

4.5 Conclusion

In this work, we have investigated how to use probabilistic numerical methods for some classes of mean field control problem via Markovian embedding. We focused on two types of Regression Monte Carlo methods (namely, Regress-Later and Control Randomization) and Quantization. We have then presented three different examples of applications.

We found that the Regression Monte Carlo algorithms perform well in problems of control of the drift. In such problems, they are much faster than Quantization for similar precision. In particular, we noticed that Regress-Later is usually more reliable than Control Randomization; often the choice of a uniform distribution of the training points on an appropriate interval is sufficient to obtain high-



$c = 100$ and $\eta = 10$. $\rho = 0.5$ and $\eta = 100$.

Figure 4.5 – Results for the systemic risk problem. Estimations of the value function at time 0 using different algorithms w.r.t. ρ and c. We took T=1, N=100, σ=0.1, κ=0.5, x_0=10.

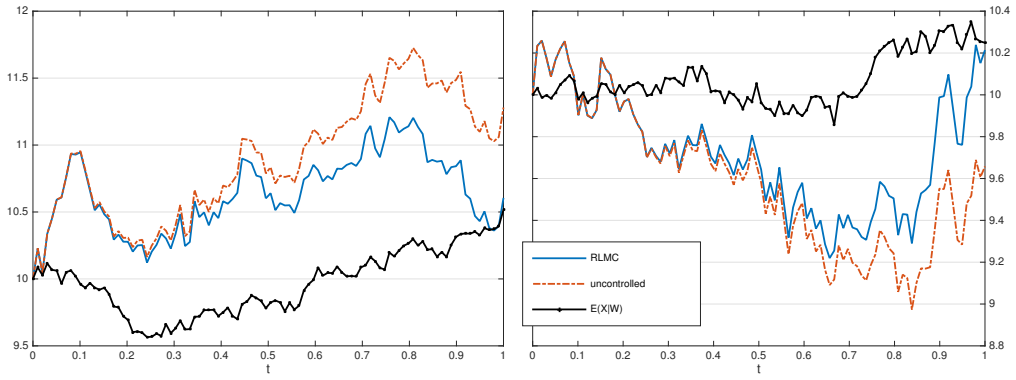


Figure 4.6 – Two realizations of $(X_t)_{t\in[0,T]}$ controlled by RLMC (curve "RLMC"), $(X_t)_{t\in[0,T]}$ naively controlled taken $\alpha = 0$ (curve "uncontrolled"), and $\bar{X}$ (curve "$E(X|W)$"). The optimal control for the systemic risk problem (computed by RLMC) is to do nothing at first, and catch $\bar{X}$ when the terminal time is getting close.

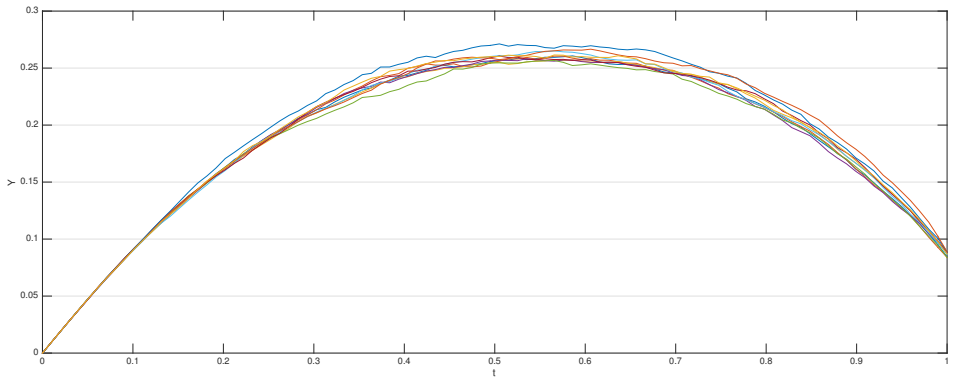


Figure 4.7 – Sample of $(Y_t)_{t\in[0,T]}$ controlled by Q. The (Q) estimated optimal control for the systemic risk problem is to initially let Y become large, and then reduce its value when the approaching the terminal time.

quality estimations. On the other hand Control Randomization is very sensitive to the choice of the distribution of the randomized control, and often a few iterations are necessary before finding a good control distribution. We have also tried to use the performance iteration or path recomputation method, but on the examples we considered, it was very time consuming and did not help much in terms of accuracy. Despite the success of Regression Monte Carlo methods in problems with control on the drift, the example of Portfolio Selection highlighted a possible weakness of these algorithms. When the control acts on the variance only, we found difficult to make the numerical scheme converge to sensible results within the computational resources available. We realized that the study of these problems and the solution via Regression Monte Carlo methods is outside the scope of this paper. This is probably related to another limitation of this family of methods: the choice of the basis functions for the regression. Indeed, for some problems, a good basis might be very large or might require several steps of trials and errors.

Quantization-based method, on the other hand, provided very stable and accurate results. A first interesting and practical feature of the Q-algorithm is that regressing the value function using quantization-based methods is local. So, first, it can be easily parallelized to provide fast results, and, second, it is easy to check at which points of the grids the estimations suffer from instability and how to change the grid to fix the problem (basically, by adding more points where the estimations need to be improved). Another interesting feature of the quantization methods is that, one can choose the grids on which to learn the value function. It is possible to exploit this feature in the case where one has, a priori, a rough idea of where the controlled process should be driven by the optimal strategy (see, e.g., the liquidation problem). In this case, one should build grids with many points located where the process is supposed to go. In the case where one has no guess of where the optimal process goes, it is always possible to use bootstrapping methods to build better grids iteratively, starting from a random guess for the grid (see, e.g., the systemic risk problem). In both cases, one has to be particularly careful with the borders of the grids that have been built. Indeed, the decisions computed by quantization-based methods at the borders might easily be wrong if the grids do not have a "good shape" at the borders. Unfortunately, the shape of the grid that should be used depends heavily on the problem under consideration. Except in special cases, it seems not possible to avoid the use of deterministic algorithms (such as gradient descent methods or extensive search) to find the optimal action at each point of the grid. A smooth expression of the conditional expectations of the quantized processes is necessary for the deterministic algorithms find optimal strategy efficiently. Once again, the use of parallel computing can alleviate the time-consuming task of searching for the optimal control at each point of the grids.

Acknowledgments. Most of this work was realized while the third author was a postdoctoral fellow at NYU Shanghai; the support of a research discretionary fund from the NYU-ECNU Institute of Mathematical Sciences and the support provided by the CEMRACS for his stay at CIRM are gratefully acknowledged. The authors's research is part of the ANR project CAESARS (ANR-15-CE05-0024).

Part II

Deep learning for Markov decision processes (MDPs)

Chapter 5

Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis[1]

Abstract This paper develops algorithms for high-dimensional stochastic control problems based on deep learning and dynamic programming. Unlike classical approximate dynamic programming approaches, we first approximate the optimal policy by means of neural networks in the spirit of deep reinforcement learning, and then the value function by Monte Carlo regression. This is achieved in the dynamic programming recursion by performance or hybrid iteration, and regress now or later/quantization methods from numerical probabilities. We provide a theoretical justification of these algorithms. Consistency and rate of convergence for the control and value function estimates are analyzed and expressed in terms of the universal approximation error of the neural networks, and of the statistical error when estimating network function. Numerical results on various applications are presented in a companion paper [BHLP18] and illustrate the performance of the proposed algorithms.

keywords Deep learning, dynamic programming, performance iteration, quantization, convergence analysis, statistical risk.

AMS 65C05, 90C39, 93E35

Contents

[1] This Chapter is based on a paper written in collaboration with Huyên Pham, Achref Bachouch and Nicolas Langrené.

5.1 Introduction

A large class of dynamic decision-making problems under uncertainty can be mathematically modeled as discrete-time stochastic optimal control problems in finite horizon. This paper is devoted to the analysis of novel probabilistic numerical algorithms based on neural networks for solving such problems. Let us consider the following discrete-time stochastic control problem over a finite horizon $N \in \mathbb{N} \setminus \{0\}$. The dynamics of the controlled state process $X^\alpha = (X_n^\alpha)_n$ valued in $\mathcal{X} \subset \mathbb{R}^d$ is given by

$$X_{n+1}^\alpha \quad = \quad F(X_n^\alpha, \alpha_n, \varepsilon_{n+1}), \quad n = 0, \ldots, N-1, \; X_0^\alpha = x_0 \in \mathbb{R}^d,$$

where $(\varepsilon_n)_n$ is a sequence of i.i.d. random variables valued in some Borel space $(E, \mathcal{B}(E))$, and defined on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$ equipped with the filtration $\mathbb{F} = (\mathcal{F}_n)_n$ generated by the noise $(\varepsilon_n)_n$ ($\mathcal{F}_0$ is the trivial σ-algebra), the control $\alpha = (\alpha_n)_n$ is an $\mathbb{F}$-adapted process valued in $\mathbb{A} \subset \mathbb{R}^q$, and F is a measurable function from $\mathbb{R}^d \times \mathbb{R}^q \times E$ into $\mathbb{R}^d$.

Given a running cost function f defined on $\mathbb{R}^d \times \mathbb{R}^q$, a terminal cost function g defined on $\mathbb{R}^d$, the cost functional associated to a control process α is

$$J(\alpha) \quad = \quad \mathbb{E}\left[\sum_{n=0}^{N-1} f(X_n^\alpha, \alpha_n) + g(X_N^\alpha)\right].$$

The set $\mathcal{C}$ of admissible control is the set of control processes α satisfying some integrability conditions ensuring that the cost functional $J(\alpha)$ is well-defined and finite. The control problem, also called Markov decision process (MDP), is formulated as

$$V_0(x_0) \quad := \quad \inf_{\alpha \in \mathcal{C}} J(\alpha),$$

and the goal is to find an optimal control $\alpha^* \in \mathcal{C}$, i.e., attaining the optimal value: $V_0(x_0) = J(\alpha^*)$. Notice that problem (5.1.1)-(5.1.2) may also be viewed as the time discretization of a continuous time stochastic control problem, in which case, F is typically the Euler scheme for a controlled diffusion process, and V_0 is the discrete-time approximation of a fully nonlinear Hamilton-Jacobi-Bellman equation.

Problem (5.1.2) is tackled by the dynamic programming approach, and we introduce the standard notations for MDP: denote by $\{P^a(x, dx'),\ a \in \mathbb{A},\ x \in \mathcal{X}\}$, the family of transition probabilities associated to the controlled (homogenous) Markov chain (5.1.1), given by

$$P^a(x, dx') \quad = \quad \mathbb{P}\left[F(x, a, \varepsilon_1) \in dx'\right]$$

and for any measurable function φ on $\mathcal{X}$:

$$P^a\varphi(x) = \int \varphi(x')P^a(x,dx') \quad = \quad \mathbb{E}\left[\varphi\big(F(x,a,\varepsilon_1)\big)\right].$$

With these notations, we have for any measurable function φ on $\mathcal{X}$, for any $\alpha \in \mathcal{C}$,

$$\mathbb{E}[\varphi(X^\alpha_{n+1})|\mathcal{F}_n] \quad = \quad P^{\alpha_n}\varphi(X^\alpha_n), \quad \forall\, n \in \mathbb{N}.$$

The optimal value $V_0(x_0)$ is then determined in backward induction starting from the terminal condition

$$V_N(x) \quad = \quad g(x), \quad x \in \mathcal{X},$$

and by the dynamic programming (DP) formula, for $n = N-1, \ldots, 0$:

$$\left\{ \begin{array}{rcl} Q_n(x,a) & = & f(x,a) + P^a V_{n+1}(x), \quad x \in \mathcal{X},\ a \in \mathbb{A}, \\ V_n(x) & = & \inf\limits_{a\in\mathbb{A}} Q_n(x,a), \end{array} \right.$$

The function Q_n is called optimal state-action value function, and V_n is the (optimal) value function. Moreover, when the infimum is attained in the DP formula at any time n by $a^*_n(x)$, we get an optimal control in feedback form given by: $\alpha^* = (a^*_n(X^*_n))_n$ where $X^* = X^{\alpha^*}$ is the Markov process defined by

$$X^*_{n+1} \quad = \quad F(X^*_n, a^*_n(X^*_n), \varepsilon_{n+1}), \quad n = 0, \ldots, N-1, \ \ X^*_0 = x_0.$$

The DP has a probabilistic formulation: it says that for any control $\alpha \in \mathcal{A}$, the value function process augmented with the cumulative costs defined by

$$\left\{S^\alpha_n \ := \ V_n(X^\alpha_n) + \sum_{k=0}^{n-1} f(X^\alpha_k, \alpha_k),\ n = 1, \ldots, N\right\}$$

is a submartingale, and a martingale for the optimal control α^*. This martingale property for the optimal control is a key observation for our algorithms described later.

Remark 5.1.1. We can deal with state/control constraints at any time, which is useful for the applications:

$$(X^\alpha_n, \alpha_n) \quad \in \quad \mathcal{S} \quad a.s., \quad n \in \mathbb{N},$$

where $\mathcal{S}$ is some given subset of $\mathbb{R}^d \times \mathbb{R}^q$. In this case, in order to ensure that the set of admissible controls is not empty, we assume that the sets

$$\mathbb{A}(x) \quad := \quad \Big\{a \in \mathbb{R}^q : (F(x,a,\varepsilon_1), a) \in \mathcal{S}\ a.s.\Big\}$$

are non empty for all $x \in \mathcal{X}$, and the DP formula reads now as

$$V_n(x) \quad = \quad \inf_{a\in\mathbb{A}(x)} \big[f(x,a) + P^a V_{n+1}(x)\big], \quad x \in \mathcal{X}.$$

in the finite horizon case, and

$$V(x) \quad = \quad \inf_{a\in\mathbb{A}(x)} \big[f(x,a) + P^a V(x)\big], \quad x \in \mathcal{X},$$

From a computational point of view, it may be more convenient to work with unconstrained state and control variable, hence by relaxing the state and control constraint and introducing into the running cost 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 $\mathcal{S}$ is in the form: $\mathcal{S} = \{(x,a) \in \mathbb{R}^d \times \mathbb{R}^q : h_k(x,a) = 0, k = 1, \ldots, p,\ h_k(x,a) \geq 0, k = p+1, \ldots, q\}$, for some functions h_k, then one can take as penalty functions:

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

where $\mu_k > 0$ are penalization coefficients (large in practice). □

The implementation of the DP formula requires the knowledge and explicit computation of the transition probabilities $P^a(x, dx')$. In situations when they are unknown, this leads to the problematic of reinforcement learning for computing the optimal control and value function by relying on simulations of the environment. The challenging tasks from a numerical point of view are then twofold:

1. *Transition probability operator.* Calculations for any $x \in \mathcal{X}$, and any $a \in \mathbb{A}$, of $P^a V_{n+1}(x)$, for $n = 0, \ldots, N-1$. This is a computational challenge in high dimension d for the state space with the "curse of dimensionality" due to the explosion of grid points in deterministic methods.

2. *Optimal control.* Computation of the infimum in $a \in \mathbb{A}$ of $f(x, a) + P^a V_{n+1}(x)$ for fixed x and n, and of $\hat{a}_n(x)$ attaining the minimum if it exists. This is also a computational challenge especially in high dimension q for the control space.

The classical probabilistic numerical methods based on DP for solving the MDP are sometimes called approximate dynamic programming methods, see e.g. [PBT96], [Pow11], and consist basically of the two following steps:

(i) Approximate at each time step n the Q_n value function defined as a conditional expectation. This can be performed by regression Monte-Carlo (RMC) techniques or quantization. RMC is typically done by least-square linear regression on a set of basis function following the popular approach by Longstaff and Schwarz [LS01] initiated for Bermudean option problem, where the suitable choice of basis functions might be delicate. Conditional expectation can be also approximated by regression on neural network as in [KKT10] for American option problem, and appears as a promising and efficient alternative in high dimension to the linear regression. The main issue in the controlled case concerns the simulation of the endogenous controlled MDP, and this can be overcome by control randomization as in [KLP14]. Alternatively, quantization method consists in approximating the noise (ε_n) by a discrete random variable on a finite grid, in order to reduce the conditional expectation to a finite sum.

(ii) Control search: Once we get an approximation $(x, a) \mapsto \hat{Q}_n(x, a)$ of the Q_n value function, the optimal control $\hat{a}_n(x)$ which achieves the minimum over $a \in \mathbb{A}$ of $Q_n(x, a)$ can be obtained either by an exhaustive search when $\mathbb{A}$ is discrete (with relatively small cardinality), or by a (deterministic) gradient-based algorithm for continuous control space (with relatively small dimension).

Recently, numerical methods by direct approximation, without DP, have been developed and made implementable thanks to the power of computers: the basic idea is to focus directly on the control approximation by considering feedback control (policy) in a parametric form:

$$a_n(x) \quad = \quad A(x; \theta_n), \quad n = 0, \ldots, N-1,$$

for some given function $A(., \theta_n)$ with parameters $\theta = (\theta_0, \ldots, \theta_{N-1}) \in \mathbb{R}^{q \times N}$, and minimize over θ the parametric functional

$$\tilde{J}(\theta) \quad = \quad \mathbb{E}\left[\sum_{n=0}^{N-1} f(X_n^A, A(x; \theta_n)) + g(X_N^A)\right],$$

where $(X_n^A)_n$ denotes the controlled process with feedback control $(A(., \theta_n))_n$. This approach was first adopted in [KPX16], who used EM algorithm for optimizing over the parameter θ, and further investigated in [HE16], [EHJ17], [HL17], who considered deep neural networks (DNN) for the parametric feedback control, and stochastic gradient descent methods (SGD) for computing the optimal parameter θ. The theoretical foundation of these DNN algorithms has been recently investigated in [HL18]. Deep learning has emerged recently in machine learning as a successful technique for dealing with high-dimensional problems in speech recognition, computer vision, etc (see e.g. [LBH15], [GBC16]). Let us mention that DNN approximation in stochastic control has already been explored in

the context of reinforcement learning (RL) (see [PBT96] and [SB98]), and called deep reinforcement learning in the artificial intelligence community [MKSR15] (see also [Li17] for a recent survey) but usually for infinite horizon (stationary) control problems.

In this paper, we combine different ideas from the mathematics (numerical probability) and the computer science (reinforcement learning) communities to propose and compare several algorithms based on dynamic programming (DP), and deep neural networks (DNN) for the approximation/learning of (i) the optimal policy, and then of (ii) the value function. Notice that this differs from the classical approach in DP recalled above, where we first approximate the Q-optimal state/control value function, and then approximate the optimal control. Our learning of the optimal policy is achieved in the spirit of [HE16] by DNN, but sequentially in time though DP instead of a global learning over the whole period $0, \ldots, N-1$. Once we get an approximation of the optimal policy, and recalling the martingale property (5.1.3), we approximate the value function by Monte-Carlo (MC) regression based on simulations of the forward process with the approximated optimal control. In particular, we avoid the issue of *a priori* endogenous simulation of the controlled process in the classical Q-approach. The MC regressions for the approximation of the optimal policy and/or value function, are performed according to different features leading to algorithmic variants: Performance iteration (PI) or hybrid iteration (HI), and regress now or regress later/quantization in the spirit of [LS01] or [GY04]. Numerical results on several applications are devoted to a companion paper [BHLP18]. The theoretical contribution of the current paper is to provide a detailed convergence analysis of our three proposed algorithms: Theorem 5.4.1 for the *NNContPI* Algo based on control learning by performance iteration with DNN, Theorem 5.4.2 for the *Hybrid-Now* Algo based on control learning by DNN and then value function learning by regress-now method, and Theorem 5.4.3 for the *Hybrid-LaterQ* Algo based on control learning by DNN and then value function learning by regress later method combined with quantization. We rely mainly on arguments from statistical learning and non parametric regression as developed notably in the book [GKKW02], for giving estimates of approximated control and value function in terms of the universal approximation error of the neural networks, and of the statistical error in the estimation of network functions.

The plan of this paper is organized as follows. We recall in Section 5.2 some basic results about deep neural networks (DNN) and stochastic optimization gradient descent methods used in DNN. Section 5.3 is devoted to the description of our three algorithms. We analyze in detail in Section 5.4 the convergence of the three algorithms. Finally the Appendix collect some Lemmas used in the proof of the convergence results.

5.2 Preliminaries on DNN and SGD

5.2.1 Neural network approximations

Deep Neural networks (DNN) aim to approximate (complex non linear) functions defined on finite-dimensional space, and in contrast with the usual additive approximation theory built via basis functions, like polynomial, they rely on composition of layers of simple functions. The relevance of neural networks comes from the universal approximation theorem and the Kolmogorov-Arnold representation theorem (see [Kol91], [Cyb89] or [Hor91]), and this has shown to be successful in numerous practical applications.

We consider here feedforward artificial network (also called multilayer perceptron) for the approximation of the optimal policy (valued in $\mathbb{A} \subset \mathbb{R}^q$) and the value function (valued in $\mathbb{R}$), both defined on the state space $\mathcal{X} \subset \mathbb{R}^d$. The architecture is depicted in Figure 5.1, and it is mathematically represented by functions

$$x \in \mathcal{X} \longmapsto \Phi(z;\theta) \in \mathbb{R}^o,$$

with $o = q$ or 1 in our context, and where $\theta \in \Theta \subset \mathbb{R}^p$ are the weights (or parameters) of the neural networks. The DNN function $\Phi = \Phi_L$ with input layer $\Phi_0 = (\Phi_0^i)_i = x \in \mathcal{X}$ composed of d units (or neurons), $L-1$ hidden layers (with layer ℓ composed of d_ℓ units), and output layer composed of

$d_L = o$ neurons is obtained by successive composition of linear combination and activation function σ_ℓ (that is a nonlinear monotone function like e.g. the sigmoid, the rectified linear unit ReLU, the exponential linear unit ELU, or the softmax):

$$\Phi_\ell \quad = \quad \sigma_\ell(w_\ell \Phi_{\ell-1} + \gamma_\ell) \ \in \ \mathbb{R}^{d_\ell}, \ \ell = 1, \dots, L,$$

for some matrix weights (w_ℓ) and vector weight (γ_ℓ), aggregating into $\theta = (w_\ell, \gamma_\ell)_{\ell=1}^L$. A key feature of neural networks is the computation of the gradient (with respect to the variable x and the weights θ) of the DNN function via a forward-backward propagation algorithm derived from chain rule composition. For example, for the sigmoid activation function $\sigma_\ell(y) = 1/(1+e^{-y})$, and noting that $\sigma'_\ell = \sigma_\ell(1-\sigma_\ell)$, we have

$$\left[\frac{\partial \Phi_\ell}{\partial z}\right]_{ij} \quad = \quad \left[w_\ell \frac{\partial \Phi_{\ell-1}}{\partial z}\right]_{ij} \Phi^i_\ell(1-\Phi^i_\ell), \ \ \ell = 1, \dots, L, \ i = 1, \dots, d_\ell, \ j = 1, \dots, d$$

while the gradient w.r.t. θ of $\mathcal{K}(\theta) = K(\Phi_L(.;\theta))$, for a real-valued differentiable function $y \in \mathbb{R}^{d_L} \mapsto K(y)$, is given in backward induction by

$$\Delta^\ell_i \quad := \quad \left[\frac{\partial \mathcal{K}}{\partial \Phi_\ell}\right]_i \Phi^i_\ell(1-\Phi^i_\ell), \quad \ell = L, \dots, 1, \ i = 1, \dots, d_\ell$$

$$\left[\frac{\partial \mathcal{K}}{\partial w_\ell}\right]_{ij} \quad = \quad \Phi^j_{\ell-1}\Delta^\ell_i, \ \left[\frac{\partial \mathcal{K}}{\partial \gamma_\ell}\right]_i \ = \ \Delta^\ell_i, \ \left[\frac{\partial \mathcal{K}}{\partial \Phi_{\ell-1}}\right]_j \ = \ \sum_{k=1}^{d_\ell} \Delta^\ell_k w^{kj}_\ell, \quad j = 1, \dots, d_{\ell-1}.$$

We refer to the online book [Nie] for a gentle introduction to neural networks and deep learning.

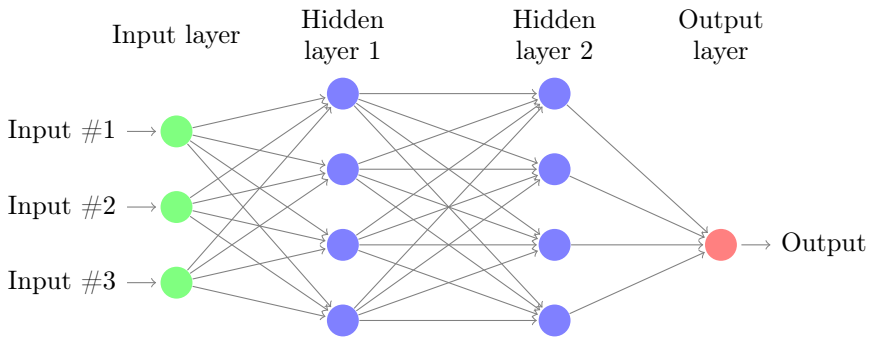


Figure 5.1 – Representation of a neural network with $d = 3$, 2 hidden layers, $d_1 = d_2 = 4$, $d_3 = 1$.

5.2.2 Stochastic optimization in DNN

Approximation by means of DNN requires a stochastic optimization with respect to a set of parameters, which can be written in a generic form as

$$\inf_\theta \mathbb{E}\big[L_n(Z_n;\theta)\big],$$

where Z_n is a random variable from which the training samples $Z_n^{(m)}$, $m = 1, \dots, M$ are drawn, and L_n is a loss function involving DNN with parameters $\theta \in \mathbb{R}^p$, and typically differentiable w.r.t. θ with known gradient $D_\theta L_n$.

Several basic algorithms are already implemented in TensorFlow for the search of infimum in (5.2.1). Given a training sample of size M, in all the following cases, the sequence $(\theta^k_n)_{k\in\mathbb{N}}$ tends to $\theta_n = \operatorname{argmin}_\theta \mathbb{E}\big[L_n(Z_n;\theta)\big]$ under suitable assumptions on the learning rate sequence $(\gamma_k)_{k=0}^\infty$.

- Batch gradient descent: (compute the gradient over the full training set). Fix an integer K, and do

$$\theta_n^{k+1} = \theta_n^k - \gamma_k \frac{1}{M} \sum_{m=1}^{M} D_\theta L_n(Z_n^{(m)}; \theta_n^k), \qquad \text{for } k = 1, \dots, K.$$

  The main problem with the Batch Gradient Descent is that the convergence is very slow and also the computation of the sum can be painful for very large training sets. Hence it makes it very stable, but too slow in most situations.

- Stochastic gradient descent (SGD): (compute the gradient over one random instance in the training set)

$$\theta_n^{m+1} = \theta_n^m - \gamma_m D_\theta L_n(Z_n^{(m)}; \theta_n^m), \quad m = 1, \dots, M-1.$$

  starting from $\theta_n^0 \in \mathbb{R}^p$, with a learning rate γ_m. The Stochastic gradient algorithm computes the gradient based on a single random instance in the training set. It is then a fast but unstable algorithm.

- Mini-batch gradient descent: (compute the gradient over random small subsets of the training set, i.e. mini-batches) let Mb be an integer than divides M. Mb stands for the number of mini-batches and should be taken much smaller than M in the applications.
  For all $k, \dots, Mb$,

  - Randomly draw a subset $\left(Z_n^{(k,m)}\right)_{m=1}^{M_{k+1}}$ of size $M_{k+1} := \frac{M}{Mb}$ in the training set.
  - iterate: $\theta_n^{k+1} = \theta_n^k - \gamma_k \frac{1}{M_{k+1}} \sum_{m=1}^{M_{k+1}} D_\theta L_n(Z_n^{(m)}; \theta_n^k)$.

  The mini-batch gradient descent is often considered to be the best trade-off between speed and stability.

The three gradient descents that we just introduced are the first three historical algorithms that has been designed to learn optimal parameters. Other methods such as the Adaptive optimization methods AdaGrad, RMSProp, and finally Adam are also available. Although not well-understood and even questioned (see e.g. [Wil+17]), the latter are often chosen by the practitioners to solve (5.2.1) and appear to provide the best results in most of the situations.
For sake of simplicity, we only refer in the sequel to the stochastic gradient descent method, when presenting our algorithms. However, we recommend to test and use different algorithms in order to know which are the ones that provide best and fastest results for a given problem.

5.3 Description of the algorithms

We propose algorithms relying on a DNN approximation of the optimal policy that we compute sequentially in time through the dynamic programming formula, and using performance or hybrid iteration. The value function is then computed by Monte-Carlo regression either by a regress now method or a regress later joint with quantization approach. These variants lead to three algorithms for MDP that we detail in this section.

Let us introduce a set $\mathcal{A}$ of neural networks for approximating optimal policies, that is a set of parametric functions $x \in \mathcal{X} \mapsto A(x;\beta) \in \mathbb{A}$, with parameters $\beta \in \mathbb{R}^l$, and a set $\mathcal{V}$ of neural networks functions for approximating value functions, that is a set of parametric functions $x \in \mathcal{X} \mapsto \Phi(x;\theta) \in \mathbb{R}$, with parameters $\theta \in \mathbb{R}^p$.

We are also given at each time n a probability measure μ_n on the state space $\mathcal{X}$, which we refer to as a training distribution. Some comments about the choice of the training measure are discussed in Section 5.3.3.

5.3.1 Control learning by performance iteration

This algorithm, refereed in short as *NNcontPI* Algo, is designed as follows:

• For $n = N-1, \dots, 0$, we keep track of the approximated optimal policies $\hat{a}_k$, $k = n+1, \dots, N-1$, and approximate the optimal policy at time n by $\hat{a}_n = A(.; \hat{\beta}_n)$ with

$$\hat{\beta}_n \quad \in \quad \arg\min_{\beta \in \mathbb{R}^l} \mathbb{E}\left[f(X_n, A(X_n;\beta)) + \sum_{k=n+1}^{N-1} f(\hat{X}_k^\beta, \hat{a}_k(\hat{X}_k^\beta)) + g(\hat{X}_N^\beta) \right],$$

where $X_n \rightsquigarrow \mu_n$, $\hat{X}_{n+1}^\beta = F(X_n, A(X_n;\beta), \varepsilon_{n+1}) \rightsquigarrow P^{A(X_n;\beta)}(X_n, dx')$, and for $k = n+1, \dots, N-1$, $\hat{X}_{k+1}^\beta = F(\hat{X}_k^\beta, \hat{a}_k(\hat{X}_k^\beta), \varepsilon_{k+1}) \rightsquigarrow P^{\hat{a}_k(\hat{X}_k^\beta)}(\hat{X}_k^\beta, dx')$. Given estimate $\hat{a}_k^M$ of $\hat{a}_k$, $k = n+1, \dots, N-1$, the approximated policy $\hat{a}_n$ is estimated by using a training sample $\left(X_n^{(m)}, (\varepsilon_{k+1}^{(m)})_{k=n}^{k=N-1}\right)$, $m = 1, \dots, M$ of $\left(X_n, (\varepsilon_{k+1})_{k=n}^{k=N-1}\right)$ for simulating $\left(X_n, (\hat{X}_{k+1}^\beta)_{k=n}^{k=N-1}\right)$, and optimizing over the parameters $\beta \in \mathbb{R}^l$ of the NN $A(.;\beta) \in \mathcal{A}$, the expectation in (5.3.1) by stochastic gradient descent method (or its variants) as described in Section (5.2.2).

▶ We then get an estimate of the optimal policy at any time $n = 0, \dots, N-1$ by:

$$\hat{a}_n^M \quad = \quad A(.; \hat{\beta}_n^M) \ \in \ \mathcal{A},$$

where $\hat{\beta}_n^M$ is the "optimal" parameter resulting from the SGD in (5.3.1) with a training sample of size M. This leads to an estimated value function given at any time n by

$$\hat{V}_n^M(x) \quad = \quad \mathbb{E}_M\left[\sum_{k=n}^{N-1} f(\hat{X}_k^{n,x}, \hat{a}_k^M(\hat{X}_k^{n,x})) + g(\hat{X}_N^{n,x}) \right],$$

where $\mathbb{E}_M$ is the expectation conditioned on the training set (used for computing $\left(\hat{a}_k^M\right)_k$), and $\left(\hat{X}_k^{n,x}\right)_{k=n}^N$, is given by: $\hat{X}_n^{n,x} = x$, $\hat{X}_{k+1}^{n,x} \rightsquigarrow P^{\hat{a}_k^M(\hat{X}_k^{n,x})}(\hat{X}_k^{n,x}, dx')$, $k = n, \dots, N-1$. The dependence of the estimated value function $\hat{V}_n^M$ upon the training samples $X_k^{(m)}$, for $m = 1, \dots, M$, used at time $k = n, \dots, N$, is emphasized through the exponent M in the notations.

Remark 5.3.1. The *NNcontPI* Algo can be viewed as a combination of the DNN algorithm designed in [HE16] and dynamic programming. In the algorithm presented in [HE16], which totally ignores the dynamic programming principle, one learns all the optimal controls $A(.;\beta_n)$, $n = 0, \dots, N-1$ at the same time, by performing one unique stochastic gradient descent. This is efficient as all the parameters of all the NN are getting trained at the same time, using the same mini-batches. However, when the number of layers of the global neural network gathering all the NN $A(.;\beta_n)$, $n = 0, \dots, N-1$ is large (say $\sum_{n=0}^{N-1} \ell_n \geq 100$, where ℓ_n is the number of layers in $A(.,\beta_n)$), then one is likely to observe vanishing or exploding gradient problems that will affect the training of the weights and biais of the first layers of the global NN (see [Gí7] for more details). Therefore, it may be more reasonable to make use of the dynamic programming structure when N is large, and learn the optimal policy sequentially as proposed in our *NNcontPI* Algo. Notice that a similar idea was already used in [GHL10] in the context of uncertain volatility model where the authors use a specific parametrization for the feedback control instead of a DNN adopted more generally here. □

Remark 5.3.2. The *NNcontPI* Algo does not require value function iteration, but instead is based on performance iteration by keeping track of the estimated optimal policies computed in backward recursion. The value function is then computed in (5.3.2) as the gain functional associated to the estimated optimal policies $(\hat{a}_k^M)_k$. Consequently, it provides usually a low bias estimate but induces possibly high variance estimate and large complexity, especially when N is large. □

5.3.2 Control learning by hybrid iteration

Instead of keeping track of all the approximated optimal policies as in the *NNcontPI* Algo, we use an approximation of the value function at time $n+1$ in order to compute the optimal policy at time n. The approximated value function is then updated at time n by relying on the martingale property (5.1.3) under the optimal control. This leads to the following generic algorithm:

Generic Hybrid Algo

1. *Initialization*: $\hat{V}_N = g$
2. For $n = N-1, \ldots, 0$,
   (i) Approximate the optimal policy at time n by $\hat{a}_n = A(.;\hat{\beta}_n)$ with

$$\hat{\beta}_n \quad \in \quad \arg\min_{\beta\in\mathbb{R}^l} \mathbb{E}\left[f(X_n, A(X_n;\beta)) + \hat{V}_{n+1}(X_{n+1}^{A(.,\beta)})\right],$$

   where $X_n \rightsquigarrow \mu_n$, $\hat{X}_{n+1}^{A(.,\beta)} = F(X_n, A(X_n;\beta), \varepsilon_{n+1}) \rightsquigarrow P^{A(X_n;\beta)}(X_n, dx')$.
   (ii) *Updating*: approximate the value function by

$$\hat{V}_n(x) \quad = \quad \mathbb{E}\left[f(X_n, \hat{a}_n(X_n)) + \hat{V}_{n+1}(X_{n+1}^{\hat{a}_n})|X_n = x\right].$$

The approximated policy $\hat{a}_n$ is estimated by using a training sample $(X_n^{(m)}, \varepsilon_{n+1}^{(m)})$, $m = 1, \ldots, M$ of (X_n, ε_{n+1}) to simulate $\left(X_n, X_{n+1}^{A(.;\beta)}\right)$, and optimizing over the parameters $\beta \in \mathbb{R}^l$ of the NN $A(.;\beta)$ $\in \mathcal{A}$, the expectation in (5.3.3) by stochastic gradient descent method (or its variants) as described in Section (5.2.2). We then get an estimate $\hat{a}_n^M = A\left(.;\hat{\beta}_n^M\right)$. The approximated value function written as a conditional expectation in (5.3.4) is estimated according to a Monte Carlo regression, either by a regress now method (in the spirit of [KKT10]) or a regress later (in the spirit of [GY04] and [BP18]) joint with quantization approach, and this leads to the following algorithmic variants detailed in the two next paragraphs.

5.3.2.1 Hybrid-Now Algo

Given an estimate $\hat{a}_n^M$ of the optimal policy at time n, and an estimate $\hat{V}_{n+1}^M$ of $\hat{V}_{n+1}$, we estimate $\hat{V}_n$ by neural networks regression, i.e.,

$$\hat{V}_n^M \quad \in \quad \arg\min_{\Phi(.;\theta)\in\mathcal{V}} \mathbb{E}\left|f(X_n, \hat{a}_n^M(X_n)) + \hat{V}_{n+1}^M(X_{n+1}^{\hat{a}_n^M}) \; - \; \Phi(X_n;\theta)\right|^2$$

using samples $X_n^{(m)}$, $X_{n+1}^{\hat{a}_n^M,(m)}$, $m = 1, \ldots, M$ of $X_n \rightsquigarrow \mu_n$, and $X_{n+1}^{\hat{a}_n^M,(m)}$ of $X_{n+1}^{\hat{a}_n^M}$. In other words, we have

$$\hat{V}_n^M \quad = \quad \Phi\left(.;\hat{\theta}_n^M\right),$$

where $\hat{\theta}_n^M$ is the "optimal" parameter resulting from the SGD in (5.3.5) with a training sample of size M.

5.3.2.2 Hybrid-LaterQ Algo

Given an estimate $\hat{a}_n^M$ of the optimal policy at time n, and an estimate $\hat{V}_{n+1}^M$ of $\hat{V}_{n+1}$, the regress-later approach for estimating $\hat{V}_n$ is achieved in two stages: (a) we first regress/interpolate the estimated value $\hat{V}_{n+1}^M\left(X_{n+1}^{\hat{a}_n^M}\right)$ at time $n+1$ by a NN (or alternatively a Gaussian process) $\Phi(X_{n+1}^{\hat{a}_n^M})$, (b) Analytical formulae are applied to the conditional expectation of this NN of future values $X_{n+1}^{\hat{a}_n^M}$ with respect to the present value X_n, and this is obtained by quantization of the noise (ε_n) driving the dynamics (5.1.1) of the state process.

The ingredients of the quantization approximation are described as follows:

- We denote by $\hat{\varepsilon}$ a K-quantizer of the E-valued random variable $\varepsilon_{n+1} \rightsquigarrow \varepsilon_1$ (typically a Gaussian random variable), that is a discrete random variable on a grid $\Gamma = \{e_1, \ldots, e_K\} \subset E^K$ defined by

$$\hat{\varepsilon} \;=\; \mathrm{Proj}_\Gamma(\varepsilon_1) \;:=\; \sum_{\ell=1}^{K} e_\ell 1_{\varepsilon_1 \in C_\ell(\Gamma)},$$

  where $C_1(\Gamma)$, ..., $C_K(\Gamma)$ are Voronoi tesselations of Γ, i.e., Borel partitions of the Euclidian space $(E, |.|)$ satisfying

$$C_\ell(\Gamma) \;\subset\; \Big\{e \in E : |e - e_\ell| \;=\; \min_{j=1,\ldots,K} |e - e_j|\Big\}.$$

  The discrete law of $\hat{\varepsilon}$ is then characterized by

$$\hat{p}_\ell \;:=\; \mathbb{P}[\hat{\varepsilon} = e_\ell] \;=\; \mathbb{P}[\varepsilon_1 \in C_\ell(\Gamma)], \quad \ell = 1, \ldots, K.$$

  The grid points (e_ℓ) which minimize the L^2-quantization error $\|\varepsilon_1 - \hat{\varepsilon}\|_2$ lead to the so-called optimal L-quantizer, and can be obtained by a stochastic gradient descent method, known as Kohonen algorithm or competitive learning vector quantization (CLVQ) algorithm, which also provides as a byproduct an estimation of the associated weights $(\hat{p}_\ell)$. We refer to [PPP04c] for a description of the algorithm, and mention that for the normal distribution, the optimal grids and the weights of the Voronoi tesselations are precomputed on the website http://www.quantize.maths-fi.com

- Recalling the dynamics (5.1.1), the conditional expectation operator is equal to

$$P^{\hat{a}_n^M(x)}W(x) \;=\; \mathbb{E}\big[W(X_{n+1}^{\hat{a}_n^M})|X_n = x\big] \;=\; \mathbb{E}\big[W(F(x, \hat{a}_n^M(x), \varepsilon_1))\big], \quad x \in \mathcal{X},$$

  that we shall approximate analytically by quantization via:

$$\widehat{P}^{\hat{a}_n^M(x)}W(x) \;:=\; \mathbb{E}\big[W(F(x, \hat{a}_n^M(x), \hat{\varepsilon}))\big] \;=\; \sum_{\ell=1}^{K} \hat{p}_\ell W\left(F(x, \hat{a}_n^M(x), e_\ell)\right).$$

The two stages of the regress-later are then detailed as follows:

(a) *(Later) interpolation of the value function*: Given a DNN $\Phi(.;\theta)$ on $\mathbb{R}^d$ with parameters $\theta \in \mathbb{R}^p$, we interpolate $\hat{V}_{n+1}^M$ by

$$\widetilde{V}_{n+1}^M(x) \;:=\; \Phi\left(x; \theta_{n+1}^M\right),$$

where θ_{n+1}^M is obtained via SGD (as described in paragraph 5.2.2) from the regression of $\hat{V}_{n+1}^M(X_{n+1}^{\hat{a}_n^M})$ against $\Phi\left(X_{n+1}^{\hat{a}_n^M}; \theta\right)$, using training samples $X_n^{(m)}$, $X_{n+1}^{\hat{a}_n^M,(m)}$, $m = 1, \ldots, M$ of $X_n \rightsquigarrow \mu_n$, and $X_{n+1}^{\hat{a}_n^M,(m)}$ of $X_{n+1}^{\hat{a}_n^M}$.

(b) *Updating/approximation of the value function*: by using the hat operator in (5.3.6) for the approximation of the conditional expectation by quantization, we calculate analytically

$$\hat{V}_n^M(x) \quad := \quad f(x,a) + \widehat{P}^{\hat{a}_n^M} \widetilde{V}_{n+1}^M(x) \;=\; f(x,a) + \sum_{\ell=1}^{K} \hat{p}_\ell \Phi\left(F(x, \hat{a}_n^M(x), e_\ell); \theta_{n+1}^M\right).$$

Remark 5.3.3. Let us discuss and compare the Algos Hybrid-Now and Hybrid-LaterQ. When regressing later, one just has to learn a deterministic function through the interpolation step (a), as the noise is then approximated by quantization for getting analytical formula. Therefore, compared to Hybrid-Now, the Hybrid-LaterQ Algo reduces the variance of the estimate $\hat{V}_n^M$. Moreover, one has a wide choice of loss functions when regressing later, e.g., MSE loss function, $L1$-loss, relative error loss, etc, while the $L2$-loss function is required to approximate of condition expectation using regress-now method. However, although quantization is quite easy and fast to implement in small dimension for the noise, it might be not efficient in high-dimension compared to Hybrid-Now. □

Remark 5.3.4. We point out that the estimated value function $\hat{V}_n^M$ in Hybrid-Now or Hybrid-LaterQ depend on training samples $X_k^{(m)}$, $m = 1, \ldots, M$, used at times $k = n, \ldots, N$, for computing the estimated optimal policies $\hat{a}_k^M$, and this is emphasized through the exponent M in the notations. □

5.3.3 Training sets design

We discuss here the choice of the training measure μ_n used to generate the training sets on which will be computed the estimations. Two cases are considered in this section. The first one is a knowledge-based selection, relevant when the controller knows with a certain degree of confidence where the process has to be driven in order to optimize her cost functional. The second case, on the other hand, is when the controller has no idea where or how to drive the process to optimize the cost functional.

Exploitation only strategy

In the knowledge-based setting, there is no need for exhaustive and expensive (in time mainly) exploration of the state space, and the controller can directly choose training sets Γ_n constructed from distributions μ_n that assign more points to the parts of the state space where the optimal process is likely to be driven.

In practice, at time n, assuming we know that the optimal process is likely to stay in the ball centered around the point m_n and with radius r_n, we choose a training measure μ_n centered around m_n as, for example $\mathcal{N}(m_n, r_n^2)$, and build the training set as sample of the latter.

Explore first, exploit later

- *Explore first:* If the agent has no idea of where to drive the process to receive large rewards, she can always proceed to an exploration step to discover favorable subsets of the state space. To do so, Γ_n, the training sets at time n, for $n = 0, \ldots, N-1$, can be built as uniform grids that cover a large part of the state space, or μ can be chosen uniform on such domain. It is essential to explore far enough to have a well understanding of where to drive and where not to drive the process.
- *Exploit later:* For $n = 0, \ldots, N-1$, the estimates for the optimal controls at time t_n that come up from the *Explore first* step are relatively good in the way that they manage to avoid the wrong areas of state space when driving the process. However, the training sets that have been used to compute the estimated optimal control are too sparse to ensure accuracy on the estimation. In order to improve the accuracy, the natural idea is to build new training sets by simulating M times the process using the estimates on the optimal strategy computed from the *Explore first* step, and then proceed to another estimation of the optimal strategies using the new training sets. This trick can be seen as a two steps algorithm that improves the estimate of the optimal control.

5.3.4 Comparison of the algorithms

We end this section with a synthetic comparison of the proposed algorithms. We emphasize the pros (+) and cons (-) of the three proposed algorithms in terms of bias estimate for the value function, variance, complexity and dimension for the state space.

Algo	Bias	Variance	Complexity	Dimension	Number of time steps N
NNContPI	+	-	-	+	--
Hybrid-Now	-	+	+	+	+
Hybrid-LaterQ	-	++	+	-	+

This table is the result of observations made when numerically solving various control problems, combined to a close look at the rates of convergence derived for the three algorithms in Theorems 5.4.1, 5.4.2 and 5.4.3. Note that the sensibility of the NNContPI and the Hybrid-LaterQ algorithms w.r.t. the number of time steps N is clearly described in the studies of their rate of convergence achieved in Theorems 5.4.1 and 5.4.3. However, we could only provide a weak result on the rate of convergence of the Hybrid algorithm (see Theorem 5.4.3), which in particular does not explain why the latter does not suffer from large value of N, unless stronger assumptions are made on the loss of the neural network estimating the optimal controls.

5.4 Convergence analysis

This section is devoted to the convergence of the estimator $\hat{V}_n^M$ of the value function V_n obtained from a training sample of size M and using DNN algorithms listed in Section 5.3.

Training samples rely on a given family of probability distributions μ_n on $\mathcal{X}$, for $n = 0, \ldots, N$, refereed to as training distribution (see Section 5.3.3 for a discussion on the choice of μ). For sake of simplicity, we consider that μ_n does not depend on n, and denote then by μ the training distribution. We shall assume that the family of controlled transition probabilities has a density w.r.t. μ, i.e.,

$$P^a(x, dx') \quad = \quad r(x, a; x')\mu(dx').$$

We shall assume that r is uniformly bounded in $(x, x', a) \in \mathcal{X}^2 \times \mathbb{A}$, and uniformly Lipschitz w.r.t. (x, a), i.e.,

(Hd) There exists some positive constants $\|r\|_\infty$ and $[r]_{_L}$ s.t.

$$\begin{aligned} |r(x, a; x')| &\leq \|r\|_\infty, \quad \forall x, x' \in \mathcal{X},\ a \in \mathbb{A}, \\ |r(x_1, a_1; x') - r(x_2, a_2; x')| &\leq [r]_{_L}(|x_1 - x_2| + |a_1 - a_2|), \quad \forall x_1, x_2 \in \mathcal{X},\ a_1, a_2 \in \mathbb{A}. \end{aligned}$$

Remark 5.4.1. Assumption **(Hd)** is usually satisfied when the state and control space are compacts. While the compactness on the control space $\mathbb{A}$ is not explicitly assumed, the compactness condition on the state space $\mathcal{X}$ turns out to be more crucial for deriving estimates on the estimation error (see Lemma 5.4.1), and will be assumed to hold true for simplicity. Actually, this compactness condition on $\mathcal{X}$ can be relaxed by truncation and localization arguments (see proposition 5.A.1 in the appendix) by considering a training distribution μ such that **(Hd)** is true and which admits a moment of order 1, i.e. $\int |y| d\mu(y) < +\infty$. □

We shall also assume some boundedness and Lipschitz condition on the reward functions:

(HR) There exists some positive constants $\|f\|_\infty$, $\|g\|_\infty$, $[f]_{_L}$, and $[f]_{_L}$ s.t.

$$\begin{aligned} |f(x, a)| &\leq \|f\|_\infty, \quad |g(x)| \leq \|g\|_\infty, \quad \forall x \in \mathcal{X},\ a \in \mathbb{A}, \\ |f(x_1, a_1) - f(x_2, a_2)| &\leq [f]_{_L}(|x_1 - x_2| + |a_1 - a_2|), \\ |g(x_1) - g(x_2)| &\leq [g]_{_L}|x_1 - x_2|, \qquad \forall x_1, x_2 \in \mathcal{X},\ a_1, a_2 \in \mathbb{A}. \end{aligned}$$

Under this boundedness condition, it is clear that the value function V_n is also bounded:

$$\|V_n\|_\infty \quad \leq \quad (N-n)\|f\|_\infty + \|g\|_\infty, \quad \forall n \in \{0, ..., N\}.$$

We shall finally assume a Lipschitz condition on the dynamics of the MDP.

(HF) For any $e \in E$, there exists $C(e)$ such that for all couples (x, a) and (x', a') in $\mathcal{X} \times \mathbb{A}$:

$$|F(x, a, e) - F(x', a', e)| \leq C(e) \left(|x - x'| + |a - a'|\right).$$

In the sequel, we define for any $M \in \mathbb{N}^*$:

$$\rho_M \quad = \quad \mathbb{E}\left[\sup_{1\leq m\leq M} C(\varepsilon^m)\right],$$

where the $(\varepsilon^m)_m$ is a i.i.d. sample of the noise ε. The rate of convergence of ρ_M toward infinity will play a crucial role to show the convergence of the algorithms.

Remark 5.4.2. A typical example when **(HF)** holds is the case where F is defined through the time discretization of an Euler scheme, i.e., as

$$F(x, a, \varepsilon) := b(x, a) + \sigma(x, a)\varepsilon,$$

with b and σ Lipschitz-continuous w.r.t. the couple (x, a), and $\varepsilon \sim \mathcal{N}(0, I_d)$, where I_d is the identity matrix of size $d \times d$. Indeed, in this case, it is straightforward to see that $C(\varepsilon) = [b]_L + [\sigma]_L\|\varepsilon\|_d$, where $[b]_L$ and $[\sigma]_L$ stand for the Lipschitz coefficients of b and σ, and $\|.\|_d$ stands for the Euclidean norm in $\mathbb{R}^d$. Moreover, one can show that:

$$\rho_M \leq [b]_L + d[\sigma]_L\sqrt{2\log(2dM)}, \tag{5.4.1}$$

which implies in particular that

$$\rho_M \underset{M\to+\infty}{=} \mathcal{O}\left(\sqrt{\log(M)}\right).$$

Let us indeed check the inequality (5.4.1). For this, let us fix some integer $M' > 0$ and let $Z := \sup_{1\leq m\leq M'} |\epsilon_1^m|$ where ϵ_1^m are i.i.d. such that $\epsilon_1^1 \sim \mathcal{N}(0, 1)$. From Jentzen inequality to the r.v. Z and the convex function $z \mapsto \exp(tz)$, where $t > 0$ will be fixed later, we get

$$\exp\left(t\mathbb{E}\left[Z\right]\right) \leq \mathbb{E}\left[\exp\left(tZ\right)\right] \leq \mathbb{E}\left[\sup_{1\leq m\leq M'} \exp\left(t|\epsilon_1^m|\right)\right] \leq \sum_{m=1}^{M'} \mathbb{E}\Big[\exp\left(t|\epsilon_1^m|\right)\Big]$$

$$\leq 2M' \exp\left(\frac{t^2}{2}\right),$$

where we used the closed-form expression of the moment generating function of the folded normal distribution[2] to write the last inequality. Hence, we have for all $t > 0$:

$$\mathbb{E}\big[Z\big] \leq \frac{\log(2M')}{t} + \frac{t}{2}.$$

We get, after taking $t = \sqrt{2\log(2M')}$:

$$\mathbb{E}\big[Z\big] \leq \sqrt{2\log(2M')}. \tag{5.4.2}$$

Since inequality $\|x\|_d \leq d\|x\|_\infty$ holds for all $x \in \mathbb{R}^d$, we derive

$$\mathbb{E}\left[\sup_{1\leq m\leq M} C(\varepsilon^m)\right] \leq [b]_L + d[\sigma]_L\mathbb{E}\left[\sup_{1\leq m\leq dM} C(\epsilon_1^m)\right],$$

and apply (5.4.2) with $M' = dM$, to complete the proof of (5.4.1). □

[2]The folded normal distribution is defined as the distribution of $|Z|$ where $Z \sim \mathcal{N}_1(\mu, \sigma)$. Its moment generating function is given by $t \mapsto \exp\left(\frac{\sigma^2t^2}{2} + \mu t\right)\left[1 - \Phi\left(-\frac{\mu}{\sigma} - \sigma t\right)\right] + \exp\left(\frac{\sigma^2t^2}{2} - \mu t\right)\left[1 - \Phi\left(\frac{\mu}{\sigma} - \sigma t\right)\right]$, where Φ is the c.d.f. of $\mathcal{N}_1(0, 1)$.

Remark 5.4.3. Under **(Hd)**, **(HR)** and **(HF)**, it is straightforward to see from the dynamic programming formula 5.1 that V_n is Lipschitz for all $n = 0, \dots, N$, with a Lipschitz coefficient $[V_n]_L$, which -can be bounded by the minimum of the two following bounds:

$$\begin{cases} [V_N]_L &= [g]_L \\ [V_n]_L &\leq [f]_L + \|V_n\|_\infty [r]_L, \quad \text{for } n = 0, \dots, N-1. \end{cases}$$

and

$$\begin{cases} [V_N]_L &= [g]_L \\ [V_n]_L &\leq \rho_1 \frac{1-\rho_1^{N-n}}{1-\rho_1} + \rho_1^{N-n}[g]_L, \quad \text{for } n = 0, \dots, N-1, \end{cases}$$

which holds since we have by standard arguments:

$$\begin{cases} [V_N]_L &= [g]_L \\ [V_n]_L &\leq [f]_L + \rho_1 [V_{n+1}]_L \quad \text{for } n = 0, \dots, N-1. \end{cases}$$

Note that we use the usual convention $\frac{1-x^p}{1-x} = p$ for $p \in \mathbb{N}^*$ and $x = 0$. The Lipschitz continuity of V_n plays a significant role to prove the convergence of the Hybrid and the LaterQ algorithms described and studied in sections 5.4.2 and 5.4.3. □

5.4.1 Control learning by performance iteration (NNcontPI)

In this paragraph, we analyze the convergence of the NN control learning by performance iteration as described in Section 5.3.1. Actually, we shall consider neural networks for the optimal policy with one hidden layer, K neurons with total variation[3] smaller than γ, kernel bounded by η, Relu activation function for the hidden layer, and activation function $\sigma_{\mathbb{A}}$ for the output layer (in order to ensure that the NN is valued in $\mathbb{A}$): this is represented by the parametric set of functions

$$\begin{aligned} {}^{\eta}\mathcal{A}_K^{\gamma} \quad := \quad & \Big\{ x \in \mathcal{X} \mapsto A(x;\beta) \;=\; (A_1(x;\beta), \dots, A_q(x;\beta)) \in \mathbb{A}, \\ & A_i(x;\beta) = \sigma_{\mathbb{A}}\Big(\sum_{j=1}^{K} c_{ij}(a_{ij}.x + b_{ij})_+ + c_{0j} \Big), \quad i = 1, \dots, q, \\ & \beta = (a_{ij}, b_{ij}, c_{ij})_{i,j},\ a_{ij} \in \mathbb{R}^d, \|a_{ij}\| \leq \eta, b_{ij}, c_{ij} \in \mathbb{R}, \sum_{i=0}^{K} |c_{ij}| \leq \gamma \Big\}, \end{aligned}$$

where $\|.\|$ is the Euclidean norm in $\mathbb{R}^d$.

Let K_M, η_M and γ_M be sequences of integers such that

$$\begin{aligned} K_M \xrightarrow[M\to\infty]{} \infty, \quad \gamma_M \xrightarrow[M\to\infty]{} \infty, \quad \eta_M \xrightarrow[M\to\infty]{} \infty, \\ \rho_M^{N-1} \gamma_M^{N-1} \eta_M^{N-2} \sqrt{\frac{\log(M)}{M}} \xrightarrow[M\to\infty]{} 0. \end{aligned} \tag{5.4.3}$$

We denote by $\mathcal{A}_M := {}^{\eta_M}\mathcal{A}_{k_M}^{\gamma_M}$ the class of neural network for policy with norm η_M on the kernel $a = (a_{ij})$, K_M neurons and norm γ_M that satisfy conditions (5.4.3).

Remark 5.4.4. In the case where F is defined in dimension d as: $F(x, a, \varepsilon) = b(x, a) + \sigma(x, a)\varepsilon$, we can use (5.4.1) to bound ρ_M^{N-n} and get:

$$\rho_M^{N-n} \underset{M\to+\infty}{=} \mathcal{O}\left(\sqrt{\log(M)}^{N-n} \right).$$

□

[3] The total variation for the class of NN $\mathcal{A}_K^{\gamma}$ is equal to $\sum_{i=0}^{K} |c_{ij}|$ with the notations above. See e.g. [Bac17] for a general definition.

Recall that the approximation of the optimal policy in the NNcontPI algorithm is computed in backward induction as follows: For $n = N-1, \dots, 0$, generate a training sample for the state $X_n^{(m)}$, $m = 1, \dots, M$ from the training distribution μ, and samples of the exogenous noise $\left(\varepsilon_k^m\right)_{m=1,k=n+1}^{M,N}$.

- Compute the approximated policy at time n

$$\hat{a}_n^M \in \underset{A \in \mathcal{A}_M}{\operatorname{argmin}} \tfrac{1}{M} \textstyle\sum_{m=1}^M \left[f(X_n^{(m)}, A(X_n^{(m)})) + \hat{Y}_{n+1}^{(m),A} \right] \tag{5.4.4}$$

  where

$$\hat{Y}_{n+1}^{(m),A} = \sum_{k=n+1}^{N-1} f\left(X_k^{(m),A}, \hat{a}_k^M\left(X_k^{(m),A}\right)\right) + g\left(X_N^{(m),A}\right), \tag{5.4.5}$$

  with $\left(X_k^{(m),A}\right)_{k=n+1}^N$ defined by induction as follows, for $m{=}1, \dots, M$:

$$\begin{cases} X_{n+1}^{(m),A} &= F\Big(X_n^m, A\big(X_n^m\big), \varepsilon_{n+1}^m\Big) \\ X_k^{(m),A} &= F\Big(X_{k-1}^{(m),A}, A\big(X_{k-1}^{(m),A}\big), \varepsilon_k^m\Big), \quad \text{for } k = n+2, \dots, N. \end{cases}$$

- Compute the estimated value function $\hat{V}_n^M$ as in (5.3.2).

Remark 5.4.5. In order to simplify the theoretical analysis, we assume that the argmin in (5.4.4) is exactly reached by running batch, mini-batch or stochastic gradient descent, which are the methods that we used to code the algorithm in our companion paper. □

Remark 5.4.6. The minimization problem in (5.4.4) is actually a problem of minimization over the parameter β (of the neural network A) of the expectation of a function of β and some noises $\left(X_n^{(m)}\right)_{m=1}^M$, $\left(\varepsilon_k^m\right)_{m=1,k=n+1}^{M,N}$, where F is iterated many times. Stochastic-gradient-based methods are chosen for such a task, although the gradient becomes more and more difficult to compute when we are going backward in time, since there are more and more iterations of F involved in the derivatives of the gradients.

The integrand is differentiable if assumption **(HF)** holds, but it is always possible to apply the stochastic-gradient-based algoritm for certain classes of non-differentiable functions F (see e.g. the gradient-descent implementation in TensorFlow which works with the non-differentiable at 0 ReLu activation functions.). □

We now state our main result about the convergence of the NNcontPI algorithm.

Theorem 5.4.1. *Assume that there exists an optimal feedback control $(a_k^{\mathrm{opt}})_{k=n}^{N-1}$ for the control problem with value function V_n, $n = 0, \dots, N$, and let $X_n \rightsquigarrow \mu$. Then, as $M \to \infty$*[4]

$$\begin{aligned} \mathbb{E}\big[\hat{V}_n^M(X_n) - V_n(X_n)\big] &= \mathcal{O}\Bigg(\frac{\rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}}{\sqrt{M}} \\ &\qquad + \sup_{n \le k \le N-1} \inf_{A \in \mathcal{A}_M} \mathbb{E}\left[|A(X_k) - a_k^{\mathrm{opt}}(X_k)| \right] \Bigg), \end{aligned} \tag{5.4.6}$$

where $\mathbb{E}$ stands for the expectation over the training set used to evaluate the approximated optimal policies $(\hat{a}_k^M)_{n \le k \le N-1}$, as well as the path $(X_n)_{n \le k \le N}$ controlled by the latter. Moreover, as $M \to \infty$[5]

$$\begin{aligned} \mathbb{E}_M\big[\hat{V}_n^M(X_n) - V_n(X_n)\big] = \mathcal{O}_{\mathbb{P}}\Bigg(& \rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}\sqrt{\frac{\log(M)}{M}} \\ & + \sup_{n \le k \le N-1} \inf_{A \in \mathcal{A}_M} \mathbb{E}\left[|A(X_k) - a_k^{\mathrm{opt}}(X_k)| \right] \Bigg), \end{aligned}$$

[4] The notation $x_M = \mathcal{O}(y_M)$ as $M \to \infty$, means that the ratio $|x_M|/|y_M|$ is bounded as M goes to infinity.

[5] The notation $x_M = \mathcal{O}_{\mathbb{P}}(y_M)$ as $M \to \infty$, means that there exists $c > 0$ such that $\mathbb{P}\big(|x_M| > c|y_M|\big) \to 0$ as M goes to infinity.

where $\mathbb{E}_M$ stands for the expectation conditioned by the training set used to estimate the optimal policies $(\hat{a}_k^M)_{n\leq k\leq N-1}$.

Remark 5.4.7. 1. The term $\frac{\rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}}{\sqrt{M}}$ should be seen as the estimation error. It is due to the approximation of the optimal controls by means of neural networks in $\mathcal{A}_M$ using *empirical* cost functional in (5.4.4). We show in section 5.B that this term disappears in the ideal case where the real cost functional (i.e. not the empirical one) is minimized.

2. The rate of convergence depends dramatically on N since it becomes exponentially slower when N goes to infinity. This is a huge drawback for this performance iteration-based algorithm. We will see in the next section that the rate of convergence of value iteration-based algorithms do not suffer from this dramatical dependence on N. □

Comment: Since we clearly have $V_n \leq \hat{V}_n^M$, estimation (5.4.6) implies the convergence in L^1 norm of the NNcontPI algorithm, under condition (5.4.3), and in the case where

$$\sup_{n\leq k\leq N}\inf_{A\in\mathcal{A}_M}\mathbb{E}\big[|A(X_k)-a_k^{\rm opt}(X_k)|\big]\xrightarrow[M\to+\infty]{}0$$

. This is actually the case under some regularity assumptions on the optimal controls, as stated in the following proposition.

Proposition 5.4.1. *The two following assertions hold:*

1. *Assume that $a_k^{\rm opt}(X_k)\in\mathbb{L}^1(\mu)$ for $k=n,...,N-1$. Then*

$$\sup_{n\leq k\leq N-1}\inf_{A\in\mathcal{A}_M}\mathbb{E}\big[|A(X_k)-a_k^{\rm opt}(X_k)|\big]\xrightarrow[M\to+\infty]{}0. \tag{5.4.7}$$

2. *Assume that the function $a_k^{\rm opt}$ is c-Lipschitz for $k=n,...,N-1$. Then*

$$\sup_{n\leq k\leq N-1}\inf_{A\in\mathcal{A}_M}\mathbb{E}\big[|A(X_k)-a_k^{\rm opt}(X_k)|\big] < c\left(\frac{\gamma_M}{c}\right)^{-2d/(d+1)}\log\left(\frac{\gamma_M}{c}\right)+\gamma_M K_M^{-(d+3)/(2d)}. \tag{5.4.8}$$

Proof. The first statement of Proposition 5.4.1 relies essentially on the universal approximation theorem, and the second assertion is stated and proved in [Bac17]. For sake of completeness, we recall the details in Section 5.E in the Appendix. □

Remark 5.4.8. Note that the second statement in the above proposition is stronger than the first one since it provides a rate of convergence of the approximation error. Fixing K_M and minimizing the r.h.s. of (5.4.8) over γ_M, results in

$$\sup_{n\leq k\leq N-1}\inf_{A\in\mathcal{A}_M}\mathbb{E}\big[|A(X_k)-a_k^{\rm opt}(X_k)|\big] < cK_M^{-\frac{1}{d}}\left(1+\frac{d+1}{2d}\log\big(K_M\big)\right),$$

when we take $\gamma_M = cK_M^{\frac{d+1}{2d}}$. Hence, for such a value of γ_M, the l.h.s. decreases to 0 with rate proportional to $\log(K_M)/\sqrt[d]{K_M}$. □

The rest of this section is devoted to the proof of Theorem 5.4.1. Let us introduce some useful notations. Denote by $\mathbb{A}^{\mathcal{X}}$ the set of Borelian functions from the state space $\mathcal{X}$ into the control space $\mathbb{A}$. For $n=0,\ldots,N-1$, and given a feedback control (policy) represented by a sequence $(A_k)_{k=n,\ldots,N-1}$, with A_k in $\mathbb{A}^{\mathcal{X}}$, we denote by $J_n^{(A_k)_{k=n}^{N-1}}$ the cost functional associated to the policy $(A_k)_k$. Notice that with this notation, we have $\hat{V}_n^M = J_n^{(\hat{a}_k^M)_{k=n}^{N-1}}$. We define the *estimation error* at time n associated to the NNContPI algorithm by

$$\varepsilon_{{\rm PI},n}^{\rm esti} := \sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]-\mathbb{E}_M\big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\big]\right|,$$

with $X_n \rightsquigarrow \mu$: It measures how well the chosen estimator (e.g. mean square estimate) can approximate a certain quantity (e.g. the conditional expectation). Of course we expect the latter to cancel when the size of the training set used to build the estimator goes to infinity. Actually, we have

Lemma 5.4.1. *For $n = 0, \dots, N-1$, we have*

$$\begin{aligned}\mathbb{E}[\varepsilon^{\text{esti}}_{\text{PI},n}] &\leq \big(\sqrt{2}+16\big)\frac{\big((N-n)\|f\|_\infty + \|g\|_\infty\big)}{\sqrt{M}} \\ &+ \frac{16\gamma_M}{\sqrt{M}}\left\{[f]_L\left(1+\rho_M\frac{1-\rho_M^{N-n-1}\big(1+\eta_M\gamma_M\big)^{N-n-1}}{1-\rho_M\big(1+\eta_M\gamma_M\big)}\right)+\big(1+\eta_M\gamma_M\big)^{N-n-1}\rho_M^{N-n}[g]_L\right\} \\ &= \mathcal{O}\left(\frac{\rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}}{\sqrt{M}}\right), \qquad \text{as } M\to\infty.\end{aligned} \tag{5.4.9}$$

This implies in particular that

$$\varepsilon^{\text{esti}}_{\text{PI},n} = \mathcal{O}_{\mathbb{P}}\left(\rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}\sqrt{\frac{\log(M)}{M}}\right), \qquad \text{as } M\to\infty, \tag{5.4.10}$$

where we remind that $\rho_M = \mathbb{E}\left[\sup_{1\leq m\leq M} C(\varepsilon^m)\right]$ is defined in **(HF)**.

Proof. The relation (5.4.9) states that the estimation error cancels when $M\to\infty$ with a rate of convergence of order $\mathcal{O}\Big(\frac{\rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}}{\sqrt{M}}\Big)$. The proof is in the spirit of the one that can be found in chapter 9 of [GKKW02]. It relies on a technique of symmetrization by a ghost sample, and a wise introduction of additional randomness by random signs. The details are postponed in Section 5.C in the Appendix. The proof of (5.4.10) follows from (5.4.9) by a direct application of Markov inequality. □

Let us also define the *approximation error* at time n associated to the NNContPI algorithm by

$$\varepsilon^{\text{approx}}_{\text{PI},n} := \inf_{A\in\mathcal{A}_M}\mathbb{E}_M\Big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\Big] - \inf_{A\in\mathbb{A}^{\mathcal{X}}}\mathbb{E}_M\Big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\Big],$$

where we recall that $\mathbb{E}_M$ denotes the expectation conditioned by the training set used to compute the estimates $(\hat{a}_k^M)_{k=n+1}^{N-1}$ and the one of $X_n \rightsquigarrow \mu$.
$\varepsilon^{\text{approx}}_{\text{PI},n}$ measures how well the regression function can be approximated by means of neural networks functions in $\mathcal{A}_M$ (notice that the class of neural networks is not dense in the set $\mathbb{A}^{\mathcal{X}}$ of all Borelian functions).

Lemma 5.4.2. *For $n = 0, \dots, N-1$, it holds as $M\to\infty$,*

$$\mathbb{E}[\varepsilon^{\text{approx}}_{\text{PI},n}] = \mathcal{O}\left(\frac{\rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}}{\sqrt{M}} + \sup_{n\leq k\leq N-1}\inf_{A\in\mathcal{A}_M}\mathbb{E}\big[|A(X_k)-a_k^{\text{opt}}(X_k)|\big]\right). \tag{5.4.11}$$

This implies in particular

$$\begin{aligned}\varepsilon^{\text{approx}}_{\text{PI},n} = \mathcal{O}_{\mathbb{P}}\Bigg(&\rho_M^{N-n-1}\gamma_M^{N-n-1}\eta_M^{N-n-2}\sqrt{\frac{\log(M)}{M}} \\ &+ \sup_{n\leq k\leq N-1}\inf_{A\in\mathcal{A}_M}\mathbb{E}\big[|A(X_k)-a_k^{\text{opt}}(X_k)|\big]\Bigg).\end{aligned} \tag{5.4.12}$$

Proof. See Section 5.D in Appendix for the proof of (5.4.11). The proof of (5.4.12) then follows by a direct application of Markov inequality. □

Proof of Theorem 5.4.1.
Step 1. Let us denote by

$$\hat{J}_{n,M}^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}} \quad := \quad \frac{1}{M}\sum_{m=1}^{M}\Big[f\big(X_n^{(m)},A(X_n^{(m)})\big)+\hat{Y}_{n+1}^{(m),A}\Big],$$

the empirical cost function, from time n to N, associated to the sequence of controls $(A,(\hat{a}_k^M)_{k=n+1}^{N-1})$ and the training set, where we recall that $\hat{Y}_{n+1}^{(m),A}$ is defined in (5.4.5). We then have

$$\begin{aligned}\mathbb{E}_M\big[\hat{V}_n^M(X_n)\big] &= \mathbb{E}_M\Big[J_n^{(\hat{a}_k^M)_{k=n}^{N-1}}(X_n)\Big]-\hat{J}_{n,M}^{(\hat{a}_k^M)_{k=n}^{N-1}}+\hat{J}_{n,M}^{(\hat{a}_k^M)_{k=n}^{N-1}}\\ &\le \varepsilon_{\mathrm{PI},n}^{\mathrm{esti}}+\hat{J}_{n,M}^{(\hat{a}_k^M)_{k=n}^{N-1}},\end{aligned} \tag{5.4.13}$$

by definition of $\hat{V}_n^M$ and $\varepsilon_{\mathrm{PI},n}^{\mathrm{esti}}$. Moreover, for any $A\in\mathcal{A}_M$,

$$\begin{aligned}\hat{J}_{n,M}^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}} &= \hat{J}_{n,M}^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}-\mathbb{E}_M\Big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\Big]+\mathbb{E}_M\Big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\Big]\\ &\le \varepsilon_{\mathrm{PI},n}^{\mathrm{esti}}+\mathbb{E}_M\Big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\Big].\end{aligned} \tag{5.4.14}$$

Recalling that

$$\hat{a}_n^M \quad = \quad \operatorname*{argmin}_{A\in\mathcal{A}_M}\hat{J}_{n,M}^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}},$$

and taking the infimum over $\mathcal{A}_M$ in the l.h.s. of (5.4.14) first, and in the r.h.s. secondly, we then get

$$\hat{J}_{n,M}^{(\hat{a}_k^M)_{k=n}^{N-1}} \quad \le \quad \varepsilon_{\mathrm{PI},n}^{\mathrm{esti}}+\inf_{A\in\mathcal{A}_M}\mathbb{E}_M\Big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\Big].$$

Plugging this last inequality into (5.4.13) yields the following estimate

$$\mathbb{E}_M\big[\hat{V}_n^M(X_n)\big]-\inf_{A\in\mathcal{A}_M}\mathbb{E}_M\Big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\Big] \quad \le \quad 2\varepsilon_{\mathrm{PI},n}^{\mathrm{esti}}. \tag{5.4.15}$$

Step 2. By definition (5.4.1) of the approximation error, using the law of iterated conditional expectations for J_n, and the dynamic programming principle for V_n with the optimal control a_n^{opt} at time n, we have

$$\begin{aligned}&\inf_{A\in\mathcal{A}_M}\mathbb{E}_M\big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\big]-\mathbb{E}_M[V_n(X_n)]\\ &\qquad=\varepsilon_{\mathrm{PI},n}^{\mathrm{approx}}+\inf_{A\in\mathbb{A}^{\mathcal{X}}}\mathbb{E}_M\big\{f(X_n,A(X_n))+\mathbb{E}_n^A\big[J_{n+1}^{(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_{n+1})\big]\big\}\\ &\qquad\quad-\mathbb{E}_M\Big[f(X_n,a_n^{\mathrm{opt}}(X_n))+\mathbb{E}_n^{a_n^{\mathrm{opt}}}\big[V_{n+1}(X_{n+1})\big]\Big]\\ &\qquad\le\varepsilon_{\mathrm{PI},n}^{\mathrm{approx}}\;+\;\mathbb{E}_M\mathbb{E}_n^{a_n^{\mathrm{opt}}}\Big[J_{n+1}^{(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_{n+1})-V_{n+1}(X_{n+1})\Big],\end{aligned}$$

where $\mathbb{E}_n^A[.]$ stands for the expectation conditioned by X_n at time n and the training set, when strategy A is followed at time n. Under the bounded density assumption in **(Hd)**, we then get

$$\begin{aligned}&\inf_{A\in\mathcal{A}_M}\mathbb{E}_M\big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\big]-\mathbb{E}_M[V_n(X_n)]\\ &\qquad\le\varepsilon_{\mathrm{PI},n}^{\mathrm{approx}}\;+\;\|r\|_\infty\int\big[J_{n+1}^{(\hat{a}_k^M)_{k=n+1}^{N-1}}(x')-V_{n+1}(x')\big]\mu(dx')\\ &\qquad\le\varepsilon_{\mathrm{PI},n}^{\mathrm{approx}}\;+\;\|r\|_\infty\mathbb{E}_M\Big[\hat{V}_{n+1}^M(X_{n+1})-V_{n+1}(X_{n+1})\Big],\ \text{with } X_{n+1}\sim\mu.\end{aligned} \tag{5.4.16}$$

Step 3. From (5.4.15) and (5.4.16), we have

$$\begin{aligned}
&\mathbb{E}_M\big[\hat{V}_n^M(X_n) - V_n(X_n)\big] = \mathbb{E}_M\big[\hat{V}_n^M(X_n)\big] - \inf_{A\in\mathcal{A}_M} \mathbb{E}_M\big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\big]\Big] \\
&+ \inf_{A\in\mathcal{A}_M} \mathbb{E}_M\big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\big] - \mathbb{E}_M[V_n(X_n)] \\
&\leq 2\varepsilon_{\mathrm{PI},n}^{\mathrm{esti}} + \varepsilon_{\mathrm{PI},n}^{\mathrm{approx}} + \|r\|_\infty \mathbb{E}_M\Big[\hat{V}_{n+1}^M(X_{n+1}) - V_{n+1}(X_{n+1})\Big]. \qquad (5.4.17)
\end{aligned}$$

By induction, this implies

$$\mathbb{E}_M\big[\hat{V}_n^M(X_n) - V_n(X_n)\big] \leq \sum_{k=n}^{N-1} \big(2\varepsilon_{\mathrm{PI},k}^{\mathrm{esti}} + \varepsilon_{\mathrm{PI},k}^{\mathrm{approx}}\big).$$

Use the estimations (5.4.10) for $\varepsilon_{\mathrm{PI},n}^{\mathrm{esti}}$ in Lemma 5.4.1, and (5.4.12) for $\varepsilon_{\mathrm{PI},n}^{\mathrm{approx}}$ in Lemma 5.4.2, and observe that $\hat{V}_n(X_n) \geq V_n(X_n)$ holds a.s., to complete the proof of (5.4.1). Finally, the proof of (5.4.6) is obtained by taking expectation in (5.4.17), and using estimations (5.4.9) and (5.4.11). □

5.4.2 Hybrid-Now algorithm

In this paragraph, we analyze the convergence of the hybrid-now algorithm as described in Section 5.3.2.1. We shall consider neural networks for the value function estimation with one hidden layer, K neurons with total variation γ, kernel bounded by η, a sigmoid activation function σ for the hidden layer, and no activation function for the output layer (i.e. the last layer): this is represented by the parametric set of functions

$$\begin{aligned}
{}^{\eta}\mathcal{V}_K^{\gamma} \quad := \quad \Big\{ & x \in \mathcal{X} \mapsto \Phi(x;\theta) \;=\; \sum_{i=1}^{K} c_i \sigma(a_i.x + b_i) + c_0, \\
& \theta = (a_i, b_i, c_i)_i, \quad \|a_i\| \leq \eta,\ b_i \in \mathbb{R},\ \sum_{i=0}^{K} |c_i| \leq \gamma \Big\}.
\end{aligned}$$

Let η_M, K_M and γ_M be integers such that:

$$\begin{aligned}
&\eta_M \xrightarrow[M\to\infty]{} \infty \quad , \quad \gamma_M \xrightarrow[M\to\infty]{} +\infty \quad , \quad K_M \xrightarrow[M\to\infty]{} \infty, \\
&\tfrac{\gamma_M^4 K_M \log(M)}{M} \xrightarrow[M\to\infty]{} 0 \quad , \quad \tfrac{\gamma_M^4 \rho_M^2 \eta_M^2 \log(M)}{M} \xrightarrow[M\to\infty]{} 0, \qquad (5.4.18)
\end{aligned}$$

where we remind that ρ_M is defined in **(HF)**.

In the sequel we denote by $\mathcal{V}_M := {}^{\eta_M}\mathcal{V}_{K_M}^{\gamma_M}$ the space of neural networks for the estimated value functions at time $n = 0, \ldots, N-1$, parametrized by the values η_M, γ_M and K_M that satisfy (5.4.18). We also consider the class $\mathcal{A}_M$ of neural networks for estimated feedback optimal control at time $n = 0, \ldots, N-1$, as described in Section 5.4.1, with the same parameters η_M, γ_M and K_M.

Recall that the approximation of the value function and optimal policy in the hybrid-now algorithm is computed in backward induction as follows:

- Initialize $\hat{V}_N^M = g$
- For $n = N-1, \ldots, 0$, generate a training sample $X_n^{(m)}$, $m = 1, \ldots, M$ from the training distribution μ, and a training sample for the exogenous noise $\varepsilon_{n+1}^{(m)}$, $m = 1, \ldots, M$.

  (i) compute the approximated policy at time n

$$\hat{a}_n^M \quad \in \quad \operatorname*{argmin}_{A\in\mathcal{A}_M} \frac{1}{M} \sum_{m=1}^{M} \big[f(X_n^{(m)}, A(X_n^{(m)})) + \hat{V}_{n+1}^M(X_{n+1}^{(m),A})\big]$$

where $X_{n+1}^{(m),A} = F(X_n^{(m)}, A(X_n^{(m)}), \varepsilon_{n+1}^{(m)}) \rightsquigarrow P^{A(X_n^{(m)})}(X_n^{(m)}, dx')$.

(ii) compute the untruncated estimation of the value function at time n

$$\tilde{V}_n^M \in \underset{\Phi\in\mathcal{V}_M}{\operatorname{argmin}} \frac{1}{M}\sum_{m=1}^{M}\Big[f(X_n^{(m)}, \hat{a}_n^M(X_n^{(m)})) + \hat{V}_{n+1}^M(X_{n+1}^{(m),\hat{a}_n^M}) - \Phi(X_n^{(m)})\Big]^2$$

and set the truncated estimated value function at time n

$$\hat{V}_n^M = \max\Big(\min\big(V_n^M, \|V_n\|_\infty\big), -\|V_n\|_\infty\Big).$$

Remark 5.4.9. Notice that we have truncated the estimated value function in (5.4.19) by an *a priori* bound on the true value function. This truncation step is natural from a practical implementation point of view, and is also used for simplifying the proof of the convergence of the algorithm. The conditions in (5.4.18) for the parameters are weaker than those required in (5.4.3) for the NNcontPI algo by performance iteration, which implies a much faster convergence w.r.t. the size of the training set. However, one should keep in mind that unlike the performance iteration procedure, the value iteration one is biased since the computation of $\hat{V}_{n+1}^M(X_{n+1}^A)$ are biased future rewards when decision A is taken at time n and estimated optimal strategies are taken at time $k \geq n+1$. □

We now state our main result about the convergence of the Hybrid-Now algorithm.

Theorem 5.4.2. *Assume that there exists an optimal feedback control $(a_k^{\mathrm{opt}})_{k=n}^{N-1}$ for the control problem with value function V_n, $n = 0, \ldots, N$, and let $X_n \rightsquigarrow \mu$. Then, as $M \to +\infty$*

$$\begin{aligned}
&\mathbb{E}_M\Big[|\hat{V}_n^M(X_n) - V_n(X_n)|\Big] \\
&\quad = \mathcal{O}_{\mathbb{P}}\Bigg(\left(\gamma_M^4 \frac{K_M \log(M)}{M}\right)^{\frac{1}{2(N-n)}} + \left(\gamma_M^4 \frac{\rho_M^2 \eta_M^2 \log(M)}{M}\right)^{\frac{1}{4(N-n)}} \\
&\qquad + \sup_{n\leq k\leq N} \inf_{\Phi\in\mathcal{V}_M} \Big(\mathbb{E}_M\Big[|\Phi(X_k) - V_k(X_k)|^2\Big]\Big)^{\frac{1}{2(N-n)}} \\
&\qquad + \sup_{n\leq k\leq N} \inf_{A\in\mathcal{A}_M} \Big(\mathbb{E}\Big[|A(X_k) - a_k^{\mathrm{opt}}(X_k)|\Big]\Big)^{\frac{1}{2(N-n)}}\Bigg),
\end{aligned}$$

where $\mathbb{E}_M$ stands for the expectation conditioned by the training set used to estimate the optimal policies $(\hat{a}_k^M)_{n\leq k\leq N-1}$.

Comment: Theorem 5.4.2 states that the estimator for the value function provided by hybrid-now algorithm converges in $\mathbb{L}^1(\mu)$ when the size of the training set goes to infinity. Note that the term $\left(\gamma_M^4 \frac{K_M \log(M)}{M}\right)^{\frac{1}{2(N-n)}}$ stands for the estimation error made by estimating *empirically* the value functions using neural networks, and $\left(\gamma_M^4 \frac{\rho_M^2 \eta_M^2 \log(M)}{M}\right)^{\frac{1}{4(N-n)}}$ stands for the estimation error made by estimating *empirically* the optimal control using neural networks. The following quantity $\sup_{n\leq k\leq N} \inf_{\Phi\in\mathcal{V}_M} \sqrt{\mathbb{E}\Big[|\Phi(X_k) - V_k(X_k)|^2\Big]}$ stands for the approximation error made by estimating the value function as a neural network function in $\mathcal{V}_M$, and the term $\sup_{n\leq k\leq N} \inf_{A\in\mathcal{A}_M} \mathbb{E}\Big[|A(X_k) - a_k^{\mathrm{opt}}(X_k)|\Big]$ is the one made by estimating the optimal control as a neural network function in $\mathcal{A}_M$.

In order to prove Theorem 5.4.2, let us first introduce the estimation error at time n associated

to the Hybrid-Now algorithm by

$$\varepsilon^{\text{esti}}_{\text{HN},n} := \sup_{A\in\mathcal{A}_M} \left| \frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)}, A(X_n^{(m)})) + \hat{Y}_{n+1}^{(m),A}\right] - \mathbb{E}^{A}_{M,n,X_n}\left[f(X_n, A(X_n)) + \hat{V}^M_{n+1}(X_{n+1})\right]\right|,$$

where $\hat{Y}_{n+1}^{(m),A} = \hat{V}^M_{n+1}(X_{n+1}^{(m),A})$, and $X_{n+1}^{(m),A} = F\left(X_n^{(m)}, A(X_n^{(m)}), \varepsilon^m_{n+1}\right)$.

We have the following bound on this estimation error:

Lemma 5.4.3. *For $n = 0, ..., N-1$, it holds:*

$$\mathbb{E}\left[\varepsilon^{\text{esti}}_{\text{HN},n}\right] \leq \frac{\left(\sqrt{2}+16\right)\left((N-n)\|f\|_\infty + \|g\|_\infty\right) + 16[f]_L}{\sqrt{M}} + 16\frac{\rho_M\eta_M\gamma_M^2}{\sqrt{M}} \underset{M\to\infty}{=} \mathcal{O}\left(\frac{\rho_M\eta_M\gamma_M^2}{\sqrt{M}}\right). \tag{5.4.20}$$

Proof. See Section 5.F in Appendix. □

Remark 5.4.10. The result stated by lemma 5.4.3 is sharper than the one stated in Lemma 5.4.1 for the performance iteration procedure. The main reason is that we can make use of the $\gamma_M\eta_M$-Lipschitz-continuity of the estimation of the value function at time $n+1$. □

We secondly introduce the approximation error at time n associated to the Hybrid Now algorithm by

$$\varepsilon^{\text{approx}}_{\text{HN},n} := \inf_{A\in\mathcal{A}_M} \mathbb{E}_M\left[f\left(X_n, A(X_n)\right) + \hat{Y}^A_{n+1}\right] - \inf_{A\in\mathbb{A}^{\mathcal{X}}} \mathbb{E}_M\left[f\left(X_n, A(X_n)\right) + \hat{Y}^A_{n+1}\right],$$

where $\hat{Y}^A_{n+1} := \hat{V}^M_{n+1}\left(F\left(X_n, A(X_n), \varepsilon_{n+1}\right)\right)$.

We have the following bound on this approximation error:

Lemma 5.4.4. *For $n = 0, ..., N-1$, it holds:*

$$\varepsilon^{\text{approx}}_{\text{HN},n} \leq \left([f]_L + \|V_{n+1}\|_\infty[r]_L\right) \inf_{A\in\mathbb{A}^{\mathcal{X}}} \mathbb{E}_M\left[\left|A(X_n) - a_n^{\text{opt}}(X_n)\right|\right] + 2\|r\|_\infty \mathbb{E}_M\left[\left|V_{n+1}(X_{n+1}) - \hat{V}^M_{n+1}(X_{n+1})\right|\right].$$

Proof. See Section 5.G in Appendix. □

Proof of Theorem 5.4.2

Observe that not only the optimal strategy but also the value function is estimated at each time step $n = 0, ..., N-1$ using neural networks in the hybrid algorithm. It spurs us to introduce the following auxiliary process $(\bar{V}^M_n)_{n=0}^N$ defined by backward induction as:

$$\begin{cases} \bar{V}^M_N(x) &= g(x), \quad \text{for } x\in\mathcal{X}, \\ \bar{V}^M_n(x) &= f(x, \hat{a}^M_n(x)) + \mathbb{E}\left[\hat{V}^M_{n+1}(F(x, \hat{a}^M_n(x), \varepsilon_{n+1}))\right], \quad \text{for } x\in\mathcal{X}, \end{cases}$$

and we notice that for $n = 0, ..., N-1$, $\bar{V}^M_n$ is the quantity estimated by $\hat{V}^M_n$.

Step 1. We state the following estimates: for $n = 0, ..., N-1$,

$$0 \leq \mathbb{E}_M\left[\bar{V}^M_n(X_n) - \inf_{a\in A}\left\{f(X_n, a) + \mathbb{E}^a_{M,n,X_n}\left[\hat{V}^M_{n+1}(X_{n+1})\right]\right\}\right] \leq 2\varepsilon^{\text{esti}}_{\text{HN},n} + \varepsilon^{\text{approx}}_{\text{HN},n}, \tag{5.4.21}$$

and,

$$\mathbb{E}_M\left[\left|\bar{V}_n^M(X_n) - \inf_{a\in A}\left\{f(X_n,a) + \mathbb{E}^a_{M,n,X_n}\left[\hat{V}^M_{n+1}(X_{n+1})\right]\right\}\right|^2\right] \tag{5.4.22}$$
$$\leq 2\left((N-n)\|f\|_\infty + \|g\|_\infty\right)\left(2\varepsilon^{\text{esti}}_{\text{HN},n} + \varepsilon^{\text{approx}}_{\text{HN},n}\right),$$

where $\mathbb{E}_{M,n,X_n}$ stands for the expectation conditioned by the training set and X_n.

Let us first show the estimate (5.4.21). Note that inequality

$$\bar{V}_n^M(X_n) - \inf_{a\in A}\left\{f(X_n,a) + \mathbb{E}^a_{M,n,X_n}\left[\hat{V}^M_{n+1}(X_{n+1})\right]\right\} \geq 0$$

holds because $\hat{a}_n^M$ cannot do better than the optimal strategy. Take its expectation to get the first inequality in (5.4.21). Moreover, we write

$$\begin{aligned}\mathbb{E}_M\left[\bar{V}_n^M(X_n)\right] &\leq \mathbb{E}_M\left[f\left(X_n,\hat{a}_n^M(X_n)\right) + \hat{V}^M_{n+1}\left(X^{\hat{a}_n^M}_{n+1}\right)\right]\\ &\leq \inf_{A\in\mathcal{A}_M}\mathbb{E}_M\left[f\left(X_n,A(X_n)\right) + \hat{V}^M_{n+1}\left(X^A_{n+1}\right)\right] + 2\varepsilon^{\text{esti}}_{\text{HN},n},\end{aligned}$$

which holds by the same arguments as those used to prove (5.4.15). We deduce that

$$\begin{aligned}\mathbb{E}_M\left[\bar{V}_n^M(X_n)\right] &\leq \inf_{A\in\mathbb{A}^{\mathcal{X}}}\mathbb{E}_M\left[f\left(X_n,A(X_n)\right) + \hat{V}^M_{n+1}\left(X^A_{n+1}\right)\right] + \varepsilon^{\text{approx}}_{\text{HN},n} + 2\varepsilon^{\text{esti}}_{\text{HN},n}\\ &\leq \mathbb{E}_M\left[\inf_{a\in A}\left\{f\left(X_n,a\right) + \mathbb{E}^a_M\left[\hat{V}^M_{n+1}\left(X_{n+1}\right)\middle|X_n\right]\right\}\right] + \varepsilon^{\text{approx}}_{\text{HN},n} + 2\varepsilon^{\text{esti}}_{\text{HN},n}.\end{aligned}$$

This completes the proof of the second inequality stated in (5.4.21).
On the other hand, noting:

$$\left|\bar{V}_n^M(X_n) - \inf_{a\in A}\left\{f(X_n,a) + \mathbb{E}^a_M\left[\hat{V}^M_{n+1}(X_{n+1})\middle|X_n\right]\right\}\right| \leq 2\left((N-n)\|f\|_\infty + \|g\|_\infty\right),$$

and using (5.4.21), we obtain the inequality (5.4.22).

Step 2. We state the following estimation: for all $n\in\{0,...,N\}$

$$\begin{aligned}&\left\|\hat{V}_n^M(X_n) - \bar{V}_n^M(X_n)\right\|_{M,1}\\ &= \mathcal{O}_{\mathbb{P}}\left(\gamma_M^2\sqrt{K_M\frac{log(M)}{M}} + \inf_{\Phi\in\mathcal{V}_M}\sqrt{\|\Phi(X_n) - V_n^M(X_n)\|_{M,1}}\right.\\ &\left. + \inf_{A\in\mathbb{A}^{\mathcal{X}}}\sqrt{\left\|A(X_n) - a_n^{\text{opt}}(X_n)\right\|_{M,1}} + \sqrt{\left\|V_{n+1}(X_{n+1}) - \hat{V}^M_{n+1}(X_{n+1})\right\|_{M,1}}\right),\end{aligned}$$

where $\|.\|_{M,p} = \left(\mathbb{E}_M\left[|.|^p\right]\right)^{\frac{1}{p}}$, i.e. $\|.\|_{M,p}$ stands for the $\mathbb{L}^p$ norm conditioned by the training set, for $p\in\{1,2\}$. The proof relies on Lemma 5.H.1 and Lemma 5.H.2 (see Section 5.H in Appendix) which are proved respectively in [Koh06] (see their Theorem 3) and [KKT10].

Let us first show the following relation:

$$\mathbb{E}_M\left[\left|\hat{V}_n^M(X_n) - \bar{V}_n^M(X_n)\right|^2\right] = \mathcal{O}_{\mathbb{P}}\left(\gamma_M^4 K_M\frac{log(M)}{M} + \inf_{\Phi\in\mathcal{V}_M}\mathbb{E}\left[\left|\Phi(X_n) - \bar{V}_n^M(X_n)\right|^2\right]\right).$$

For this, take $\delta_M = \gamma_M^4 K_M \frac{\log(M)}{M}$, let $\delta > \delta_M$, and denote

$$\Omega_g := \left\{ f - g : f \in \mathcal{V}_M, \frac{1}{M} \sum_{m=1}^{M} \left| f(x_m) - g(x_m) \right|^2 \leq \frac{\delta}{\gamma_M^2} \right\}$$

Apply Lemma 5.H.2 to obtain:

$$\begin{aligned}
&\int_{c_2\delta/\gamma_M^2}^{\sqrt{\delta}} \log\left(\mathcal{N}_2\left(\frac{u}{4\gamma_M}, \Omega_g, x_1^M \right) \right)^{1/2} \mathrm{d}u \\
&\qquad \leq \int_{c_2\delta/\gamma_M^2}^{\sqrt{\delta}} \log\left(\mathcal{N}_2\left(\frac{u}{4\gamma_M}, \mathcal{V}_M, x_1^M \right) \right)^{1/2} \mathrm{d}u \\
&\qquad \leq \int_{c_2\delta/\gamma_M^2}^{\sqrt{\delta}} \big((4d+9)K_M + 1\big)^{1/2} \left[\log\left(\frac{48e\gamma_M^2\big(K_M+1\big)}{u} \right) \right]^{1/2} \mathrm{d}u \\
&\qquad \leq \int_{c_2\delta/\gamma_M^2}^{\sqrt{\delta}} \big((4d+9)K_M + 1\big)^{1/2} \left[\log\left(48e\frac{\gamma_M^4}{\delta}\big(K_M+1\big) \right) \right]^{1/2} \mathrm{d}u \\
&\qquad \leq \sqrt{\delta}\big((4d+9)K_M + 1\big)^{1/2} \left[\log\left(48e\gamma_M^4 M\big(K_M+1\big) \right) \right]^{1/2} \\
&\qquad \leq c_5\sqrt{\delta}\sqrt{K_M}\sqrt{\log(M)}, \qquad (5.4.23)
\end{aligned}$$

where $\mathcal{N}_2(\varepsilon, \mathcal{V}, x_1^M)$ stands for the ε-covering number of $\mathcal{V}$ on x_1^M, which is introduced in section 5.H, and where the last line holds since we assumed $\frac{M\delta_M}{\gamma_M^2} \xrightarrow[M\to 0]{} 0$. Since $\delta > \delta_M := \gamma_M^4 K_M \frac{\log(M)}{M}$, we then have $\sqrt{\delta}\sqrt{K_M}\sqrt{\log(M)} \leq \frac{\sqrt{M}\delta}{\gamma_M^2}$, which implies that (5.H.1) holds by (5.4.23). Therefore, by application of Lemma 5.H.1, it holds:

$$\mathbb{E}_M\left[\left| \tilde{V}_n^M(X_n) - \bar{V}_n^M(X_n) \right|^2 \right] = \mathcal{O}_{\mathbb{P}}\left(\gamma_M^4 K_M \frac{\log(M)}{M} + \inf_{\Phi\in\mathcal{V}_M} \mathbb{E}\left[\left| \Phi(X_n) - \bar{V}_n^M(X_n) \right|^2 \right] \right).$$

It remains to note that $\mathbb{E}_M\left[\left| \hat{V}_n^M(X_n) - \bar{V}_n^M(X_n) \right|^2 \right] \leq \mathbb{E}_M\left[\left| \tilde{V}_n^M(X_n) - \bar{V}_n^M(X_n) \right|^2 \right]$ always holds, and this completes the proof of (5.4.2).

Next, let us show

$$\begin{aligned}
&\inf_{\Phi\in\mathcal{V}_M} \left\| \Phi(X_n) - \bar{V}_n(X_n) \right\|_{M,2} \\
= \;& \mathcal{O}\Bigg(\gamma_M^2 \sqrt{\frac{K_M \log(M)}{M}} + \sup_{n\leq k\leq N} \inf_{\Phi\in\mathcal{V}_M} \left\| \Phi(X_n) - V_n(X_n) \right\|_{M,2} \\
&+ \inf_{A\in\mathcal{A}_M} \mathbb{E}_M\left[\left| A(X_n) - a_n^{\mathrm{opt}}(X_n) \right| \right] + \left\| V_{n+1}(X_{n+1}) - \hat{V}_{n+1}^M(X_{n+1}) \right\|_{M,2} \Bigg).
\end{aligned}$$

For this, take some arbitrary $\Phi \in \mathcal{V}_M$ and split

$$\left\| \Phi(X_n) - \bar{V}_n^M(X_n) \right\|_{M,2} \leq \left\| \Phi(X_n) - V_n(X_n) \right\|_{M,2} + \left\| V_n(X_n) - \bar{V}_n^M(X_n) \right\|_{M,2},$$

so that:

$$\begin{aligned}
\inf_{\Phi\in\mathcal{V}_M} \left\| \Phi(X_n) - \bar{V}_n^M(X_n) \right\|_{M,2} \;\leq\; & \inf_{\Phi\in\mathcal{V}_M} \left\| \Phi(X_n) - V_n(X_n) \right\|_{M,2} \qquad (5.4.25) \\
& + \left\| V_n(X_n) - \bar{V}_n^M(X_n) \right\|_{M,2}.
\end{aligned}$$

To bound the last term in the r.h.s. of (5.4.25), we write

$$\begin{aligned}\left\|V_n(X_n)-\bar{V}_n^M(X_n)\right\|_{M,2} &\\ \leq \left\|V_n(X_n)-\inf_{a\in A}\left\{f\left(X_n,a\right)+\mathbb{E}_M^a\left[\hat{V}_{n+1}^M\left(X_{n+1}\right)\left|X_n\right.\right]\right\}\right\|_{M,2} &\\ +\left\|\inf_{a\in A}\left\{f\left(X_n,a\right)+\mathbb{E}_M^a\left[\hat{V}_{n+1}^M\left(X_{n+1}\right)\left|X_n\right.\right]\right\}-\bar{V}_n^M(X_n)\right\|_{M,2}&\end{aligned}$$

Use the dynamic programming principle, assumption **(Hd)** and (5.4.22) to get:

$$\begin{aligned}\left\|V_n(X_n)-\bar{V}_n^M(X_n)\right\|_{M,2} \leq \|r\|_\infty \left\|V_{n+1}(X_{n+1})-\hat{V}_{n+1}^M\left(X_{n+1}\right)\right\|_{M,2} &\\ +\sqrt{2\left((N-n)\|f\|_\infty+\|g\|_\infty\right)\left(2\varepsilon_{\mathrm{HN},n}^{\mathrm{esti}}+\varepsilon_{\mathrm{HN},n}^{\mathrm{approx}}\right)}.&\end{aligned}$$

We then notice that

$$\begin{aligned}\left|V_{n+1}(X_{n+1})-\hat{V}_{n+1}^M(X_{n+1})\right|^2 &\\ \leq 2\|r\|_\infty\left((N-n)\|f\|_\infty+\|g\|_\infty\right)\left|V_{n+1}(X_{n+1})-\hat{V}_{n+1}^M(X_{n+1})\right|&\end{aligned}$$

holds a.s., so that

$$\begin{aligned}\left\|V_n(X_n)-\bar{V}_n^M(X_n)\right\|_{M,2} &\\ \leq \sqrt{2\|r\|_\infty\left((N-n)\|f\|_\infty+\|g\|_\infty\right)\left\|V_{n+1}(X_{n+1})-\hat{V}_{n+1}^M\left(X_{n+1}\right)\right\|_{M,1}} &\\ +\sqrt{2\left((N-n)\|f\|_\infty+\|g\|_\infty\right)\left(2\varepsilon_{\mathrm{HN},n}^{\mathrm{esti}}+\varepsilon_{\mathrm{HN},n}^{\mathrm{approx}}\right)},&\end{aligned}$$

and use Lemma 5.4.4 to bound $\varepsilon_{\mathrm{HN},n}^{\mathrm{approx}}$. By plugging into (5.4.25), and using the estimations in Lemmas 5.4.3 and 5.4.4, we obtain the estimate (5.4.24). Together with (5.4.2), this proves the required estimate(5.4.2). By induction, we get as $M\to\infty$,

$$\begin{aligned}\mathbb{E}_M\left[\left|\hat{V}_n^M(X_n)-V_n(X_n)\right|\right] &= \mathcal{O}_{\mathbb{P}}\Bigg(\left(\gamma_M^4\frac{K_M\log(M)}{M}\right)^{\frac{1}{2(N-n)}}+\left(\gamma_M^4\frac{\rho_M^2\eta_M^2\log(M)}{M}\right)^{\frac{1}{4(N-n)}} &\\ &+\sup_{n\leq k\leq N}\inf_{\Phi\in\mathcal{V}_M}\left(\mathbb{E}_M\left[\left|\Phi(X_k)-V_k(X_k)\right|^2\right]\right)^{\frac{1}{2(N-n)}} &\\ &+\sup_{n\leq k\leq N}\inf_{A\in\mathcal{A}_M}\left(\mathbb{E}\left[\left|A(X_k)-a_k^{\mathrm{opt}}(X_k)\right|\right]\right)^{\frac{1}{2(N-n)}}\Bigg),&\end{aligned}$$

which completes the proof of Theorem 5.4.2. □

5.4.3 Hybrid-LaterQ algorithm

In this paragraph, we analyze the convergence of the Hybrid-LaterQ algorithm described in Section 5.3.2.2.

We shall make the following assumption on F to ensure the convergence of the Hybrid-LaterQ algorithm.

(HF-LQ) Assume F to be such that:

1. (Estimation error Assumption) **(HF)** holds, i.e. for all $e\in E$, there exists $C(e)>0$ such that for all couples (x,a) and (x',a') in $\mathcal{X}\times\mathbb{A}$:

$$\left|F(x,a,\varepsilon)-F(x',a',\varepsilon)\right|\leq C(\varepsilon)\left(|x-x'|+|a-a'|\right).$$

Recall that for all integer $M > 0$, ρ_M is defined as

$$\rho_M = \mathbb{E}\Big[\sup_{1\leq m\leq M} C(\varepsilon^m)\Big],$$

where the $(\varepsilon^m)_m$ is a i.i.d. sample of the noise ε.

2. (Quantization Assumption) There exists a constant $[F]_L > 0$ such that for all $(x, a) \in \mathcal{X} \times \mathbb{A}$ and all pair of r.v. $(\varepsilon, \varepsilon')$, it holds:

$$\|F(x, a, \varepsilon) - F(x, a, \varepsilon')\|_2 \leq [F]_L \|\varepsilon - \varepsilon'\|_2.$$

As for the hybrid-now algorithm, we shall consider neural networks for the value function estimation with one hidden layer, K neurons with total variation γ, kernel bounded by η, a sigmoid activation function σ for the hidden layer, and no activation function for the output layer (i.e. the last layer), which is represented by the parametric set of function ${}^{\eta}\mathcal{V}_K^{\gamma}$. Let η_M, K_M and γ_M be integers such that:

$$\begin{array}{ccccc} K_M \xrightarrow[M\to\infty]{} \infty & , & \gamma_M \xrightarrow[M\to\infty]{} \infty & , & \eta_M \xrightarrow[M\to\infty]{} \infty \\ & & \rho_M \eta_M \gamma_M^2 \sqrt{\frac{\log(M)}{M}} \xrightarrow[M\to\infty]{} 0. & & \end{array} \tag{5.4.26}$$

In the sequel we denote by $\mathcal{V}_M := {}^{\eta_M}\mathcal{V}_{K_M}^{\gamma_M}$ the space of neural networks parametrized by the values η_M, γ_M and K_M that satisfy (5.4.26). We also consider the class $\mathcal{A}_M$ of neural networks for estimated feedback optimal control at time $n = 0, \ldots, N-1$, as described in Section 5.4.1, with the same parameters η_M, γ_M and K_M.

Recall that the approximation of the value function and optimal policy in the Hybrid-LaterQ algorithm is computed in backward induction as follows:

- Initialize $\hat{V}_N^M = g$
- For $n = N-1, \ldots, 0$, generate a training sample $X_n^{(m)}$, $m = 1, \ldots, M$ from the training distribution μ, and a training sample for the exogenous noise $\varepsilon_{n+1}^{(m)}$, $m = 1, \ldots, M$.

  (i) compute the approximated policy at time n

$$\hat{a}_n^M \in \operatorname*{argmin}_{A\in\mathcal{A}_M} \frac{1}{M}\sum_{m=1}^{M} \big[f(X_n^{(m)}, A(X_n^{(m)})) + \hat{V}_{n+1}^M(X_{n+1}^{(m),A})\big]$$

  where $X_{n+1}^{(m),A} = F(X_n^{(m)}, A(X_n^{(m)}), \varepsilon_{n+1}^{(m)}) \rightsquigarrow P^{A(X_n^{(m)})}(X_n^{(m)}, dx')$.

  (ii) compute an untruncated interpolation of the value function at time $n+1$

$$\tilde{V}_{n+1}^M \in \operatorname*{argmin}_{\Phi\in\mathcal{V}_M} \tfrac{1}{M}\textstyle\sum_{m=1}^{M} \Big[\hat{V}_{n+1}^M(X_{n+1}^{(m),\hat{a}_n^M}) - \Phi\big(X_{n+1}^{(m),\hat{a}_n^M}\big)\Big]^2, \tag{5.4.27}$$

  and set the truncated interpolation of the value function at time $n+1$

$$\tilde{V}_{n+1}^{\mathrm{trun}} = \max\Big(\min\big(\tilde{V}_{n+1}^M, \|V_{n+1}\|_\infty\big), -\|V_{n+1}\|_\infty\Big).$$

  (iii) update/compute the estimated value function

$$\hat{V}_n^M(x) = f(x, \hat{a}_n^M(x)) + \sum_{\ell=1}^{L} p_\ell \tilde{V}_{n+1}^{\mathrm{trun}}\big(F(x, \hat{a}_n^M(x), e_\ell)\big),$$

  where $\hat{\varepsilon}_n$ is a L-optimal quantizer of ε_n on the grid $\{e_1, \ldots, e_L\}$ with weights $(p_1, \ldots, p_L)$.

Remark 5.4.11. 1. It is straightforward to see that the neuronal network functions in $\mathcal{V}_M$ are Lipschitz with Lipschitz coefficient bounded by $\eta_M\gamma_M$. We highly rely on this property to show the convergence of the Hybrid-LaterQ algorithm.

2. Note that (5.4.27) is an interpolation step. In the pseudo-code above, we decided to interpolate the value function $\tilde{V}_n^Q$ using neural networks in $\mathcal{V}_M$ by reducing an empirical quadratic norm. However, we could have chosen other families of functions and other loss criterion to minimize. Gaussian processes have been recently reconsidered to interpolate functions, see [LM18]. □

We now state our main result about the convergence of the Hybrid-LaterQ algorithm.

Theorem 5.4.3. *Assume that there exists an optimal feedback control $(a_k^{\rm opt})_{k=n}^{N-1}$ for the control problem with value function V_n, $n = 0, \dots, N$. Take $X_n \rightsquigarrow \mu$, and let L_M be a sequence of integers such that*

$$L_M \xrightarrow[M\to\infty]{} \infty, \quad \text{and} \quad \frac{\eta_M\gamma_M}{L_M^{1/d}} \xrightarrow[M\to\infty]{} 0.$$

Take L_M points for the optimal quantization of the exogenous noise. Then, it holds as $M \to \infty$:

$$\mathbb{E}_M\big[|\hat{V}_n^M(X_n) - V_n(X_n)|\big] = \mathcal{O}_{\mathbb{P}}\Bigg(\rho_M\eta_M\gamma_M^2\sqrt{\frac{\log(M)}{M}} + \frac{\eta_M\gamma_M}{L_M^{1/d}} + \sup_{n\le k\le N}\inf_{A\in\mathcal{A}_M}\mathbb{E}\big[|A(X_k) - a_k^{\rm opt}(X_k)|\big] + \sup_{n+1\le k\le N}\inf_{\Phi\in\mathcal{V}_M}\mathbb{E}\big[|\Phi(X_k) - V_k(X_k)|\big]\Bigg). \tag{5.4.28}$$

Comment: Theorem 5.4.3, combined to Proposition 5.4.1, show that estimator $\hat{V}_n^M$ provided by Hybrid-LaterQ algorithm is consistent, i.e. converges in $\mathbb{L}^1$ toward the value function V_n at time n when the number of points for the regression and quantization goes to infinity.

Remark 5.4.12. Note that the dimension d of the state space appears (explicitly) in the quantization error written in (5.4.28), as well as (implicitly) in the approximation errors associated to the value functions and optimal control learning. See for example (5.4.8) for an explicit dependence of the approximation error on d. □

In order to prove Theorem 5.4.3, let us introduce the estimation error at time n associated to the Hybrid-LaterQ algorithm by

$$\varepsilon_{{\rm LQ},n}^{\rm esti} := \sup_{A\in\mathcal{A}_M}\Bigg|\frac{1}{M}\sum_{m=1}^{M}\Big[f(X_n^{(m)}, A(X_n^{(m)})) + \hat{Y}_{n+1}^{(m),A}\Big] - \mathbb{E}_{M,n,X_n}^{A}\Big[f(X_n, A(X_n)) + \hat{V}_{n+1}^{M}\big(X_{n+1}\big)\Big]\Bigg|,$$

where $\hat{Y}_{n+1}^{(m),A} = \hat{V}_{n+1}^{M}\big(X_{n+1}^{(m),A}\big)$, and $\mathbb{E}_{M,n,X_n}^{A}[.]$ stands for the expectation conditioned by the training set and X_n when decision A has been taken at time n.

We have the following bound on this estimation error:

Lemma 5.4.5. *For $n = 0, \dots, N-1$, it holds:*

$$\mathbb{E}\big[\varepsilon_{{\rm LQ},n}^{\rm esti}\big] \le \frac{\big(\sqrt{2}+16\big)\big((N-n)\|f\|_\infty + \|g\|_\infty\big) + 16[f]_L}{\sqrt{M}} + 16\frac{\rho_M\eta_M\gamma_M^2}{\sqrt{M}} \underset{M\to\infty}{=} \mathcal{O}\Big(\frac{\rho_M\eta_M\gamma_M^2}{\sqrt{M}}\Big).$$

Moreover,

$$\mathbb{E}_M\left[\varepsilon^{\text{esti}}_{\text{LQ},n}\right] \underset{M\to\infty}{=} \mathcal{O}\left(\rho_M\eta_M\gamma_M^2\sqrt{\frac{\log(M)}{M}}\right).$$

Remark 5.4.13. The result stated in Lemma 5.4.5 is the same as the one stated in Lemma 5.4.3 for the hybrid-now algorithm. This result can actually be proved using the same arguments, so we omit the proof here. □

Next, we introduce the approximation error at time n associated to the Hybrid-LaterQ algorithm by

$$\varepsilon^{\text{approx}}_{\text{LQ},n} = \inf_{A\in\mathcal{A}_M}\mathbb{E}_M\Big[f\big(X_n,A(X_n)\big)+\hat{Y}^A_{n+1}\Big] - \inf_{A\in\mathbb{A}^{\mathcal{X}}}\mathbb{E}_M\Big[f\big(X_n,A(X_n)\big)+\hat{Y}^A_{n+1}\Big],$$

where $\hat{Y}^A_{n+1} := \hat{V}^M_{n+1}\left(F\left(X_n,A(X_n),\varepsilon_{n+1}\right)\right)$.

We have the following bound on this approximation error, which is similar to the one stated in Lemma 5.4.4 for the Hybrid-Now algorithm. The proof is similar and is thus omitted here.

Lemma 5.4.6. *For $n=0,...,N-1$, it holds:*

$$\begin{aligned}\varepsilon^{\text{approx}}_{\text{LQ},n} \le ([f]_L+[r]_L\|V_{n+1}\|_\infty)\inf_{A\in\mathbb{A}^{\mathcal{X}}}\mathbb{E}_M\left[\left|A(X_n)-a^{\text{opt}}_n(X_n)\right|\right]\\ +\ 2\|r\|_\infty\mathbb{E}_M\left[\left|V_{n+1}(X_{n+1})-\hat{V}^M_{n+1}(X_{n+1})\right|\right].\end{aligned}$$

Proof of Theorem 5.4.3.

We split the L^1 norm as follows:

$$\begin{aligned}\left\|\hat{V}^M_n(X_n)-V_n(X_n)\right\|_{M,1} \le \left\|\hat{V}^M_n-\bar{V}^M_n\right\|_{M,1}+\left\|\bar{V}^M_n-\bar{V}^{\text{opt}}_n\right\|_{M,1}\\ +\left\|\bar{V}^{\text{opt}}_n-V_n\right\|_{M,1},\end{aligned} \tag{5.4.29}$$

where $(\bar{V}^M_n)_n$ is defined as:

$$\begin{cases}\bar{V}^M_N(x) &= g(x)\\ \bar{V}^M_n(x) &= f(x,\hat{a}^M_n(x))+\mathbb{E}_M\left[\tilde{V}^{\text{trunc}}_{n+1}\big(F(x,\hat{a}^M_n(x),\varepsilon_{n+1})\big)\right],\quad n=0,\dots,N-1,\end{cases}$$

and $(\bar{V}^{\text{opt}}_n)_n$ is defined as:

$$\begin{cases}\bar{V}^{\text{opt}}_N(x) &= g(x)\\ \bar{V}^{\text{opt}}_n(x) &= \inf_{a\in A}\left\{f(x,a)+\mathbb{E}_M\left[\tilde{V}^{\text{trunc}}_{n+1}\big(F(x,a,\varepsilon_{n+1})\big)\right]\right\},\ n=0,\dots,N-1.\end{cases}$$

Recall that $\|.\|_{M,p}=\left(\mathbb{E}_M\left[|.|^p\right]\right)^{\frac{1}{p}}$ stands for the $\mathbb{L}^p$-norm conditioned by the training set, for $p\in\{1,2\}$.

Step 1: The first term in the r.h.s. of (5.4.29) is the quantization error. We show that

$$\left\|\hat{V}^M_n-\bar{V}^M_n\right\|_{M,1}=\mathcal{O}_{\mathbb{P}}\left(\frac{\eta_M\gamma_M}{L_M^{1/d}}\right),\qquad \text{as } M\to\infty. \tag{5.4.30}$$

Denote by $\varepsilon^Q_p := \|\hat{V}^M_n(X_n)-\bar{V}^M_n(X_n)\|_p$ the $\mathbb{L}^p$-quantization error, for $p\in\{1,2\}$. Since $\tilde{V}^{\text{trunc}}_n$ is Lipschitz, for $n\in\{0,...,N\}$, with its Lipschitz coefficient bounded by $\eta_M\gamma_M$, we thus get:

$$\varepsilon^Q_2 := \|\hat{V}^M_n(X_n)-\bar{V}^M_n(X_n)\|_2 \le \eta_M\gamma_M[F]_L\|\hat{\varepsilon}_{n+1}-\varepsilon_{n+1}\|_2,$$

from assumption **(HF-LQ)**. Now, recall by Zador theorem about optimal quantization (see [GL00]) that there exists some positive constant C such that

$$\lim_{M\to+\infty}\left(L_M^{\frac{2}{d}}\|\hat{\varepsilon}_{n+1}-\varepsilon_{n+1}\|_2^2\right) \;=\; C.$$

By using Zador theorem in (**(HF-LQ)**) and with inequality $\varepsilon_1^Q \le \varepsilon_2^Q$, we obtain the bound (5.4.30) for the quantization error.

Step 2: We show: as $M\to\infty$,

$$\left\|\tilde{V}_{n+1}^{\text{trunc}}(X_{n+1})-\hat{V}_{n+1}^M(X_{n+1})\right\|_{M,1} = \mathcal{O}_{\mathbb{P}}\Bigg(\sqrt{\frac{\log(M)}{M}}$$

$$+\inf_{\Phi\in\mathcal{V}_M}\|\Phi(X_{n+1})-V_{n+1}(X_{n+1})\|_{M,1}+\left\|V_{n+1}(X_{n+1})-\hat{V}_{n+1}^M(X_{n+1})\right\|_2^2\Bigg).$$

Denote by

$$\hat{R}_{n+1}\left(\tilde{V}_{n+1}^{\text{trunc}}\right)=\frac{1}{M}\sum_{m=1}^M\left|\tilde{V}_{n+1}^{\text{trunc}}(X_{n+1}^{(m)})-\hat{V}_{n+1}^M(X_{n+1}^{(m)})\right|^2$$

the empirical quadratic risk, and by

$$R_{n+1}\left(\tilde{V}_{n+1}^{\text{trunc}}\right)=\mathbb{E}_M\left[\left|\tilde{V}_{n+1}^{\text{trunc}}(X_{n+1})-\hat{V}_{n+1}^M(X_{n+1})\right|^2\right]$$

its associated quadratic risk. From the central limit theorem, we have

$$\frac{\hat{R}_{n+1}\left(\tilde{V}_{n+1}^{\text{trunc}}\right)-R_{n+1}\left(\tilde{V}_{n+1}^{\text{trunc}}\right)}{\sigma_{M,n+1}\sqrt{M}}\xrightarrow[M\to\infty]{\mathcal{L}}\mathcal{N}(0,1)$$

where $\sigma_{M,n+1}$ is the standard variation conditioned by the training set, defined as

$$\sigma_{M,n+1}^2=\text{Var}_M\left(\left|\tilde{V}_{n+1}^{\text{trunc}}(X_{n+1})-\hat{V}_{n+1}^M(X_{n+1})\right|^2\right).$$

Use inequality $\tilde{V}_{n+1}^{\text{trunc}}(X_{n+1})-\hat{V}_{n+1}^M(X_{n+1})\le(N-n)\|f\|_\infty+\|g\|_\infty$ to bound $\sigma_{M,n+1}$ by a constant that does not depend on M, and get

$$R_{n+1}\left(\tilde{V}_{n+1}^{\text{trunc}}\right)=\mathcal{O}_{\mathbb{P}}\left(\sqrt{\frac{\log(M)}{M}}+\frac{1}{M}\sum_{m=1}^M\left|\tilde{V}_{n+1}^{\text{trunc}}(X_{n+1}^{(m)})-\hat{V}_{n+1}^M(X_{n+1}^{(m)})\right|^2\right),$$

which, after noticing that

$$\frac{1}{M}\sum_{m=1}^M\left|\tilde{V}_{n+1}^{\text{trunc}}(X_{n+1}^{(m)})-\hat{V}_{n+1}^M(X_{n+1}^{(m)})\right|^2\le\frac{1}{M}\sum_{m=1}^M\left|\tilde{V}_{n+1}^M(X_{n+1}^{(m)})-\hat{V}_{n+1}^M(X_{n+1}^{(m)})\right|^2,$$

implies:

$$R_{n+1}\left(\tilde{V}_{n+1}^{\text{trunc}}\right)=\mathcal{O}_{\mathbb{P}}\left(\sqrt{\frac{\log(M)}{M}}+\inf_{\Phi\in\mathcal{V}_M}\frac{1}{M}\sum_{m=1}^M\left|\Phi(X_{n+1}^{(m)})-\hat{V}_{n+1}^M(X_{n+1}^{(m)})\right|^2\right). \tag{5.4.31}$$

Once again from the central limit theorem, we derive:

$$\inf_{\Phi\in\mathcal{V}_M}\frac{1}{M}\sum_{m=1}^M\left|\Phi(X_{n+1}^{(m)})-\hat{V}_{n+1}^M(X_{n+1}^{(m)})\right|^2$$

$$=\mathcal{O}_{\mathbb{P}}\left(\sqrt{\frac{\log(M)}{M}}+\inf_{\Phi\in\mathcal{V}_M}\left\|\Phi(X_{n+1})-\hat{V}_{n+1}^M(X_{n+1})\right\|_2^2\right). \tag{5.4.32}$$

Indeed, first write

$$\begin{aligned}
&\mathbb{P}\left(\inf_{\Phi\in\mathcal{V}_M}\frac{1}{M}\sum_{m=1}^{M}\left|\Phi(X_{n+1}^{(m)})-\hat{V}_{n+1}^{M}(X_{n+1}^{(m)})\right|^2\right.\\
&\qquad\left.\leq\sqrt{\frac{\log(M)}{M}}+\inf_{\Phi\in\mathcal{V}_M}\left\|\Phi(X_{n+1})-\hat{V}_{n+1}^{M}(X_{n+1})\right\|_2^2\right)\\
&\leq\mathbb{P}\left(\frac{1}{M}\sum_{m=1}^{M}\left|\Phi(X_{n+1}^{(m)})-\hat{V}_{n+1}^{M}(X_{n+1}^{(m)})\right|^2\right.\\
&\qquad\left.\leq\sqrt{\frac{\log(M)}{M}}+\inf_{\Phi\in\mathcal{V}_M}\left\|\Phi(X_{n+1})-\hat{V}_{n+1}^{M}(X_{n+1})\right\|_2^2\right)\ \text{for all } \Phi\in\mathcal{V}_M,\\
&\leq\mathbb{P}\left(\frac{1}{M}\sum_{m=1}^{M}\left|\tilde{\Phi}(X_{n+1}^{(m)})-\hat{V}_{n+1}^{M}(X_{n+1}^{(m)})\right|^2\right.\\
&\qquad\left.\leq\sqrt{\frac{\log(M)}{M}}+\left\|\tilde{\Phi}(X_{n+1})-\hat{V}_{n+1}^{M}(X_{n+1})\right\|_2^2\right),
\end{aligned}$$

where $\tilde{\Phi}=\operatorname*{argmin}_{\Phi\in\mathcal{V}_M}\left\|\Phi(X_{n+1})-\hat{V}_{n+1}^{M}(X_{n+1})\right\|_2^2$. Then apply the Central limit theorem to get (5.4.32).

Plugging (5.4.32) into (5.4.31) leads to

$$R_{n+1}\left(\tilde{V}_{n+1}^{\text{trunc}}\right)=\mathcal{O}_{\mathbb{P}}\left(\sqrt{\frac{\log(M)}{M}}+\inf_{\Phi\in\mathcal{V}_M}\left\|\Phi(X_{n+1})-\hat{V}_{n+1}^{M}(X_{n+1})\right\|_2^2\right).$$

Apply the triangular inequality to finally obtain:

$$\begin{aligned}
R_{n+1}\left(\tilde{V}_{n+1}^{\text{trunc}}\right)=\mathcal{O}_{\mathbb{P}}\Bigg(&\sqrt{\frac{\log(M)}{M}}+\inf_{\Phi\in\mathcal{V}_M}\|\Phi(X_{n+1})-V_{n+1}(X_{n+1})\|_2^2\\
&+\left\|V_{n+1}(X_{n+1})-\hat{V}_{n+1}^{M}(X_{n+1})\right\|_2^2\Bigg).
\end{aligned}$$

It remains to notice that

$$\begin{aligned}
&\left\|V_{n+1}(X_{n+1})-\hat{V}_{n+1}^{M}(X_{n+1})\right\|_2^2\\
&\qquad\leq((N-n-1)\|f\|_\infty+\|g\|_\infty)\left\|V_{n+1}(X_{n+1})-\hat{V}_{n+1}^{M}(X_{n+1})\right\|_{M,1},
\end{aligned}$$

to obtain inequality (**(HF-LQ)**).

Step 3: let us show

$$\begin{aligned}
\left\|\bar{V}_n^M-\bar{V}_n^{\text{opt}}\right\|_{M,1}=\mathcal{O}_{\mathbb{P}}\Bigg(&\rho_M\eta_M\gamma_M^2\sqrt{\frac{\log(M)}{M}}+\inf_{A\in\mathcal{A}_M}\left\|A(X_n)-a_n^{\text{opt}}(X_n)\right\|_{M,1}\\
&+\left\|\tilde{V}_{n+1}^{\text{trunc}}(X_n)-V_n(X_n)\right\|_{M,1}\Bigg).
\end{aligned}\tag{5.4.33}$$

Note that once again it holds

$$\left\|\bar{V}_n^M-\bar{V}_n^{\text{opt}}\right\|_{M,1}\leq 2\varepsilon_n^{\text{esti}}+\varepsilon_n^{\text{approx}},$$

which can be shown by similar arguments as those used to prove of inequality (5.4.21). It remains to bound the estimation and approximation errors by using estimations (5.4.5) and (5.4.6) to get (5.4.33).

Step 4: We show

$$\begin{aligned}\left\|\bar{V}_n^{\rm opt}(X_n) - V_n(X_n)\right\|_{M,1} \le \|r\|_\infty \left\|\hat{V}_{n+1}^M(X_{n+1}) - V_{n+1}(X_{n+1})\right\|_{M,1} \\ + \|r\|_\infty \left\|\tilde{V}_{n+1}^{\rm trunc}(X_{n+1}) - \hat{V}_{n+1}^M(X_{n+1})\right\|_{M,1},\end{aligned} \tag{5.4.34}$$

where $X_{n+1} \sim \mu$. For this, denote by $(\bar{V}_n')_{0\le n\le N}$ the following auxiliary process:

$$\left\{\begin{array}{rcl}\bar{V}_N'(x) &=& g(x)\\ \bar{V}_n'(x) &=& \inf_{a\in A}\left\{f(x,a) + \mathbb{E}_M\left[\hat{V}_{n+1}^M\big(F(x,a,\varepsilon_{n+1})\big)\right]\right\}, \quad n = 0,\dots,N-1,\end{array}\right.$$

and notice that we have under assumption **(Hd)**:

$$\begin{aligned}\left\|\bar{V}_n^{\rm opt}(X_n) - V_n(X_n)\right\|_{M,1} &\le \left\|\bar{V}_n^{\rm opt}(X_n) - \bar{V}_n'(X_n)\right\|_{M,1} + \left\|\bar{V}_n'(X_n) - V_n(X_n)\right\|_{M,1}\\ &\le \|r\|_\infty \left\|\hat{V}_{n+1}^M(X_{n+1}) - V_{n+1}(X_{n+1})\right\|_{M,1}\\ &\quad + \|r\|_\infty \left\|\tilde{V}_{n+1}^{\rm trunc}(X_{n+1}) - \hat{V}_{n+1}^M(X_{n+1})\right\|_{M,1},\end{aligned}$$

as stated in (5.4.34).

Step 5 Conclusion: By plugging (5.4.30), (5.4.33) and (5.4.34) into (5.4.29), we derive the following bound for the l.h.s. of (5.4.29):

$$\begin{aligned}\left\|\hat{V}_n^M(X_n) - V_n(X_n)\right\|_{M,1} = \mathcal{O}_{\mathbb{P}}\Bigg(&\frac{\eta_M\gamma_M}{L_M^{1/d}} + \rho_M\eta_M\gamma_M^2\sqrt{\frac{\log(M)}{M}}\\ &+ \inf_{\Phi\in\mathcal{V}_M}\|\Phi(X_{n+1}) - V_{n+1}(X_{n+1})\|_{M,1} + \inf_{A\in\mathcal{A}_M}\left\|A(X_n) - a_n^{\rm opt}(X_n)\right\|_{M,1}\\ &+ \left\|\hat{V}_{n+1}^M(X_n) - V_{n+1}(X_{n+1})\right\|_{M,1}\Bigg), \quad \text{as } M\to+\infty.\end{aligned}$$

By induction, we get for $n = 0,\dots,N-1$:

$$\begin{aligned}\left\|\hat{V}_n^M(X_n) - V_n(X_n)\right\|_{M,1} = \mathcal{O}_{\mathbb{P}}\Bigg(&\frac{\eta_M\gamma_M}{L_M^{1/d}} + \rho_M\eta_M\gamma_M^2\sqrt{\frac{\log(M)}{M}}\\ &+ \sup_{n\le k\le N}\inf_{A\in\mathcal{A}_M}\left\|A(X_k) - a_k^{\rm opt}(X_k)\right\|_{M,1} + \sup_{n+1\le k\le N}\inf_{\Phi\in\mathcal{V}_M}\|\Phi(X_k) - V_k(X_k)\|_{M,1}\Bigg),\end{aligned}$$

which is the result stated in Theorem 5.4.3. □

5.5 Conclusion

This paper develops new machine learning algorithms for high-dimensional Markov decision process. We propose and compare three algorithms based on dynamic programming, performance/hybrid iteration and neural networks for approximating the control and value function. The main theoretical contribution is to provide a detailed convergence analysis for each of these algorithms: by using least squares neural network regression, we state error estimates in terms of the universal approximation error of neural networks, and of the statistical risk when estimating the network function. Numerical tests on various applications are presented in a companion paper [BHLP18].

5.A Localization

In this section, we show how to relax the boundedness condition on the state space by a localization argument.

Let $R > 0$. Consider the localized state space $\bar{B}_d^{\mathcal{X}}(0,R) := \mathcal{X} \cap \{\|x\|_d \leq R\}$, where $\|.\|_d$ is the Euclidean norm of $\mathbb{R}^d$. Let $\left(\bar{X}_n\right)_{0\leq n\leq N}$ be the Markov chain defined by its transition probabilities as

$$\mathbb{P}\left(\bar{X}_{n+1} \in O \middle| \bar{X}_n = x, a\right) = \int_O r(x,a;y) d\pi_R \circ \mu(y), \qquad \text{for } n = 0, \dots, N-1,$$

for all Borelian O in $\bar{B}_d^{\mathcal{X}}(0,R)$, where π_R is the Euclidean projection of $\mathbb{R}^d$ on $\bar{B}_d^{\mathcal{X}}(0,R)$, and $\pi_R \circ \mu$ is the pushforward measure of μ. Notice that the transition probability of $\bar{X}$ admits the same density r, for which **(Hd)** holds, w.r.t. $\pi_R \circ \mu$.

Define $(\bar{V}_n^R)_n$ as the value function associated to the following stochastic control problem for $\left(\bar{X}_n\right)_{n=0}^N$:

$$\begin{cases} \bar{V}_N^R(x) &= g(x), \\ \bar{V}_n^R(x) &= \inf\limits_{\alpha \in \mathcal{C}} \mathbb{E}\left[\sum\limits_{k=n}^{N-1} f\left(\bar{X}_k, \alpha_k\right) + g\left(\bar{X}_N\right)\right], \text{ for } n = 0, \dots, N-1, \end{cases} \tag{5.A.1}$$

for $x \in \bar{B}_d^{\mathcal{X}}(0,R)$. By the dynamic programming principle, $(\bar{V}_n^R)_n$ is solution of the following Bellman backward equation:

$$\begin{cases} \bar{V}_N^R(x) = g(x) \\ \bar{V}_n^R(x) = \inf\limits_{a \in \mathbb{A}} \left\{ f(x,a) + \mathbb{E}_n^a\left[\bar{V}_{n+1}^R\big(\pi_R\big(F(x,a,\varepsilon_{n+1})\big)\big)\right]\right\}, \qquad \forall x \in B_d^{\mathcal{X}}(0,R), \end{cases}$$

where, again, π_R is the Euclidean projection on $B_d^{\mathcal{X}}(0,R)$.

Lemma 5.A.1. *$\bar{V}_n^R$ is Lipschitz for $n \in \{0, ..., N\}$, and moreover the following bound on its Lipschitz coefficient holds:*

$$[\bar{V}_n^R]_L \leq \frac{1}{[F]_L^n} \frac{1 - [F]_L^{N-n}}{1 - [F]_L} [f]_L + [F]_L^{N-n} [g]_L, \tag{5.A.2}$$

with the standard convention $\frac{1-x^n}{1-x} := n$ if $x = 1$.

Proof: Using the dynamic programming equation, assumption **(HR)**, and also the fact that π_R is 1-Lipschitz as a projection on the convex $B_d(0,R)$ straightforwardly yields:

$$\begin{cases} [\bar{V}_N^R]_L &\leq \|g\|_L, \\ [\bar{V}_n^R]_L &\leq [f]_L + [F]_L \left[\bar{V}_{n+1}^R\right]_L, \qquad \text{for } n = \{0, ..., N-1\}. \end{cases}$$

Proceed by induction to prove (5.A.2).

□

We shall assume two conditions on the measure μ.
(Hloc) μ is such that:

$$\mathbb{E}\left[|\pi_R(X) - X|\right] \xrightarrow[R\to\infty]{} 0 \quad \text{and} \quad \mathbb{P}\left(|X| > R\right) \xrightarrow[R\to\infty]{} 0, \quad \text{where } X \sim \mu.$$

Using the dominated convergence theorem, it is straightforward to see that **(Hloc)** holds if μ admits a moment of order 1.

Proposition 5.A.1. *Let $X_n \sim \mu$. It holds:*

$$\mathbb{E}\Big[\Big|\bar{V}_n^R\big(\pi_R(X_n)\big) - V_n\big(X_n\big)\Big|\Big] \leq \|V\|_\infty \Big([r]_L \mathbb{E}\left[|\pi_R(X_n) - X_n|\right] + 2\mathbb{P}\left(|X_n| > R\right)\Big) \frac{1 - \|r\|_\infty^{N-n}}{1 - \|r\|_\infty} + [g]_L \|r\|_\infty^{N-n} \mathbb{E}\left[|\pi_R(X_n) - X_n|\right],$$

*where we denote $\|V\|_\infty = \sup_{0\leq k\leq N} \|V_k\|_\infty$, and use the convention $\frac{1-x^p}{1-x} = p$ for $x = 0$ and $p > 1$. Consequently, for all $n = 0, ..., N$, we get under **(Hloc)**:*

$$\mathbb{E}\Big[\Big|\bar{V}_n^R\big(\pi_R(X_n)\big) - V_n\big(X_n\big)\Big|\Big] \xrightarrow[R\to\infty]{} 0, \qquad \text{where } X_n \sim \mu.$$

Comment: Proposition 5.A.1 states that if $\mathcal{X}$ is not bounded, the control problem (5.A.1) associated to a bounded controlled process $\bar{X}$ can be as close as desired, in $\mathbb{L}^1(\mu)$ sense, to the original control problem by taking R large enough. Moreover, as stated before, the transition probability of $\bar{X}$ admits the same density r as X w.r.t. the pushforward measure $\pi_R \circ \mu$.

Proof of Proposition 5.A.1. Take $X_n \sim \mu$ and write:

$$\mathbb{E}\Big[\big|\bar{V}_n^R(\pi_R(X_n)) - V_n(X_n)\big|\Big] \leq \mathbb{E}\left[\big|\bar{V}_n^R(X_n) - V_n(X_n)\big|\mathbb{1}_{|X_n|\leq R}\right] + \mathbb{E}\left[\big|\bar{V}_n^R(\pi_R(X_n)) - V_n(X_n)\big|\mathbb{1}_{|X_n|\geq R}\right]. \tag{5.A.3}$$

Let us first bound the first term in the r.h.s. of (5.A.3), by showing that, for $n = 0, \ldots, N-1$:

$$\mathbb{E}\left[\big|\bar{V}_n^R(X_n) - V_n(X_n)\big|\mathbb{1}_{|X_n|\leq R}\right] \leq \|r\|_\infty \mathbb{E}\Big[\big|\bar{V}_{n+1}^R(\pi_R(X_{n+1})) - V_{n+1}(X_{n+1})\big|\Big] + [r]_L \|V_{n+1}\|_\infty \mathbb{E}\left[|\pi_R(X_{n+1}) - X_{n+1}|\right], \text{ with } X_{n+1} \sim \mu.$$

Take $x \in \bar{B}_d(0, R)$ and notice that

$$\begin{aligned}
\big|\bar{V}_n^R(x) - V_n(x)\big| &\leq \inf_{a\in A} \left\{ \int_A \big|\bar{V}_{n+1}^R\left(\pi_R(y)\right) - V_{n+1}(y)\big|\, r\left(x, a; \pi_R(y)\right) d\mu(y) \right. \\
&\qquad \left. + \int |V_{n+1}(y)|\, |r(x, a; \pi_R(y)) - r(x, a; y)|\, d\mu(y) \right\} \\
&\leq \|r\|_\infty \mathbb{E}\left[\big|\bar{V}_{n+1}^R\left(\pi(X_{n+1})\right) - V_{n+1}\left(X_{n+1}\right)\big|\right] \\
&\quad + [r]_L \|V_{n+1}\|_\infty \mathbb{E}\left[|\pi_R(X_{n+1}) - X_{n+1}|\right], \qquad \text{where } X_{n+1} \sim \mu.
\end{aligned}$$

It remains to inject this bound in the expectation to obtain **(Hloc)**.

To bound the second term in the r.h.s. of (5.A.3), notice that

$$\big|\bar{V}_n^R(\pi_R(X_n)) - V_n(X_n)\big| \leq 2\|V_n\|_\infty$$

holds a.s., which implies:

$$\mathbb{E}\left[\big|\bar{V}_n^R(\pi_R(X_n)) - V_n(X_n)\big|\mathbb{1}_{|X_n|\geq R}\right] \leq 2\|V_n\|_\infty \mathbb{P}\left(|X_n| > R\right).$$

Use the dynamic programming equation and the standard inequality $|\inf_i a_i - \inf_i b_i| \leq \sup_i |a_i - b_i|$ to write:

$$\begin{aligned}
\mathbb{E}\Big[\big|\bar{V}_n^R(X_n) - V_n(X_n)\big|\Big] &\leq \mathbb{E}\left[\sup_{a\in\mathbb{A}} \Big|\bar{V}_{n+1}^R\Big(\pi\big(F(X_n, a, \varepsilon_{n+1})\big)\Big) - V_{n+1}\big(F(X_n, a, \varepsilon_{n+1})\big)\Big|\right] \\
&\quad + \mathbb{E}\Big[\big|V_n\big(\pi_R(X_n)\big) - V_n(X_n)\big|\Big] \\
&\leq 3\Big(\Big(\big[\bar{V}_n^R\big]_L^2 + [V_n]_L^2\Big)\mathbb{E}\Big[\big|\pi_R(X_n) - X_n\big|^2\Big] + \mathbb{E}\Big[\big|\bar{V}_n^R\big(\pi_R(X_n)\big) - V_n\big(\pi_R(X_n)\big)\big|^2\Big]\Big).
\end{aligned} \tag{5.A.4}$$

Use the programming equation and assumption **(Hd)** to bound the last term in the r.h.s. of (5.A.4):

$$\begin{aligned}
\mathbb{E}\Big[\big|\bar{V}_n^R\big(\pi_R(X_n)\big) - V_n\big(\pi_R(X_n)\big)\big|\Big] &\leq \mathbb{E}\Big[\sup_{a\in A}\mathbb{E}^a_{n,X_n}\Big[\Big|\bar{V}_{n+1}^R\Big(\pi_R\big(F(\pi_R(X_n),a,\varepsilon_{n+1})\big)\Big) \\
&\qquad - V_{n+1}\Big(\pi_R\big(F(\pi_R(X_n),a,\varepsilon_{n+1})\big)\Big)\Big|\Big]\Big] \\
&\leq \|r\|_\infty \mathbb{E}\Big[\big|\bar{V}_{n+1}^R\big(\pi_R(X_{n+1})\big) - V_{n+1}(\pi_R(X_{n+1}))\big|\Big] \\
&\leq \|r\|_\infty\Big(\mathbb{E}\Big[\big|\bar{V}_{n+1}^R\big(\pi_R(X_{n+1})\big) - V_{n+1}(X_{n+1})\big|\Big] \\
&\qquad + \mathbb{E}\Big[\big|V_{n+1}(X_{n+1}) - V_{n+1}(\pi_R(X_{n+1}))\big|\Big]\Big) \\
&\leq \|r\|_\infty\Big([V_{n+1}]_L\mathbb{E}\Big[\big|\pi_R(X_n) - X_n\big|\Big] + \mathbb{E}\Big[\big|\bar{V}_{n+1}^R\big(\pi_R(X_{n+1})\big) - V_{n+1}(X_{n+1})\big|\Big]\Big). \quad (5.A.5)
\end{aligned}$$

Inject (5.A.5) in (5.A.4) to obtain:

$$\begin{aligned}
\mathbb{E}\Big[\big|\bar{V}_n^R(\pi_R(X_n)) - V_n(X_n)\big|\Big] &\leq \Big([V_n]_L + \|r\|_\infty[V_{n+1}]_L\Big)\mathbb{E}\Big[\big|\pi_R(X_n) - X_n\big|\Big] \\
&\quad + \|r\|_\infty\mathbb{E}\Big[\big|\bar{V}_{n+1}^R\big(\pi_R(X_{n+1})\big) - V_{n+1}(X_{n+1})\big|\Big].
\end{aligned}$$

Hence, by induction it holds

$$\begin{aligned}
\mathbb{E}\Big[\big|\bar{V}_n^R(\pi_R(X_n)) - V_n(X_n)\big|\Big] \leq \mathbb{E}\Big[\big|\pi_R(X_1) - X_1\big|\Big]\Bigg(\sum_{k=n}^{N-1}\Big([V_k]_L + \|r\|_\infty\big[V_{k+1}\big]_L\Big)\|r\|_\infty^{k-n} \\
+ \|r\|_\infty^{N-n}[g]_L\Bigg).
\end{aligned}$$

Plugging **(Hloc)** and **(Hloc)** into (5.A.3) yields:

$$\begin{aligned}
\mathbb{E}\Big[\big|\bar{V}_n^R(\pi_R(X_n)) - V_n(X_n)\big|\Big] &\leq \|r\|_\infty\mathbb{E}\Big[\big|\bar{V}_{n+1}^R(\pi_R(X_{n+1})) - V_{n+1}(X_{n+1})\big|\Big] \\
&\quad + [r]_L\|V_{n+1}\|_\infty\mathbb{E}\left[|\pi_R(X_{n+1}) - X_{n+1}|\right] + 2\|V_n\|_\infty\mathbb{P}\left(|X_n| > R\right),
\end{aligned}$$

with X_n and X_{n+1} i.i.d. following the law μ. The result stated in proposition 5.A.1 then follows by induction. □

5.B Forward evaluation of the optimal controls in $\mathcal{A}_M$

We evaluate in this section the real performance of the best controls in $\mathcal{A}_M$.

Let $(a_n^{\mathcal{A}_M})_{n=0}^{N-1}$ be the sequence of optimal controls in the class of neural networks $\mathcal{A}_M$, and denote by $(J_n^{\mathcal{A}_M})_{0\leq n\leq N}$ the cost functional sequence associated to $(a_n^{\mathcal{A}_M})_{n=0}^{N-1}$ and characterized as solution of the Bellman equation:

$$\begin{cases} J_N^{\mathcal{A}_M}(x) = g(x) \\ J_n^{\mathcal{A}_M}(x) = \inf\limits_{A\in\mathcal{A}_M}\Big\{f(x,A(x)) + \mathbb{E}^A_{n,X_n}[J_{n+1}^{\mathcal{A}_M}(X_{n+1})]\Big\}, \end{cases}$$

where $\mathbb{E}^A_{n,X_n}[\cdot]$ stands for the expectation conditioned by X_n and when the control A is applied at time n.

In this section, we are interested in comparing $J_n^{\mathcal{A}_M}$ to V_n. Note that $V_n(x) \leq J_n^{\mathcal{A}_M}(x)$ holds for all $x \in \mathcal{X}$, since $\mathcal{A}_M$ is included in the set of the Borelian functions of $\mathcal{X}$. We can actually show the following:

Proposition 5.B.1. *Assume that there exists a sequence of optimal feedback controls* $(a_n^{\text{opt}})_{0\leq n\leq N-1}$ *for the control problem with value function* V_n, $n=0,\dots,N$. *Then it holds, as* $M\to\infty$*:*

$$\mathbb{E}\left[J_n^{\mathcal{A}_M}(X_n)-V_n(X_n)\right]=\mathcal{O}\left(\sup_{n\leq k\leq N-1}\inf_{A\in\mathcal{A}_M}\mathbb{E}\left[|A(X_k)-a_k^{\text{opt}}(X_k)|\right]\right). \tag{5.B.1}$$

Remark 5.B.1. Notice that there is no estimation error term in (5.B.1), since the optimal strategies in $\mathcal{A}_M$ are defined as those minimizing the real cost functionals in $\mathcal{A}_M$, and not the empirical ones. □

Proof of Proposition 5.B.1. Let $n\in\{0,...,N-1\}$, and $X_n\sim\mu$. Take $A\in\mathcal{A}_M$, and denote $J_n^A(X_n)=f(x,A(x))+\mathbb{E}_{n,X_n}^A[J_{n+1}^{\mathcal{A}_M}(X_{n+1})]$. Clearly, we have $J_n^{\mathcal{A}_M}=\min_{A\in\mathcal{A}_M}J_n^A$. Moreover:

$$\begin{aligned}
\mathbb{E}\Big[J_n^A(X_n)-V_n(X_n)\Big]&\leq\mathbb{E}\Big[|f(X_n,A(X_n))-f(X_n,a_n^{\text{opt}}(X_n))|\Big]\\
&\quad+\mathbb{E}\Big[|J_{n+1}^{\mathcal{A}_M}(F(X_n,A(X_n),\varepsilon_{n+1}))-V_{n+1}(F(X_n,a_n^{\text{opt}}(X_n),\varepsilon_{n+1}))|\Big]\\
&\leq[f]_L\mathbb{E}\Big[|a_n^{\text{opt}}(X_n)-A(X_n)|\Big]\\
&\quad+\mathbb{E}\Big[|V_{n+1}(F(X_n,A(X_n),\varepsilon_{n+1}))-V_{n+1}(F(X_n,a_n^{\text{opt}}(X_n),\varepsilon_{n+1}))|\Big]\\
&\quad+\mathbb{E}\Big[|J_{n+1}^{\mathcal{A}_M}(F(X_n,A(X_n),\varepsilon_{n+1}))-V_{n+1}(F(X_n,A(X_n),\varepsilon_{n+1}))|\Big].
\end{aligned}\tag{5.B.2}$$

Applying assumption **(Hd)** to bound the last term in the r.h.s. of (5.B.2) yields

$$\begin{aligned}
\mathbb{E}\Big[J_n^A(X_n)-V_n(X_n)\Big]&\leq\big([f]_L+\|V_{n+1}\|_\infty[r]_L\big)\mathbb{E}\Big[|a_n^{\text{opt}}(X_n)-A(X_n)|\Big]\\
&\quad+\|r\|_\infty\mathbb{E}\Big[|J_{n+1}^{\mathcal{A}_M}(X_{n+1})-V_{n+1}(X_{n+1})|\Big],
\end{aligned}$$

which holds for all $A\in\mathcal{A}_M$, so that:

$$\begin{aligned}
\mathbb{E}\Big[J_n^{\mathcal{A}_M}(X_n)-V_n(X_n)\Big]&\leq\big([f]_L+\|V_{n+1}\|_\infty[r]_L\big)\inf_{A\in\mathcal{A}_M}\mathbb{E}\Big[|a_n^{\text{opt}}(X_n)-A(X_n)|\Big]\\
&\quad+\|r\|_\infty\mathbb{E}\Big[|J_{n+1}^{\mathcal{A}_M}(X_{n+1})-V_{n+1}(X_{n+1})|\Big].
\end{aligned}$$

(5.B.1) then follows directly by induction. □

5.C Proof of Lemma 5.4.1

The proof is divided into four steps.

Step 1: Symmetrization by a ghost sample. We take $\varepsilon>0$ and show that for $M>2\frac{\left((N-n)\|f\|_\infty+\|g\|_\infty\right)^2}{\varepsilon^2}$, the following holds:

$$\begin{aligned}
&\mathbb{P}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^M\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]-\mathbb{E}\Big[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\Big]\right|>\varepsilon\right]\\
&\leq 2\mathbb{P}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^M\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}-f(X_n^{'(m)},A(X_n^{'(m)}))-\hat{Y}_{n+1}^{'(m),A}\right]\right|>\frac{\varepsilon}{2}\right],
\end{aligned}\tag{5.C.1}$$

where:

- $(X_k^{'(m)})_{1\leq m\leq M,n\leq k\leq N}$ is a copy of $(X_k^{(m)})_{1\leq m\leq M,n\leq k\leq N}$ generated from an independent copy of the exogenous noises $(\varepsilon_k^{'(m)})_{1\leq m\leq M,n\leq k\leq N}$, and independent copy of initial positions at time n, $(X_n^{'(m)})_{m=1}^M$, following the same control $\hat{a}_k^M$ at time $k=n+1,\dots,N-1$, and control A at time n,

- We remind that $Y_{n+1}^{(m),A}$ has already been defined in (5.4.5), and we similarly define

$$Y_{n+1}'^{(m),A} := \sum_{k=n+1}^{N-1} f(X_k'^{(m),A}, \hat{a}_k^M(X_k'^{(m),A})) + g(X_N'^{(m),A}).$$

Let $A^* \in \mathcal{A}_M$ be such that:

$$\left| \frac{1}{M} \sum_{m=1}^{M} \left[f(X_n^{(m)}, A^*(X_n^{(m)})) + \hat{Y}_{n+1}^{(m),A^*} \right] - \mathbb{E}\left[J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n) \right] \right| > \varepsilon$$

if such a function exists, and an arbitrary function in $\mathcal{A}_M$ if such a function does not exist. Note that $\frac{1}{M}\sum_{m=1}^{M} \left[f(X_n^{(m)}, A^*(X_n^{(m)})) + \hat{Y}_{n+1}^{(m),A^*} \right] - \mathbb{E}\left[J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n) \right]$ is a r.v., which implies that A^* also depends on $\omega \in \Omega$. Denote by $\mathbb{P}_M$ the probability conditioned by the training set of exogenous noises $(\varepsilon_k^{(m)})_{1\le m\le M, n\le k\le N}$ and initial positions $(X_k^{(m)})_{1\le m\le M, n\le k\le N}$, and recall that $\mathbb{E}_M$ stands for the expectation conditioned by the latter. Application of Chebyshev's inequality yields

$$\begin{aligned}
&\mathbb{P}_M\left[\left| \mathbb{E}_M\left[J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n') \right] - \frac{1}{M}\sum_{m=1}^{M} \left[f(X_n'^{(m)}, A^*(X_n'^{(m)})) + \hat{Y}_{n+1}'^{(m),A^*} \right] \right| > \frac{\varepsilon}{2} \right] \\
&\quad \le \frac{\mathrm{Var}_M\left[J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n') \right]}{M(\varepsilon/2)^2} \le \frac{\left((N-n)\|f\|_\infty + \|g\|_\infty\right)^2}{M\varepsilon^2},
\end{aligned}$$

where we have used $0 \le \left| J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n') \right| \le (N-n)\|f\|_\infty + \|g\|_\infty$ which implies

$$\begin{aligned}
&\mathrm{Var}_M\left[J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n') \right] = \mathrm{Var}_M\left[J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n') - \frac{(N-n)\|f\|_\infty + \|g\|_\infty}{2} \right] \\
&\le \mathbb{E}\left[\left(J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n') - \frac{(N-n)\|f\|_\infty + \|g\|_\infty}{2} \right)^2 \right] \le \frac{\left((N-n)\|f\|_\infty + \|g\|_\infty\right)^2}{4}.
\end{aligned}$$

Thus, for $M > 2\frac{\left((N-n)\|f\|_\infty + \|g\|_\infty\right)^2}{\varepsilon^2}$, we have

$$\begin{aligned}
&\mathbb{P}_M\left[\left| \mathbb{E}_M\left[J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n) \right] - \frac{1}{M}\sum_{m=1}^{M} \left[f(X_n'^{(m)}, A^*(X_n'^{(m)})) + \hat{Y}_{n+1}'^{(m),A^*} \right] \right| \le \frac{\varepsilon}{2} \right] \\
&\qquad \ge \frac{1}{2}. \qquad (5.C.2)
\end{aligned}$$

Hence:

$$\begin{aligned}
&\mathbb{P}\left[\sup_{A\in\mathcal{A}_M} \left| \frac{1}{M}\sum_{m=1}^{M} \left[f(X_n^{(m)}, A(X_n^{(m)})) + \hat{Y}_{n+1}^{(m),A} \right.\right.\right. \\
&\qquad\qquad \left.\left.\left. - f(X_n'^{(m)}, A(X_n'^{m})) - \hat{Y}_{n+1}'^{(m),A} \right] \right| > \frac{\varepsilon}{2} \right] \\
&\ge \mathbb{P}\left[\left| \frac{1}{M}\sum_{m=1}^{M} \left[f(X_n^{(m)}, A^*(X_n^{(m)})) + \hat{Y}_{n+1}^{(m),A^*} \right.\right.\right. \\
&\qquad\qquad \left.\left.\left. - f(X_n'^{(m)}, A^*(X_n'^{(m)})) - \hat{Y}_{n+1}'^{(m),A^*} \right] \right| > \frac{\varepsilon}{2} \right] \\
&\ge \mathbb{P}\left[\left| \frac{1}{M}\sum_{m=1}^{M} \left[f(X_n^{(m)}, A^*(X_n^{(m)})) + \hat{Y}_{n+1}^{(m),A^*} \right] - \mathbb{E}_M\left[J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n) \right] \right| > \varepsilon, \right. \\
&\qquad \left. \left| \frac{1}{M}\sum_{m=1}^{M} \left[f(X_n'^{(m)}, A^*(X_n'^{(m)})) + \hat{Y}_{n+1}'^{(m),A^*} \right] - \mathbb{E}_M\left[J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n) \right] \right| \le \frac{\varepsilon}{2} \right].
\end{aligned}$$

Observe that $\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A^*(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A^*}\right]-\mathbb{E}_M\left[J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\right]$ is measurable w.r.t. the σ-algebra generated by the training set, so that conditioning by the training set and injecting (5.C.2) yields

$$\begin{aligned}
&2\mathbb{P}\Bigg[\sup_{A\in\mathcal{A}_M}\Bigg|\frac{1}{M}\sum_{m=1}^{M}\Big[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\\
&\qquad\qquad-f(X_n'^{(m)},A(X_n'^{(m)}))-\hat{Y}_{n+1}'^{(m),A}\Big]\Bigg|>\frac{\varepsilon}{2}\Bigg]\\
&\geq\mathbb{P}\left[\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A^*(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A^*}\right]-\mathbb{E}_M\left[J_n^{A^*,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\right]\right|>\varepsilon\right]\\
&=\mathbb{P}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]-\mathbb{E}\left[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\right]\right|>\varepsilon\right]
\end{aligned}$$

for $M>2\frac{\left((N-n)\|f\|_\infty+\|g\|_\infty\right)^2}{\varepsilon^2}$, where we use the definition of A^* to go from the second-to-last to the last line. The proof of (5.C.1) is then completed.

Step 2: We show that

$$\begin{aligned}
&\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]-\mathbb{E}\left[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\right]\right|\right]\\
&\leq 4\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}-f(X_n'^{(m)},A(X_n'^{(m)}))-\hat{Y}_{n+1}'^{(m),A}\right]\right|\right]\\
&\quad+\mathcal{O}\left(\frac{1}{\sqrt{M}}\right).
\end{aligned}\tag{5.C.3}$$

Indeed, let $M'=\sqrt{2}\frac{(N-n)\|f\|_\infty+\|g\|_\infty}{\sqrt{M}}$, and notice

$$\begin{aligned}
&\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]-\mathbb{E}\left[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\right]\right|\right]\\
&=\int_0^\infty\mathbb{P}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]-\mathbb{E}\left[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\right]\right|>\varepsilon\right]\mathrm{d}\varepsilon\\
&=\int_0^{M'}\mathbb{P}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]-\mathbb{E}\left[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\right]\right|>\varepsilon\right]\mathrm{d}\varepsilon\\
&\quad+\int_{M'}^\infty\mathbb{P}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]-\mathbb{E}\left[J_n^{A,(\hat{a}_k^M)_{k=n+1}^{N-1}}(X_n)\right]\right|>\varepsilon\right]\mathrm{d}\varepsilon\\
&\leq\sqrt{2}\frac{(N-n)\|f\|_\infty+\|g\|_\infty}{\sqrt{M}}\\
&\quad+4\int_0^\infty\mathbb{P}\Bigg[\sup_{A\in\mathcal{A}_M}\Bigg|\frac{1}{M}\sum_{m=1}^{M}\Big[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\\
&\qquad\qquad-f(X_n'^{(m)},A(X_n'^{(m)}))-\hat{Y}_{n+1}'^{(m),A}\Big]\Bigg|>\varepsilon\Bigg]\mathrm{d}\varepsilon.
\end{aligned}\tag{5.C.4}$$

The second term in the r.h.s. of (5.C.4) comes from (5.C.1). It remains to write the latter as an expectation to obtain (5.C.3).

Step 3: Introduction of additional randomness by random signs.
Let $(r_m)_{1\le m\le M}$ be i.i.d. Rademacher r.v.[6]. We show that:

$$\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}-f(X_n'^{(m)},A(X_n'^{(m)}))-\hat{Y}_{n+1}'^{(m),A}\right]\right|\right]$$
$$\le 4\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]\right|\right]. \tag{5.C.5}$$

Since for each $m=1,...,M$ the set of exogenous noises $(\varepsilon_k'^{(m)})_{n\le k\le N}$ and $(\varepsilon_k^{(m)})_{n\le k\le N}$ are i.i.d., their joint distribution remain the same if one randomly interchanges the corresponding components. Thus, it holds for $\varepsilon\ge 0$:

$$\begin{aligned}
&\mathbb{P}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}-f(X_n'^{m},A(X_n'^{m}))-\hat{Y}_{n+1}'^{(m),A}\right]\right|>\varepsilon\right]\\
&=\mathbb{P}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\Big[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right.\right.\\
&\qquad\left.\left.-f(X_n'^{m},A(X_n'^{m}))-\hat{Y}_{n+1}'^{(m),A}\Big]\right|>\varepsilon\right]\\
&\le\mathbb{P}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\Big[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\Big]\right|>\frac{\varepsilon}{2}\right]\\
&\quad+\mathbb{P}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\Big[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\Big]\right|>\frac{\varepsilon}{2}\right]\\
&\le 2\mathbb{P}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\Big[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\Big]\right|>\frac{\varepsilon}{2}\right].
\end{aligned}$$

It remains to integrate on $\mathbb{R}_+$ w.r.t. ε to get (5.C.5).

Step 4: We show that

$$\begin{aligned}
&\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\Big[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\Big]\right|\right]\\
&\qquad\le\frac{(N-n)\|f\|_\infty+\|g\|_\infty}{\sqrt{M}}\\
&\qquad\quad+\left([f]_L+[f]_L\sum_{k=n+1}^{N-1}\big(1+\eta_M\gamma_M\big)^{k-n}[F]_L^{k-n}+\eta_M^{N-n}\gamma_M^{N-n}[F]_L^{N-n}[g]_L\right)\\
&\qquad=\mathcal{O}\left(\frac{\gamma_M^{N-n}\eta_M^{N-n}}{\sqrt{M}}\right),\qquad\text{as } M\to\infty.
\end{aligned}\tag{5.C.6}$$

Adding and removing the cost obtained by control 0 at time n yields:

$$\begin{aligned}
&\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\Big(f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\Big)\right|\right]\\
&\le\mathbb{E}\left[\left|\frac{1}{M}\sum_{m=1}^{M}r_m\Big(f(X_n^{(m)},0)+\hat{Y}_{n+1}^{(m),0}\Big)\right|\right]\\
&\quad+\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\Big(f(X_n^{(m)},A(X_n^{(m)}))-f(X_n^{(m)},0)+\hat{Y}_{n+1}^{(m),A}-\hat{Y}_{n+1}^{(m),0}\Big)\right|\right].
\end{aligned}\tag{5.C.7}$$

[6]The probability mass function of a Rademacher r.v. is by definition $\frac{1}{2}\delta_{-1}+\frac{1}{2}\delta_1$.

We now bound the first term of the r.h.s. of (5.C.7). By Cauchy-Schwartz inequality, and recalling that $(r_m)_{1\leq m\leq M}$ are i.i.d. with zero mean such that $r_m^2 = 1$, we get

$$\begin{aligned}\mathbb{E}\left[\left|\frac{1}{M}\sum_{m=1}^{M} r_m\Big(f(X_n^{(m)},0)+\hat{Y}_{n+1}^{(m),0}\Big)\right|\right] &\leq \frac{1}{M}\sqrt{\mathbb{E}\left[\left|\sum_{m=1}^{M} r_m\Big(f(X_n^{(m)},0)+\hat{Y}_{n+1}^{(m),0}\Big)\right|^2\right]}\\ &\leq \frac{1}{\sqrt{M}}\big((N-n)\|f\|_\infty+\|g\|_\infty\big)\end{aligned}$$

Turn now to the second term of (5.C.7). By the Lipschitz continuity of f, it stands:

$$\begin{aligned}&\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m\Big(f(X_n^{(m)},A(X_n^{(m)}))-f(X_n^{(m)},0)+\hat{Y}_{n+1}^{(m),A}-\hat{Y}_{n+1}^{(m),0}\Big)\right|\right]\\ &\leq [f]_L\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m A(X_n^{(m)})\right|\right]+\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m\Big(\hat{Y}_{n+1}^{(m),A}-\hat{Y}_{n+1}^{(m),0}\Big)\right|\right]\\ &\leq \left([f]_L+[f]_L\sum_{k=n+1}^{N-1}\big(1+\eta_M\gamma_M\big)^{k-n}\mathbb{E}\left[\sup_{1\leq m\leq M}\prod_{j=n+1}^{k} C\left(\varepsilon_j^m\right)\right]\right.\\ &\qquad\left.+[g]_L\left(1+\eta_M^{N-n}\gamma_M^{N-n}\right)\mathbb{E}\left[\sup_{1\leq m\leq M}\prod_{j=n+1}^{N} C\left(\varepsilon_j^m\right)\right]\right)\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m A(X_n^{(m)})\right|\right]\end{aligned}$$

where we condition by the exogenous noise, use assumption **(HF-PI)** and the $\eta_M\gamma_M$-Lipschitz continuity of the estimated optimal controls at time k, for $k = n+1,\dots,N-1$. Now, notice first that

$$\mathbb{E}\left[\sup_{1\leq m\leq M}\prod_{k=n+1}^{N} C\left(\varepsilon_k^m\right)\right] \leq \prod_{k=n+1}^{N}\mathbb{E}\left[\sup_{1\leq m\leq M} C\left(\varepsilon_k^m\right)\right] \leq \rho_M^{N-n},$$

and moreover:

$$\begin{aligned}\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m A(X_n^{(m)})\right|\right] &\leq \gamma_M\mathbb{E}\left[\sup_{|v|_2\leq 1/R}\left|\sum_{m=1}^{M} r_m (v^T X_n^{(m)})_+\right|\right]\\ &\leq \gamma_M\mathbb{E}\left[\sup_{|v|_2\leq 1/R}\left|\sum_{m=1}^{M} r_m v^T X_n^{(m)}\right|\right],\end{aligned}$$

where $R>0$ is a bound for the state space (see e.g. the discussion on the Frank-Wolfe step p.10 of [Bac17] for a proof of this inequality), which implies by Cauchy-Schwarz inequality:

$$\begin{aligned}\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m A(X_n^{(m)})\right|\right] &\leq \frac{\gamma_M}{R}\sqrt{\mathbb{E}\left[\left|\sum_{m=1}^{M} r_m X_n^{(m)}\right|^2\right]}\\ &\leq \gamma_M\sqrt{M}\end{aligned}$$

since the $(r_m)_m$ are i.i.d. Rademacher r.v. Plug first (5.C.9) and (5.C.10) into (5.C) to obtain

$$\begin{aligned}&\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m\Big(f(X_n^{(m)},A(X_n^{(m)}))-f(X_n^{(m)},0)+\hat{Y}_{n+1}^{(m),A}-\hat{Y}_{n+1}^{(m),0}\Big)\right|\right]\\ &\leq \left([f]_L+[f]_L\sum_{k=n+1}^{N-1}\big(1+\eta_M\gamma_M\big)^{k-n}\rho_M^{k-n}+[g]_L\left(1+\eta_M^{N-n}\gamma_M^{N-n}\right)\rho_M^{N-n}\right)\gamma_M\sqrt{M}.\end{aligned} \tag{5.C.11}$$

Plug then (5.C.8) and (5.C.11) into (5.C.7) to get (5.C.6).

Step 5: Conclusion

Plug (5.C.6) into (5.C.5) and combine it with (5.C.3) to obtain the bound on the estimation error, as stated in (5.4.9) of Lemma 5.4.1. □

5.D Proof of Lemma 5.4.2

Let $(\hat{a}_k^M)_{k=n+1}^{N-1}$ be the sequence of estimated controls at time $k = n+1, ..., N-1$. Take $A \in \mathcal{A}_M$ and remind that we denote by $J_n^{A,(\hat{a})_{k=n+1}^{N-1}}$ the cost functional associated to the control A at time n, and $\hat{a}_k^M$ at time $k = n+1, \dots, N-1$. The latter is characterized as solution of the Bellman equation

$$\begin{cases} J_N^{A,(\hat{a})_{k=n+1}^{N-1}}(x) = g(x) \\ J_n^{A,(\hat{a})_{k=n+1}^{N-1}}(x) = f(x, A(x)) + \mathbb{E}_{n,x}^A \left[J_{n+1}^{A,(\hat{a})_{k=n+1}^{N-1}}(X_{n+1}) \right], \end{cases}$$

where $\mathbb{E}_{n,x}^A[\cdot]$ stands for the expectation conditioned by $\{X_n = x\}$ when feedback control A is followed at time n.

Take $n \in \{1, ..., N\}$. It holds:

$$\begin{aligned} \varepsilon_{\mathrm{PI},n}^{\mathrm{approx}} &:= \inf_{A \in \mathcal{A}_M} \mathbb{E}_M \left[J_n^{A,(\hat{a})_{k=n+1}^{N-1}}(X_n) \right] - \inf_{A \in \mathbb{A}^{\mathcal{X}}} \mathbb{E}_M \left[J_n^{A,(\hat{a})_{k=n+1}^{N-1}}(X_n) \right] \\ &= \inf_{A \in \mathcal{A}_M} \mathbb{E}_M \left[J_n^{A,(\hat{a})_{k=n+1}^{N-1}}(X_n) \right] - \mathbb{E}\left[V_n(X_n) \right] \\ &\quad + \mathbb{E}\left[V_n(X_n) \right] - \inf_{A \in \mathbb{A}^{\mathcal{X}}} \mathbb{E}_M \left[J_n^{A,(\hat{a})_{k=n+1}^{N-1}}(X_n) \right] \\ &\leq \inf_{A \in \mathcal{A}_M} \mathbb{E}_M \left[J_n^{A,(\hat{a})_{k=n+1}^{N-1}}(X_n) \right] - \mathbb{E}\left[V_n(X_n) \right], \end{aligned} \tag{5.D.1}$$

where the last inequality stands because the value function is smaller than the cost functional associated to any other strategy. We then apply the dynamic programming principle to obtain:

$$\begin{aligned} &\min_{A \in \mathcal{A}_M} \mathbb{E}_M \left[J_n^{A,(\hat{a})_{k=n+1}^{N-1}}(X_n) \right] - \mathbb{E}\left[V_n(X_n) \right] \\ &\qquad \leq \inf_{A \in \mathcal{A}_M} \mathbb{E}_M \left[f\big(X_n, A(X_n)\big) + \mathbb{E}_n^A \left[J_{n+1}^{(\hat{a})_{k=n+1}^{N-1}}(X_{n+1}) \right] \right] \\ &\qquad\quad - \mathbb{E}\left[f\big(X_n, a_n^{\mathrm{opt}}(X_n)\big) + \mathbb{E}_n^{a^{\mathrm{opt}}} \left[V_{n+1}(X_{n+1}) \right] \right]. \end{aligned} \tag{5.D.2}$$

To bound the r.h.s. of (5.D.2), first observe that for $A \in \mathcal{A}_M$:

$$\begin{aligned} &\mathbb{E}_M \left[f\big(X_n, A(X_n)\big) + \mathbb{E}_n^A \left[J_{n+1}^{(\hat{a})_{k=n+1}^{N-1}}(X_{n+1}) \right] \right] \\ &\qquad\qquad - \mathbb{E}\Big[f\big(X_n, a_n^{\mathrm{opt}}(X_n)\big) + \mathbb{E}_n^{a^{\mathrm{opt}}} \left[V_{n+1}(X_{n+1}) \right] \Big] \\ &\leq \mathbb{E}\left[\left| f\big(X_n, A(X_n)\big) - f\big(X_n, a_n^{\mathrm{opt}}(X_n)\big) \right| \right] \\ &\qquad\qquad + \mathbb{E}_M \Big[\mathbb{E}_n^A J_{n+1}^{(\hat{a})_{k=n+1}^{N-1}}(X_{n+1}) - \mathbb{E}_n^{a^{\mathrm{opt}}} V_{n+1}(X_{n+1}) \Big] \\ &\leq \big([f]_L + \|V_{n+1}\|_\infty [r]_L \big) \mathbb{E}\left[\left| A(X_n) - a_n^{\mathrm{opt}}(X_n) \right| \right] \\ &\qquad\qquad + \|r\|_\infty \mathbb{E}_M \Big[J_{n+1}^{(\hat{a})_{k=n+1}^{N-1}}(X_{n+1}) - V_{n+1}(X_{n+1}) \Big], \end{aligned} \tag{5.D.3}$$

where we used twice assumption **(Hd)** at the second-last line of (5.D.3). Inject inequality

$$\mathbb{E}_M \Big[J_{n+1}^{(\hat{a})_{k=n+1}^{N-1}}(X_{n+1}) \Big] \leq \inf_{A \in \mathcal{A}_M} \mathbb{E}_M \Big[J_{n+1}^{A,(\hat{a})_{k=n+2}^{N-1}}(X_{n+1}) \Big] + 2\varepsilon_{n+1}^{esti}$$

into (5.D.3) to obtain:

$$\begin{aligned}\mathbb{E}_M\left[f\big(X_n,A(X_n)\big)+\mathbb{E}_n^A\left[J_{n+1}^{(\hat{a})_{k=n+1}^{N-1}}(X_{n+1})\right]\right]&\\ -\mathbb{E}\Big[f\big(X_n,a_n^{\rm opt}(X_n)\big)+\mathbb{E}_n^{a^{\rm opt}}\left[V_{n+1}(X_{n+1})\right]\Big]&\\ \leq\big([f]_L+\|V_{n+1}\|_\infty[r]_L\big)\mathbb{E}\left[|A(X_n)-a_n^{\rm opt}(X_n)|\right]&\\ +\|r\|_\infty\inf_{A\in\mathcal{A}_M}\mathbb{E}_M\left[J_{n+1}^{A,(\hat{a})_{k=n+2}^{N-1}}(X_{n+1})-V_{n+1}(X_{n+1})\right]+2\|r\|_\infty\varepsilon_{n+1}^{esti}.&\qquad(5.D.4)\end{aligned}$$

Plugging (5.D.4) into (5.D.2) yields

$$\begin{aligned}\inf_{A\in\mathcal{A}_M}\mathbb{E}_M\left[J_n^{A,(\hat{a})_{k=n+1}^{N-1}}(X_n)\right]-\mathbb{E}\left[V_n(X_n)\right]&\\ \leq\|r\|_\infty\inf_{A\in\mathcal{A}_M}\mathbb{E}_M\left[J_{n+1}^{A,(\hat{a})_{k=n+2}^{N-1}}(X_{n+1})-V_{n+1}(X_{n+1})\right]+2\|r\|_\infty\varepsilon_{n+1}^{esti}&\\ +\big([f]_L+\|V_{n+1}\|_\infty[r]_L\big)\inf_{A\in\mathcal{A}_M}\mathbb{E}\left[|A(X_n)-a_n^{\rm opt}(X_n)|\right],&\end{aligned}$$

which implies by induction, as $M\to+\infty$:

$$\begin{aligned}\mathbb{E}\left[\inf_{A\in\mathcal{A}_M}\mathbb{E}_M\left[J_n^{A,(\hat{a})_{k=n+1}^{N-1}}(X_n)\right]-\mathbb{E}\left[V_n(X_n)\right]\right]&\\ =\mathcal{O}\Big(\sup_{n+1\leq k\leq N-1}\mathbb{E}\left[\varepsilon_k^{esti}\right]+\sup_{n\leq k\leq N-1}\inf_{A\in\mathcal{A}_M}\mathbb{E}\left[|A(X_n)-a_n^{\rm opt}(X_n)|\right]\Big).&\end{aligned}$$

We now use Lemma 5.4.1 to bound the expectations of the $\varepsilon_{{\rm PI},k}^{\rm esti}$ for $k=n+1,\dots,N-1$, and plug the result into (5.D.1) to complete the proof of Lemma 5.4.2. □

5.E Function approximation by neural networks

We assume $a_n^{\rm opt}(X_n)\in\mathbb{L}^2(\mu)$, and show the relation (5.4.7) in Proposition 5.4.1.

The universal approximator theorem applies for

$$\mathcal{A}_\infty:=\bigcup_{M=1}^{\infty}\mathcal{A}_M,$$

and states that for all $\varepsilon>0$, there exists a neural network a^* in $\mathcal{A}_\infty$ such that:

$$\sup_{n\leq k\leq N-1}\|a_k^{\rm opt}-a^*\|_\infty<\frac{\varepsilon}{\mathcal{V}_d(\mathcal{X})},$$

where $\mathcal{V}_d(\mathcal{X})$ stands for the volume of compact set $\mathcal{X}$ seen as a compact of the euclidean space $\mathbb{R}^d$. By integrating, we then get:

$$\sup_{n\leq k\leq N-1}\int_{\mathcal{X}}\big|a_k^{\rm opt}(x)-a^*(x)\big|d\mu(x)<\varepsilon,$$

Also, notice that $\big(\mathcal{A}_M\big)_{M\geq1}$ is increasing, which implies that $\mathcal{A}_\infty=\lim_{M\to+\infty}\mathcal{A}_M$, and gives the existence of $M>0$, that depends on ε, such that $a^*\in\mathcal{A}_M$.

Therefore, we have shown that for $n=0,...,N-1$

$$\sup_{n\leq k\leq N-1}\inf_{A\in\mathcal{A}_M}\mathbb{E}\big[|A(X_k)-a_k^{\rm opt}(X_k)|\big]\xrightarrow[M\to\infty]{}0,\qquad\text{with } X_k\sim\mu,$$

which is the required result stated in (5.4.7). □

We now show (5.4.8) of proposition 5.4.1:
As stated in section 4.7 of [Bac17]: proposition 6 in [Bac17] shows that we can approximate a c-Lipschitz function by a γ_1-norm less than γ_M and uniform error less than $c\left(\frac{\gamma_M}{c}\right)^{-2d/(d+1)}\log\frac{\gamma_M}{c}$, and proposition 1 in [Bac17] shows that a function with γ_1 less than γ_M may be approximated with K_M neurons with uniform error $\gamma_M K_M^{-(d+3)/(2d)}$.
Thus, given K_M and γ_M, there exists a neural network a^* in $\mathcal{V}_M$ such that

$$\|a^* - a^{\text{opt}}\|_\infty \leq c\left(\frac{\gamma_M}{c}\right)^{-2d/(d+1)}\log\left(\frac{\gamma_M}{c}\right) + \gamma_M K_M^{-(d+3)/(2d)}.$$

□

5.F Proof of Lemma 5.4.3

We prove Lemma 5.4.3 in four steps. Since the proof is very similar to the one of Lemma 5.4.1, we only detail the arguments that are modified.

Step 1: Symmetrization by a ghost sample. We take $\varepsilon > 0$ and show that for $M > 2\frac{\left((N-n)\|f\|_\infty+\|g\|_\infty\right)^2}{\varepsilon^2}$, it holds

$$\begin{aligned}
&\mathbb{P}\Bigg[\sup_{A\in\mathcal{A}_M}\Bigg|\frac{1}{M}\sum_{m=1}^{M}\Big[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\Big] \\
&\qquad\qquad -\mathbb{E}\Big[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\Big]\Bigg| > \varepsilon\Bigg] \\
&\leq 2\mathbb{P}\Bigg[\sup_{A\in\mathcal{A}_M}\Bigg|\frac{1}{M}\sum_{m=1}^{M}\Big[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A} \\
&\qquad\qquad -f(X_n'^{(m)},A(X_n'^{(m)}))-\hat{Y}_{n+1}'^{(m),A}\Big]\Bigg| > \frac{\varepsilon}{2}\Bigg],
\end{aligned}$$

where:

- $\left(X_n'^{(m)}\right)_{m=1}^M$ is a i.i.d. copy of $\left(X_n^{(m)}\right)_{m=1}^M$,
- $\left(\varepsilon_{n+1}'^{m}\right)_{m=1}^M$ is a i.i.d. copy of $\left(\varepsilon_{n+1}^{m}\right)_{m=1}^M$,
- we define

$$\hat{Y}_{n+1}^{(m),A} := \hat{V}_{n+1}^M\left(F\left(X_n^{(m)},A\left(X_n^{(m)}\right),\varepsilon_{n+1}^m\right)\right),$$

and

$$\hat{Y}_{n+1}'^{(m),A} := \hat{V}_{n+1}^M\left(F\left(X_n'^{(m)},A\left(X_n'^{(m)}\right),\varepsilon_{n+1}'^m\right)\right).$$

Proof: Since $\hat{V}_n^M$ the estimated value function at time n, for n=0, ..., $N-1$, is bounded by construction (we truncated the estimation at the last step of the pseudo-code of the Hybrid algorithm), the proof is the same as the one in step 1 of Lemma 5.4.1. □

Step 2: The following result holds

$$\begin{aligned}
&\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]-\mathbb{E}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]\right|\right]\\
&\leq 4\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}-f(X_n'^{(m)},A(X_n'^{(m)}))-\hat{Y}_{n+1}'^{(m),A}\right]\right|\right]\\
&\quad+\mathcal{O}\left(\frac{1}{\sqrt{M}}\right).
\end{aligned}\tag{5.F.1}$$

Proof: same as step 2 in the proof of Lemma 5.4.1. □

Step 3: Introduction of additional randomness by random signs.
The following result holds:

$$\begin{aligned}
&\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}-f(X_n'^{(m)},A(X_n'^{(m)}))-\hat{Y}_{n+1}'^{(m),A}\right]\right|\right]\\
&\quad\leq 4\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]\right|\right].
\end{aligned}\tag{5.F.2}$$

Proof: same as step 3 in the proof of Lemma 5.4.1. □

Step 4: We show that

$$\begin{aligned}
\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\left[f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right]\right|\right]&\leq\frac{(N-n)\|f\|_\infty+\|g\|_\infty}{\sqrt{M}}\\
&\quad+\left([f]_L+\rho_M\gamma_M\eta_M\right)\frac{\gamma_M}{\sqrt{M}}\\
&=\mathcal{O}\left(\frac{\rho_M\gamma_M^2\eta_M}{\sqrt{M}}\right),\quad\text{as } M\to+\infty.
\end{aligned}\tag{5.F.3}$$

Adding and removing the cost obtained by control 0 at time n yields:

$$\begin{aligned}
&\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\left(f(X_n^{(m)},A(X_n^{(m)}))+\hat{Y}_{n+1}^{(m),A}\right)\right|\right]\\
&\leq\mathbb{E}\left[\left|\frac{1}{M}\sum_{m=1}^{M}r_m\left(f(X_n^{(m)},0)+\hat{Y}_{n+1}^{(m),0}\right)\right|\right]\\
&\quad+\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\frac{1}{M}\sum_{m=1}^{M}r_m\left(f(X_n^{(m)},A(X_n^{(m)}))-f(X_n^{(m)},0)+\hat{Y}_{n+1}^{(m),A}-\hat{Y}_{n+1}^{(m),0}\right)\right|\right].
\end{aligned}\tag{5.F.4}$$

The first term in the r.h.s. in (5.F.4) is bounded as in the proof of Lemma 5.4.1 by

$$\frac{(N-n)\|f\|_\infty+\|g\|_\infty}{\sqrt{M}}.$$

We use the Lipschitz-continuity of f as follows, to bound its second term:

$$\begin{aligned}
&\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m\Big(f(X_n^{(m)},A(X_n^{(m)}))-f(X_n^{(m)},0)+\hat{Y}_{n+1}^{(m),A}-\hat{Y}_{n+1}^{(m),0}\Big)\right|\right]\\
&\leq [f]_L\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m A(X_n^{(m)})\right|\right]+\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m\Big(\hat{Y}_{n+1}^{(m),A}-\hat{Y}_{n+1}^{(m),0}\Big)\right|\right],\\
&\leq ([f]_L+\rho_M\eta_M\gamma_M)\,\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m A(X_n^{(m)})\right|\right]
\end{aligned}$$

where we condition by the exogenous noise, use assumption **(HF)**, and the $\eta_M\gamma_M$-Lipschitz continuity of the estimated value fonction at time $n+1$.

By using the same arguments as those presented in the proof of Lemma 5.4.1, we can first bound $\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m A(X_n^{(m)})\right|\right]$ as follows:

$$\mathbb{E}\left[\sup_{A\in\mathcal{A}_M}\left|\sum_{m=1}^{M} r_m A(X_n^{(m)})\right|\right]\leq \gamma_M\sqrt{M},$$

and then conclude that (5.F.3) holds.

Step 5: Conclusion
Combining(5.F.1),(5.F.2) and (5.F.3) results in the bound on the estimation error as stated in (5.4.20). □

5.G Proof of Lemma 5.4.4

We divide the proof of Lemma 5.4.4 into two steps.

First write

$$\begin{aligned}
\varepsilon_{\mathrm{HN},n}^{\mathrm{approx}} \leq \inf_{A\in\mathcal{A}_M}\mathbb{E}_M\left[f\left(X_n,A(X_n)\right)+\hat{V}_{n+1}^{M}\left(X_{n+1}^{A}\right)\right]-\mathbb{E}\left[V_n(X_n)\right]\\
+\mathbb{E}\left[V_n(X_n)\right]-\inf_{A\in\mathbb{A}^{\mathcal{X}}}\mathbb{E}_M\left[f\left(X_n,A(X_n)\right)+\hat{V}_{n+1}^{M}\left(X_{n+1}^{A}\right)\right].
\end{aligned}\tag{5.G.1}$$

Step 1: We show

$$\begin{aligned}
&\inf_{A\in\mathcal{A}_M}\mathbb{E}_M\left[f\left(X_n,A(X_n)\right)+\hat{V}_{n+1}^{M}\left(X_{n+1}^{A}\right)\right]-\mathbb{E}\left[V_n(X_n)\right]\\
&\qquad\leq ([f]_L+\|V_{n+1}\|_\infty[r]_L)\inf_{A\in\mathbb{A}^{\mathcal{X}}}\mathbb{E}_M\left[\left|A(X_n)-a_n^{\mathrm{opt}}(X_n)\right|\right]\\
&\qquad\quad+\|r\|_\infty\mathbb{E}_M\left[\left|V_{n+1}(X_{n+1})-\hat{V}_{n+1}^{M}(X_{n+1})\right|\right].
\end{aligned}\tag{5.G.2}$$

Take $A\in\mathcal{A}_M$, and apply the dynamic programming principle to write

$$\begin{aligned}
&\mathbb{E}_M\left[f\left(X_n,A(X_n)\right)+\hat{V}_{n+1}^{M}\left(X_{n+1}^{A}\right)\right]-\mathbb{E}\left[V_n(X_n)\right]\\
&\leq [f]_L\mathbb{E}_M\left[\left|A(X_n)-a_n^{\mathrm{opt}}(X_n)\right|\right]+\mathbb{E}_M\left[\mathbb{E}_M\left[\hat{V}_{n+1}^{M}(X_{n+1}^{A})\right]-\mathbb{E}_M\left[V_{n+1}\left(X_{n+1}^{a_n^{\mathrm{opt}}}\right)\right]\right]\\
&\leq ([f]_L+\|V_{n+1}\|_\infty[r]_L)\,\mathbb{E}_M\left[\left|A(X_n)-a_n^{\mathrm{opt}}(X_n)\right|\right]+\mathbb{E}_M\left[\left|\hat{V}_{n+1}^{M}(X_{n+1}^{A})-V_{n+1}(X_{n+1}^{A})\right|\right],
\end{aligned}$$

where we used **(Hd)** at the second-to-last line. By using one more time assumption **(Hd)**, we then get:

$$\begin{aligned}\mathbb{E}_M\left[f\left(X_n, A(X_n)\right) + \hat{V}_{n+1}^M\left(X_{n+1}^A\right)\right] - \mathbb{E}\left[V_n(X_n)\right] \\ \leq \left([f]_L + \|V_{n+1}\|_\infty [r]_L\right) \mathbb{E}_M\left[\left|A(X_n) - a_n^{\text{opt}}(X_n)\right|\right] \\ + \|r\|_\infty \mathbb{E}_M\left[\left|\hat{V}_{n+1}^M(X_{n+1}) - V_{n+1}(X_{n+1})\right|\right], \quad \text{with } X_{n+1} \sim \mu,\end{aligned}$$

which is the result stated in (5.G.2).

Step 2: We show

$$\begin{aligned}\mathbb{E}\left[V_n(X_n)\right] - \inf_{A \in \mathbb{A}^{\mathcal{X}}} \mathbb{E}_M\left[f\left(X_n, A(X_n)\right) + \hat{V}_{n+1}^M\left(X_{n+1}^A\right)\right] \\ \leq \|r\|_\infty \mathbb{E}_M\left[\left|V_{n+1}(X_{n+1}) - \hat{V}_{n+1}^M(X_{n+1})\right|\right].\end{aligned} \tag{5.G.3}$$

Write

$$\begin{aligned}&\mathbb{E}\left[V_n(X_n)\right] - \inf_{A \in \mathbb{A}^{\mathcal{X}}} \mathbb{E}_M\left[f\left(X_n, A(X_n)\right) + \hat{V}_{n+1}^M\left(X_{n+1}^A\right)\right] \\ &\leq \inf_{A \in \mathbb{A}^{\mathcal{X}}} \mathbb{E}_M\left[f\left(X_n, A(X_n)\right) + V_{n+1}\left(X_{n+1}^A\right)\right] - \inf_{A \in \mathbb{A}^{\mathcal{X}}} \mathbb{E}_M\left[f\left(X_n, A(X_n)\right) + \hat{V}_{n+1}^M\left(X_{n+1}^A\right)\right] \\ &\leq \inf_{A \in \mathbb{A}^{\mathcal{X}}} \mathbb{E}_M\left[V_{n+1}\left(X_{n+1}^A\right) - \hat{V}_{n+1}^M\left(X_{n+1}^A\right)\right] \leq \|r\|_\infty \mathbb{E}_M\left[\left|V_{n+1}(X_{n+1}) - \hat{V}_{n+1}^M(X_{n+1})\right|\right]\end{aligned}$$

which completes the proof of (5.G.3).

Step 3 Conclusion:
We complete the proof of Lemma 5.4.4 by plugging (5.G.2) and (5.G.3) into (5.G.1). □

5.H Some useful Lemmas for the proof of Theorem 5.4.2

Fix $M \in \mathbb{N}^*$, let $x_1, \ldots, x_M \in \mathbb{R}^d$, and set $x^M = (x_1, \ldots, x_M)$. Define the distance $d_2(f, g)$ between $f : \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^d \to \mathbb{R}$ by

$$d_2(f, g) = \left(\frac{1}{M} \sum_{m=1}^{M} |f(x_m) - g(x_m)|^2\right)^{1/2}.$$

An ε-cover of $\mathcal{V}$ (w.r.t. the distance d_2) is a set of functions $f_1, \ldots, f_P : \mathbb{R}^d \to \mathbb{R}$ such that

$$\min_{p=1,\ldots,P} d_2\left(f, f_p\right) < \varepsilon, \quad \text{for } f \in \mathcal{V}.$$

Let $\mathcal{N}_2(\varepsilon, \mathcal{V}, x^M)$ denote the size of the smallest ε-cover of $\mathcal{V}$ w.r.t. the distance d_2, and set by convention $\mathcal{N}_2(\varepsilon, \mathcal{V}, x^M) = +\infty$ if there does not exist any ε-cover of $\mathcal{V}$ of finite size. $\mathcal{N}_2(\varepsilon, \mathcal{V}, x^M)$ is called ^{2}-ε-covering number of $\mathcal{V}$ on x^M.

Lemma 5.H.1. *Let (X, Y) be a random variable. Assume $|Y| \leq L$ a.s. and let*

$$m(x) = \mathbb{E}[Y|X = x].$$

Assume $Y - m(X)$ is sub-Gaussian in the sense that

$$\max_{m=1,\ldots,M} c^2 \mathbb{E}\left[e^{(Y-m(X))^2/c^2} - 1|X\right] \leq \sigma^2 \quad \text{a.s.}$$

for some $c, \sigma > 0$. Let $\gamma_M, L \geq 1$ and assume that the regression function is bounded by L and that $\gamma_M \xrightarrow[M\to+\infty]{} +\infty$. Set

$$\hat{m}_M = \operatorname*{argmin}_{\Phi \in \mathcal{V}_M} \frac{1}{M} \sum_{m=1}^{M} \left|\Phi(x_i) - \bar{Y}_m\right|^2$$

for some $\mathcal{V}_M$ of functions $\Phi : \mathbb{R}^d \to [-\gamma_M, \gamma_M]$ and some random variables $\bar{Y}_1, ..., \bar{Y}_M$ which are bounded by L, and denote

$$\Omega_g := \left\{ f - g : f \in \mathcal{V}_M, \frac{1}{M} \sum_{m=1}^{M} \left|f(x_m) - g(x_m)\right|^2 \leq \frac{\delta}{\gamma_M^2} \right\}.$$

Then there exists constants $c_1, c_2 > 0$ which depend only on σ and c such that for any $\delta_M > 0$ with

$$\delta_M \xrightarrow[M\to+\infty]{} 0, \quad \text{and} \quad \frac{M\delta_M}{\gamma_M} \xrightarrow[M\to+\infty]{} +\infty$$

and

$$c_1 \frac{\sqrt{M}\delta}{\gamma_M^2} \geq \int_{c_2\delta/\gamma_M^2}^{\sqrt{\delta}} \log\left(\mathcal{N}_2\left(\frac{u}{4\gamma_M}, \Omega_g, x_1^M \right) \right)^{1/2} \mathrm{d}u \tag{5.H.1}$$

for all $\delta \geq \delta_M$ and all $g \in \mathcal{V}_M \cup \{m\}$ we have as $M \to +\infty$:

$$\mathbb{E}\left[\left|\bar{m}_M(X) - m(X)\right|^2 \right] = \mathcal{O}_{\mathbb{P}}\left(\frac{1}{M} \sum_{m=1}^{M} \left|Y_m - \bar{Y}_m\right|^2 + \delta_M + \inf_{\Phi \in \mathcal{V}_M} \mathbb{E}\left[\left|\Phi(X) - m(X)\right|^2 \right] \right).$$

Proof. See [Koh06]. □

Lemma 5.H.2. *Let $\mathcal{V}_M$ be defined as in Section 5.4.2. For any $\varepsilon > 0$, we have*

$$\mathcal{N}_2\left(\varepsilon, \mathcal{V}_M, (X_n^{(m)})_{1\leq m\leq M}\right) \leq \left(\frac{12e\gamma_M\left(K_M + 1\right)}{\varepsilon} \right)^{(4d+9)K_M+1}.$$

Proof. See Lemma 5.1 in [KKT10]. □

Chapter 6

Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications[1]

Abstract This paper presents several numerical applications of deep learning-based algorithms that have been introduced in [HPBL18]. Numerical and comparative tests using TensorFlow illustrate the performance of our different algorithms, namely control learning by performance iteration (algorithms NNcontPI and ClassifPI), control learning by hybrid iteration (algorithms Hybrid-Now and Hybrid-LaterQ), on the 100-dimensional nonlinear PDEs examples from [EHJ17] and on quadratic backward stochastic differential equations as in [CR16]. We also performed tests on low-dimension control problems such as an option hedging problem in finance, as well as energy storage problems arising in the valuation of gas storage and in microgrid management. Numerical results and comparisons to quantization-type algorithms Qknn, as an efficient algorithm to numerically solve low-dimensional control problems, are also provided; and the corresponding codes are available on https://github.com/comeh/.

Key words: Deep learning, algorithms, performance iteration, value iterations, Monte-Carlo, quantization.

Contents

[1] This Chapter is based on a paper written in collaboration with Huyên Pham, Achref Bachouch and Nicolas Langrené.

6.1 Introduction

This paper is devoted to the numerical resolution of discrete-time stochastic control problem over a finite horizon. The dynamics of the controlled state process $X = (X_n)_n$ valued in $\mathbb{R}^d$ is given by

$$X_{n+1} = F(X_n, \alpha_n, \varepsilon_{n+1}), \quad n = 0, \ldots, N-1, \; X_0 = x_0 \in \mathbb{R}^d,$$

where $(\varepsilon_n)_n$ is a sequence of i.i.d. random variables valued in some Borel space $(E, \mathcal{B}(E))$, and defined on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$ equipped with the filtration $\mathbb{F} = (\mathcal{F}_n)_n$ generated by the noise $(\varepsilon_n)_n$ ($\mathcal{F}_0$ is the trivial σ-algebra), the control $\alpha = (\alpha_n)_n$ is an $\mathbb{F}$-adapted process valued in $\mathbb{A} \subset \mathbb{R}^q$, and F is a measurable function from $\mathbb{R}^d \times \mathbb{R}^q \times E$ into $\mathbb{R}^d$. Given a running cost function f defined on $\mathbb{R}^d \times \mathbb{R}^q$ and a terminal cost function g defined on $\mathbb{R}^d$, the cost functional associated with a control process α is

$$J(\alpha) = \mathbb{E}\left[\sum_{n=0}^{N-1} f(X_n, \alpha_n) + g(X_N)\right].$$

The set $\mathcal{A}$ of admissible controls is the set of control processes α satisfying some integrability conditions ensuring that the cost functional $J(\alpha)$ is well-defined and finite. The control problem, also called Markov decision process (MDP), is formulated as

$$V_0(x_0) := \inf_{\alpha \in \mathcal{A}} J(\alpha),$$

and the goal is to find an optimal control $\alpha^* \in \mathcal{A}$, i.e., attaining the optimal value: $V_0(x_0) = J(\alpha^*)$. Notice that problem (6.1.1)-(6.1.3) may also be viewed as the time discretization of a continuous time stochastic control problem, in which case, F is typically the Euler scheme for a controlled diffusion process.

It is well-known that the global dynamic optimization problem (6.1.3) can be reduced to local optimization problems via the dynamic programming (DP) approach, which allows to determine the value function in a backward recursion by

$$\begin{aligned} V_N(x) &= g(x), \quad x \in \mathbb{R}^d, \\ V_n(x) &= \inf_{a \in \mathbb{A}} Q_n(x, a), \\ \text{with } Q_n(x, a) &= f(x, a) + \mathbb{E}\big[V_{n+1}(X_{n+1}) \big| X_n = x, \alpha_n = a\big], \quad (x, a) \in \mathbb{R}^d \times \mathbb{A}. \end{aligned}$$

Moreover, when the infimum is attained in the DP formula (6.1.4) at any time n by $a_n^*(x) \in \arg\min_{a \in \mathbb{A}} Q_n(x, a)$, we get an optimal control in feedback form (policy) given by: $\alpha^* = (a_n^*(X_n^*))_n$ where X^* is the Markov process defined by

$$X_{n+1}^* = F(X_n^*, a_n^*(X_n^*), \varepsilon_{n+1}), \quad n = 0, \ldots, N-1, \; X_0^* = x_0.$$

The practical implementation of the DP formula may suffer from the curse of dimensionality and large complexity when the state space dimension d and the control space dimension are high. In [HPBL18], we proposed algorithms relying on deep neural networks for approximating/learning the optimal policy and then eventually the value function by performance/policy iteration or hybrid iteration with Monte Carlo regressions now or later. This research led to three algorithms, namely algorithms NNcontPI, Hybrid-Now and Hybrid-LaterQ that are recalled in Section 6.2. In Section 6.3, we perform some numerical and comparative tests for illustrating the efficiency of our different algorithms, on 100-dimensional nonlinear PDEs examples as in [EHJ17] and quadratic Backward Stochastic Differential equations as in [CR16]. We present numerical results for an option hedging problem in finance, and energy storage problems arising in the valuation of gas storage and in microgrid management. Numerical results and comparisons to quantization-type algorithms Qknn, introduced in this paper as an efficient algorithm to numerically solve low-dimensional control problems, are also provided. Finally, we conclude in Section 6.4 with some comments about possible extensions and improvements of our algorithms.

6.2 Algorithms

We introduce in this section four neural network-based algorithms for solving the discrete-time stochastic control problem (6.1.1)-(6.1.3). The convergence of these algorithms have been analyzed in detail in our companion paper [HPBL18]. We also introduce at the end of this section a quantization and k-nearest-neighbor-based algorithm (Qknn) that will be used as benchmark when testing our algorithms on low-dimensional control problems.

We are given a class of deep neural networks (DNN) for the control policy represented by the parametric functions $x \in \mathbb{R}^d \mapsto A(x;\beta) \in \mathbb{A}$, with parameters $\beta \in \mathbb{R}^q$, and a class of DNN for the value function represented by the parametric functions: $x \in \mathbb{R}^d \mapsto \Phi(x;\theta) \in \mathbb{R}$, with parameters $\theta \in \mathbb{R}^p$. Recall that these DNN functions A and Φ are compositions of linear combinations and nonlinear activation functions, see [GBC16].

Additionally, we shall be given a sequence of probability measures on the state space $\mathbb{R}^d$, that we call training measure and denoted $(\mu_n)_{n=0}^{N-1}$, and which should be seen as dataset providers to learn the optimal strategies and the value functions at time $n = 0, \ldots, N-1$.

Remark 6.2.1 (Training sets design)**.** Two cases are considered for the choice of the training measure μ_n used to generate the training sets on which will be computed the estimates at time n. The first one is a knowledge-based selection, relevant when the controller knows with a certain degree of confidence where the process has to be driven in order to optimize her cost functional. The second case is when the controller has no idea where or how to drive the process to optimize the cost functional.

(1) Exploitation only strategy

In the knowledge-based setting, there is no need for exhaustive and expensive (in time mainly) exploration of the state space, and the controller can take a training measure μ_n that assigns more points in the region of the state space that is likely to be visited by the optimally-driven process.

In practice, at time n, assuming we know that the optimal process is likely to lie in a region $\mathcal{D}$, we choose a training measure in which the density assigns lot of weight to the points of $\mathcal{D}$, for example $\mathcal{U}(\mathcal{D})$, the uniform distribution in $\mathcal{D}$.

(2) Explore first, exploit later When the controller has no idea where or how to drive the process to optimize the cost functional, we suggest the latter to adopt the following two-step approach:

(i) *Explore first:* If the agent has no idea of where to drive the process to receive large rewards, she can always proceed to an exploration step to discover favorable subsets of the state space. To do so, the training sets Γ_n at time n, for $n = 0, \ldots, N-1$, can be built as uniform grids that cover a large part of the state space, or μ can be chosen uniform on such domain. It is essential to explore far enough to acquire a good understanding of where to drive and where not to drive the process.

(ii) *Exploit later:* The estimates for the optimal controls at time n, $n = 0, \ldots, N-1$, that come up from the *Explore first* step, are relatively good in the way that they manage to avoid the poor regions when driving the process. However, the training sets that have been used to compute the estimated optimal control are too sparse to ensure accuracy on the estimation. In order to improve the accuracy, the natural idea is to build new training sets by simulating M times the process using the estimates on the optimal strategy computed from the *Explore first* step, and then proceed to another estimation of the optimal strategies using the new training sets. This trick can be seen as a two-step algorithm that improves the final estimate of the optimal control. □

Remark 6.2.2 (Choice of Neural Networks)**.** Unless otherwise specified, we use Feedback Neural Networks with two or three hidden layers and d+10 neurons per hidden layer. We tried sigmoid, tanh, ReLU and ELU activation functions and noticed that ELU is most often the one providing the best results in our applications. We normalize the input data of each neural network in order to speed up the training of the latter. □

Remark 6.2.3 (Neural Networks Training)**.** We use the Adam optimizer, as implemented in TensorFlow, with learning rate set to 0.001 or 0.005, which are the default values in TensorFlow, to train the optimal strategy and the value function computed from the algorithms described later. In order to force the weights of biases of the neurons to stay small, we use an $\mathbb{L}^2$ regularization with parameter mainly set to 0.01, but the value can change in order to make sure that the regularization term is neither too strong or too weak when added to the loss when training neural networks.

We consider a large enough number of mini-batches of size 64 or 128 for the training, depending essentially empirically on the dimension of the problem. We use at least 10 epochs[2] and stop the training when the loss computed on a validation set of size 100 stops decreasing. We noticed that taking more than one epoch really improves the quality of the estimates. □

Remark 6.2.4 (Constraints)**.** The proposed algorithms can deal with state and control constraints at any time, which is useful in several applications:

$$(X_n^\alpha, \alpha_n) \quad \in \quad \mathcal{S} \quad a.s., \quad n \in \mathbb{N},$$

where $\mathcal{S}$ is some given subset of $\mathbb{R}^d \times \mathbb{R}^q$. In this case, in order to ensure that the set of admissible controls is not empty, we assume that the sets

$$\mathbb{A}(x) \quad := \quad \Big\{ a \in \mathbb{R}^q : (F(x, a, \varepsilon_1), a) \in \mathcal{S} \ a.s. \Big\}$$

are non empty for all $x \in \mathcal{S}$, and the DP formula now reads

$$V_n(x) \quad = \quad \inf_{a \in \mathbb{A}(x)} \big[f(x,a) + P^a V_{n+1}(x) \big], \quad x \in \mathcal{S}.$$

From a computational point of view, it may be more convenient to work with unconstrained state/control variables, hence by relaxing the state/control constraint and introducing into the running cost 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 $\mathcal{S}$ is in the form: $\mathcal{S} = \{(x,a) \in \mathbb{R}^d \times \mathbb{R}^q : h_k(x,a) = 0, k = 1, \ldots, p, \ h_k(x,a) \geq 0, k = p+1, \ldots, q\}$, for some functions h_k, then one can take as penalty functions:

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

where $\mu_k > 0$ are penalization coefficients (large in practice). □

6.2.1 Control Learning by Performance Iteration

We present in this section Algorithm 6.1, which combines an optimal policy estimation by neural networks and the dynamic programming principle. We rely on the performance iteration procedure, i.e. paths are always recomputed up to the terminal time N.

6.2.1.1 Algorithm NNContPI

Our first algorithm, refereed to as NNContPI, and described in Algorithm 6.1, is well-designed for control problems with continuous control space such as $\mathbb{R}^q$ or a ball in $\mathbb{R}^q$. The main idea is to first parametrize the optimal control using a neural network in which the activation function for the output layer takes values in the control space. For example, one can take the identity function as activation function for the output layer if the control space is $\mathbb{R}^q$; or the sigmoid function if the control space is $[0,1]$, or more generally a bounded set. Then, it remains to learn the optimal parameters of the neural network using a training set, which is given as initial positions of law μ_n and random noises.

[2] We denote by epoch one pass of the full training set.

Algorithm 6.1 NNContPI

Input: the training distributions $(\mu_n)_{n=0}^{N-1}$;

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

2: Compute

$$\hat{\beta}_n \in \underset{\beta \in \mathbb{R}^q}{\operatorname{argmin}}\, \mathbb{E}\left[f\big(X_n, A(X_n;\beta)\big) + \sum_{k=n+1}^{N-1} f\big(X_k^\beta, \hat{a}_k(X_k^\beta)\big) + g\big(X_N^\beta\big) \right]$$

where $X_n \sim \mu_n$ and where $\big(X_k^\beta\big)_{k=n+1}^{N}$ is defined by induction as:

$$\begin{cases} X_{n+1}^\beta &= F\big(X_n, A\big(X_n;\beta\big), \varepsilon_{n+1}\big) \\ X_{k+1}^\beta &= F\big(X_k^\beta, \hat{a}_k\big(X_k^\beta;\beta\big), \varepsilon_{k+1}\big), \quad \text{for } k = n+1, \dots, N-1. \end{cases}$$

for $m = 1, \dots, M$;

3: Set $\hat{a}_n = A(.;\hat{\beta}_n)$; ▷ $\hat{a}_n$ is the estimate of the optimal policy at time n

4: **end for**

Output: estimates of the optimal strategy $(\hat{a}_n)_{n=0}^{N-1}$;

6.2.1.2 Algorithm ClassifPI

In the special case where the control space $\mathbb{A}$ is finite, i.e., $\mathrm{Card}(\mathbb{A}) = L < \infty$ with $\mathbb{A} = \{a_1, \dots, a_L\}$, a classification method can be used: consider a DNN that takes state x as input and returns a probability vector $p(x;\beta) = (p_\ell(x;\beta))_{\ell=1}^{L}$ with parameters β. Algorithm 6.2, presented below, is based on this idea, and is called ClassifPI.

Algorithm 6.2 ClassifPI

Input: the training distributions $(\mu_n)_{n=0}^{N-1}$;

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

2: Compute

$$\hat{a}_n(x) = a_{\hat{\ell}_n(x)} \text{ with } \hat{\ell}_n(x) \in \arg\max_{\ell=1,\dots,L} p_\ell(x;\hat{\beta}_n)$$

$$\hat{\beta}_n \in \underset{\beta \in \mathbb{R}^q}{\arg\min}\, \mathbb{E}\left[\sum_{\ell=1}^{L} p_\ell(X_n;\beta) \Big(f(X_n, a_\ell) + \sum_{k=n+1}^{N-1} f\big(X_k^\ell, \hat{a}_k(X_k^\ell)\big) + g(X_N^\ell) \Big) \right],$$

where $X_n \sim \mu_n$ on $\mathbb{R}^d$, $X_{n+1}^\ell = F(X_n, a_\ell, \varepsilon_{n+1})$, $X_{k+1}^\ell = F(X_k^\ell, \hat{a}_k(X_k^\ell), \varepsilon_{k+1})$, for $k = n+1, \dots, N-1$ and $\ell = 1, \dots, L$;

3: Set $\hat{a}_n = a_{\hat{\ell}_n(x)}$ with $\hat{\ell}_n(x) \in \arg\max_{\ell=1,\dots,L} p_\ell(x;\hat{\beta}_n)$; ▷ $\hat{a}_n$ is the estimate of the optimal policy at time n

4: **end for**

Output: estimates of the optimal strategy $(\hat{a}_n)_{n=0}^{N-1}$;

Note that, when using Algorithms 6.1 and 6.2, the estimate of the optimal strategy at time n highly relies on the estimates of the optimal strategy at time $n+1, \dots, N-1$, that have been computed previously. In particular, the practitioner who wants to use Algorithms 6.1 and 6.2 needs to keep track of the estimates of the optimal strategy at time $n+1, \dots, N-1$ in order to compute the estimate of the optimal strategy at time n.

Remark 6.2.5. In practice, for $n = N-1, ..., 0$, one should approximate the expectations (6.2.1) and (6.2.3) by its empirical mean, i.e. consider a sample from μ_n for the initial position at time n, and other samples from the law ε_k, for $k = n+1, \dots, N$, in order to generate a finite number of paths

$(X_k^\beta)_{k=n+1}^N$ on which to estimate the expectations (6.2.1) and (6.2.3) using Monte Carlo method. □

6.2.2 Control and value function learning by double DNN

We present in this section two algorithms, which in contrast with Algorithms 6.1 or 6.2, only keep track on the estimates of the value function and optimal control at time $n+1$ in order to build an estimate of the value function and optimal control at time n.

6.2.2.1 Regress Now (Hybrid-Now)

The Algorithm 6.3, 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.

Algorithm 6.3 Hybrid-Now

Input: the training distributions $(\mu_n)_{n=0}^{N-1}$;

Set $\hat{V}_N = g$;
for $n = N-1, \dots, 0$ **do**
  Compute:

$$\hat{\beta}_n \in \underset{\beta\in\mathbb{R}^q}{\operatorname{argmin}}\, \mathbb{E}\Big[f\big(X_n, A(X_n;\beta)\big) + \hat{V}_{n+1}(X_{n+1}^\beta)\Big]$$

  where $X_n \sim \mu$, and $X_{n+1}^\beta = F\big(X_n, A(X_n;\beta)\big), \varepsilon_{n+1})$;
  Set $\hat{a}_n = A(.;\hat{\beta}_n)$; ▷ $\hat{a}_n$ is the estimate of the optimal policy at time n;
  Compute

$$\hat{\theta}_n \in \underset{\theta\in\mathbb{R}^p}{\operatorname{argmin}}\, \mathbb{E}\left[\Big(\big(f(X_n, \hat{a}_n(X_n)) + \hat{V}_{n+1}(X_{n+1}^{\hat{\beta}_n}) - \Phi(X_n;\theta)\big)\Big)^2\right];$$

  Set $\hat{V}_n = \Phi(.;\hat{\theta}_n)$; ▷ $\hat{V}_n$ is the estimate of the value function at time n
end for

Output:
– estimate of the optimal strategy $(\hat{a}_n)_{n=0}^{N-1}$;
– estimate of the value function $(\hat{V}_n)_{n=0}^{N-1}$;

Remark 6.2.6. One can combine different features from Algorithms 6.1, 6.2 and 6.3 to solve specific problems, as it has been done for example in Section 6.3.4, where we designed Algorithm 6.6 to solve a Smart Grid Management problem. □

6.2.2.2 Regress Later and Quantization (Hybrid-LaterQ)

The Algorithm 6.4, called Hybrid-LaterQ, combines regress-later and quantization methods to build estimates of the value function. The main idea behind Algorithm 6.4 is to first interpolate the value function at time $n+1$ by a set of basis functions, which is in the spirit of the regress-later-based algorithms, and secondly regress the interpolation at time n using quantization. The usual regress-later approach requires the ability to compute closed-form conditional expectations, which limits the stochastic dynamics and regression bases that can be considered. The use of quantization avoids this limitation and makes the regress-later algorithm more generally applicable.

Let us first recall the basic ingredients of quantization. We denote by $\hat{\varepsilon}$ a K-quantizer of the $\mathbb{R}^d$-valued random variable $\varepsilon_{n+1} \rightsquigarrow \varepsilon_1$ (typically a Gaussian random variable), that is a discrete random

variable on a grid $\Gamma = \{e_1, \ldots, e_L\} \subset (\mathbb{R}^d)^L$ defined by

$$\hat{\varepsilon} \quad = \quad \mathrm{Proj}_\Gamma(\varepsilon_1) \; := \; \sum_{\ell=1}^{L} e_\ell 1_{\varepsilon_1 \in C_\ell(\Gamma)},$$

where $C_1(\Gamma)$, ..., $C_K(\Gamma)$ are Voronoi tesselations of Γ, i.e., Borel partitions of the Euclidian space $(E, |.|)$ satisfying

$$C_\ell(\Gamma) \quad \subset \quad \Big\{ e \in E : |e - e_\ell| \; = \; \min_{j=1,\ldots,L} |e - e_j| \Big\}.$$

The discrete law of $\hat{\varepsilon}$ is then characterized by

$$\hat{p}_\ell \quad := \quad \mathbb{P}[\hat{\varepsilon} = e_\ell] \; = \; \mathbb{P}[\varepsilon_1 \in C_\ell(\Gamma)], \quad \ell = 1, \ldots, L.$$

The grid points (e_ℓ) which minimize the L^2-quantization error $\|\varepsilon_1 - \hat{\varepsilon}\|_2$ lead to the so-called optimal L-quantizer, and can be obtained by a stochastic gradient descent method, known as Kohonen algorithm or competitive learning vector quantization (CLVQ) algorithm, which also provides as a byproduct an estimation of the associated weights $(\hat{p}_\ell)$. We refer to [PPP04c] for a description of the algorithm, and mention that for the normal distribution, the optimal grids and the weights of the Voronoi tesselations are precomputed on the website http://www.quantize.maths-fi.com.

Quantization is mainly used in Algorithm 6.4 to efficiently approximate the expectations: recalling the dynamics (6.1.1), the conditional expectation operator for any functional W is equal to

$$P^{\hat{a}_n^M(x)} W(x) \; = \; \mathbb{E}\big[W(X_{n+1}^{\hat{a}_n^M}) | X_n = x\big] \quad = \quad \mathbb{E}\big[W(F(x, \hat{a}_n^M(x), \varepsilon_1))\big], \quad x \in \mathbb{R}^d,$$

that we shall approximate analytically by quantization via:

$$\widehat{P}^{\hat{a}_n^M(x)} W(x) \quad := \quad \mathbb{E}\big[W(F(x, \hat{a}_n^M(x), \hat{\varepsilon}))\big] \; = \; \sum_{\ell=1}^{K} \hat{p}_\ell W\left(F(x, \hat{a}_n^M(x), e_\ell)\right).$$

Observe that the solution to (6.2.7) actually provides a neural network $\Phi(.; \hat{\theta}_{n+1})$ that interpolates $\tilde{V}_{n+1}$. Hence the Algorithm 6.4 contains an interpolation step, and moreover, any kind of distance in $\mathbb{R}^d$ can be chosen as a loss to compute $\hat{\theta}_{n+1}$. In (6.2.7), we decide to take the $\mathbb{L}^2$-loss, mainly because it is the one that worked the best in our applications.

Remark 6.2.7 (Quantization)**.** In dimension 1, we used the optimal grids and weights with $L = 21$ points, to quantize the reduced and centered normal law $\mathcal{N}(0, 1)$; and took 100 points to quantize the reduced and centered normal law in dimension 2, i.e. $\mathcal{N}_2(0, 1)$. All the grids and weights for the optimal quantization of the normal law in dimension d are available in http://www.quantize.maths-fi.com for $d = 1, \ldots, 100$. □

6.2.2.3 Some remarks on Algorithms 6.3 and 6.4

As in Remark 6.2.5, all the expectations written in our pseudo-codes in Algorithm 6.3 and 6.4 should be approximated by empirical mean using a finite training set.

Algorithms 6.3 or 6.4 are quite efficient to use in the usual case where the value function and the optimal control at time n are very close to the value function and the optimal control at time $n+1$, which happens e.g. when the value function and the optimal control are approximations of the time discretization of a continuous in time value function and an optimal control. In this case, it is recommended to follow the two steps procedure:

(i) initialize the parameters (i.e. weights and bias) of the neural network approximations of the value function and the optimal control at time n to the ones of the neural network approximations of the value function and the optimal control at time $n+1$.

Algorithm 6.4 Hybrid-LaterQ

Input:

– the training distributions $(\mu_n)_{n=0}^{N-1}$;

– The grid $\{e_1, \ldots, e_L\}$ of L points in E, with weights $p_1, \ldots, p_L$ for the quantization of the noise ε_n;

1: Set $\hat{V}_N = g$;

2: **for** $n = N-1, \ldots, 0$ **do**

3: Compute:

$$\hat{\beta}_n \quad \in \quad \underset{\beta \in \mathbb{R}^q}{\operatorname{argmin}}\, \mathbb{E}\Big[f\big(X_n, A(X_n;\beta)\big) + \hat{V}_{n+1}(X_{n+1}^{\beta}) \Big]$$

where $X_n \sim \mu_n$, and $X_{n+1}^{\beta} = F\big(X_n, A(X_n;\beta)\big), \varepsilon_{n+1})$;

4: Set $\hat{a}_n = A(.;\hat{\beta}_n)$; ▷ $\hat{a}_n$ is the estimate of the optimal policy at time n

5: Compute

$$\hat{\theta}_n \quad \in \quad \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}}\, \mathbb{E}\left[\Big(\hat{V}_{n+1}(X_{n+1}^{\hat{\beta}_n}) - \Phi(X_{n+1};\theta) \Big)^2 \right]$$

and set $\tilde{V}_{n+1} = \Phi(.;\hat{\theta}_{n+1})$; ▷ interpolation at time $n+1$

6: Set

$$\hat{V}_n(x) \quad = \quad f(x, \hat{a}_n(x)) + \sum_{\ell=1}^{L} p_\ell \tilde{V}_{n+1}\big(F(x, \hat{a}_n(x), e_\ell)\big);$$

▷ $\hat{V}_n$ is the estimate by quantization of the value function at time n

7:

8: **end for**

Output:

– estimate of the optimal strategy $(\hat{a}_n)_{n=0}^{N-1}$;

– estimate of the value function $(\hat{V}_n)_{n=0}^{N-1}$;

(ii) take a very small learning rate parameter, for the Adam optimizer, that guarantees the stability of the parameters' updates from the gradient-descent based learning procedure.

Doing so, one obtains stable estimates of the value function and optimal control, which is desirable. We highlight the fact that this stability procedure cannot be implemented in most of the algorithms proposed in the literature (for example the ones presented in [Bal+19] which are based on regress-now, regress-later or quantization).

6.2.3 Quantization with k-nearest-neighbors: Qknn-algorithm

Algorithm 6.5 presents the pseudo-code of an algorithm based on the quantization and k-nearest neighbors methods, called Qknn, which will be the benchmark in all the low-dimensional control problems that will be considered in Section 6.3 to test NNContPI, ClassifPI, Hybrid-Now and Hybrid-Later. Also, comparisons of Algorithm 6.5 to other well-known algorithms on various control problems in low-dimension are performed in [Bal+19], which show in particular that Algorithm 6.5 works very well to solve low-dimensional control problems. Actually, in our experiments, Algorithm 6.5 always outperforms the other algorithms based either on regress-now or regress-later methods whenever the dimension of the problem is low enough for Algorithm 6.5 to be feasible.

Just as it has been done in Section 6.2.2.2, we consider an L-optimal quantizer of the noise ε_n, i.e. a discrete random variable $\hat{\varepsilon}_n$ valued in a grid $\{e_1, \dots, e_L\}$ of L points in E, and with weights $p_1, \dots, p_L$. We also consider grids Γ_k, $k = 0, \dots, N$ of points in $\mathbb{R}^d$, which are assumed to properly cover the region of $\mathbb{R}^d$ that is likely to be visited by the optimally driven process X at time $k = 0, \dots, N-1$. These grids can be viewed as samples of well-chosen training distributions where more points are taken in the region that is likely to be visited by the optimally driven controlled process (see Remark 6.2.1 for details on the choice of the training measure).

Remark 6.2.8. The estimate of the Q-value at time n given by (6.2.8) is not continuous w.r.t. the control variable a, which might cause some stability issues when running Qknn, especially during the optimization procedure (6.2.9). We refer to Section 3.2.2. in [Bal+19] for a detailed presentation of an extension of Algorithm 6.5 where the estimates of the Q value function Q_n is continuous w.r.t. the control variable. □

Algorithm 6.5 Qknn

Input:
– Grids Γ_k, $k = 0, \dots, N$ in $\mathbb{R}^d$;
– Grid $\{e_1, \dots, e_L\}$ of L points in E, with weights $p_1, \dots, p_L$ for the quantization of ε_n;

Set $\hat{V}_N = g$;
for $n = N-1, \dots, 0$ **do**
    Compute for $(z, a) \in \Gamma_n \times A$,

$$\hat{Q}_n(z,a) \quad = \quad f(z,a) + \sum_{\ell=1}^{L} p_\ell \widehat{V}_{n+1}\Big(\mathrm{Proj}_{\Gamma_{n+1}}\big(F(z,a,e_\ell)\big)\Big),$$

    where $\mathrm{Proj}_{\Gamma_{n+1}}$ is the Euclidean projection over Γ_{n+1};
    ▷ $\hat{Q}_n$ is the approximated Q-value[3] at time n
    Compute the optimal control at time n

$$\hat{A}_n(z) \in \underset{a \in A}{\mathrm{argmin}} \big[\hat{Q}_n(z,a)\big], \quad \forall z \in \Gamma_n; \tag{6.2.9}$$

    ▷ use classical optimization algorithms of deterministic functions for this step
    Set $\widehat{V}_n(z) = \hat{Q}_n\big(z, \hat{A}_n(z)\big), \quad \forall z \in \Gamma_n$;
    ▷ $\widehat{V}_n$ is the estimate by quantization of the value function
end for

Output:
– estimate of the optimal strategy $(\hat{a}_n)_{n=0}^{N-1}$;
– estimate of the value function $(\hat{V}_n)_{n=0}^{N-1}$;

[3]The Q-value at time n, denoted by Q_n, is defined as the function that takes the couple state-action (x, a) as argument, and returns the expected optimal reward earned from time n to time N when the process X is at state x and action a is chosen at time n; i.e. $Q_n : \mathbb{R}^d \times \mathbb{R}^q \ni (x, a) \mapsto f(x, a) + \mathbb{E}^a_{n,x}[V_{n+1}(X_{n+1})]$.

6.3 Numerical applications

In this section, we test the Neural-Networks-based algorithms presented in Section 6.2 on different examples. In high-dimension, we took the same example as already considered in [EHJ17] so that we can directly compare our results to theirs. In low-dimension, we compared the results of our algorithms to the ones provided by Qknn, which has been introduced in Section 6.2 as an excellent benchmark for low-dimensional control problems.

6.3.1 A semilinear PDE

We consider the following semilinear PDE with quadratic growth in the gradient:

$$\left\{ \begin{array}{rcl} \frac{\partial v}{\partial t} + \Delta_x v \ - \ |D_x v|^2 & = & 0, \quad (t,x) \in [0,T) \times \mathbb{R}^d, \\ v(T,x) & = & g(x), \quad x \in \mathbb{R}^d \, . \end{array} \right. \tag{6.3.1}$$

By observing that for any $p \in \mathbb{R}^d$, $-|p|^2 = \inf_{a \in \mathbb{R}^d} [|a|^2 + 2a.p]$, the PDE (6.3.1) can be written as a Hamilton-Jacobi-Bellman equation

$$\left\{ \begin{array}{rcl} \frac{\partial v}{\partial t} + \Delta_x v \ + \ \inf\limits_{a \in \mathbb{R}^d} \left[|a|^2 + 2a.D_x v \right] & = & 0, \quad (t,x) \in [0,T) \times \mathbb{R}^d, \\ v(T,x) & = & g(x), \quad x \in \mathbb{R}^d, \end{array} \right.$$

hence associated with the stochastic control problem

$$v(t,x) \ = \ \inf_{\alpha \in \mathcal{A}} \mathbb{E} \left[\int_t^T |\alpha_s|^2 ds + g(X_T^{t,x,\alpha}) \right],$$

where $X = X^{t,x,\alpha}$ is the controlled process governed by

$$dX_s \ = \ 2\alpha_s ds \ + \ \sqrt{2} dW_s, \quad t \leq s \leq T, \ X_t = x,$$

W is a d-dimensional Brownian motion, and the control process α is valued in $A = \mathbb{R}^d$. The time discretization (with time step $h = T/N$) of the control problem (6.3.2) leads to the discrete-time control problem (6.1.1)-(6.1.2)-(6.1.3) with

$$X_{n+1}^\alpha \ = \ X_n^\alpha + 2\alpha_n h + \sqrt{2h} \varepsilon_{n+1} \ =: \ F(X_n^\alpha, \alpha_n, \varepsilon_{n+1}), \quad n = 0, \ldots, N-1,$$

where $(\varepsilon_n)_n$ is a sequence of i.i.d. random variables of law $\mathcal{N}(0, \mathbf{I}_d)$, and the cost functional

$$J(\alpha) \ = \ \mathbb{E} \left[\sum_{n=0}^{N-1} h |\alpha_n|^2 \ + \ g(X_N^\alpha) \right].$$

On the other hand, it is known that an explicit solution to (6.3.1) (or equivalently (6.3.1)) can be obtained via a Hopf-Cole transformation (see e.g. [CR16]), and is given by

$$v(t,x) \ = \ - \ln \Big(\mathbb{E} \Big[\exp \big(- g(x + \sqrt{2} W_{T-t}) \big) \Big] \Big), \quad (t,x) \in [0,T] \times \mathbb{R}^d \, .$$

We choose to run tests on two different examples that have already been considered in the literature:

Test 1 Some recent numerical results have been obtained in [EHJ17] (see Section 4.3 in [EHJ17]) when $T = 1$ and $g(x) = \ln(\frac{1}{2}(1 + |x|^2))$ in dimension $d = 100$ (see Table 2 and Figure 3 in [EHJ17]). Their method is based on neural network regression to solve the BSDE representation associated to the PDE (6.3.1), and provide estimates of the value function at time 0 and state 0 for different values of a coefficient γ. We plotted the results of the Hybrid-Now algorithm in Figure 6.1. Hybrid-Now

Table 6.1 – Value function at time 0 and state 0 w.r.t. γ, computed with the Y&R, Hybrid-Now, Hybrid-Later and Qknn algorithms. Bench reports the MC estimates of the closed-form formula (6.3.3).

γ	Y&R	Hybrid-LaterQ	Hybrid-Now	Qknn	Bench
1.0	-0.402	-0.456	-0.460	-0.461	-0.464
0.5	-0.466	-0.495	-0.507	-0.508	-0.509
0.1	-0.573	-0.572	-0.579	-0.581	-0.586
0.0	-0.620	-1.000	-1.000	-1.000	-1.000

took one hour to achieves a relative error of 0.11%, using a 4-cores 3GHz intel Core i7 CPU. We want to highlight the fact that the algorithm presented in [EHJ17] only needed 330 seconds to provide a relative error of 0.17%. However, to our experience, it is difficult to reduce the relative error from 0.17% to 0.11% using their algorithm. Also, we believe that the computation time of our algorithm can easily be alleviated; some ideas in that direction are discussed in Section 6.4.

We also considered the same problem in dimension $d = 2$, for which we plotted the first component of X w.r.t. time in Figure 6.3, for five different paths of the Brownian motion, where for each ω, the agent follows either the naive ($\alpha = 0$) or the Hybrid-Now strategy. One can see that both strategies are very similar when the terminal time is far; but the Hybrid-Now strategy clearly forces X to get closer to 0 when the terminal time gets closer, in order to reduce the terminal cost.

Implementation details on the algorithms presented in Section 6.2: As one can guess from the representation of v in (6.3.2), it is probably optimal to drive the process X around 0. Hence we decided to take $\mu_n := (\frac{nT}{N})^{1/2}\mathcal{N}_d(0, I_d)$ as a training measure at time n to learn the optimal strategy and value function at time n, for $n = 0, \dots, N-1$.

Test 2 Tests of the algorithms have also been run in dimension 1 with the terminal cost $g(x) = -x^\gamma \mathbb{1}_{0\le x\le 1} - \mathbb{1}_{1\le x}$ and $\gamma \in (0, 1)$. This problem has already been considered in [Ric10], where the author used the algorithm presented in [Ric11], which is based on the BSDE representation of the PDE (6.3.1). Their estimates of the value function at time 0 and state 0, when $\gamma = 1, 0.5, 0.1, 0$, are available in [Ric10], and have been reported in column $Y\&R$ of Table 6.1. Also, the exact values for the value function have been computed for these values of γ by Monte Carlo using the closed-form formula (6.3.3), and are reported in the column Bench of Table 6.1. Tests of the Hybrid-Now and Hybrid-LaterQ algorithms have been run, and the estimates of the value function at time 0 and state $x = 0$ are reported in the Hybrid-Now and Hybrid-LaterQ columns. We also tested Qknn and reported its results in column Qknn. Note that Qknn is particularly well-suited to 1-dimensional control problems. In particular, it is not time-consuming since the dimension of the state space is d=1. Actually, it provides the fastest results, which is not surprising since the other algorithms need time to learn the optimal strategy and value function through gradient-descent method at each time step $n = 0, \dots, N-1$. Moreover, Table 6.1 reveals that Qknn is the most accurate algorithm on this example, probably because it uses local methods in space to estimate the conditional expectation that appears in the expression of the Q-value.

We end this paragraph by giving some implementation details for the different algorithms as part of **Test 2**:

- *Y&R:* The algorithm Y&R converged only when using a Lipschitz version of g. The following approximation was used to obtained the results in Table 6.1:

$$g_N(x) = \begin{cases} g(x) & \text{if } x \notin [0, N^{\frac{-1}{1-\gamma}}] \\ -Nx & \text{otherwise.} \end{cases}$$

- *Hybrid-Now:* We used $N = 40$ time steps for the time-discretization of $[0, T]$. The value functions and optimal controls at time $n = 0, \dots, N-1$ are estimated using neural networks with 3 hidden layers and 10+5+5 neurons.

Figure 6.1 – Relative error of the Hybrid-Now estimate of the value function at time 0 w.r.t the number of mini-batches used to build the Hybrid-Now estimators of the optimal strategy. The value functions have been computed running three times a forward Monte Carlo with a sample of size 10,000, following the optimal strategy estimated by the Hybrid-Now algorithm.

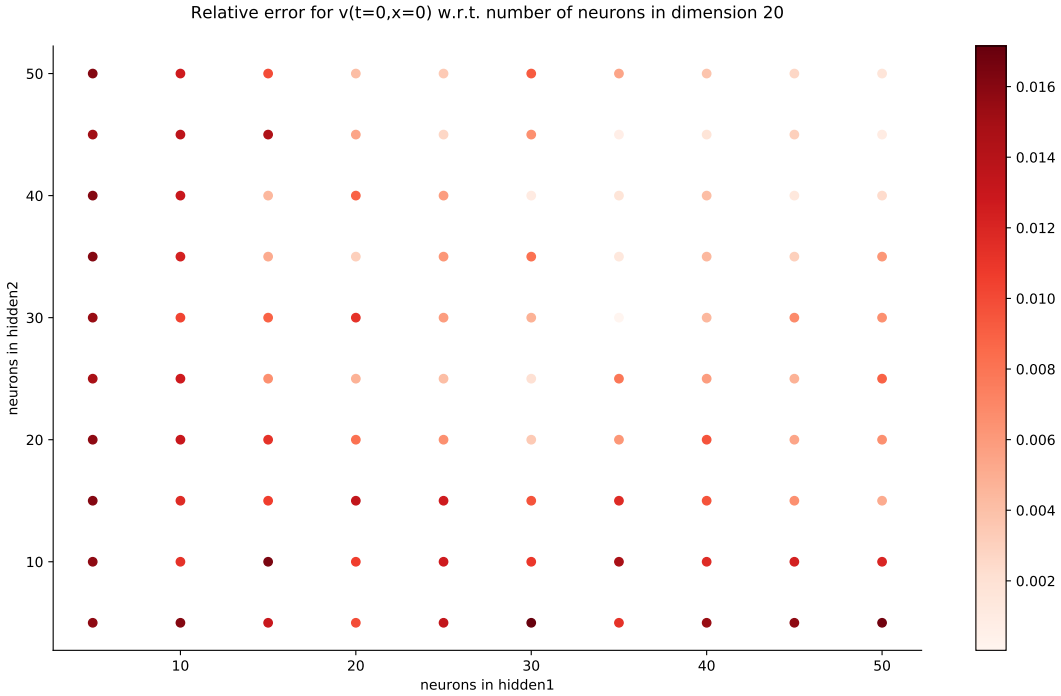


Figure 6.2 – Relative error of the Hybrid-Now estimate of the value function at time 0 w.r.t the number of neurons in the first and second hidden layers. On can observe that taking roughly 35 neurons for the first and 35 neurons for the second hidden layers led to the best estimates of the value function.

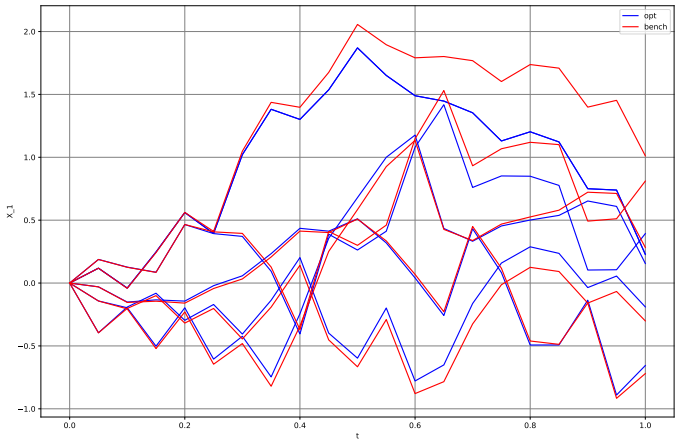


Figure 6.3 – Pathwise comparison of the first component of X w.r.t. time when the agent follows the optimal strategy estimated by the Hybrid-Now algorithm (opt) and the naive strategy $\alpha = 0$ (bench). The dimension of the semilinear control problem has been set to d=2. Observe that, as expected, the strategy designed by the Hybrid-Now algorithm is to not influence the diffusion of X when the terminal time is far in order to avoid any running cost; and to try to make X small when the terminal time gets close, in order to minimize the terminal cost.

- *Hybrid-LaterQ:* We used $N = 40$ time steps for the time-discretization of $[0, T]$. The value functions and optimal controls at time $n = 0, \dots, N-1$ are estimated using neural networks with 3 hidden layers containing 10+5+5 neurons; and 51 points for the quantization of the exogenous noise.
- *Qknn:* We used $N = 40$ time steps for the time-discretization of $[0, T]$. We take 51 points to quantize the exogenous noise, $\varepsilon_n \sim \mathcal{N}(0, 1)$, for $n = 0, \dots, N$; and decided to use the 200 points of the optimal grid of $\mathcal{N}_2(0, 1)$ for the state space discretization.

The main conclusion regarding the results in this semilinear PDE problem is that Hybrid-Now provides better estimates of the solution to the PDE in dimension d=100 than the previous results available in [EHJ17] but requires more time to do so.

Hybrid-Now and Hybrid-Later provide better results than those available in [Ric11] to solve the PDE in dimension 2; but are outperformed by Qknn, which is arguably very accurate.

6.3.2 Option hedging

Our second example comes from a classical hedging problem in finance. We consider an investor who trades in q stocks with (positive) price process $(P_n)_n$, and we denote by (α_n) valued in $\mathbb{A} \subset \mathbb{R}^q$ the amount held in these assets on the period $(n, n+1]$. We assume for simplicity that the price of the riskless asset is constant equal to 1 (zero interest rate). It is convenient to introduce the return process as: $R_{n+1} = \mathrm{diag}(P_n)^{-1}(P_{n+1} - P_n)$, $n = 0, \dots, N-1$, so that the self-financed wealth process of the investor with a portfolio strategy α, and starting from some capital w_0, is governed by

$$\mathcal{W}^{\alpha}_{n+1} \quad = \quad \mathcal{W}^{\alpha}_n + \alpha_n . R_{n+1}, \quad n = 0, \dots, N-1, \quad \mathcal{W}^{\alpha}_0 = w_0.$$

Given an option payoff $h(P_N)$, the objective of the agent is to minimize over her portfolio strategies α her expected square replication error

$$V_0 \quad = \quad \inf_{\alpha \in \mathcal{A}} \mathbb{E}\Big[\ell\big(h(P_N) - \mathcal{W}^{\alpha}_N\big)\Big],$$

where ℓ is a convex function on $\mathbb{R}$. Assuming that the returns R_n, $n = 1, \dots, N$ are i.i.d, we are in a $(q+1)$-dimensional framework of Section 5.1 with $X^\alpha = (\mathcal{W}^\alpha, P)$ with $\varepsilon_n = R_n$ valued in $E \subset \mathbb{R}^q$, with the dynamics function

$$F(w,p,a,r) \quad = \quad \begin{cases} w + a.r \\ p + \mathrm{diag}(p)r, \end{cases} \qquad x = (w,p) \in \mathbb{R} \times \mathbb{R}^q,\ a \in \mathbb{R}^q,\ r \in E,$$

the running cost function $f = 0$ and the terminal cost $g(w,p) = \ell(h(p) - w)$. We test our algorithm in the case of a square loss function, i.e. $\ell(w) = w^2$, and when there is no portfolio constraints $\mathbb{A} = \mathbb{R}^q$, and compare our numerical results with the explicit solution derived in [BKL01]: denote by $\nu(dr)$ the distribution of R_n, by $\bar{\nu} = \mathbb{E}[R_n] = \int r\nu(dr)$ its mean, and by $\bar{M}_2 = \mathbb{E}[R_n R_n^\intercal]$ assumed to be invertible; we then have

$$V_n(w,p) \quad = \quad K_n w^2 - 2Z_n(p)w + C_n(p)$$

where the functions $K_n > 0$, $Z_n(p)$ and $C_n(p)$ are given in backward induction, starting from the terminal condition

$$K_N \ = \ 1, \ \ Z_N(p) \ = \ h(p), \ \ C_N(p) \ = \ h^2(p),$$

and for $n = N-1, \dots, 0$, by

$$\begin{aligned}
K_n &= K_{n+1}\big(1 - \bar{\nu}^\intercal \bar{M}_2^{-1} \bar{\nu}\big), \\
Z_n(p) &= \int Z_{n+1}(p + \mathrm{diag}(p)r)\nu(dr) - \bar{\nu}^\intercal \bar{M}_2^{-1} \int Z_{n+1}(p + \mathrm{diag}(p)r) r \nu(dr), \\
C_n(p) &= \int C_{n+1}(p + \mathrm{diag}(p)r)\nu(dr) \\
&\qquad - \frac{1}{K_{n+1}} \Big(\int Z_{n+1}(p + \mathrm{diag}(p)r) r \nu(dr) \Big)^\intercal \bar{M}_2^{-1} \Big(\int Z_{n+1}(p + \mathrm{diag}(p)r) r \nu(dr) \Big),
\end{aligned}$$

so that $V_0 = K_0 w_0^2 - 2Z_0(p_0)w_0 + C_0(p_0)$, where p_0 is the initial stock price. Moreover, the optimal portfolio strategy is given in feedback form by $\alpha_n^* = a_n^*(\mathcal{W}_n^*, P_n)$, where $a_n^*(w,s)$ is the function

$$a_n^*(w,p) = \bar{M}_2^{-1} \left[\frac{\int Z_{n+1}(p + \mathrm{diag}(p)r) r \nu(dr)}{K_{n+1}} - \bar{\nu} w \right],$$

and $\mathcal{W}^*$ is the optimal wealth associated with α^*, i.e., $\mathcal{W}_n^* = \mathcal{W}_n^{\alpha^*}$. Moreover, the initial capital w_0^* that minimizes $V_0 = V_0(w_0, p_0)$, and called (quadratic) hedging price is given by

$$w_0^* \quad = \quad \frac{Z_0(p_0)}{K_0}.$$

Test Take $N = 6$, and consider one asset $q = 1$ with returns modeled by a trinomial tree:

$$\nu(dr) \quad = \quad \pi_+ \delta_{r_+} + \pi_0 \delta_0 + \pi_- \delta_{r_-}, \quad \pi_0 + \pi_+ + \pi_- = 1,$$

with $r_+ = 5\%$, $r_- = -5\%$, $\pi_+ = 60\%$, $\pi_- = 30\%$. Take $p_0 = 100$, and consider the call option $h(p) = (p - \kappa)_+$ with $\kappa = 100$. The price of this option is defined as the initial value of the portfolio that minimizes the terminal quadratic loss of the agent when the latter follows the optimal strategy associated with the initial value of the portfolio. In this test, we want to determine the price of the call and the associated optimal strategy using different algorithms.

Remark 6.3.1. The option hedging problem is linear-quadratic, hence belongs to the class of problems where the agent has ansatzes on the optimal control and the value function. Indeed, we expect here the optimal control to be affine w.r.t. w and the value function to be quadratic w.r.t. w. For these kind of problems, the algorithms presented in Section 6.2 can easily be adapted so that the expressions of the estimators satisfy the ansatzes. See (6.3.4) and (6.3.5) for the option hedging problem. □

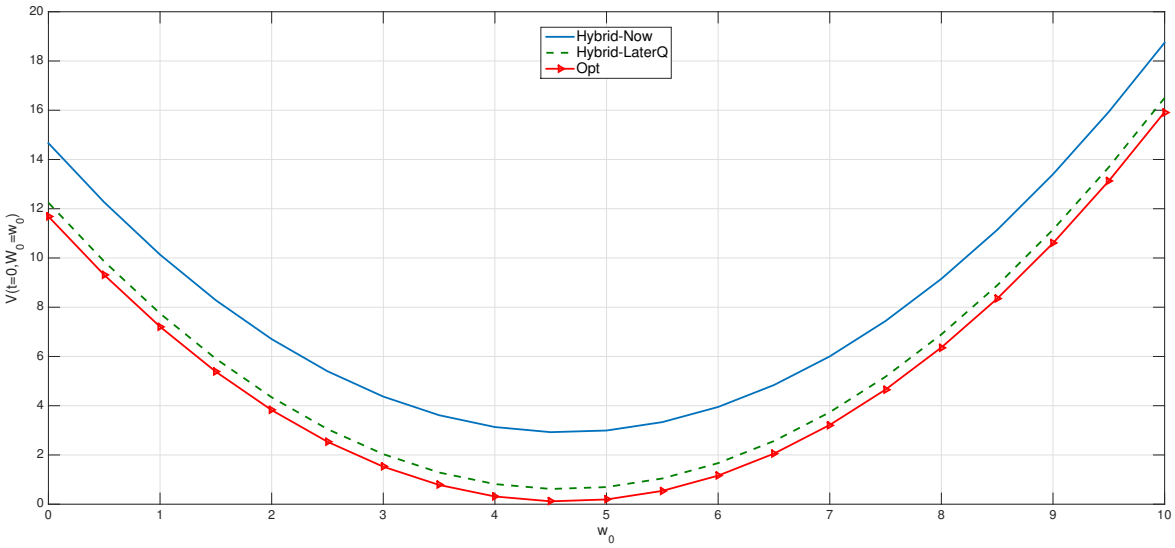


Figure 6.4 – Estimates of the value function at time 0 w.r.t. w_0 using Hybrid-Now (blue line) or Hybrid-LaterQ (green dashes). We draw the value function in red for comparison. One can observe that all the algorithms estimate the price to be 4.5, but Hybrid-LaterQ is better than Hybrid-Now at reducing the quadratic risk.

Numerical results In Figure 6.4, we plot the value function at time 0 w.r.t w_0, the initial value of the portfolio, when the agent follows the theoretical optimal strategy (benchmark), and the optimal strategy estimated by the Hybrid-Now or Hybrid-LaterQ algorithms. We perform forward Monte Carlo using 10,000 samples to approximate the lower bound of the value function at time 0 (see [HL17] for details on how to get an approximation of the upper-bound of the value function via duality). One can observe that while all the algorithms give a call option price approximately equal to 4.5, Hybrid-LaterQ clearly provides a better strategy than Hybrid-Now to reduce the quadratic risk of the terminal loss.

We plot in Figure 6.5 three different paths of the value of the portfolio w.r.t the time n, when the agent follows either the theoretical optimal strategy (red), or the estimated one using Hybrid-Now (blue) or Hybrid-LaterQ (green). We set $w_0 = 100$ for these simulations.

Comments on Hybrid-Now and Hybrid-LaterQ The Option Hedging problem belongs to the class of the linear-quadratic control problems for which we expect the optimal control to be affine w.r.t. w and the value function to be quadratic w.r.t. w. It is then natural to consider the following classes of controls $\mathcal{A}_M$ and functions $\mathcal{F}_M$ to properly approximate the optimal controls and the values functions at time n=$0, \dots, N-1$:

$$\mathcal{A}_M := \left\{(w,p) \mapsto A(x;\beta)\cdot\left(1,w\right)^\intercal; \quad \beta \in \mathbb{R}^p\right\}, \tag{6.3.4}$$

$$\mathcal{F}_M := \left\{(w,p) \mapsto \Phi(x;\theta)\cdot\left(1,w,w^2\right)^\intercal; \quad \theta \in \mathbb{R}^p\right\}, \tag{6.3.5}$$

where β describes the parameters (weights+bias) associated with the neural network A and θ describes those associated with the neural network Φ. The notation $^\intercal$ stands for the transposition, and $\cdot$ for the inner product. Note that there are 2 (resp. 3) neurons in the output layer of A (resp. Φ), so that the inner product is well-defined in (6.3.5) and (6.3.4).

6.3.3 Valuation of energy storage

We present a discrete-time version of the energy storage valuation problem studied in [CL10]. We consider a commodity (gas) that has to be stored in a cave, e.g. salt domes or aquifers. The manager of

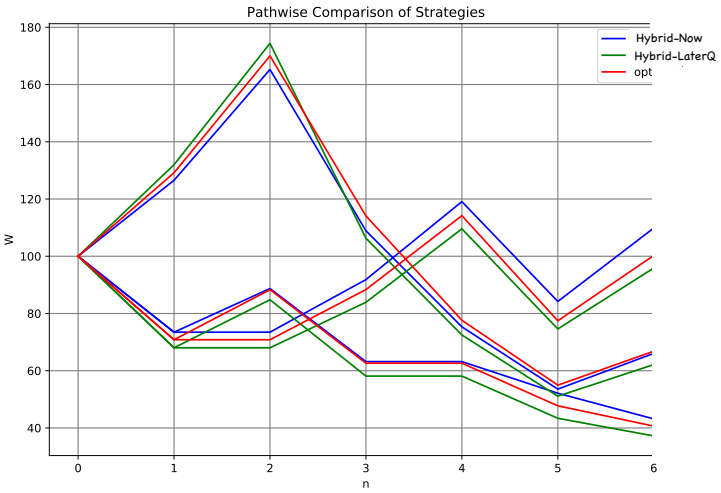


Figure 6.5 – Three simulations of the agent's wealth w.r.t. the time n when, for each ω, the latter follows the theoretical optimal strategy (red), the estimated one using Hybrid-Now (blue) and the one using Hybrid-LaterQ (green). We took $w_0 = 100$. Observe that the process is driven similarly to the optimally controlled process, when the agent follows the estimated optimal strategy using Hybrid-LaterQ or Hybrid-Now.

such a cave aims to maximize the real options value by optimizing over a finite horizon N the dynamic decisions to inject or withdraw gas as time and market conditions evolve. We denote by (P_n) the gas price, which is an exogenous real-valued Markov process modeled by the following mean-reverting process:

$$P_{n+1} \quad = \quad \bar{p}(1-\beta) + \beta P_n + \xi_{n+1},$$

where $\beta < 1$, and $\bar{p} > 0$ is the stationary value of the gas price. The current inventory in the gas storage is denoted by $(C_n^\alpha)_n$ and depends on the manager's decisions represented by a control process $\alpha = (\alpha_n)$ valued in $\{-1, 0, 1\}$: $\alpha_n = 1$ (resp. -1) means that she injects (resp. withdraws) gas with an injection (resp. withdrawal) rate $a_{in}(C_n^\alpha)$ (resp. $a_{out}(C_n^\alpha)$) requiring (causing) a purchase (resp. sale) of $b_{in}(C_n^\alpha) \geq a_{in}(C_n^\alpha)$ (resp. $b_{out}(C_n^\alpha) \leq a_{out}(C_n^\alpha)$), and $\alpha_n = 0$ means that she is doing nothing. The difference between b_{in} and a_{in} (resp. b_{out} and a_{out}) indicates gas loss during injection/withdrawal. The evolution of the inventory is then governed by

$$C_{n+1}^\alpha \quad = \quad C_n^\alpha + h(C_n^\alpha, \alpha_n), \quad n = 0, \ldots, N-1, \; C_0^\alpha \; = \; c_0,$$

where we set

$$h(c,a) \quad = \quad \begin{cases} a_{in}(c) & \text{for } \; a = 1 \\ 0 & \text{for } \; a = 0 \\ -a_{out}(c) & \text{for } \; a = -1, \end{cases}$$

and we have the physical inventory constraint:

$$C_n^\alpha \quad \in \quad [C_{min}, C_{max}], \quad n = 0, \ldots, N.$$

The running gain of the manager at time n is $f(P_n, C_n^\alpha, \alpha_t)$ given by

$$f(p,c,a) \quad = \quad \begin{cases} -b_{in}(c)p - K_1(c) & \text{for } \; a = 1 \\ -K_0(c) & \text{for } \; a = 0 \\ b_{out}(c)p - K_{-1}(c) & \text{for } \; a = -1, \end{cases}$$

and $K_i(c)$ represents the storage cost in each regime $i = -1, 0, 1$. The problem of the manager is then to maximize over α the expected total profit

$$J(\alpha) \quad = \quad \mathbb{E}\left[\sum_{n=0}^{N-1} f(P_n, C_n^\alpha, \alpha_n) + g(P_N, C_N^\alpha)\right],$$

where a common choice for the terminal condition is

$$g(p, c) \quad = \quad -\mu p(c_0 - c)_+,$$

which penalizes for having less gas than originally, and makes this penalty proportional to the current price of gas ($\mu > 0$). We are then in the 2-dimensional framework of Section 5.1 with $X^\alpha = (P, C^\alpha)$, and the set of admissible controls in the dynamic programming loop is given by:

$$A_n(c) = \big\{a \in \{-1, 0, 1\} : c + h(c, a) \in [C_{min}, C_{max}],\ c \in [C_{min}, C_{max}]\big\},\ \ n = 0, \ldots, N-1.$$

Test We fixed the parameters as follows, to run our numerical tests:

$$\begin{aligned} a_{in}(c) \ = \ b_{in}(c) \ &= \ 0.06, \qquad a_{out}(c) \ = \ b_{out}(c) \ = \ 0.25 \\ K_i(c) \ &= \ 0.01c \end{aligned}$$

$C_{max} = 8$, $C_{min} = 0$, $c_0 = 4$, $\bar{p} = 5$, $\beta = 0.5$, $\xi_{n+1} \rightsquigarrow \mathcal{N}(0, \sigma^2)$ with $\sigma^2 = 0.05$, and $\mu = 2$ in the terminal penalty function, N =30.

Numerical results We plotted in Figure 6.6 the estimates of the value function at time 0 w.r.t. a_{in} using Qknn, as well as the reward function (6.3.8) associated to the naive do-nothing strategy $\alpha = 0$ (see Bench in figure 6.6). As expected, the naive strategy performs well when a_{in} is small compared to a_{out}, since, in this case, it takes time to fill the cave, so that the agent is likely to do nothing in order to avoid any penalization at terminal time. When a_{in} is of the same order as a_{out}, it is easy to fill up and empty the cave, so the agent has more freedom to buy and sell gas in the market without worrying about the terminal cost. Observe that the value function is not monotone, due to the fact that the C component in the state space takes its value in a bounded and discrete set (see (6.3.7)).

Table 6.2 provides the estimates of the value function using the ClassifPI, Hybrid-Now and Qknn algorithms. Observe first that the estimates provided by Qknn are larger than those provided by the other algorithms, meaning that Qknn outperforms the other algorithms. The second best algorithm is ClassifPI, while Hybrid-Now performs poorly and clearly suffers from instability, due to the discontinuity of the running rewards w.r.t. the control variable.

Finally, Figures 6.7, 6.8, 6.9 provide the optimal decisions w.r.t. (P, C) at times 5, 10, 15, 20, 25, 29 estimated respectively by the Qknn, ClassifPI and Hybrid-Now algorithms. As expected, one can observe on each plot that the optimal strategy is to inject gas when the price is low, to sell gas when the price is high, and to make sure to have a volume of gas greater than c_0 in the cave when the terminal time is getting closer to minimize the terminal cost.

Let us now comment on the implementation of the algorithms:

- *Qknn:* Table 6.2 shows that once again, due to the low-dimensionality of the problem, Qknn provides the best value function estimates. The estimated optimal strategies, shown on Figure 6.7, are very good estimates of the theoretical ones. The three decision regions on Figure 6.7 are natural and easy to interpret: basically it is optimal to sell when the price is high, and to buy when it is low. However, a closer look reveals that the waiting region (where it is optimal to do nothing) has an unusual triangular-based shape, due essentially to the discreteness of the space on which the C component of the state space takes its values. We expect this shape to be very hard to reproduce with the DNN-based algorithms proposed in Section 6.2.

Table 6.2 – $V(0, P_0, C_0)$ estimates for different values of a_{in}, using the optimal strategy provided by the ClassifPI , Hybrid-Now and Qknn algorithms, with $a_{out} = 0.25$, $P_0 = 4$ and $C_0 = 4$.

a_{in}	Hybrid-Now	ClassifPI	Qknn	$\alpha = 0$
0.06	-0.99	-0.71	-0.66	-1.20
0.10	-0.70	-0.38	-0.34	-1.20
0.20	-0.21	0.01	0.12	-1.20
0.30	-0.10	0.37	0.37	-1.20
0.40	0.10	0.51	0.69	-1.20

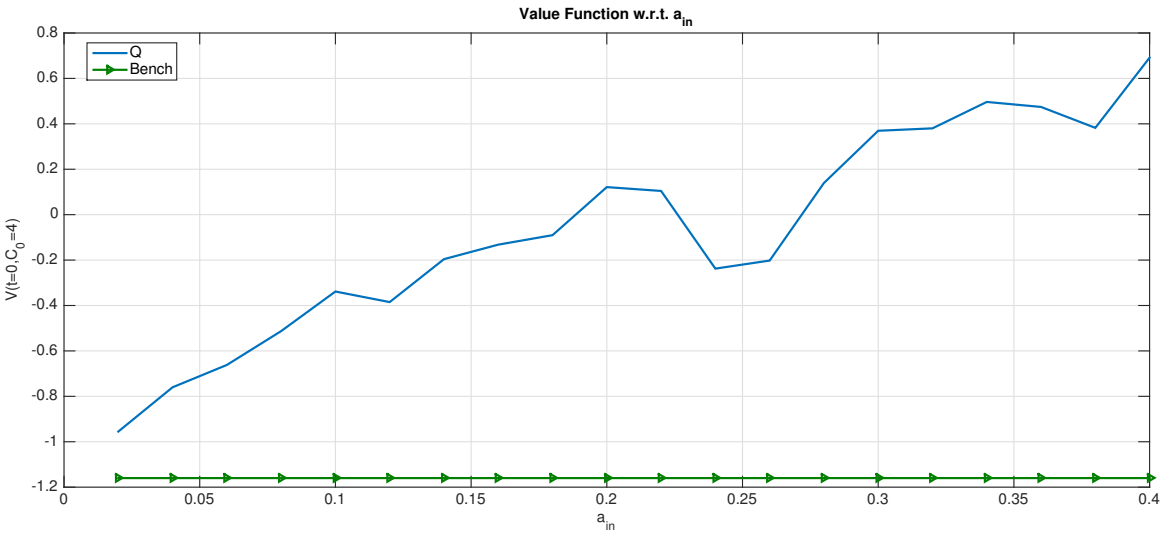


Figure 6.6 – Estimate of the value function at time 0 w.r.t. a_{in}, when the agent follows the optimal strategy estimated by Qknn, by running a forward Monte Carlo with a sample of size 100,000 (blue). We also plotted the cost functional associated with the naive passive strategy $\alpha = 0$ (Bench). See that for small values of a_{in} such as 0.06, doing nothing is a reasonable strategy. Observe also that the value function is not monotone w.r.t. a_{in} which is due to the dynamics of C (6.3.7).

- *ClassifPI:* As shown on Figure 6.8, the ClassifPI algorithm manages to provide accurate estimates for the optimal controls at time $n = 0, \dots, N-1$. However, the latter is not able to catch the particular triangular-based shape of the waiting region, which explains why Qknn performs better.
- *Hybrid-Now:* As shown on Figure 6.9, Hybrid-Now only manages to provide relatively poor estimates, compared to ClassifPI and Qknn, of the three different regions at time $n = 0, \dots, N-1$. In particular, the regions suffer from instability.

We end this paragraph by providing some implementation details for the different algorithms we tested.

- *Qknn:* We used the extension of Algorithm 6.5 introduced in the paragraph "semi-linear interpolation" of the Section 3.2.2. in [Bal+19] and used projection of each state on its k=2-nearest neighbors to get an estimate of the value function which is continuous w.r.t. the control variable at each time $n = 0, \dots, N-1$. The optimal control is computed at each point of the grids using the Brent algorithm, which is a deterministic function optimizer already implemented in Python[4].

[4]We could have chosen other algorithms to optimize the Q-value, but, in our tests, Brent was faster than the other choices that we tried, such as GoldenSearch, and always provided accurate estimates of the optimal controls.

- *Implementation details for the neural network-based algorithms:* We use neural networks with two hidden layers, ELU activation functions[5] and $20+20$ neurons . The output layer contains 3 neurons with softmax activation function for the ClassifPI algorithm and no activation function for the Hybrid-Now one. We use a training set of size M=60,000 at each time step. Note that given the expression of the terminal cost, the ReLU activation functions (Rectified Linear Units) could have been deemed a better choice to capture the shape of the value functions, but our tests revealed that ELU activation functions provide better results. At time $n = 0, \dots, N-1$, we took $\mu_n = \mathcal{U}(C_{min}, C_{max})$ as training measure.

  We did not use the pre-train trick discussed in Section 6.2.2.3, which explains the instability in the decisions that can be observed in Figure 6.9.

The main conclusion of our numerical comparisons on this energy storage example is that ClassifPI, the DNN-based classification algorithm designed for stochastic control problems with discrete control space, appears to be more accurate than the more general Hybrid-Now. Nevertheless, ClassifPI was not able to capture the unusual triangle-based shape of the optimal control as well as Qknn did.

6.3.4 Microgrid management

Finally, we consider a discrete-time model for power microgrid inspired by the continuous-time models developed in [Hey+18] and [JP15]; see also [Ala+19]. The microgrid consists of a photovoltaic (PV) power plant, a diesel generator and a battery energy storage system (BES), hence using a mix of fuel and renewable energy sources. These generation units are decentralized, i.e., installed at a rather small scale (a few kW power), and physically close to electricity consumers. The PV produces electricity from solar panels with a generation pattern $(P_n)_n$ depending on the weather conditions. The diesel generator has two modes: on and off. Turning it on consumes fuel, and produces an amount of power α_n. The BES can store energy for later use but has limited capacity and power. The aim of the microgrid management is to find the optimal planning that meets the power demand, denoted by $(D_n)_n$, while minimizing the operational costs due to the diesel generator. We denote by

$$R_n \quad = \quad D_n - P_n,$$

the residual demand of power: when $R_n > 0$, one should provide power through diesel or battery, and when $R_n < 0$, one can store the surplus power in the battery.

The optimal control problem over a fixed horizon N is formulated as follows. At any time $n = 0, \dots, N-1$, the microgrid manager decides the power production of the diesel generator, either by turning it off: $\alpha_n = 0$, or by turning it on, hence generating a power α_n valued in $[A_{min}, A_{max}]$ with $0 < A_{min} < A_{max} < \infty$. There is a fixed cost $\kappa > 0$ associated with switching from the on/off mode to the other one off/on, and we denote by M_n^α the mode valued in $\{0 = \text{off}, 1 = \text{on}\}$ of the generator right before time n, i.e., $M_{n+1}^\alpha = 1_{\alpha_n \neq 0}$.

When the diesel generator and renewable provide a surplus of power, the excess can be stored into the battery (up to its limited capacity) for later use, and in case of power insufficiency, the battery is discharged for satisfying the power demand. The input power process $\mathcal{I}^\alpha$ for charging the battery is then given by

$$\mathcal{I}_n^\alpha \quad = \quad (\alpha_n - R_n)_+ \wedge (C_{max} - C_n^\alpha),$$

where C_{max} is the maximum capacity of the battery with current charge C^α, while the output power process O^α for discharging the battery is given by

$$O_n^\alpha \quad = \quad (R_n - \alpha_n)_+ \wedge C_n^\alpha.$$

[5] The Exponential Linear Unit (ELU) activation function is defined as $x \mapsto \begin{cases} \exp(x) - 1 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$.

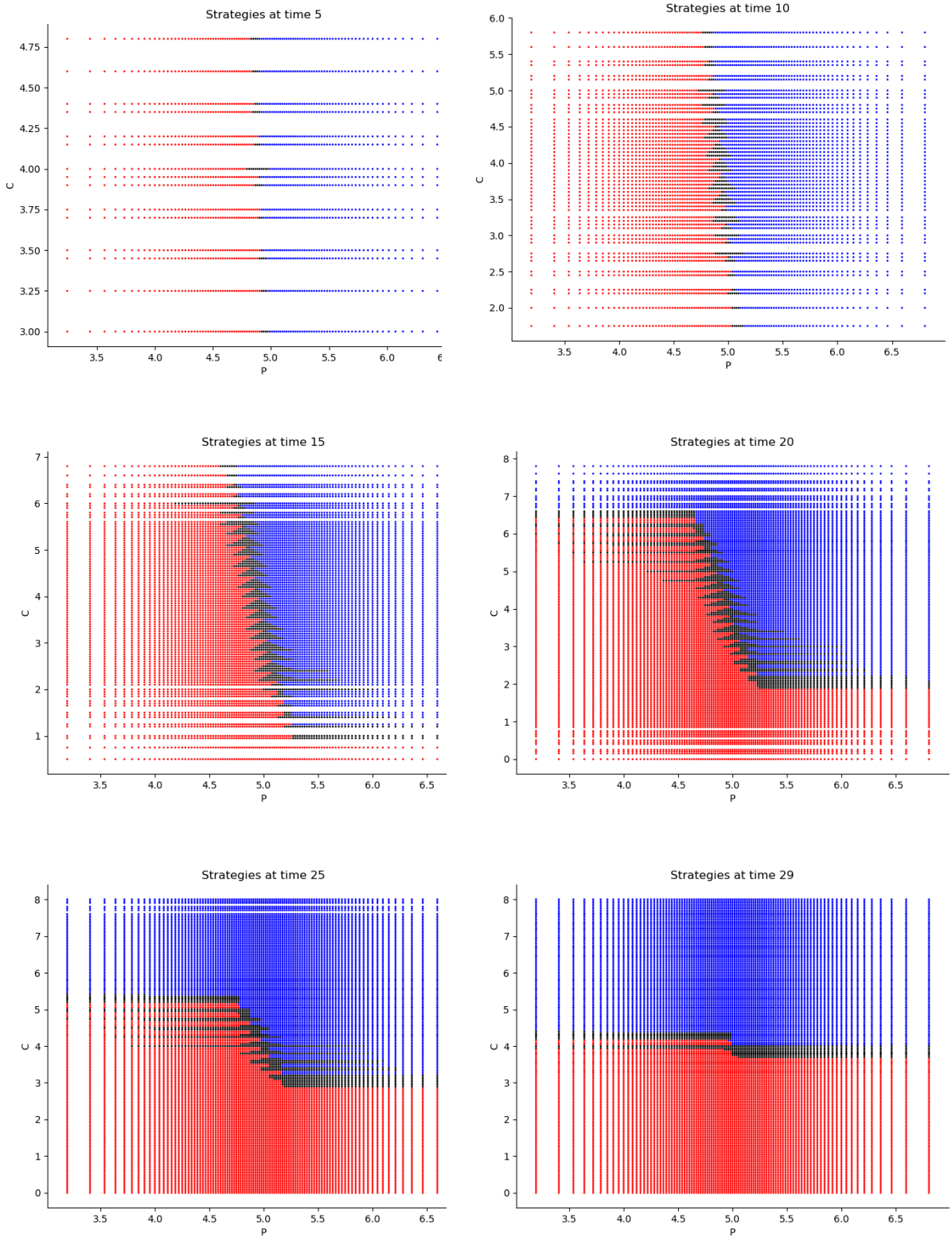


Figure 6.7 – Estimated optimal decisions at times 5, 10, 15, 20, 25, 29 w.r.t. (P,C) for the energy storage valuation problem using **Qknn**. Injection (a=-1) in red, store (a=0) in black and withdraw (a=1) in blue.

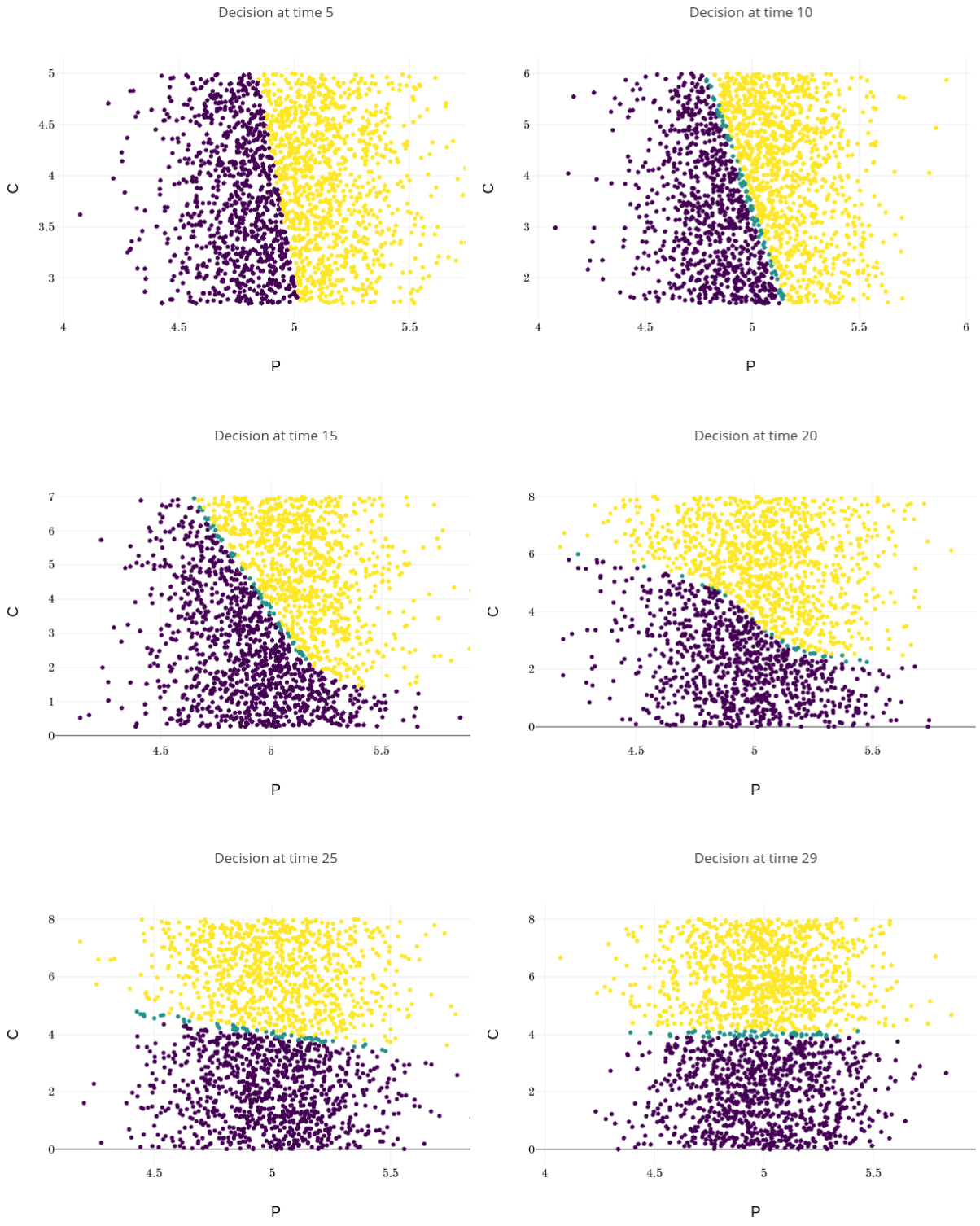


Figure 6.8 – Estimated optimal decisions at times 5, 10, 15, 20, 25, 29 w.r.t. (P,C) for the energy storage valuation problem using **ClassifPI**. Injection (a=-1) in purple, store (a=0) in blue and withdraw (a=1) in yellow.

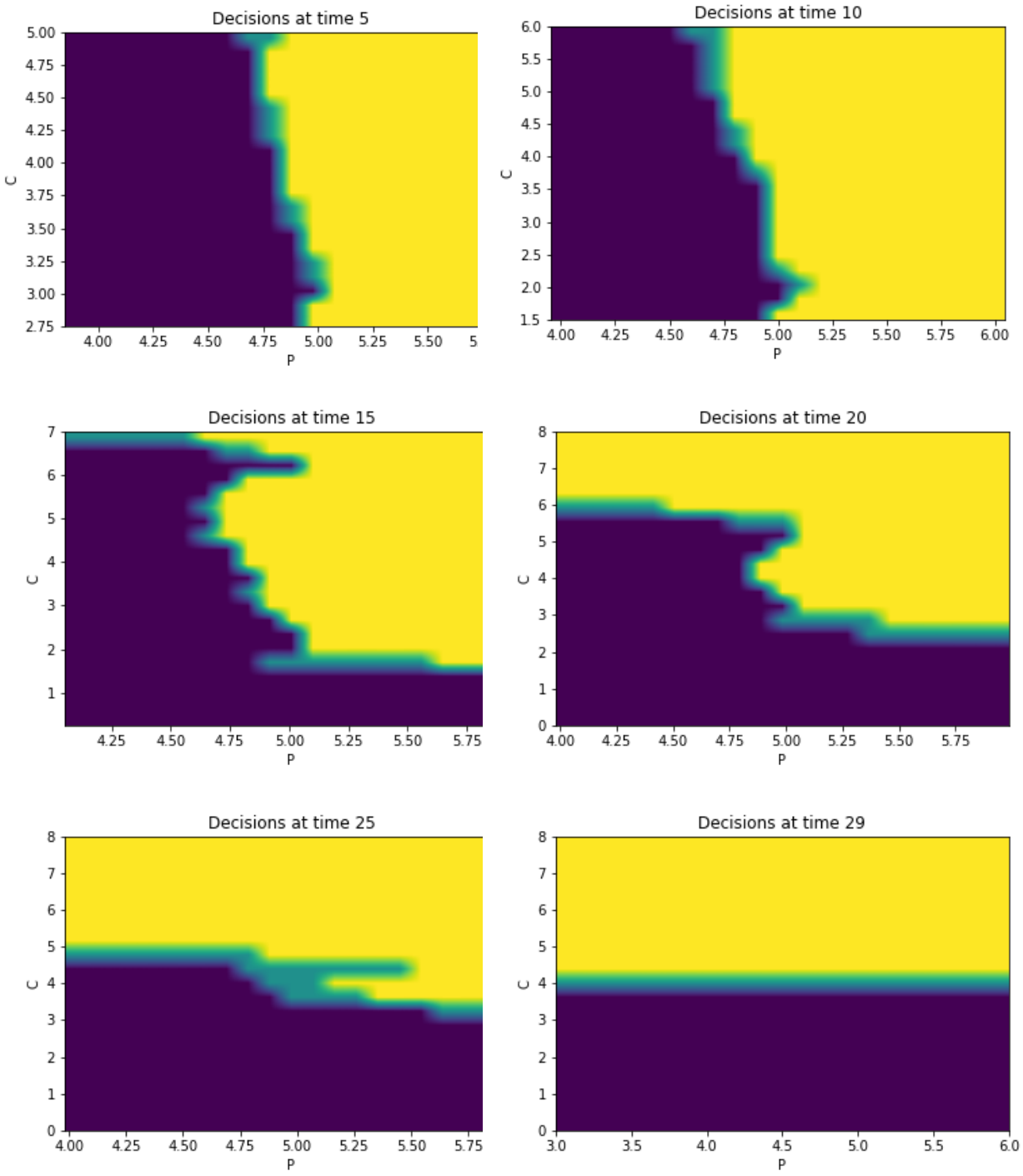


Figure 6.9 – Estimated optimal decisions at times 5, 10, 15, 20, 25, 29 w.r.t. (P,C) for the energy storage valuation problem using **Hybrid-Now**. Injection (a=-1) in purple, store (a=0) in blue and withdraw (a=1) in yellow. Observe the instability in the decisions which come from the fact that we did not pre-train the neural networks (see Section 6.2.2.3)

Here, we denote $p_+ = \max(p, 0)$. Assuming for simplicity that the battery is fully efficient, the capacity charge $(C_n^\alpha)_n$ of the BES, valued in $[0, C_{max}]$, evolves according to the dynamics

$$C_{n+1}^\alpha \quad = \quad C_n^\alpha + \mathcal{I}_n^\alpha - O_n^\alpha.$$

The imbalance process defined by

$$S_n^\alpha \quad = \quad R_n - \alpha_n + \mathcal{I}_n^\alpha - O_n^\alpha$$

represents how well we are doing for satisfying electricity supply: the ideal situation occurs when $S_n^\alpha = 0$, i.e., perfect balance between demand and generation. When $S_n^\alpha > 0$, this means that demand is not satisfied, i.e., there is missing power in the microgrid, and when $S_n^\alpha < 0$, there is an excess of electricity. In order to ensure that there is no missing power, we impose the following constraint on the admissible control:

$$S_n^\alpha \ \leq \ 0, \quad \text{i.e.} \quad \alpha_n \ \geq \ R_n - C_n^\alpha,$$

but penalize the excess of electricity when $S_n^\alpha < 0$ with a proportional cost $Q^- > 0$. We model the residual demand as a mean-reverting process:

$$R_{n+1} \quad = \quad \bar{R}(1-\varrho) + \varrho R_n + \varepsilon_{n+1},$$

where $(\varepsilon_n)_n$ are i.i.d., $\bar{R} \in \mathbb{R}$, and $\varrho < 1$. The goal of the microgrid manager is to find the optimal (admissible) decision α that minimizes the functional cost

$$J(\alpha) \quad = \quad \mathbb{E}\left[\sum_{n=0}^{N-1} \ell(\alpha_n) + \kappa 1_{\{M_n^\alpha \neq M_{n+1}^\alpha\}} + Q^-(S_n^\alpha)_-\right],$$

where $\ell(.)$ is the cost function for fuel consumption: $\ell(0) = 0$, and e.g. $\ell(a) = Ka^\gamma$, with $K > 0$, $\gamma > 0$. This stochastic control problem fits into the 3-dimensional framework of Section 5.1 (see also Remark 6.2.4) with control α valued in $\mathbb{A} = \{0\} \times [A_{min}, A_{max}]$, $X^\alpha = (C^\alpha, M^\alpha, R)$, noise ε_{n+1}, starting from an initial value $(C_0^\alpha, M_0^\alpha, R_0) = (c_0, 0, r_0)$ on the state space $[0, C_{max}] \times \{0, 1\} \times \mathbb{R}$, with dynamics function

$$F(x, a, e) \quad = \quad \begin{pmatrix} F^1(x,a) := c + (a-r)_+ \wedge (C_{max} - c) - (r-a)_+ \wedge c \\ 1_{a \neq 0} \\ \bar{R}(1-\varrho) + \varrho r + e \end{pmatrix},$$

for $x = (c, m, r) \in [0, C_{max}] \times \{0, 1\} \times \mathbb{R}$, $a \in \{0\} \times [A_{min}, A_{max}]$, $e \in \mathbb{R}$, running cost function

$$\begin{aligned} f(x, a) \quad &= \quad \ell(a) + \kappa 1_{m=1_{a=0}} + Q^- S(x, a)_-, \\ S(x, a) \quad &= \quad r - a + (a-r)_+ \wedge (C_{max} - c) - (r-a)_+ \wedge c, \end{aligned}$$

zero terminal cost $g = 0$, and control constraint

$$\begin{aligned} \mathbb{A}_n(x) \quad &= \quad \Big\{a \in \{0\} \times [A_{min}, A_{max}] : S(x, a) \leq 0\Big\} \\ &= \quad \Big\{a \in \{0\} \times [A_{min}, A_{max}] : r - c \leq a\Big\}. \end{aligned}$$

Remark 6.3.2. The state/space constraint is managed in our NN-based algorithm by introducing a penalty function into the running cost (see Remark 6.2.4): $f(x, a) \leftarrow f(x, a) + L(x, a)$

$$L(x, a) \quad = \quad Q^+\Big(r - c - a\Big)_+$$

with large Q^+ taken much larger than Q^-. Doing so, the NN-based estimate of the optimal control learns not to take any forbidden decision. □

The control space $\{0\} \cup [A_{\min}, A_{\max}]$ is a mix between a discrete space and a continuous space, which is challenging for algorithms with neural networks. We actually use a mixture of classification and standard DNN for the control: $(p_0(x;\theta), \pi(x;\beta))$ valued in $[0,1] \times [A_{min}, A_{max}]$, where $p_0(x;\theta)$ is the probability of turning off in state x, and $\pi(x;\beta)$ is the amount of power when turning on with probability $1 - p_0(x;\theta)$. In other words,

$$X_{n+1} = \begin{cases} F(X_n, 0, \varepsilon_{n+1}) & \text{with probability } p_0(X_n;\theta_n) \\ F(X_n, \pi(X_n;\beta_n), \varepsilon_{n+1}) & \text{with probability } 1 - p_0(X_n;\theta_n) \end{cases}$$

The pseudo-code of algorithm, designed for this problem, is written in Algorithm 6.6, and we refer to the latter as ClassifHybrid, in the sequel. Note in particular that it is an Hybrid version of ClassifPI.

Algorithm 6.6 ClassifHybrid

Input: the training distributions $(\mu_n)_{n=0}^{N-1}$;

1: Set $\hat{V}_N = g$;
2: **for** $n = N-1, \dots, 0$ **do**
3: Compute

$$(\hat{\beta}_n^0, \hat{\beta}_n^1) \in \underset{\beta^0,\beta^1}{\operatorname{argmax}}\, \mathbb{E}\Big[p_0(X_n;\beta^0) \Big[f(X_n,0) + \hat{V}_{n+1}\Big(f(\hat{X}_{n+1}^0)\Big)\Big] + (1 - p_0(X_n;\beta^0)) \Big[f(X_n, \pi(X_n;\beta^1)) + \hat{V}_{n+1}\Big(\hat{X}_{n+1}^{1,\beta^1}\Big)\Big]\Big],$$

where $X_n \rightsquigarrow \mu_n$, $\hat{X}_{n+1}^0 = F(X_n, 0, \varepsilon_{n+1})$, and $\hat{X}_{n+1}^{1,\beta^1} = F(X_n, \pi(X_n;\beta^1), \varepsilon_{n+1})$

4: Compute

$$\hat{\theta}_n \in \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}\Big[p_0\big(X_n;\hat{\beta}_n^0\big) \Big[f(X_n,0) + \hat{V}_{n+1}\Big(f(\hat{X}_{n+1}^0)\Big) - \Phi(.;\theta)\Big]^2 + \big(1 - p_0\big(X_n;\hat{\beta}_n^0\big)\big) \Big[f(X_n, \pi(X_n;\beta_n^1)) + \hat{V}_{n+1}\Big(\hat{X}_{n+1}^{1,\hat{\beta}_n^1}\Big) - \Phi(.;\theta)\Big]^2\Big];$$

5: Set $\hat{V}_n = \Phi(.;\hat{\theta}_n)$; ▷ $\hat{V}_n$ is the estimate of the value function at time n
6: **end for**

Output:
– estimate of the optimal strategy $(\hat{a}_n)_{n=0}^{N-1}$;
– estimate of the value function $(\hat{V}_n)_{n=0}^{N-1}$;

Test We took the following parameters to compare Qknn and ClassifHybrid:

N	= 30 or 200,	$\bar{R}$	= 0.1,	ϱ	= 0.9,	σ	= 0.2,
$C_{\min}$	= 0,	$C_{\max}$	= 1 or 4,	C_0	= 0,	K	= 2,
γ	= 2,	κ	= 0.2,	Q^-	= 10,	R_0	= 0.1,
$A_{\min}$	= 0.05,	$A_{\max}$	= 10	Q^+	= 1000.		

Results Figure 6.10 shows the Qknn-estimated optimal decisions to take at times $n = 1, 10, 28$ in the cases where $m = M_n = 0$ and $m = M_n = 1$. If the generator is off at time n, i.e. $m = 0$, the blue curve separates the region where it is optimal to keep it off and the one where it is optimal to generate power. If the generator is on at time n, i.e. $m = 1$, the blue curve separates the region where it is optimal to turn it off and the one where it is optimal to generate power. A colorscale is

Table 6.3 – Estimates of the value function at time 0 and state ($C_0 = 0, M_0 = 0, R_0 = 0.1$), for $N = 30$ and $C_{max} = 1$, using Qknn and ClassifHybrid algorithms. Note that ClassifHybrid got a better result than Qknn on this problem.

	Mean	std
ClassifHybrid	33.34	0.31
Qknn	35.37	0.34

available on the right to inform how much power it is optimal to generate in both cases. Observe that the optimal decisions are quite intuitive: for example, if the demand is high and the battery is empty, then it is optimal to generate a lot of energy. Moreover, it is optimal to turn the generator off if the demand is negative or if the battery is charged enough to meet the demand.
We plot in Figure 6.11 the estimated optimal decisions at times $n = 1, 10, 28$, using the Hybrid-Now algorithm, with $N = 30$ time steps. See that the decisions are similar to the ones given using Qknn. Note that the plots in Figure 6.10 and 6.11 look much better than the ones obtained in [Ala+19] in which algorithms based on regress-now or regress-later are used (see in particular Figure 4 in [Ala+19]); hence Qknn and ClassifHybrid are more stable than the algorithms proposed in [Ala+19].

We report in Table 6.3 the result for the estimates of the value function with N=30 time steps, obtained by running 10 times a forward Monte Carlo with 10,000 simulations using the optimal strategy estimated using Qknn and ClassifHybrid algorithms. Observe that Hybrid-Now performs better than Qknn. However, Qknn run in less than a minute whereas Hybrid-Now needed seven minutes to run.
We also report in Table 6.4 the result for the estimatest of the value function with N=200 time steps, obtained by running 20 times a forward Monte Carlo with 10,000 simulations using the Qknn-estimated optimal strategy.

Figure 6.12 shows two simulations of (C, M, R) controlled using the Qknn-estimated optimal strategy, where $N = 200$ has been chosen. Observe in particular the natural behavior of the Qknn-decisions which consists in turning the generator on when the demand cannot be met by the battery, and turn it off when the demand is negative or when the battery is charged enough to meet the demand. Note that the plots are similar to the ones plotted in Figure 9 of [Ala+19].

Comments on Qknn: Note that there is no need to use a penalization method with the Qknn-algorithm to constrain the control to stay in $\mathbb{A}_n(x)$, where x is the state at time n, since, for all state x, we can simply search for the optimal control associated in $\mathbb{A}_n(x)$, using e.g. the Brent algorithm. For $n = 0, \dots, N-1$, we took the training set as follows: $\Gamma_n := \Gamma_C \times \{0, 1\} \times \Gamma_R^n$; where $\Gamma_C := \{C_{min} + \frac{i}{50}(C_{max} - C_{min}), i = 0, \dots, 50\}$, $\Gamma_R^n := \rho^n R_0 + \sigma \frac{1-\rho^n}{1-\rho} \Gamma_1$ and where Γ_1 is the optimal grid for the quantization of $\mathcal{N}(0, 1)$, available in http://www.quantize.maths-fi.com, with 51 points. This choice of training points for the C component corresponds to the exploration procedure discussed in Remark 6.2.1, whereas we chose the best grid with 51 points for the (uncontrolled) R component.
Comments on ClassifHybrid: We took 100 mini-batches of size 300 and took 100 epochs to run the algorithm. We chose the following training distribution at time n: $\mu_n = \mathcal{U}(C_{min}, C_{max}) \times \mathcal{U}(\{0, 1\}) \times \mathbb{P}_{R_n}$, where $\mathbb{P}_{R_n}$ is the law of the (uncontrolled) residual demand at time t_n. Note that such a choice of training distribution means that we want to explore all the available states for the controlled components of the controlled process (C, M, R) in order to learn the optimal strategy globally.

The microgrid management problem is very challenging for our algorithms because the control space $\{0\} \cup [a_{\min}, a_{\max}]$ is a mix of discrete and continuous space, moreover the choice of the optimal control is subject to constraints. We designed ClassifHybrid, an Hybrid version of ClassifPI, to solve this problem. ClassifHybrid provided very good estimates and actually managed to perform better than Qknn.

Table 6.4 – Qknn-estimates of the value function at time 0 and state ($C_0 = 0, M_0 = 0, R_0 = 0.1$), for $N = 200$.

Mean	Standard Deviation
231.8	1.2

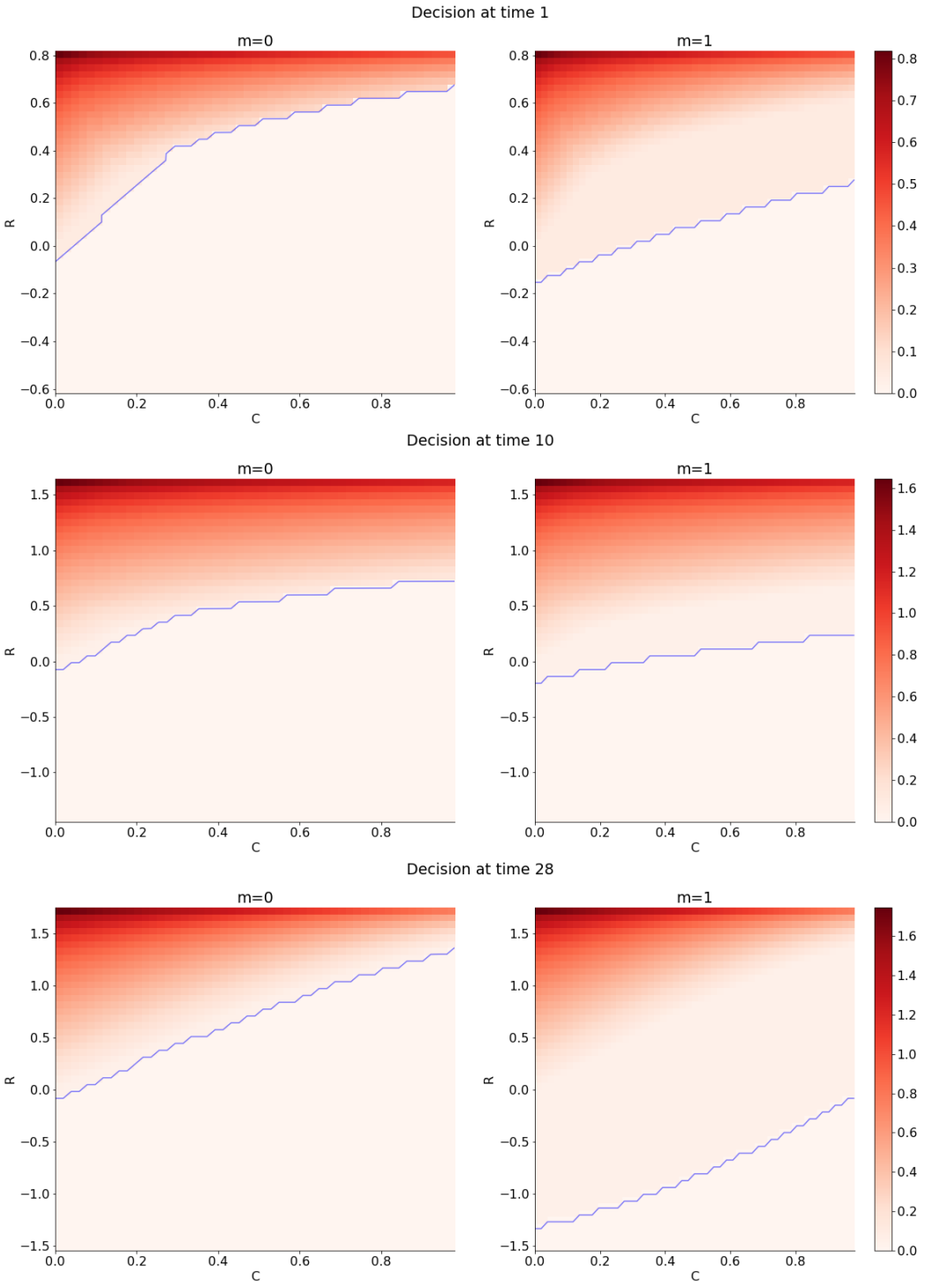


Figure 6.10 – Estimated optimal decisions at time 1, 10 and 28, using Qknn, with $N = 30$ time steps. The region under the blue line is the one where it is optimal to turn the generator off if m=1 (i.e. the generator was on at time n-1), or keep it off if $m = 0$ (i.e. the generator was off at time n-1).

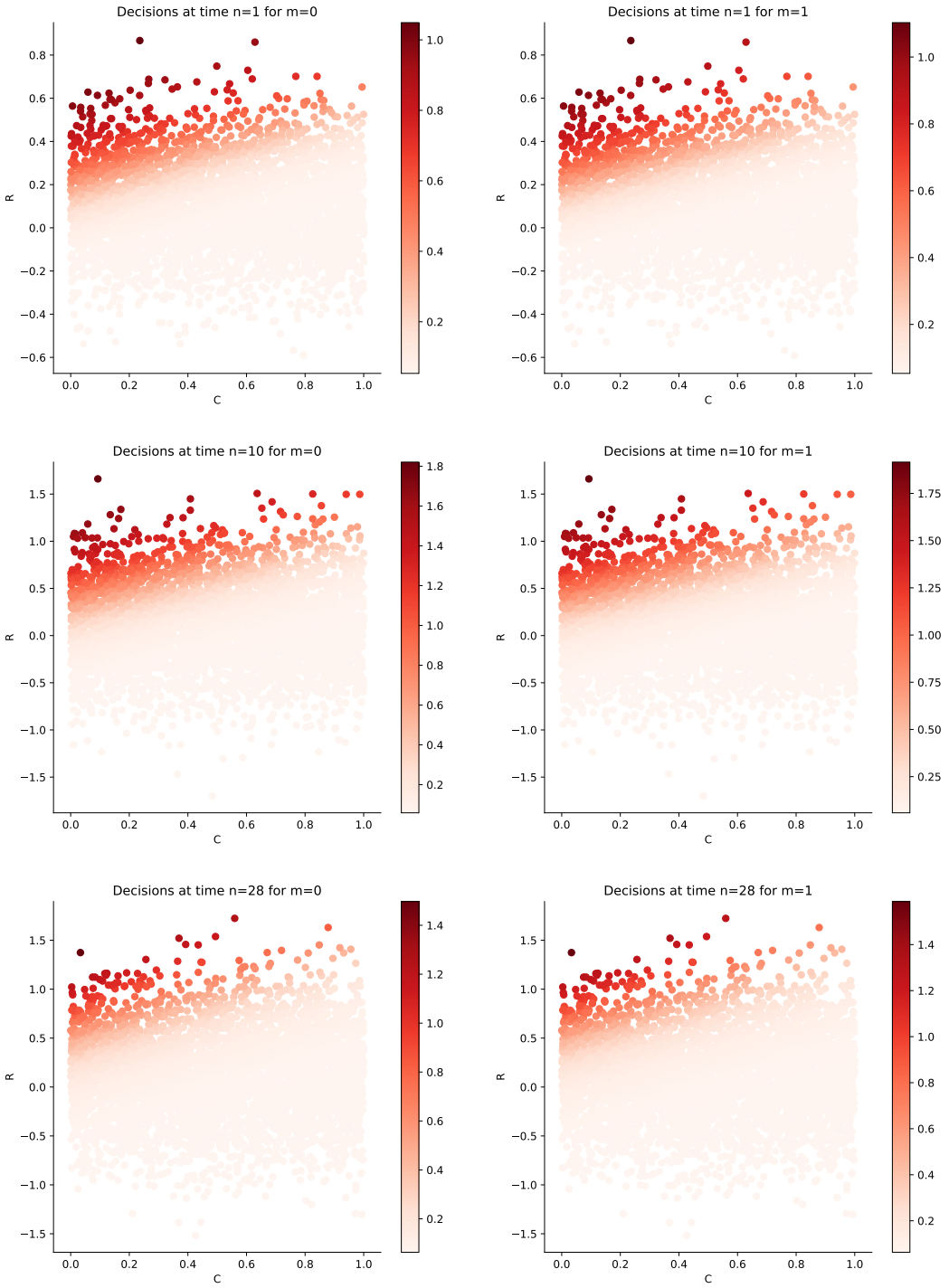


Figure 6.11 – Estimated optimal decisions at time 1, 10 and 28, using ClassifHybrid, with $N = 30$ time steps.

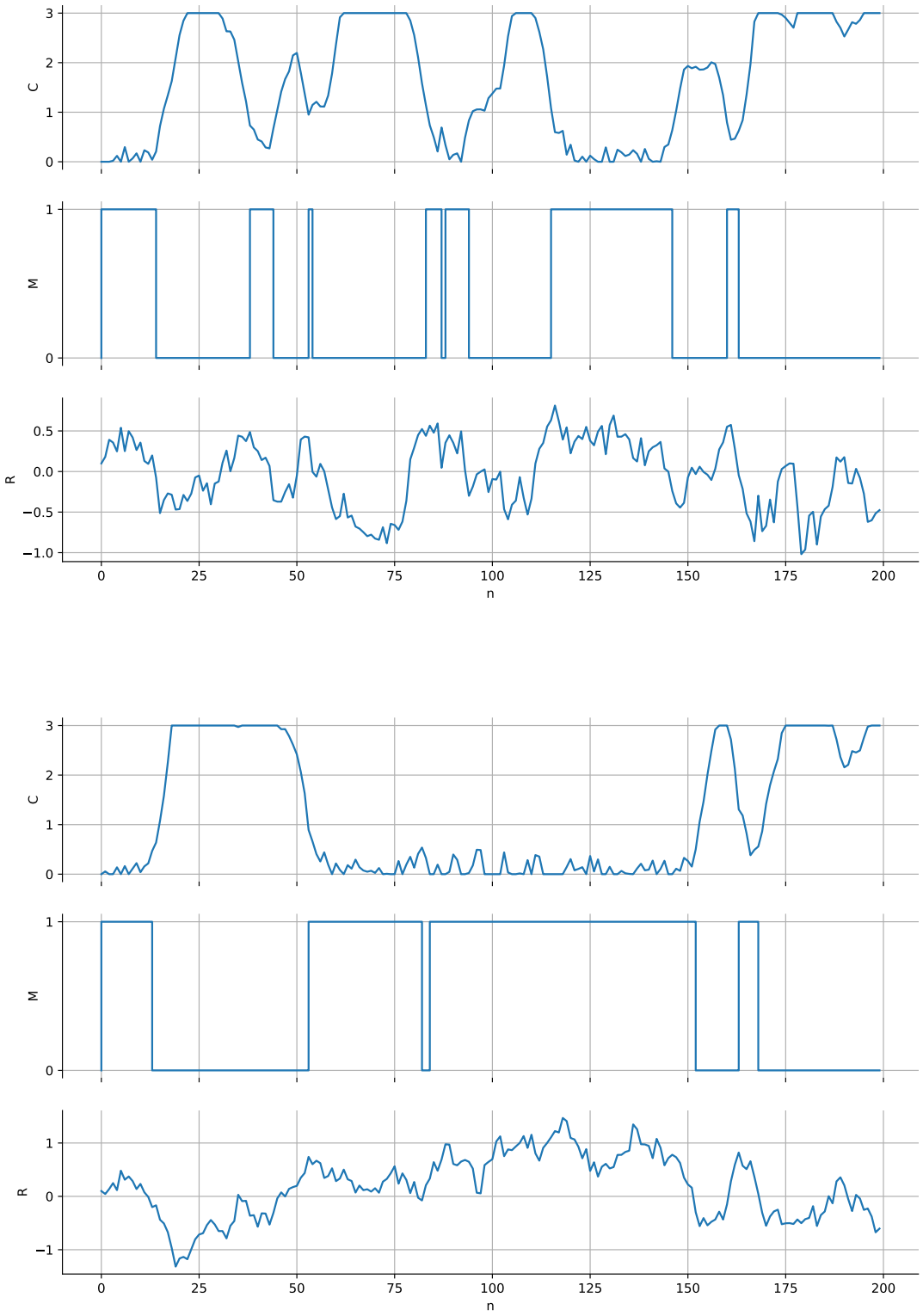


Figure 6.12 – Two simulations of (C, M, R) optimally controlled using Qknn, with $N = 200$ and $C_{\max} = 4$.

6.4 Discussion and conclusion

Our proposed algorithms are well-designed and provide accurate estimates of optimal control and value function associated with various high-dimensional control problems. Also, when tested on low-dimensional problems, they performed as well as the Monte Carlo-based or quantization-based methods, which have shown their efficiency in low dimension, see e.g. [Bal+19] and [Ala+19].

The presented algorithms suffer from a rather high time-consuming cost due to the expensive training of $2(N-1)$ neural networks to learn the value functions and optimal controls at times $n = 0, \dots, N-1$. However, the agent can easily alleviate the computation time. A first trick consists in reducing the number of neural networks by partially or totally ignoring the dynamic programming principle (DPP), as it has been done e.g. in [EHJ17]. The use of one unique Recurrent Neural Networks (RNN) (in the case where the DPP is totally ignored) or a few of them (in the partial-ignored case) can also be considered to learn the optimal controls, either all at the same time (first case), or group by group in a backward way (second case). We refer to [CMW18] for algorithms in this spirit. Another trick consists in learning faster the value functions and optimal controls at times $n = 0, \dots, N-1$ by pre-training the neural networks. The way to proceed in that direction is to initialize at time n the weights and bias of the value function estimator $\hat{V}_n$ to the ones of $\hat{V}_{n+1}$. We then rely on the continuity of the value function w.r.t. the time n to expect that the weights will not change much from time n to $n+1$, hence do not need to be train for a long time anymore. Another benefit from the pre-training task is to get the stability of the estimates w.r.t. the time, which is also a very pleasant feature.

Part III

Deep learning for BSDEs and PDEs

Chapter 7

Some machine learning schemes for high-dimensional nonlinear PDEs[1]

Abstract We propose new machine learning schemes for solving high dimensional nonlinear partial differential equations (PDEs). Relying on the classical backward stochastic differential equation (BSDE) representation of PDEs, our algorithms estimate simultaneously the solution and its gradient by deep neural networks. These approximations are performed at each time step from the minimization of loss functions defined recursively by backward induction. The methodology is extended to variational inequalities arising in optimal stopping problems. We analyze the convergence of the deep learning schemes and provide error estimates in terms of the universal approximation of neural networks. Numerical results show that our algorithms give very good results till dimension 50 (and certainly above), for both PDEs and variational inequalities problems. For the PDEs resolution, our results are very similar to those obtained by the recent method in [EHJ17] when the latter converges to the right solution or does not diverge. Numerical tests indicate that the proposed methods are not stuck in poor local minima as it can be the case with the algorithm designed in [EHJ17], and no divergence is experienced. The only limitation seems to be due to the inability of the considered deep neural networks to represent a solution with a too complex structure in high dimension.

Key words: Deep neural networks, nonlinear PDEs in high dimension, optimal stopping problem, backward stochastic differential equations.

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

Contents

[1] This Chapter is based on a paper written in collaboration with Huyên Pham and Xavier Warin.

7.1 Introduction

This paper is devoted to the resolution in high dimension of nonlinear parabolic partial differential equations (PDEs) of the form

$$\begin{cases} \partial_t u + \mathcal{L}u + f(.,.,u,\sigma^\intercal D_x u) = 0, & \text{on } [0,T)\times\mathbb{R}^d, \\ u(T,.) = g, & \text{on } \mathbb{R}^d, \end{cases} \tag{7.1.1}$$

and a second-order generator $\mathcal{L}$ defined by

$$\mathcal{L}u := \frac{1}{2}\,\mathrm{Tr}\left(\sigma\sigma^\intercal D_x^2 u\right) + \mu.D_x u.$$

Here 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 $\mathcal{L}$ is the generator associated to the forward diffusion process:

$$\mathcal{X}_t = x_0 + \int_0^t \mu(s,\mathcal{X}_s)\,\mathrm{d}s + \int_0^t \sigma(s,\mathcal{X}_s)\,\mathrm{d}W_s, \quad 0\le t\le T, \tag{7.1.2}$$

with 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.

Due to the so called "curse of dimensionality", the resolution of nonlinear PDEs in high dimension has always been a challenge for scientists. Until recently, only the BSDE (Backward Stochastic Differential Equation) approach first developed in [PP90] was available to tackle this problem: using the time discretization scheme proposed in [BT04], some effective algorithms based on regressions manage to solve non linear PDEs in dimension above 4 (see [GLW05; LGW06]). However this approach is still not implementable in dimension above 6 or 7 : the number of basis functions used for the regression still explodes with the dimension.

Quite recently some new methods have been developed for this problem, and three kind of methodologies have emerged:

- Some are based on the Feyman-Kac representation of the PDE. Branching techniques [HL+16] have been studied and shown to be convergent but only for small maturities and some small nonlinearities. Some effective techniques based on nesting Monte Carlo have been studied in [War18b; War18a]: the convergence is proved for semi-linear equations. Still based on this Feyman-Kac representation some machine learning techniques permitting to solve a fixed point problem have been used recently in [CWNMW18]: numerical results show that it is efficient and some partial demonstrations justify why it is effective.

- Another class of methods is based on the BSDE approach and the curse of dimensionality issue is partially avoided by using some machine learning techniques. [HJE17; EHJ17] propose a deep learning based technique called *Deep BSDE*. Based on an Euler discretization of the forward underlying SDE $\mathcal{X}_t$, the idea is to view the BSDE as a forward SDE, and the algorithm tries to learn the values u and $z = \sigma^\intercal Du$ at each time step of the Euler scheme by minimizing a global loss function between the forward simulation of u till maturity T and the target $g(\mathcal{X}_T)$.

- At last, using some machine learning representation of the solution, [SS18] proposes to use the automatic numerical differentiation of the solution to solve the PDE on a finite domain.

Like the second methodology, our approach relies on BSDE representation of the PDE and deep learning approximations: we first discretize the BSDE associated to the PDE by an Euler scheme, but in contrast with [EHJ17], we adopt a classical backward resolution technique. On each time step, we propose to use some machine learning techniques to estimate simultaneously the solution and its gradient by minimizing a loss function defined recursively by backward induction, and solving this local problem by a stochastic gradient algorithm. Two different schemes are designed to deal with the local problems:

(1) The first one tries the estimate the solution and its gradient by a neural network.

(2) The second one tries only to approximate the solution by a neural network while its gradient is estimated directly with some numerical differentiation techniques.

The proposed methodology is then extended to solve some variational inequalities, i.e., free boundary problems related to optimal stopping problems.

Convergence analysis of the two schemes for PDEs and variational inequalities is provided and shows that the approximation error goes to zero as we increase the number of time steps and the number of neurons/layers whenever the gradient descent method used to solve the local problems is not trapped in a local minimum.

In the last part of the paper, we test our algorithms on different examples. When the solution is easy to represent by a neural network, we can solve the problem in quite high dimension (at least 50 in our numerical tests). We show that the proposed methodology is superior to the algorithm proposed in [HJE17] that often does not converge or is trapped in a local minimum far away from the true solution. We then show that when the solution has a very complex structure, we can still solve the problem but only in moderate dimension: the neural network used is not anymore able to represent the solution accurately in very high dimension. Finally, we illustrate numerically that the method is effective to solve some system of variational inequalities: we consider the problem of American options and show that it can be solved very accurately in high dimension (we tested until 40).

The outline of the paper is organized as follows. In Section 7.2, we give a brief and useful reminder for neural networks. We describe in Section 7.3 our two numerical schemes and compare with the algorithm in [HJE17]. Section 7.4 is devoted to the convergence analysis of our machine learning algorithms, and we present in Section 7.5 several numerical tests.

7.2 Neural networks as function approximators

Multilayer (also called deep) neural networks are designed to approximate unknown or large class of functions. In contrast to additive approximation theory with weighted sum over basis functions, e.g. polynomials, neural networks rely on the composition of simple functions, and appear to provide an efficient way to handle high-dimensional approximation problems, in particular thanks to the increase in computer power for finding the "optimal" parameters by (stochastic) gradient descent methods.

We shall consider feedforward (or artificial) neural networks, which represent the basic type of deep neural networks. Let us recall some notation and basic definitions that will be useful in our context. We fix the input dimension $d_0 = d$ (here the dimension of the state variable x), the output dimension d_1 (here $d_1 = 1$ for approximating the real-valued solution to the PDE, or $d_1 = d$ for approximating the vector-valued gradient function), the global number $L + 1 \in \mathbb{N} \setminus \{1, 2\}$ of layers 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 $\mathbb{R}^d$ to $\mathbb{R}^{d_1}$ defined as the composition

$$x \in \mathbb{R}^d \longmapsto A_L \circ \varrho \circ A_{L-1} \circ \ldots \circ \varrho \circ A_1(x) \in \mathbb{R}^{d_1}. \tag{7.2.1}$$

Here A_ℓ, $\ell = 1, \dots, L$ are affine transformations: A_1 maps from $\mathbb{R}^d$ to $\mathbb{R}^m$, $A_2, \dots, A_{L-1}$ 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., $\varrho(x_1, \dots, x_m) = (\varrho(x_1), \dots, \varrho(x_m))$. Standard examples of activation functions are the sigmoid, the ReLu, the Elu, tanh.

All these matrices $\mathcal{W}_\ell$ and vectors β_ℓ, $\ell = 1, \dots, L$, are the parameters of the neural network, and can be identified with an element $\theta \in \mathbb{R}^{N_m}$, where $N_m = \sum_{\ell=0}^{L-1} m_\ell(1 + m_{\ell+1}) = d(1+m) + m(1+m)(L-2) + m(1+d_1)$ is the number of parameters, where we fix d_0, d_1, L, but allow growing number m of hidden neurons. We denote by Θ_m the set of possible parameters: in the sequel, we shall consider either the case when there are no constraints on parameters, i.e., $\Theta_m = \mathbb{R}^{N_m}$, or when the total variation norm of the neural networks is smaller than γ_m, i.e.,

$$\Theta_m = \Theta_m^\gamma := \big\{\theta = (\mathcal{W}_\ell, \beta_\ell)_\ell : |\mathcal{W}_l| \le \gamma_m, \ \ \ell = 1, \dots, L\big\}, \quad \text{with } \gamma_m \nearrow \infty, \text{ as } m \to \infty.$$

We denote by $\Phi_m(.;\theta)$ the neural network function defined in (7.2.1), and by $\mathcal{NN}^\varrho_{d,d_1,L,m}(\Theta_m)$ the set of all such neural networks $\Phi_m(.;\theta)$ for $\theta \in \Theta_m$, and set

$$\mathcal{NN}^\varrho_{d,d_1,L} = \bigcup_{m\in\mathbb{N}} \mathcal{NN}^\varrho_{d,d_1,L,m}(\Theta_m) = \bigcup_{m\in\mathbb{N}} \mathcal{NN}^\varrho_{d,d_1,L,m}(\mathbb{R}^{N_m}),$$

as the class of all neural networks within a fixed structure given by d, d_1, L and ϱ.

The fundamental result of Hornick et al. [HSW89] justifies the use of neural networks as function approximators:

Universal approximation theorem (I): $\mathcal{NN}^\varrho_{d,d_1,L}$ is dense in $L^2(\nu)$ for any finite measure ν on $\mathbb{R}^d$, whenever ϱ is continuous and non-constant.

Moreover, we have a universal approximation result for the derivatives in the case of a single hidden layer, i.e. $L = 2$, and when the activation function is a smooth function, see [HSW90].

Universal approximation theorem (II): Assume that ϱ is a (non constant) C^k function. Then, $\mathcal{NN}^\varrho_{d,d_1,2}$ approximates any function and its derivatives up to order k, arbitrary well on any compact set of $\mathbb{R}^d$.

7.3 Deep learning-based schemes for semi-linear PDEs

The starting point for our probabilistic numerical schemes to the PDE (7.1.1) is the well-known (see [PP90]) nonlinear Feynman-Kac formula via the pair (Y, Z) of $\mathbb{F}$-adapted processes valued in $\mathbb{R} \times \mathbb{R}^d$, solution to the BSDE

$$Y_t = g(\mathcal{X}_T) + \int_t^T f(s, \mathcal{X}_s, Y_s, Z_s)\, \mathrm{d}s - \int_t^T Z_s^\intercal\, \mathrm{d}W_s, \quad 0 \le t \le T, \tag{7.3.1}$$

related to the solution u of (7.1.1) via

$$Y_t = u(t, \mathcal{X}_t), \quad 0 \le t \le T,$$

and when u is smooth:

$$Z_t = \sigma^\intercal(t, \mathcal{X}_t) D_x u(t, \mathcal{X}_t), \quad 0 \le t \le T.$$

7.3.1 The deep BSDE scheme of [HJE17]

The DBSDE algorithm proposed in [HJE17; EHJ17] starts from the BSDE representation (7.3.1) of the solution to (7.1.1), but rewritten in forward form as:

$$\begin{aligned} u(t,\mathcal{X}_t) = \; & u(0,x_0) - \int_0^t f(s,\mathcal{X}_s,u(s,\mathcal{X}_s),\sigma^\intercal(s,\mathcal{X}_s)D_x u(s,\mathcal{X}_s))\,\mathrm{d}s \\ & + \int_0^t D_x u(s,\mathcal{X}_s)^\intercal \sigma(s,\mathcal{X}_s)\,\mathrm{d}W_s, \qquad 0 \le t \le T. \end{aligned} \tag{7.3.2}$$

The forward process $\mathcal{X}$ in equation (7.1.2), when it is not simulatable, is numerically approximated by an Euler scheme $X = X^\pi$ on a time grid: $\pi = \{t_0 = 0 < t_1 < \ldots < t_N = T\}$, with modulus $|\pi| = \max_{i=0,\ldots,N-1} \Delta t_i$, $\Delta t_i := t_{i+1} - t_i$, and defined as

$$X_{t_{i+1}} \;=\; X_{t_i} + \mu(t_i,X_{t_i})\Delta t_i + \sigma(t_i,X_{t_i})\Delta W_{t_i}, \quad i = 0,\ldots,N-1,\; X_0 = x_0, \tag{7.3.3}$$

where we set $\Delta W_{t_i} := W_{t_{i+1}} - W_{t_i}$. To alleviate notations, we omit the dependence of $X = X^\pi$ on the time grid π as there is no ambiguity (recall that we use the notation $\mathcal{X}$ for the forward diffusion process). The approximation of equation (7.1.1) is then given formally from the Euler scheme associated to the forward representation (7.3.2) by

$$u(t_{i+1},X_{t_{i+1}}) \approx F(t_i,X_{t_i},u(t_i,X_{t_i}),\sigma^\intercal(t_i,X_{t_i})D_x u(t_i,X_{t_i}),\Delta t_i,\Delta W_{t_i}) \tag{7.3.4}$$

with

$$F(t,x,y,z,h,\Delta) := y - f(t,x,y,z)h + z^\intercal\Delta.$$

In [HJE17; EHJ17], the numerical approximation of $u(t_i,X_{t_i})$ is designed as follows: starting from an estimation $\mathcal{U}_0$ of $u(0,X_0)$, and then using at each time step t_i, $i = 0,\ldots,N-1$, a multilayer neural network $x \in \mathbb{R}^d \mapsto \mathcal{Z}_i(x;\theta_i)$ with parameter θ_i for the approximation of $x \mapsto \sigma^\intercal(t_i,x)D_x u(t_i,x)$:

$$\mathcal{Z}_i(x;\theta_i) \approx \sigma^\intercal(t_i,x)D_x u(t_i,x), \tag{7.3.5}$$

one computes estimations $\mathcal{U}_i$ of $u(t_i;X_{t_i})$ by forward induction via:

$$\mathcal{U}_{i+1} = F(t_i,X_{t_i},\mathcal{U}_i,\mathcal{Z}_i(X_{t_i};\theta_i),\Delta t_i,\Delta W_{t_i}),$$

for $i = 0,\ldots,N-1$. This algorithm forms a global deep neural network composed of the neural networks (7.3.5) of each period, by taking as input data (in machine learning language) the paths of $(X_{t_i})_{i=0,\ldots,N}$ and $(W_{t_i})_{i=0,\ldots,N}$, and giving as output $\mathcal{U}_N = \mathcal{U}_N(\theta)$, which is a function of the input and of the total set of parameters $\theta = (\mathcal{U}_0,\theta_0,\ldots,\theta_{N-1})$. The output aims to match the terminal condition $g(X_{t_N})$ of the BSDE, and one then optimizes over the parameter θ the expected square loss function:

$$\theta \mapsto \; \mathbb{E}\big|g(X_{t_N}) - \mathcal{U}_N(\theta)\big|^2.$$

This is obtained by stochastic gradient descent-type (SGD) algorithms relying on training input data.

7.3.2 New schemes: DBDP1 and DBDP2

The proposed scheme is defined from a backward dynamic programming type relation, and has two versions:

(1) First version:

- Initialize from an estimation $\widehat{\mathcal{U}}_N^{(1)}$ of $u(t_N,.)$ with $\widehat{\mathcal{U}}_N^{(1)} = g$
- For $i = N-1,\dots,0$, given $\widehat{\mathcal{U}}_{i+1}^{(1)}$, use a pair of deep neural networks $(\mathcal{U}_i(.;\theta),\mathcal{Z}_i(.;\theta)) \in \mathcal{NN}_{d,1,L,m}^{\varrho}(\mathbb{R}^{N_m}) \times \mathcal{NN}_{d,d,L,m}^{\varrho}(\mathbb{R}^{N_m})$ for the approximation of $(u(t_i,.),\sigma^\intercal(t_i,.)D_x u(t_i,.))$, and compute (by SGD) the minimizer of the expected quadratic loss function

$$\begin{cases} \hat{L}_i^{(1)}(\theta) := \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}^{(1)}(X_{t_{i+1}}) - F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta), \mathcal{Z}_i(X_{t_i};\theta), \Delta t_i, \Delta W_{t_i})\Big|^2 \\ \quad \theta_i^* \in \arg\min_{\theta\in\mathbb{R}^{N_m}} \hat{L}_i^1(\theta). \end{cases} \tag{7.3.6}$$

  Then, update: $\widehat{\mathcal{U}}_i^{(1)} = \mathcal{U}_i(.;\theta_i^*)$, and set $\widehat{\mathcal{Z}}_i^{(1)} = \mathcal{Z}_i(.;\theta_i^*)$.

(2) Second version:

- Initialize with $\widehat{\mathcal{U}}_N^{(2)} = g$
- For $i = N-1,\dots,0$, given $\widehat{\mathcal{U}}_{i+1}^{(2)}$, use a deep neural network $\mathcal{U}_i(.;\theta) \in \mathcal{NN}_{d,1,L,m}^{\varrho}(\Theta_m)$, and compute (by SGD) the minimizer of the expected quadratic loss function

$$\begin{cases} \hat{L}_i^{(2)}(\theta) := \mathbb{E}\Big|\widehat{\mathcal{U}}_{i+1}^{(2)}(X_{t_{i+1}}) - \\ \qquad\qquad F(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta), \sigma^\intercal(t_i,X_{t_i})\hat{D}_x\mathcal{U}_i(X_{t_i};\theta), \Delta t_i, \Delta W_{t_i})\Big|^2 \\ \quad \theta_i^* \in \arg\min_{\theta\in\Theta_m} \hat{L}_i^2(\theta), \end{cases} \tag{7.3.7}$$

  where $\hat{D}_x\mathcal{U}_i(.;\theta)$ is the numerical differentiation of $\mathcal{U}_i(.;\theta)$. Then, update: $\widehat{\mathcal{U}}_i^{(2)} = \mathcal{U}_i(.;\theta_i^*)$, and set $\widehat{\mathcal{Z}}_i^{(2)} = \sigma^\intercal(t_i,.)\hat{D}_x\mathcal{U}_i(.;\theta_i^*)$.

Remark 7.3.1. For the first version of the scheme, one can use independent neural networks, respectively for the approximation of $u(t_i,.)$ and for the approximation of $\sigma^\intercal(t_i,.)D_x u(t_i,.)$. In other words, the parameters are divided into a pair $\theta = (\xi,\eta)$ and we consider neural networks $\mathcal{U}_i(.;\xi)$ and $\mathcal{Z}_i(.;\eta)$. □

In the sequel, we refer to the first and second version of the new scheme above as DBDP1 and DBDP2, where the acronym DBDP stands for deep learning backward dynamic programming.

The intuition behind DBDP1 and DBDP2 is the following. For simplicity, take $f = 0$, so that $F(t,x,y,z,h,\Delta) = y + z^\intercal\Delta$. The solution u to the PDE (7.1.1) should then approximately satisfy (see (7.3.4))

$$u(t_{i+1}, X_{t_{i+1}}) \approx\ u(t_i, X_{t_i}) + D_x u(t_i, X_{t_i})^\intercal \sigma(t_i, X_{t_i})\Delta W_{t_i}.$$

Consider the first scheme DBDP1, and suppose that at time $i+1$, $\widehat{\mathcal{U}}_{i+1}^{(1)}$ is an estimation of $u(t_{i+1,.})$. The quadratic loss function at time i is then approximately equal to

$$\begin{aligned} \hat{L}_i^{(1)}(\theta) &\approx\ \mathbb{E}\Big|u(t_{i+1}, X_{t_{i+1}}) - \mathcal{U}_i(X_{t_i};\theta) - \mathcal{Z}_i(X_{t_i};\theta)^\intercal\Delta W_{t_i}\Big|^2 \\ &\approx\ \mathbb{E}\Big[\Big|u(t_i,X_{t_i}) - \mathcal{U}_i(X_{t_i};\theta)\Big|^2 + \Delta t_i\Big|\sigma^\intercal(t_i,X_{t_i})D_x u(t_i,X_{t_i}) - \mathcal{Z}_i(X_{t_i};\theta)\Big|^2\Big]. \end{aligned}$$

Therefore, by minimizing over θ this quadratic loss function, via 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 $\sigma^\intercal(t_i,)D_x u(t_i,)$ in view of the universal approximation theorem [HSW90]. Similarly, the second scheme DPDP2, which uses only neural network on the value functions, learns $u(t_i,.)$ by means of the neural network $\mathcal{U}_i$, and $\sigma^\intercal(t_i,)D_x u(t_i,)$ via $\sigma^\intercal(t_i,)\hat{D}_x\mathcal{U}_i$. The rigorous arguments for the convergence of these schemes will be derived in the next section.

The advantages of our two schemes, compared to the Deep BSDE algorithm, are the following:

- we solve smaller problems that are less prone to be trapped in local mimimizer. The memory needed in [HJE17] can be a problem when taking too many time steps.
- at each time step, we initialize the weights and bias of the neural network to the weights and bias of the previous time step treated : this methodology always used in iterative solvers in PDE methods permits to have a starting point close to the solution, and then to avoid local minima too far away from the true solution. Besides the number of gradient iterations to achieve is rather small after the first resolution step.

The small disadvantage is due to the Tensorflow structure. As it is done in python, the global graph creation takes much time as it is repeated for each time step and the global resolution is a little bit time consuming : as the dimension of the problem increases, the time difference decreases and it becomes hard to compare the computational time for a given accuracy when the dimension is above 5.

7.3.3 Extension to variational inequalities: scheme RDBDP

Let us consider a variational inequality in the form

$$\begin{cases} \min\big[-\partial_t u - \mathcal{L}u - f(t,x,u,\sigma^\intercal D_x u), u-g\big] = 0, & t\in[0,T),\ x\in\mathbb{R}^d, \\ u(T,x) = g(x), & x\in\mathbb{R}^d. \end{cases} \tag{7.3.8}$$

which arises, e.g., in optimal stopping problem and American option pricing in finance. It is known, see e.g. [EK+97], that such variational inequality is related to reflected BSDE of the form

$$\begin{aligned} Y_t =&\ g(\mathcal{X}_T) + \int_t^T f(s,\mathcal{X}_s,Y_s,Z_s)\,\mathrm{d}s - \int_t^T Z_s^\intercal\,\mathrm{d}W_s + K_T - K_t, \\ Y_t \geq&\ g(X_t), \quad 0\leq t\leq T, \end{aligned} \tag{7.3.9}$$

where K is an adapted non-decreasing process satisfying

$$\int_0^T \big(Y_t - g(X_t)\big)dK_t = 0.$$

The extension of our DBDP1 scheme for such variational inequality, and refereed to as RDBDP scheme, becomes

- Initialize $\widehat{\mathcal{U}}_N = g$
- For $i = N-1,\dots,0$, given $\widehat{\mathcal{U}}_{i+1}$, use a pair of (multilayer) neural network $(\mathcal{U}_i(.;\theta),\mathcal{Z}_i(.;\theta)) \in \mathcal{NN}^{\varrho}_{d,1,L,m}(\mathbb{R}^{N_m})\times\mathcal{NN}^{\varrho}_{d,d,L,m}(\mathbb{R}^{N_m})$, and compute (by SGD) the minimizer of the expected quadratic loss function

$$\begin{cases} \hat{L}_i(\theta) := \mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) - F(t_i,X_{t_i},\mathcal{U}_i(X_{t_i};\theta),\mathcal{Z}_i(X_{t_i};\theta),\Delta t_i,\Delta W_{t_i})\big|^2 \\ \theta_i^* \in \arg\min_{\theta\in\mathbb{R}^{N_m}} \hat{L}_i(\theta). \end{cases} \tag{7.3.10}$$

  Then, update: $\widehat{\mathcal{U}}_i = \max\big[\mathcal{U}_i(.;\theta_i^*), g\big]$, and set $\hat{\mathcal{Z}}_i = \mathcal{Z}(.;\theta_i^*)$.

Remark 7.3.2. As for the scheme DBDP1, two neural networks can be used respectively for the approximation of $u(t_i,.)$ and for the approximation of $\sigma^\intercal(t_i,.)D_x u(t_i,.)$. □

7.4 Convergence analysis

The main goal of this section is to prove convergence of the DBDP schemes towards the solution (Y,Z) to the BSDE (7.3.1) (or reflected BSDE (7.3.9) for variational inequalities), and to provide a rate of convergence that depends on the approximation errors by neural networks.

7.4.1 Convergence of DBDP1

We assume the standard Lipschitz conditions on μ and σ, which ensures the existence and uniqueness of an adapted solution $\mathcal{X}$ to the forward SDE (7.1.2) satisfying for any $p > 1$,

$$\mathbb{E}\Big[\sup_{0\le t\le T}|\mathcal{X}_t|^p\Big] < C_p(1+|x_0|^p), \tag{7.4.1}$$

for some constant C_p depending only on p, b, σ and T. Moreover, we have the well-known error estimate with the Euler scheme $X = X^\pi$ defined in (7.3.3) with a time grid $\pi = \{t_0 = 0 < t_1 < \ldots < t_N = T\}$, with modulus $|\pi|$ s.t. $N|\pi|$ is bounded by a constant depending only on T (hence independent of N):

$$\max_{i=0,\ldots,N-1}\mathbb{E}\Big[|\mathcal{X}_{t_{i+1}} - X_{t_{i+1}}|^2 + \sup_{t\in[t_i,t_{i+1}]}|\mathcal{X}_t - X_{t_i}|^2\Big] = O(|\pi|). \tag{7.4.2}$$

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

We shall make the standing usual assumptions on the driver f and the terminal data g.

(H1) (i) There exists a constant $[f]_L > 0$ such that the driver f satisfies:

$$|f(t_2,x_2,y_2,z_2) - f(t_1,x_1,y_1,z_1)| \le [f]_L\Big(|t_2-t_1|^{1/2} + |x_2-x_1| + |y_2-y_1| + |z_2-z_1|\Big),$$

for all (t_1,x_1,y_1,z_1) and $(t_2,x_2,y_2,z_2) \in [0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^d$. Moreover,

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

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

Recall that Assumption **(H1)** ensures the existence and uniqueness of an adapted solution (Y,Z) to (7.3.1) satisfying

$$\mathbb{E}\Big[\sup_{0\le t\le T}|Y_t|^2 + \int_0^T |Z_t|^2\,\mathrm{d}t\Big] < \infty.$$

From the linear growth condition on f in **(H1)**, and (7.4.1), we also see that

$$\mathbb{E}\Big[\int_0^T |f(t,\mathcal{X}_t,Y_t,Z_t)|^2\,\mathrm{d}t\Big] < \infty. \tag{7.4.3}$$

Moreover, we have the standard L^2-regularity result on Y:

$$\max_{i=0,\ldots,N-1}\mathbb{E}\Big[\sup_{t\in[t_i,t_{i+1}]}|Y_t - Y_{t_i}|^2\Big] = O(|\pi|). \tag{7.4.4}$$

Let us also introduce the L^2-regularity of Z:

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

where $\mathbb{E}_i$ denotes the conditional expectation given $\mathcal{F}_{t_i}$. Since $\bar{Z}$ is a L^2-projection of Z, we know that $\varepsilon^Z(\pi)$ converges to zero when $|\pi|$ goes to zero. Moreover, as shown in [Zha04], when the terminal condition g is also Lipschitz, we have

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

Let us first investigate the convergence of the scheme DBDP1 in (7.3.6), and define (implicitly)

$$\begin{cases} \widehat{\mathcal{V}}_{t_i} & := & \mathbb{E}_i\big[\widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}})\big] + f(t_i, X_{t_i}, \widehat{\mathcal{V}}_{t_i}, \overline{\widehat{\mathcal{Z}}_{t_i}})\Delta t_i \\ \overline{\widehat{\mathcal{Z}}_{t_i}} & := & \frac{1}{\Delta t_i}\mathbb{E}_i\Big[\widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}})\Delta W_{t_i}\Big], \end{cases} \tag{7.4.5}$$

for $i = 0, \dots, N-1$. Notice that $\widehat{\mathcal{V}}_{t_i}$ is well-defined for $|\pi|$ small enough (recall that f is Lipschitz) by a fixed point argument. By the Markov property of the discretized forward process $(X_{t_i})_{i=0,\dots,N}$, we note that there exists some deterministic functions $\hat{v}_i$ and $\overline{\hat{z}_i}$ s.t.

$$\widehat{\mathcal{V}}_{t_i} \;=\; \hat{v}_i(X_{t_i}), \text{ and } \quad \overline{\widehat{\mathcal{Z}}_{t_i}} \;=\; \overline{\hat{z}_i}(X_{t_i}), \qquad i = 0, \dots, N-1. \tag{7.4.6}$$

Moreover, by the martingale representation theorem, there exists an $\mathbb{R}^d$-valued square integrable process $(\widehat{\mathcal{Z}}_t)_t$ such that

$$\widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}}) = \; \widehat{\mathcal{V}}_{t_i} - f(t_i, X_{t_i}, \widehat{\mathcal{V}}_{t_i}, \overline{\widehat{\mathcal{Z}}_{t_i}})\Delta t_i + \int_{t_i}^{t_{i+1}} \widehat{\mathcal{Z}}_s^{\intercal}\, \mathrm{d}W_s, \tag{7.4.7}$$

and by Itô isometry, we have

$$\overline{\widehat{\mathcal{Z}}_{t_i}} = \; \frac{1}{\Delta t_i}\mathbb{E}_i\Big[\int_{t_i}^{t_{i+1}} \widehat{\mathcal{Z}}_s\, \mathrm{d}s\Big], \qquad i = 0, \dots, N-1.$$

Let us now define a measure of the (squared) error for the DBDP1 scheme by

$$\mathcal{E}\big[(\widehat{\mathcal{U}}^{(1)}, \widehat{\mathcal{Z}}^{(1)}), (Y, Z)\big] := \max_{i=0,\dots,N-1} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}^{(1)}_i(X_{t_i})\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \big|Z_t - \widehat{\mathcal{Z}}^{(1)}_i(X_{t_i})\big|^2 dt\Big].$$

Our first main result gives an error estimate of the DBDP1 scheme in terms of the L^2-approximation errors of $\hat{v}_i$ and $\overline{\hat{z}_i}$ by neural networks $\mathcal{U}_i$ and $\mathcal{Z}_i$, $i = 0, \dots, N-1$, assumed to be independent (see Remark 7.3.1), and defined as

$$\varepsilon_i^{\mathcal{N},v} := \inf_{\xi} \mathbb{E}\big|\hat{v}_i(X_{t_i}) - \mathcal{U}_i(X_{t_i}; \xi)\big|^2, \qquad \varepsilon_i^{\mathcal{N},z} := \inf_{\eta} \mathbb{E}\big|\overline{\hat{z}_i}(X_{t_i}) - \mathcal{Z}_i(X_{t_i}; \eta)\big|^2.$$

Here, we fix the structure of the neural networks with input dimension d, output dimension $d_1 = 1$ for $\mathcal{U}_i$, and $d_1 = d$ for $\mathcal{Z}_i$, number of layers L, and m neurons for the hidden layers, and the parameters vary in the whole set $\mathbb{R}^{N_m}$ where N_m is the number of parameters. From the universal approximation theorem (I) ([HSW89]), we know that $\varepsilon_i^{NN,v}$ and $\varepsilon_i^{NN,z}$ converge to zero as m goes to infinity, hence can be made arbitrary small for sufficiently large number of neurons.

Theorem 7.4.1. (Consistency of DBDP1) *Under* **(H1)**, *there exists a constant $C > 0$, independent of π, such that*

$$\begin{aligned} \mathcal{E}\big[(\widehat{\mathcal{U}}^{(1)}, \widehat{\mathcal{Z}}^{(1)}), (Y, Z)\big] \le\; C\Big(&\mathbb{E}\big|g(\mathcal{X}_T) - g(X_T)\big|^2 + |\pi| + \varepsilon^Z(\pi) \\ &+ \sum_{i=0}^{N-1}\big(N\varepsilon_i^{\mathcal{N},v} + \varepsilon_i^{\mathcal{N},z}\big)\Big). \end{aligned} \tag{7.4.8}$$

Remark 7.4.1. The error contributions for the DBDP1 scheme in the r.h.s. of estimation (7.4.8) consists of four terms. The first three terms correspond to the time discretization of BSDE, similarly as in [BT04], [GLW05], namely (i) the strong approximation of the terminal condition (depending on the forward scheme and the terminal data g), and converging to zero, as $|\pi|$ goes to zero, with a rate $|\pi|$ when g is Lipschitz by (7.4.2) (see [Avi09] for irregular g), (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, which converges to zero, as $|\pi|$ goes to zero, with a rate $|\pi|$ when g is Lipschitz. Finally, the better the neural networks are able to approximate/learn the functions $\hat{v}_i$ and $\overline{\hat{z}_i}$ at each time $i = 0, \dots, N-1$, the smaller is the last term in the error estimation. □

Proof of Theorem 7.4.1.

In the following, C will denote a positive generic constant independent of π, and that may take different values from line to line.

Step 1. Fix $i \in \{0, \dots, N-1\}$, and observe by (7.3.1), (7.4.5) that

$$Y_{t_i} - \widehat{\mathcal{V}}_{t_i} = \mathbb{E}_i\big[Y_{t_{i+1}} - \widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}})\big] + \mathbb{E}_i\Big[\int_{t_i}^{t_{i+1}} f(t, \mathcal{X}_t, Y_t, Z_t) - f(t_i, X_{t_i}, \widehat{\mathcal{V}}_{t_i}, \overline{\widehat{Z}_{t_i}})\,\mathrm{d}t\Big].$$

By using Young inequality: $(a+b)^2 \le (1+\gamma\Delta t_i)a^2 + (1+\frac{1}{\gamma\Delta t_i})b^2$ for some $\gamma > 0$ to be chosen later, Cauchy-Schwarz inequality, the Lipschitz condition on f in **(H1)**, and the estimation (7.4.2) on the forward process, we then have

$$\begin{aligned}
\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{V}}_{t_i}\big|^2 \le\ & (1+\gamma\Delta t_i)\mathbb{E}\Big|\mathbb{E}_i\big[Y_{t_{i+1}} - \widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}})\big]\Big|^2 \\
& + 4[f]_L^2\Delta t_i\big(1+\frac{1}{\gamma\Delta t_i}\big)\Big\{|\Delta t_i|^2 + \mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Y_t - \widehat{\mathcal{V}}_{t_i}\big|^2\,\mathrm{d}t\Big] \\
& \qquad + \mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Z_t - \overline{\widehat{Z}_{t_i}}\big|^2\,\mathrm{d}t\Big]\Big\} \\
\le\ & (1+\gamma\Delta t_i)\mathbb{E}\Big|\mathbb{E}_i\big[Y_{t_{i+1}} - \widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}})\big]\Big|^2 \qquad (7.4.9)\\
& + 4\frac{[f]_L^2}{\gamma}(1+\gamma\Delta t_i)\Big\{C|\pi|^2 + 2\Delta t_i\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{V}}_{t_i}\big|^2 + \mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Z_t - \overline{\widehat{Z}_{t_i}}\big|^2\,\mathrm{d}t\Big]\Big\},
\end{aligned}$$

where we use in the last inequality the L^2-regularity (7.4.4) of Y.

Recalling the definition of $\bar{Z}$ as a L^2-projection of Z, we observe that

$$\mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Z_t - \overline{\widehat{Z}_{t_i}}\big|^2\,\mathrm{d}t\Big] = \mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Z_t - \bar{Z}_{t_i}\big|^2\,\mathrm{d}t\Big] + \Delta t_i\mathbb{E}\big|\bar{Z}_{t_i} - \overline{\widehat{Z}_{t_i}}\big|^2. \qquad (7.4.10)$$

By multiplying equation (7.3.1) between t_i and t_{i+1} by ΔW_{t_i}, and using Itô isometry, we have together with (7.4.5)

$$\begin{aligned}
\Delta t_i\big(\bar{Z}_{t_i} - \overline{\widehat{Z}_{t_i}}\big) =\ & \mathbb{E}_i\big[\Delta W_{t_i}\big(Y_{t_{i+1}} - \widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}})\big)\big] + \mathbb{E}_i\Big[\Delta W_{t_i}\int_{t_i}^{t_{i+1}} f(t, \mathcal{X}_t, Y_t, Z_t)\,\mathrm{d}t\Big] \\
=\ & \mathbb{E}_i\Big[\Delta W_{t_i}\Big(Y_{t_{i+1}} - \widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}}) - \mathbb{E}_i\big[Y_{t_{i+1}} - \widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}})\big]\Big)\Big] \\
& + \mathbb{E}_i\Big[\Delta W_{t_i}\int_{t_i}^{t_{i+1}} f(t, \mathcal{X}_t, Y_t, Z_t)\,\mathrm{d}t\Big].
\end{aligned}$$

By Cauchy-Schwarz inequality, and law of iterated conditional expectations, this implies

$$\begin{aligned}
\Delta t_i\mathbb{E}\big|\bar{Z}_{t_i} - \overline{\widehat{Z}_{t_i}}\big|^2 \le\ & 2d\Big(\mathbb{E}\big|Y_{t_{i+1}} - \widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}})\big|^2 - \mathbb{E}\Big|\mathbb{E}_i\big[Y_{t_{i+1}} - \widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}})\big]\Big|^2\Big) \\
& + 2d\Delta t_i\mathbb{E}\Big[\int_{t_i}^{t_{i+1}}|f(t, \mathcal{X}_t, Y_t, Z_t)|^2\,\mathrm{d}t\Big]. \qquad (7.4.11)
\end{aligned}$$

Then, by plugging (7.4.10) and (7.4.11) into (7.4.9), and choosing $\gamma = 8d[f]_L^2$, we have

$$\begin{aligned}
\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{V}}_{t_i}\big|^2 \le\ & C\Delta t_i\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{V}}_{t_i}\big|^2 + (1+\gamma\Delta t_i)\mathbb{E}\big|Y_{t_{i+1}} - \widehat{\mathcal{U}}^{(1)}_{i+1}(X_{t_{i+1}})\big|^2 + C|\pi|^2 \\
& + C\mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Z_t - \bar{Z}_{t_i}\big|^2\,\mathrm{d}t\Big] + C\Delta t_i\mathbb{E}\Big[\int_{t_i}^{t_{i+1}}|f(t, \mathcal{X}_t, Y_t, Z_t)|^2\,\mathrm{d}t\Big],
\end{aligned}$$

and thus for $|\pi|$ small enough:

$$\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{V}}_{t_i}\big|^2 \le (1 + C|\pi|)\mathbb{E}\big|Y_{t_{i+1}} - \widehat{\mathcal{U}}_{i+1}^{(1)}(X_{t_{i+1}})\big|^2 + C|\pi|^2 + C\mathbb{E}\Big[\int_{t_i}^{t_{i+1}} \big|Z_t - \bar{Z}_{t_i}\big|^2 \,\mathrm{d}t\Big] + C|\pi|\mathbb{E}\Big[\int_{t_i}^{t_{i+1}} |f(t, \mathcal{X}_t, Y_t, Z_t)|^2 \,\mathrm{d}t\Big]. \quad (7.4.12)$$

Step 2. By using Young inequality in the form: $(a+b)^2 \ge (1-|\pi|)a^2 + (1-\frac{1}{|\pi|})b^2 \ge (1-|\pi|)a^2 - \frac{1}{|\pi|}b^2$, we have

$$\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{V}}_{t_i}\big|^2 = \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i}) + \widehat{\mathcal{U}}_i^{(1)}(X_{t_i}) - \widehat{\mathcal{V}}_{t_i}\big|^2 \ge (1-|\pi|)\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2 - \frac{1}{|\pi|}\mathbb{E}\big|\widehat{\mathcal{U}}_i^{(1)}(X_{t_i}) - \widehat{\mathcal{V}}_{t_i}\big|^2. \quad (7.4.13)$$

By plugging this last inequality into (7.4.12), we then get for $|\pi|$ small enough

$$\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2 \le (1 + C|\pi|)\mathbb{E}\big|Y_{t_{i+1}} - \widehat{\mathcal{U}}_{i+1}^{(1)}(X_{t_{i+1}})\big|^2 + C|\pi|^2 + C\mathbb{E}\Big[\int_{t_i}^{t_{i+1}} \big|Z_t - \bar{Z}_{t_i}\big|^2 \,\mathrm{d}t\Big] + C|\pi|\mathbb{E}\Big[\int_{t_i}^{t_{i+1}} |f(t, \mathcal{X}_t, Y_t, Z_t)|^2 \,\mathrm{d}t\Big] + CN\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2.$$

From discrete Gronwall's lemma (or by induction), and recalling the terminal condition $Y_{t_N} = g(\mathcal{X}_T)$, $\widehat{\mathcal{U}}_i^{(1)}(X_{t_N}) = g(X_T)$, the definition $\varepsilon^Z(\pi)$ of the L^2-regularity of Z, and (7.4.3), this yields

$$\max_{i=0,\dots,N-1} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2 \le C\mathbb{E}\big|g(\mathcal{X}_T) - g(X_T)\big|^2 + C|\pi| + C\varepsilon^Z(\pi) + CN\sum_{i=0}^{N-1}\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2. \quad (7.4.14)$$

Step 3. Fix $i \in \{0, \dots, N-1\}$. By using relation (7.4.7) in the expression of the expected quadratic loss function in (7.3.6), and recalling the definition of $\overline{\widehat{Z}_{t_i}}$ as a L^2-projection of $\widehat{Z}_t$, we have for all parameters $\theta = (\xi, \eta)$ of the neural networks $\mathcal{U}_i(.;\xi)$ and $\mathcal{Z}_i(.;\eta)$

$$\hat{L}_i^{(1)}(\theta) = \tilde{L}_i(\theta) + \mathbb{E}\Big[\int_{t_i}^{t_{i+1}} \big|\widehat{Z}_t - \overline{\widehat{Z}_{t_i}}\big|^2 \,\mathrm{d}t\Big] \quad (7.4.15)$$

with

$$\tilde{L}_i(\theta) := \mathbb{E}\Big|\widehat{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi) + \big(f(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\xi), \mathcal{Z}_i(X_{t_i};\eta)) - f(t_i, X_{t_i}, \widehat{\mathcal{V}}_{t_i}, \overline{\widehat{Z}_{t_i}})\big)\Delta t_i\Big|^2 + \Delta t_i \mathbb{E}\big|\overline{\widehat{Z}_{t_i}} - \mathcal{Z}_i(X_{t_i};\eta)\big|^2.$$

By using Young inequality: $(a+b)^2 \le (1+\gamma\Delta t_i)a^2 + (1+\frac{1}{\gamma\Delta t_i})b^2$, together with the Lipschitz condition on f in **(H1)**, we clearly see that

$$\tilde{L}_i(\theta) \le (1 + C\Delta t_i)\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi)\big|^2 + C\Delta t_i \mathbb{E}\big|\overline{\widehat{Z}_{t_i}} - \mathcal{Z}_i(X_{t_i};\eta)\big|^2. \quad (7.4.16)$$

On the other hand, using Young inequality in the form: $(a+b)^2 \ge (1-\gamma\Delta t_i)a^2 + (1-\frac{1}{\gamma\Delta t_i})b^2 \ge (1-\gamma\Delta t_i)a^2 - \frac{1}{\gamma\Delta t_i}b^2$, together with the Lipschitz condition on f, we have

$$\tilde{L}_i(\theta) \ge (1-\gamma\Delta t_i)\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi)\big|^2 - \frac{2\Delta t_i [f]_L^2}{\gamma}\Big(\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi)\big|^2 + \mathbb{E}\big|\overline{\widehat{Z}_{t_i}} - \mathcal{Z}_i(X_{t_i};\eta)\big|^2\Big) + \Delta t_i \mathbb{E}\big|\overline{\widehat{Z}_{t_i}} - \mathcal{Z}_i(X_{t_i};\eta)\big|^2.$$

By choosing $\gamma = 4[f]_L^2$, this yields

$$\tilde{L}_i(\theta) \geq \ (1 - C\Delta t_i)\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi)\big|^2 + \frac{\Delta t_i}{2}\mathbb{E}\big|\overline{\widehat{Z}_{t_i}} - \mathcal{Z}_i(X_{t_i};\eta)\big|^2. \tag{7.4.17}$$

Step 4. Fix $i \in \{0,\dots,N-1\}$, and take $\theta_i^* = (\xi_i^*,\eta_i^*) \in \arg\min_\theta \hat{L}_i^{(1)}(\theta)$ so that $\widehat{\mathcal{U}}_i^{(1)} = \mathcal{U}_i(.;\xi_i^*)$, and $\widehat{\mathcal{Z}}_i^{(1)} = \mathcal{Z}_i(.;\eta_i^*)$. By (7.4.15), notice that $\theta_i^* \in \arg\min_\theta \tilde{L}_i(\theta)$. From (7.4.17) and (7.4.16), we then have for all $\theta = (\xi,\eta)$

$$\begin{aligned}
&(1 - C\Delta t_i)\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2 + \frac{\Delta t_i}{2}\mathbb{E}\big|\overline{\widehat{Z}_{t_i}} - \widehat{\mathcal{Z}}_i^{(1)}(X_{t_i})\big|^2 \\
&\quad \leq \tilde{L}_i(\theta_i^*) \ \leq \tilde{L}_i(\theta) \ \leq \ (1 + C\Delta t_i)\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi)\big|^2 + C\Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_{t_i}} - \mathcal{Z}_i(X_{t_i};\eta)\big|^2.
\end{aligned}$$

For $|\pi|$ small enough, and recalling (7.4.6), this implies

$$\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_{t_i}} - \widehat{\mathcal{Z}}_i^{(1)}(X_{t_i})\big|^2 \leq \ C\varepsilon_i^{\mathcal{N},v} + C\Delta t_i\varepsilon_i^{\mathcal{N},z}. \tag{7.4.18}$$

Plugging this last inequality into (7.4.14), we obtain

$$\begin{aligned}
\max_{i=0,\dots,N-1}\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2 \leq \ & C\mathbb{E}\big|g(\mathcal{X}_T) - g(X_T)\big|^2 + C|\pi| + C\varepsilon^Z(\pi) \\
& + \ C\sum_{i=0}^{N-1}\big(N\varepsilon_i^{\mathcal{N},v} + \varepsilon_i^{\mathcal{N},z}\big),
\end{aligned} \tag{7.4.19}$$

which proves the consistency of the Y-component in (7.4.8).

Step 5. Let us finally prove the consistency of the Z-component. From (7.4.10) and (7.4.11), we have for any $i = 0,\dots,N-1$:

$$\begin{aligned}
\mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Z_t - \overline{\widehat{Z}_{t_i}}\big|^2\,\mathrm{d}t\Big] \leq \ & \mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Z_t - \bar{Z}_{t_i}\big|^2\,\mathrm{d}t\Big] + 2d|\pi|\mathbb{E}\Big[\int_{t_i}^{t_{i+1}}|f(t,\mathcal{X}_t,Y_t,Z_t)|^2\,\mathrm{d}t\Big] \\
& + \ 2d\Big(\mathbb{E}\big|Y_{t_{i+1}} - \widehat{\mathcal{U}}_{i+1}^{(1)}(X_{t_{i+1}})\big|^2 - \mathbb{E}\Big|\mathbb{E}_i\big[Y_{t_{i+1}} - \widehat{\mathcal{U}}_{i+1}^{(1)}(X_{t_{i+1}})\big]\Big|^2\Big)
\end{aligned}$$

By summing over $i = 0,\dots,N-1$, we get (recall (7.4.3))

$$\begin{aligned}
\mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}}\big|Z_t - \overline{\widehat{Z}_{t_i}}\big|^2\,\mathrm{d}t\Big] \leq \ & \varepsilon^Z(\pi) + C|\pi| + 2d\mathbb{E}\big|g(\mathcal{X}_T) - g(X_T)\big|^2 \\
& + \ 2d\sum_{i=0}^{N-1}\Big(\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2 - \mathbb{E}\Big|\mathbb{E}_i\big[Y_{t_{i+1}} - \widehat{\mathcal{U}}_{i+1}^{(1)}(X_{t_{i+1}})\big]\Big|^2\Big)
\end{aligned} \tag{7.4.20}$$

where we change the indices in the last summation. Now, from (7.4.9), (7.4.13), we have

$$\begin{aligned}
&2d\Big(\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2 - \mathbb{E}\Big|\mathbb{E}_i\big[Y_{t_{i+1}} - \widehat{\mathcal{U}}_{i+1}^{(1)}(X_{t_{i+1}})\big]\Big|^2\Big) \\
&\quad \leq \Big(\frac{1+\gamma|\pi|}{1-|\pi|} - 1\Big)\mathbb{E}\Big|\mathbb{E}_i\big[Y_{t_{i+1}} - \hat{\mathcal{U}}_{i+1}^{(1)}(X_{t_{i+1}})\big]\Big|^2 \\
&\qquad + \ \frac{8d[f]_L^2}{\gamma}\frac{1+\gamma|\pi|}{1-|\pi|}\Big\{C|\pi|^2 + |\pi|\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{V}}_{t_i}\big|^2 + \mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Z_t - \overline{\widehat{Z}_{t_i}}\big|^2\,\mathrm{d}t\Big]\Big\} \\
&\qquad + \ \frac{2d}{|\pi|(1-|\pi|)}\mathbb{E}\big|\widehat{\mathcal{U}}_i^{(1)}(X_{t_i}) - \widehat{\mathcal{V}}_{t_i}\big|^2.
\end{aligned}$$

We now choose $\gamma = 24d[f]_L^2$ so that $\frac{8d[f]_L^2}{\gamma}(1+\gamma|\pi|)/(1-|\pi|) \leq 1/2$ for $|\pi|$ small enough, and by plugging into (7.4.20), we obtain (note also that $\big[(1+\gamma|\pi|)/(1-|\pi|)-1\big] = O(|\pi|)$):

$$\begin{aligned}
\frac{1}{2}\mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}}\big|Z_t-\overline{\widehat{Z}_{t_i}}\big|^2\,\mathrm{d}t\Big] \leq\ & \varepsilon^Z(\pi)+C|\pi|+C\max_{i=0,\dots,N}\mathbb{E}\big|Y_{t_i}-\widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2 \\
& + \frac{1}{2}|\pi|\sum_{i=0}^{N-1}\mathbb{E}\big|Y_{t_i}-\widehat{\mathcal{V}}_{t_i}\big|^2 + CN\sum_{i=0}^{N-1}\mathbb{E}\big|\widehat{\mathcal{U}}_i^{(1)}(X_{t_i})-\widehat{\mathcal{V}}_{t_i}\big|^2 \\
\leq\ & C\varepsilon^Z(\pi)+C|\pi|+C\max_{i=0,\dots,N}\mathbb{E}\big|Y_{t_i}-\widehat{\mathcal{U}}_i^{(1)}(X_{t_i})\big|^2 \\
& + CN\sum_{i=0}^{N-1}\mathbb{E}\big|\widehat{\mathcal{U}}_i^{(1)}(X_{t_i})-\widehat{\mathcal{V}}_{t_i}\big|^2 \\
\leq\ & C\mathbb{E}\big|g(\mathcal{X}_T)-g(X_T)\big|^2+C|\pi|+C\varepsilon^Z(\pi) \\
& + C\sum_{i=0}^{N-1}\big(N\varepsilon_i^{\mathcal{N},v}+\varepsilon_i^{\mathcal{N},z}\big),
\end{aligned} \tag{7.4.21}$$

where we used (7.4.12) and (7.4.3) in the second inequality, and (7.4.18) and (7.4.19) in the last inequality.

By writing that

$$\mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Z_t-\widehat{\mathcal{Z}}_i^{(1)}(X_{t_i})\big|^2\,\mathrm{d}t\Big] \leq 2\mathbb{E}\Big[\int_{t_i}^{t_{i+1}}\big|Z_t-\overline{\widehat{Z}_{t_i}}\big|^2\,\mathrm{d}t\Big]+2\Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_{t_i}}-\widehat{\mathcal{Z}}_i^{(1)}(X_{t_i})\big|^2,$$

and using (7.4.18), (7.4.21), we obtain after summation over $i=0,\dots,N-1$, the required error estimate for the Z-component as in (7.4.19), and this ends the proof. □

7.4.2 Convergence of DBDP2

We shall consider neural networks with one hidden layer, m neurons with total variation smaller than γ_m (see Section 7.2), a C^3 activation function ϱ with linear growth condition, and bounded derivatives, e.g., a sigmoid activation function, or a tanh function: this class of neural networks is then represented by the parametric set of functions

$$\mathcal{NN}^{\varrho}_{d,1,2,m}(\Theta^\gamma_m) := \left\{ x\in\mathbb{R}^d \mapsto \mathcal{U}(x;\theta)=\sum_{i=1}^m c_i\varrho(a_i.x+b_i)+b_0,\ \theta=(a_i,b_i,c_i,b_0)_{i=1}^m \in \Theta^\gamma_m \right\},$$

with

$$\Theta^\gamma_m := \left\{ \theta=(a_i,b_i,c_i,b_0)_{i=1}^m : \max_{i=1,\dots,m}|a_i|\leq\gamma_m,\ \sum_{i=1}^m|c_i|\leq\gamma_m \right\},$$

for some sequence $(\gamma_m)_m$ converging to ∞, as m goes to infinity, and such that

$$\frac{\gamma_m^6}{N} \xrightarrow[m,N\to\infty]{} 0. \tag{7.4.22}$$

Notice that the neural networks in $\mathcal{NN}^{\varrho}_{d,1,2,m}(\Theta^\gamma_m)$ have their first, second and third derivatives uniformly bounded w.r.t. the state variable x. More precisely, there exists some constant C depending only on d and the derivatives of ϱ s.t. for any $\mathcal{U}\in\mathcal{NN}^{\varrho}_{d,1,2,m}(\Theta^\gamma_m)$,

$$\begin{cases}
\sup_{x\in\mathbb{R}^d,\theta\in\Theta^\gamma_m}\big|D_x\mathcal{U}(x;\theta)\big| \leq C\gamma_m^2, & \sup_{x\in\mathbb{R}^d,\theta\in\Theta^\gamma_m}\big|D_x^2\mathcal{U}(x;\theta)\big| \leq C\gamma_m^3, \\
& \text{and} \quad \sup_{x\in\mathbb{R}^d,\theta\in\Theta^\gamma_m}\big|D_x^3\mathcal{U}(x;\theta)\big| \leq C\gamma_m^4.
\end{cases} \tag{7.4.23}$$

Let us investigate the convergence of the scheme DBDP2 in (7.3.7) with neural networks in $\mathcal{NN}^{\varrho}_{d,1,2,m}(\Theta^{\gamma}_m)$, and define for $i = 0, \dots, N-1$:

$$\left\{ \begin{array}{rcl} \widehat{\mathcal{V}}_{t_i} & := & \mathbb{E}_i\big[\widehat{\mathcal{U}}^{(2)}_{i+1}(X_{t_{i+1}})\big] + f(t_i, X_{t_i}, \widehat{\mathcal{V}}_{t_i}, \overline{\widehat{\mathcal{Z}}_{t_i}})\Delta t_i \;=\; \hat{v}_i(X_{t_i}), \\ \overline{\widehat{\mathcal{Z}}_{t_i}} & := & \frac{1}{\Delta t_i}\mathbb{E}_i\Big[\widehat{\mathcal{U}}^{(2)}_{i+1}(X_{t_{i+1}})\Delta W_{t_i}\Big] \;=\; \overline{\hat{z}_i}(X_{t_i}). \end{array} \right. \tag{7.4.24}$$

A measure of the (squared) error for the DBDP2 scheme is defined similarly as in DBDP1 scheme:

$$\mathcal{E}\big[(\widehat{\mathcal{U}}^{(2)}, \widehat{\mathcal{Z}}^{(2)}), (Y, Z)\big] := \max_{i=0,\dots,N-1} \mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}^{(2)}_i(X_{t_i})\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \big|Z_t - \widehat{\mathcal{Z}}^{(2)}_i(X_{t_i})\big|^2 dt\Big].$$

Our second main result gives an error estimate of the DBDP2 scheme in terms of the L^2-approximation errors of $\hat{v}_i$ and its derivative (which exists under assumption detailed below) by neural networks $\mathcal{U}_i \in \mathcal{NN}^{\varrho}_{d,1,2,m}(\Theta^{\gamma}_m)$, $i = 0, \dots, N-1$, and defined as

$$\varepsilon^{\mathcal{N},m}_i := \inf_{\theta\in\Theta^{\gamma}_m} \Big\{\mathbb{E}\big|\hat{v}_i(X_{t_i}) - \mathcal{U}_i(X_{t_i};\theta)\big|^2 + \Delta t_i \mathbb{E}\big|\sigma^{\intercal}(t_i, X_{t_i})\big(D_x\hat{v}_i(X_{t_i}) - D_x\mathcal{U}_i(X_{t_i};\theta)\big)\big|^2\Big\},$$

which are expected to be small in view of the universal approximation theorem (II), see discussion in Remark 7.4.2.

We also require the additional conditions on the coefficients:

(H2) (i) The functions $x \mapsto \mu(t,.)$, $\sigma(t,.)$ are C^1 with bounded derivatives uniformly w.r.t. $(t,x) \in [0,T]\times\mathbb{R}^d$.

(ii) The function $(x,y,z) \mapsto f(t,.)$ is C^1 with bounded derivatives uniformly w.r.t. (t,x,y,z) in $[0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^d$.

Theorem 7.4.2. (Consistency of DBDP2) *Under* **(H1)**-**(H2)**, *there exists a constant $C > 0$, independent of π, such that*

$$\mathcal{E}\big[(\widehat{\mathcal{U}}^{(2)}, \widehat{\mathcal{Z}}^{(2)}), (Y, Z)\big] \le C\Big(\mathbb{E}\big|g(\mathcal{X}_T) - g(X_T)\big|^2 + \frac{\gamma_m^6}{N} + \varepsilon^Z(\pi) + N\sum_{i=0}^{N-1}\varepsilon^{\mathcal{N},m}_i\Big). \tag{7.4.25}$$

Proof. For simplicity of notations, we assume $d = 1$, and only detail the arguments that differ from the proof of Theorem 7.4.8. From (7.4.24), and the Euler scheme (7.3.3), we have

$$\begin{aligned} \hat{v}_i(x) &= \tilde{v}_i(x) + \Delta t_i f(t_i, x, \hat{v}_i(x), \overline{\hat{z}_i}(x)), \quad \tilde{v}_i(x) := \mathbb{E}\big[\hat{u}_{i+1}(X^x_{t_{i+1}})\big], \quad x \in \mathbb{R}^d, \\ \overline{\hat{z}_i}(x) &= \frac{1}{\Delta t_i}\mathbb{E}\big[\hat{u}_{i+1}(X^x_{t_{i+1}})\Delta W_{t_i}\big], \quad X^x_{t_{i+1}} = x + \mu(t_i,x)\Delta t_i + \sigma(t_i,x)\Delta W_{t_i}. \end{aligned}$$

Under assumption **(H2)**(i), and recalling that $\hat{u}_{i+1} = \mathcal{U}_{i+1}(.;\theta^*_{i+1})$ is C^2 with bounded derivatives, we see that $\tilde{v}_i$ is C^1 with

$$\begin{aligned} D_x\tilde{v}_i(x) &= \mathbb{E}\Big[\big(1 + D_x\mu(t_i,x)\Delta t_i + D_x\sigma(t_i,x)\Delta W_{t_i}\big)D_x\hat{u}_{i+1}(X^x_{t_{i+1}})\Big] \\ &= \mathbb{E}\big[D_x\hat{u}_{i+1}(X^x_{t_{i+1}})\big] + \Delta t_i\, R_i(x) \\ R_i(x) &:= D_x\mu(t_i,x)\mathbb{E}\big[D_x\hat{u}_{i+1}(X^x_{t_{i+1}})\big] + \sigma(t_i,x)D_x\sigma(t_i,x)\mathbb{E}\big[D^2_x\hat{u}_{i+1}(X^x_{t_{i+1}})\big], \end{aligned} \tag{7.4.26}$$

where we use integration by parts in the second equality. Similarly, we have

$$\left\{ \begin{array}{rl} \overline{\hat{z}_i}(x) = & \sigma(t_i,x)\mathbb{E}\big[D_x\hat{u}_{i+1}(X^x_{t_{i+1}})\big], \\ D_x\overline{\hat{z}_i}(x) = & D_x\sigma(t_i,x)\mathbb{E}\big[D_x\hat{u}_{i+1}(X^x_{t_{i+1}})\big] + \sigma(t_i,x)\mathbb{E}\big[D^2_x\hat{u}_{i+1}(X^x_{t_{i+1}})\big] + \Delta t_i\,\sigma(t_i,x)G_i(x) \\ G_i(x) := & D_x\mu(t_i,x)\mathbb{E}\big[D^2_x\hat{u}_{i+1}(X^x_{t_{i+1}})\big] + \sigma(t_i,x)D_x\sigma(t_i,x)\mathbb{E}\big[D^3_x\hat{u}_{i+1}(X^x_{t_{i+1}})\big]. \end{array} \right. \tag{7.4.27}$$

Denoting by $\hat{f}_i(x) = f(t_i, x, \hat{v}_i(x), \overline{\hat{z}_i}(x))$, it follows by the implicit function theorem, and for $|\pi|$ small enough, that $\hat{v}_i$ is C^1 with derivative given by

$$D_x\hat{v}_i(x) = D_x\tilde{v}_i(x) + \Delta t_i\Big(D_x\hat{f}_i(x) + D_y\hat{f}_i(x)D_x\hat{v}_i(x) + D_z\hat{f}_i(x)D_x\overline{\hat{z}_i}(x)\Big)$$

and thus by (7.4.26)-(7.4.27)

$$\big(1 - \Delta t_i D_y\hat{f}_i(x)\big)\sigma(t_i,x)D_x\hat{v}_i(x) = \overline{\hat{z}_i}(x) + \Delta t_i\sigma(t_i,x)\Big(R_i(x) + D_x\hat{f}_i(x) + D_z\hat{f}_i(x)D_x\overline{\hat{z}_i}(x)\Big).$$

Under **(H2)**, by the linear growth condition on σ, and using the bounds on the derivatives of the neural networks in $\mathcal{NN}^{\varrho}_{d,1,2,m}(\Theta^{\gamma}_m)$ in (7.4.23), we then have

$$\mathbb{E}\Big|\sigma(t_i,X_{t_i})D_x\hat{v}_i(X_{t_i}) - \overline{\widehat{Z}_{t_i}}\Big|^2 \le C(\gamma_m^6 + |\pi|^2\gamma_m^8)|\pi|^2. \tag{7.4.28}$$

Next, by the same arguments as in Steps 3 and 4 in the proof of Theorem 7.4.1 (see in particular (7.4.18)), we have for $|\pi|$ small enough,

$$\begin{aligned}\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \widehat{\mathcal{U}}_i^{(2)}(X_{t_i})\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_{t_i}} - \widehat{\mathcal{Z}}_i^{(2)}(X_{t_i})\big|^2 \\ \le C\mathbb{E}\big[\big|\hat{v}_i(X_{t_i}) - \mathcal{U}_i(X_{t_i};\theta)\big|^2\big] + C\Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_{t_i}} - \sigma(t_i,X_{t_i})\hat{D}_x\mathcal{U}_i(X_{t_i};\theta)\big|^2,\end{aligned}$$

for all $\theta \in \Theta^N$, and then with (7.4.28), and by definition of $\varepsilon_i^{NN,v,2}$:

$$\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \widehat{\mathcal{U}}_i^{(2)}(X_{t_i})\big|^2 + \Delta t_i\mathbb{E}\big|\overline{\widehat{Z}_{t_i}} - \widehat{\mathcal{Z}}_i^{(2)}(X_{t_i})\big|^2 \le C\varepsilon_i^{NN,v,2} + C(\gamma_m^6 + |\pi|^2\gamma_m^8)|\pi|^3. \tag{7.4.29}$$

On the other hand, by the same arguments as in Steps 1 and 2 in the proof of Theorem 7.4.1 (see in particular (7.4.14)), we have

$$\begin{aligned}\max_{i=0,\dots,N-1}\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i^{(2)}(X_{t_i})\big|^2 \le\ & C\mathbb{E}\big|g(\mathcal{X}_T) - g(X_T)\big|^2 + C|\pi| + C\varepsilon^Z(\pi) \\ &+ CN\sum_{i=0}^{N-1}\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \widehat{\mathcal{U}}_i^{(2)}(X_{t_i})\big|^2.\end{aligned}$$

Plugging (7.4.29) into this last inequality, together with (7.4.22), gives the required estimation (7.4.25) for the Y-component. Finally, by following the same arguments as in Step 5 in the proof of (7.4.1), we obtain the estimation (7.4.25) for the Z-component. □

Remark 7.4.2. The universal approximation theorem (II) [HSW90] is valid on compact sets, and one cannot conclude *a priori* that the error of network approximation $\varepsilon_i^{\mathcal{N},m}$ converge to zero as m goes to infinity. Instead, we have to proceed into two steps:

(i) Localize the error by considering

$$\varepsilon_i^{\mathcal{N},m,K} := \inf_{\theta\in\Theta^{\gamma}_m}\mathbb{E}\big[\Delta_i(X_{t_i};\theta)1_{|X_{t_i}|\le K}\big],$$

where we set $\Delta_i(x;\theta) := |\hat{v}_i(x) - \mathcal{U}_i(x;\theta)|^2 + \Delta t_i\big|\sigma^{\intercal}(t_i,x)\big(D_x\hat{v}_i(x) - D_x\mathcal{U}_i(x;\theta)\big)\big|^2$.

(ii) Consider an increasing family of neural networks $\Theta_m^{\gamma^{N-1}} \subset \ldots \subset \Theta_m^{\gamma^i} \subset \ldots \subset \Theta_m^{\gamma^0}$ on which to minimize the approximation errors by backward induction at times t_i, $i = N-1,\dots,0$, and where, γ_m^i is defined by

$$\gamma_m^i := \gamma_{\varphi^{N-1-i}(m)},$$

with $\varphi : \mathbb{N} \to \mathbb{N}$ an increasing function, and where we use the notation $\varphi^k := \varphi \circ \ldots \circ \varphi$ (composition k times).

The localized approximation error at time t_i, for $0 \le i \le N-1$, should then be rewritten as

$$\varepsilon_{i,N}^{\mathcal{N},m,K} := \inf_{\theta \in \Theta_m^{\gamma_i}} \mathbb{E}\big[\Delta_i(X_{t_i};\theta) 1_{|X_{t_i}| \le K}\big],$$

and the non-localized one as

$$\varepsilon_{i,N}^{\mathcal{N},m} := \inf_{\theta \in \Theta_m^{\gamma_i}} \mathbb{E}\big[\Delta_i(X_{t_i};\theta)\big].$$

Note that $\varepsilon_{i,N}^{\mathcal{N},m,K}$ converges to zero, as m goes to infinity, for any $K > 0$, as claimed by the universal approximation theorem (II) [HSW90]. On the other hand, from the expressions of $\hat{v}_i$, $D_x\hat{v}_i$ in the above proof of Theorem 7.4.2, we see under **(H1)-(H2)**, and from (7.4.23) that for all $x \in \mathbb{R}^d$, $\theta \in \Theta_m^{\gamma_i}$, $i = 0, \ldots, N-1$:

$$|\Delta_i(x;\theta)| \le C(1+|x|^2)\gamma^4_{\varphi^{N-1}(m)},$$

for some positive constant C independent of m, π. We deduce by Cauchy-Schwarz and Chebyshev's inequalities that for all $K > 0$, and $\theta \in \Theta_m^{\gamma_i}$, $i = 0, \ldots, N-1$,

$$\mathbb{E}\big[\Delta_i(X_{t_i};\theta) 1_{|X_{t_i}|>K}\big] \le \big\|\Delta_i(X_{t_i};\theta)\big\|_2 \frac{\big\|X_{t_i}\big\|_2}{K} \le C(1+|x_0|^3)\frac{\gamma^4_{\varphi^{N-1}(m)}}{K},$$

where we used (7.4.1) in the last inequality. This shows that

$$\varepsilon_{i,N}^{\mathcal{N},m} \le \varepsilon_{i,N}^{\mathcal{N},m,K} + C\frac{\gamma^4_{\varphi^{N-1}(m)}}{K}, \quad \forall K > 0,$$

and thus, in theory, the error $\varepsilon_{i,N}^{\mathcal{N},m}$ can be made arbitrary small by suitable choices of large m and K. □

7.4.3 Convergence of RDBDP

In this paragraph, we study the convergence of machine learning schemes for the variational inequality (7.3.8).

We first consider the case when f does not depend on z, so that the component $Y_t = u(t, \mathcal{X}_t)$ solution to the reflected BSDE (7.3.9) admits a Snell envelope representation, and we shall focus on the error on Y by proposing an alternative to scheme (7.3.10), refereed to as RDBDPbis scheme, which only uses neural network for learning the function u:

- Initialize $\widehat{\mathcal{U}}_N = g$
- For $i = N-1, \ldots, 0$, given $\widehat{\mathcal{U}}_{i+1}$, use a deep neural network $\mathcal{U}_i(.;\theta) \in \mathcal{NN}^{\varrho}_{d,1,L,m}(\mathbb{R}^{N_m})$, and compute (by SGD) the minimizer of the expected quadratic loss function

$$\begin{cases} \bar{L}_i(\theta) := \mathbb{E}\big|\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) - \mathcal{U}_i(X_{t_i};\theta) + f(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\theta))\Delta t_i\big|^2 \\ \theta_i^* \in \arg\min_{\theta \in \mathbb{R}^{N_m}} \bar{L}_i(\theta). \end{cases} \tag{7.4.30}$$

  Then, update: $\widehat{\mathcal{U}}_i = \max\big[\mathcal{U}_i(.;\theta_i^*), g\big]$.

Let us also define from the scheme (7.4.30)

$$\begin{cases} \tilde{\mathcal{V}}_{t_i} := \mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}})\big] + f(t_i, X_{t_i}, \tilde{\mathcal{V}}_{t_i})\Delta t_i = \tilde{v}_i(X_{t_i}), \\ \widehat{\mathcal{V}}_{t_i} := \max[\tilde{\mathcal{V}}_{t_i}; g(X_{t_i})], \quad i = 0, \ldots, N-1. \end{cases} \tag{7.4.31}$$

Our next result gives an error estimate of the scheme (7.4.30) in terms of the L^2-approximation errors of $\tilde{v}_i$ by neural networks $\mathcal{U}_i$, $i = 0, \ldots, N-1$, and defined as

$$\tilde{\varepsilon}_i^{\mathcal{N}} := \inf_{\theta \in \mathbb{R}^{N_m}} \mathbb{E}\big|\tilde{v}_i(X_{t_i}) - \mathcal{U}_i(X_{t_i};\theta)\big|^2.$$

Theorem 7.4.3. (Case f independent of z: Consistency of RDBDPbis) *Let Assumption* **(H1)** *hold, with g Lipschitz. Then, there exists a constant $C > 0$, independent of π, such that*

$$\max_{i=0,\dots,N-1} \left\| Y_{t_i} - \hat{\mathcal{U}}_i(X_{t_i}) \right\|_2 \leq C\Big(|\pi|^{\frac{1}{2}} + \sum_{i=0}^{N-1} \sqrt{\tilde{\varepsilon}_i^{\mathcal{N}}} \Big), \tag{7.4.32}$$

where $\|.\|_2$ is the L^2-norm on $(\Omega, \mathcal{F}, \mathbb{P})$.

Remark 7.4.3. The estimation (7.4.32) implies that

$$\max_{i=0,\dots,N-1} \mathbb{E}\left| Y_{t_i} - \hat{\mathcal{U}}_i(X_{t_i}) \right|^2 \leq C\Big(|\pi| + N \sum_{i=0}^{N-1} \tilde{\varepsilon}_i^{\mathcal{N}} \Big),$$

which is of the same order than the error estimate in Theorem 7.4.1 when g is Lipschitz. □

Proof. Let us introduce the discrete-time approximation of the reflected BSDE

$$\begin{cases} Y_{t_N}^\pi = g(X_{t_N}) \\ \tilde{Y}_{t_i}^\pi = \mathbb{E}_i[Y_{t_{i+1}}^\pi] + f(t_i, X_{t_i}, \tilde{Y}_{t_i}^\pi)\Delta t_i \\ Y_{t_i}^\pi = \max\left[\tilde{Y}_{t_i}^\pi; g(X_{t_i})\right], \quad i = 0, \dots, N-1. \end{cases} \tag{7.4.33}$$

It is known, see [BP03], [BT04] that

$$\max_{i=0,\dots,N-1} \left\| Y_{t_i} - Y_{t_i}^\pi \right\|_2 = O(|\pi|^{\frac{1}{2}}). \tag{7.4.34}$$

Fix $i = 0, \dots, N-1$. From (7.4.31), (7.4.33), we have

$$\begin{aligned} |\tilde{Y}_{t_i}^\pi - \tilde{\mathcal{V}}_{t_i}| &\leq \mathbb{E}_i\left| Y_{t_{i+1}}^\pi - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) \right| + \Delta t_i \left| f(t_i, X_{t_i}, \tilde{Y}_{t_i}^\pi) - f(t_i, X_{t_i}, \tilde{\mathcal{V}}_{t_i}) \right| \\ &\leq \mathbb{E}_i\left| Y_{t_{i+1}}^\pi - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) \right| + [f]_L \Delta t_i |\tilde{Y}_{t_i}^\pi - \tilde{\mathcal{V}}_{t_i}|, \end{aligned}$$

from the Lipschitz condition on f in **(H1)**, and then for $|\pi|$ small enough

$$\left\| \tilde{Y}_{t_i}^\pi - \tilde{\mathcal{V}}_{t_i} \right\|_2 \leq (1 + C|\pi|) \left\| Y_{t_{i+1}}^\pi - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) \right\|_2.$$

By Minkowski inequality, this yields for all θ

$$\left\| \tilde{Y}_{t_i}^\pi - \mathcal{U}_i(X_{t_i}; \theta) \right\|_2 \leq (1 + C|\pi|) \left\| Y_{t_{i+1}}^\pi - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) \right\|_2 + \left\| \tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i}; \theta) \right\|_2. \tag{7.4.35}$$

On the other hand, by the martingale representation theorem, there exists an $\mathbb{R}^d$-valued square integrable process $(\tilde{Z}_t)_t$ such that

$$\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) = \tilde{\mathcal{V}}_{t_i} - f(t_i, X_{t_i}, \tilde{\mathcal{V}}_{t_i})\Delta t_i + \int_{t_i}^{t_{i+1}} \tilde{Z}_s^\intercal \, \mathrm{d}W_s,$$

and the expected squared loss function of the DBDP3 scheme can be written as

$$\bar{L}_i(\theta) = \tilde{L}_i(\theta) + \mathbb{E}\Big[\int_{t_i}^{t_{i+1}} \left| \tilde{Z}_t \right|^2 \mathrm{d}t \Big]$$

with

$$\sqrt{\tilde{L}_i(\theta)} := \left\| \tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i}; \theta) + \big(f(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i}; \theta)) - f(t_i, X_{t_i}, \tilde{\mathcal{V}}_{t_i}) \big) \Delta t_i \right\|_2.$$

From the Lipschitz condition on f, and by Minkowski inequality, we have for all θ

$$(1 - [f]_L \Delta t_i) \left\| \tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i}; \theta) \right\|_2 \leq \sqrt{\tilde{L}_i(\theta)} \leq (1 + [f]_L \Delta t_i) \left\| \tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i}; \theta) \right\|_2.$$

Take now $\theta_i^* \in \arg\min_\theta \bar{L}_i(\theta) = \arg\min_\theta \tilde{L}_i(\theta)$. Then, from the above relations, we have

$$(1 - [f]_L \Delta t_i)\big\|\tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\theta_i^*)\big\|_2 \;\le\; (1 + [f]_L \Delta t_i)\big\|\tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\theta)\big\|_2,$$

for all θ, and so

$$\big\|\tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi_i^*)\big\|_2 \le\; (1 + C|\pi|)\sqrt{\tilde{\varepsilon}_i^{\mathcal{N}}}. \tag{7.4.36}$$

Recalling that $\widehat{\mathcal{V}}_{t_i} = \max[\tilde{\mathcal{V}}_{t_i}; g(X_{t_i})]$, and $\hat{\mathcal{U}}_i(X_{t_i}) = \max[\mathcal{U}_i(X_{t_i};\xi_i^*); g(X_{t_i})]$, and since $|\max(a,c) - \max(b,c)| \le |a-b|$, we deduce that

$$\mathbb{E}\big|\widehat{\mathcal{V}}_{t_i} - \hat{\mathcal{U}}_i(X_{t_i})\big|^2 \le\; C\big(\varepsilon_i^{NN,\tilde{v}} + \Delta t_i \varepsilon_i^{NN,\tilde{z}}\big).$$

By taking $\theta = \theta_i^*$ in (7.4.35), recalling that $\widehat{\mathcal{U}}_i(X_{t_i}) = \max[\mathcal{U}_i(X_{t_i};\theta_i^*); g(X_{t_i})]$, $Y_{t_i}^\pi = \max[\tilde{Y}_{t_i}^\pi; g(X_{t_i})]$, and since $|\max(a,c) - \max(b,c)| \le |a-b|$, we obtain by using (7.4.36)

$$\big\|Y_{t_i}^\pi - \widehat{\mathcal{U}}_i(X_{t_i})\big\|_2 \le\; (1 + C|\pi|)\Big(\big\|Y_{t_{i+1}}^\pi - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}})\big\|_2 + \sqrt{\tilde{\varepsilon}_i^{\mathcal{N}}}\Big).$$

By induction, this yields

$$\max_{i=0,\dots,N-1} \big\|Y_{t_i}^\pi - \widehat{\mathcal{U}}_i(X_{t_i})\big\|_2 \le\; C\sum_{i=0}^{N-1}\sqrt{\tilde{\varepsilon}_i^{\mathcal{N}}},$$

and we conclude with (7.4.34). □

We finally turn to the general case when f may depend on z, and study the convergence of the RDBDP scheme (7.3.10) towards the variational inequality (7.3.8) related to the solution (Y,Z) of the reflected BSDE (7.3.9) by showing an error estimate for

$$\mathcal{E}\big[(\widehat{\mathcal{U}},\widehat{\mathcal{Z}}),(Y,Z)\big] := \max_{i=0,\dots,N-1}\mathbb{E}\big|Y_{t_i} - \widehat{\mathcal{U}}_i(X_{t_i})\big|^2 + \mathbb{E}\Big[\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}}\big|Z_t - \widehat{\mathcal{Z}}_i(X_{t_i})\big|^2 dt\Big].$$

Let us define from the scheme (7.3.10)

$$\begin{cases} \tilde{\mathcal{V}}_{t_i} := \mathbb{E}_i\big[\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}})\big] + f(t_i, X_{t_i}, \tilde{\mathcal{V}}_{t_i}, \overline{\tilde{\mathcal{Z}}_{t_i}})\Delta t_i \;=\; \tilde{v}_i(X_{t_i}), \\ \overline{\tilde{\mathcal{Z}}_{t_i}} := \dfrac{1}{\Delta t_i}\mathbb{E}_i\Big[\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}})\Delta W_{t_i}\Big] \;=\; \tilde{z}_i(X_{t_i}), \\ \widehat{\mathcal{V}}_{t_i} := \max[\tilde{\mathcal{V}}_{t_i}; g(X_{t_i})], \quad i = 0,\dots,N-1. \end{cases} \tag{7.4.37}$$

Our final main result gives an error estimate of the RDBDP scheme in terms of the L^2-approximation errors of $\tilde{v}_i$ and $\tilde{z}_i$ by neural networks $\mathcal{U}_i$ and $\mathcal{Z}_i$, $i = 0,\dots,N-1$, assumed to be independent (see Remark 7.3.1), and defined as

$$\varepsilon_i^{\mathcal{N},\tilde{v}} := \inf_\xi \mathbb{E}\big|\tilde{v}_i(X_{t_i}) - \mathcal{U}_i(X_{t_i};\xi)\big|^2, \qquad \varepsilon_i^{\mathcal{N},\tilde{z}} := \inf_\eta \mathbb{E}\big|\tilde{z}_i(X_{t_i}) - \mathcal{Z}_i(X_{t_i};\eta)\big|^2.$$

The result is obtained under one of the following additional assumptions

(H3) g is C^1, and g, $D_x g$ are Lipschitz.

or

(H4) σ is C^1, with σ, $D_x\sigma$ both Lipschitz, and g is C^2, with g, $D_x g$, $D_x^2 g$ all Lipschitz.

Theorem 7.4.4. (Consistency of RDBDP) *Let Assumption* **(H1)** *hold. There exists a constant* $C > 0$*, independent of* π*, such that*

$$\mathcal{E}\big[(\widehat{\mathcal{U}},\widehat{\mathcal{Z}}),(Y,Z)\big] \le C\Big(\varepsilon(\pi) + \sum_{i=0}^{N-1}\big(N\varepsilon_i^{\mathcal{N},\tilde{v}} + \varepsilon_i^{\mathcal{N},\tilde{z}}\big)\Big), \tag{7.4.38}$$

with $\varepsilon(\pi) = O(|\pi|^{\frac{1}{2}})$ *under* **(H3)**, *and* $\varepsilon(\pi) = O(|\pi|)$ *under* **(H4)**.

Proof. Let us introduce the discrete-time approximation of the reflected BSDE

$$\begin{cases} Y_{t_N}^\pi &= g(X_{t_N}) \\ Z_{t_i}^\pi &= \dfrac{1}{\Delta t_i}\mathbb{E}_i\big[Y_{t_{i+1}}^\pi \Delta W_{t_i}\big], \\ \tilde{Y}_{t_i}^\pi &= \mathbb{E}_i[Y_{t_{i+1}}^\pi] + f(t_i, X_{t_i}, \tilde{Y}_{t_i}^\pi, Z_{t_i}^\pi)\Delta t_i \\ Y_{t_i}^\pi &= \max\big[\tilde{Y}_{t_i}^\pi; g(X_{t_i})\big], \quad i = 0,\dots,N-1. \end{cases} \tag{7.4.39}$$

It is known from [BC08] that

$$\begin{cases} \max\limits_{i=0,\dots,N-1}\mathbb{E}\big|Y_{t_i} - Y_{t_i}^\pi\big|^2 = \varepsilon(\pi) \\ \mathbb{E}\Big[\sum\limits_{i=0}^{N-1}\int_{t_i}^{t_{i+1}}\big|Z_t - Z_{t_i}^\pi\big|^2 dt\Big] = O(|\pi|^{\frac{1}{2}}), \end{cases} \tag{7.4.40}$$

with $\varepsilon(\pi) = O(|\pi|^{\frac{1}{2}})$ under **(H3)**, and $\varepsilon(\pi) = O(|\pi|)$ under **(H4)**.

Fix $i = 0,\dots,N-1$. By writing that

$$\tilde{Y}_{t_i}^\pi - \tilde{\mathcal{V}}_{t_i} = \mathbb{E}_i\big[Y_{t_{i+1}} - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}})\big] + \Delta t_i\Big(f(t_i, X_{t_i}, \tilde{Y}_{t_i}^\pi, Z_{t_i}^\pi) - f(t_i, X_{t_i}, \tilde{\mathcal{V}}_{t_i}, \overline{\tilde{Z}_{t_i}})\Big),$$

and proceeding similarly as in Step 1 in the proof of Theorem 7.4.1, we have by Young inequality and Lipschitz condition on f

$$\begin{aligned}\mathbb{E}\big|\tilde{Y}_{t_i}^\pi - \tilde{\mathcal{V}}_{t_i}\big|^2 \le\ & (1+\gamma\Delta t_i)\mathbb{E}\Big|\mathbb{E}_i\big[Y_{t_{i+1}}^\pi - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}})\big]\Big|^2 \\ & + 2\frac{[f]_L^2}{\gamma}(1+\gamma\Delta t_i)\Big\{\Delta t_i\mathbb{E}\big|\tilde{Y}_{t_i}^\pi - \tilde{\mathcal{V}}_{t_i}\big|^2 + \Delta t_i\mathbb{E}\big|Z_{t_i}^\pi - \overline{\tilde{Z}_{t_i}}\big|^2\Big\}. \end{aligned} \tag{7.4.41}$$

From (7.4.37), (7.4.39), Cauchy-Schwarz inequality, and law of iterated conditional expectations, we have similarly as in Step 1 in the proof of Theorem 7.4.1:

$$\Delta t_i\mathbb{E}\big|Z_{t_i}^\pi - \overline{\tilde{Z}_{t_i}}\big|^2 \le 2d\Big(\mathbb{E}\big|Y_{t_{i+1}}^\pi - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}})\big|^2 - \mathbb{E}\Big|\mathbb{E}_i\big[Y_{t_{i+1}}^\pi - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}})\big]\Big|^2\Big).$$

Then, by plugging into (7.4.41) and choosing $\gamma = 4d[f]_L^2$, we have for $|\pi|$ small enough:

$$\mathbb{E}\big|\tilde{Y}_{t_i}^\pi - \tilde{\mathcal{V}}_{t_i}\big|^2 \le (1 + C|\pi|)\mathbb{E}\big|Y_{t_{i+1}}^\pi - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}})\big|^2.$$

Next, by using Young inequality as in Step 2 in the proof of Theorem 7.4.1, we obtain for all $\theta = (\xi,\zeta)$:

$$\mathbb{E}\big|\tilde{Y}_{t_i}^\pi - \mathcal{U}_i(X_{t_i};\xi)\big|^2 \le (1 + C|\pi|)\mathbb{E}\big|Y_{t_{i+1}}^\pi - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}})\big|^2 + CN\mathbb{E}\big|\tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi)\big|^2. \tag{7.4.42}$$

On the other hand, by the martingale representation theorem, there exists an $\mathbb{R}^d$-valued square integrable process $(\tilde{Z}_t)_t$ such that

$$\widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}}) = \tilde{\mathcal{V}}_{t_i} - f(t_i, X_{t_i}, \tilde{\mathcal{V}}_{t_i}, \overline{\tilde{Z}_{t_i}})\Delta t_i + \int_{t_i}^{t_{i+1}}\tilde{Z}_s^\intercal\, \mathrm{d}W_s,$$

and the expected squared loss function of the RDBDP scheme can be written as

$$\hat{L}_i(\theta) = \tilde{L}_i(\theta) + \mathbb{E}\Big[\int_{t_i}^{t_{i+1}} \big|\tilde{Z}_t - \overline{\tilde{Z}_{t_i}}\big|^2 \,\mathrm{d}t\Big],$$

where we notice by Itô isometry that $\overline{\tilde{Z}_{t_i}} = \frac{1}{\Delta t_i}\mathbb{E}_i\Big[\int_{t_i}^{t_{i+1}} \tilde{Z}_t dt\Big]$, and

$$\begin{aligned}\tilde{L}_i(\theta) := \ & \mathbb{E}\Big|\tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi) + \big(f(t_i, X_{t_i}, \mathcal{U}_i(X_{t_i};\xi), \mathcal{Z}_i(X_{t_i};\eta)) - f(t_i, X_{t_i}, \tilde{\mathcal{V}}_{t_i}, \overline{\tilde{Z}_{t_i}})\big)\Delta t_i\Big|^2 \\ & + \ \Delta t_i \mathbb{E}\big|\overline{\tilde{Z}_{t_i}} - \mathcal{Z}_i(X_{t_i};\eta)\big|^2.\end{aligned}$$

By the same arguments as in Step 3 in the proof of Theorem 7.4.1, using Lipschitz condition on f and Young inequality, we show that for all $\theta = (\xi, \eta)$

$$\begin{aligned}&(1 - C\Delta t_i)\mathbb{E}\big|\tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi)\big|^2 + \frac{\Delta t_i}{2}\mathbb{E}\big|\overline{\tilde{Z}_{t_i}} - \mathcal{Z}_i(X_{t_i};\eta)\big|^2 \\ &\qquad \le \tilde{L}_i(\theta) \ \le \ (1 + C\Delta t_i)\mathbb{E}\big|\tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi)\big|^2 + C\Delta t_i \mathbb{E}\big|\overline{\tilde{Z}_{t_i}} - \mathcal{Z}_i(X_{t_i};\eta)\big|^2.\end{aligned}$$

By taking $\theta_i^* = (\xi_i^*, \eta_i^*) \in \arg\min_\theta \hat{L}_i(\theta) = \arg\min_\theta \tilde{L}_i(\theta)$, it follows that for $|\pi|$ small enough

$$\mathbb{E}\big|\tilde{\mathcal{V}}_{t_i} - \mathcal{U}_i(X_{t_i};\xi_i^*)\big|^2 + \Delta t_i \mathbb{E}\big|\overline{\tilde{Z}_{t_i}} - \mathcal{Z}_i(X_{t_i};\eta_i^*)\big|^2 \le C\varepsilon_i^{\mathcal{N},\tilde{v}} + C\Delta t_i \varepsilon_i^{\mathcal{N},\tilde{z}}.$$

By plugging into (7.4.42), recalling that $\widehat{\mathcal{U}}_i(X_{t_i}) = \max[\mathcal{U}_i(X_{t_i};\xi_i^*); g(X_{t_i})]$, $Y_{t_i}^\pi = \max[\tilde{Y}_{t_i}^\pi; g(X_{t_i})]$, and since $|\max(a,c) - \max(b,c)| \le |a-b|$, we obtain

$$\mathbb{E}\big|Y_{t_i}^\pi - \widehat{\mathcal{U}}_i(X_{t_i})\big|^2 \le \ (1 + C|\pi|)\mathbb{E}\big|Y_{t_{i+1}}^\pi - \widehat{\mathcal{U}}_{i+1}(X_{t_{i+1}})\big|^2 + CN\big(\varepsilon_i^{\mathcal{N},\tilde{v}} + \Delta t_i \varepsilon_i^{\mathcal{N},\tilde{z}}\big),$$

and then by induction

$$\max_{i=0,\dots,N-1} \mathbb{E}\big|Y_{t_i}^\pi - \widehat{\mathcal{U}}_i(X_{t_i})\big|^2 \le \ C\sum_{i=0}^{N-1}\big(N\varepsilon_i^{\mathcal{N},\tilde{v}} + \varepsilon_i^{\mathcal{N},\tilde{z}}\big).$$

Combining with (7.4.40), this proves the error estimate (7.4.38) for the Y-component. The error estimate (7.4.38) for the Z-component is proved along the same arguments as in Step 5 in the proof of Theorem 7.4.1, and is omitted here. □

7.5 Numerical results

In the first two subsections, we compare our schemes DBDP1 (7.3.6), DBDP2 (7.3.7) and the scheme proposed by [HJE17] on some examples of PDEs and BSDEs.

We first test our algorithms on some PDEs with bounded solutions and quite a simple structure (see section 7.5.1), and then try to solve some PDEs with unbounded solutions and more complex structures (see section 7.5.2). Our goal is to emphasize that solutions with simple structure easily represented by a neural network can be evaluated by our method even in very high-dimension, whereas the solution with complex structure can only be evaluated in moderate dimension.

Finally, we apply the scheme described in section 7.3.3 to an American option problem and show its accuracy in high dimension (see section 7.5.3).

If not specified, we use in the sequel a fully connected feedforward network with two hidden layers, and $d + 10$ neurons on each hidden layer, to implement our schemes (7.3.6) and (7.3.7). We choose tanh as activation function for the hidden layers in order to avoid some explosion while calculating the numerical gradient Z in scheme (7.3.7) and choose identity function as activation function for the output layer. We renormalize the data before entering the network. We use Adam Optimizer, implemented in TensorFlow and mini-batch with 1000 trajectories for the stochastic gradient descent.

Table 7.1 – Estimate of $u(0, x_0)$ where $d = 1$ and $x_0 = 1$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 1.4686938.

	Averaged value	Standard deviation
DBDP1	1.46332	0.01434
DBDP2	1.4387982	0.01354

7.5.1 PDEs with bounded solution and simple structure

We begin with a simple example in dimension one. It is not hard to find test cases where the scheme proposed in [HJE17] fails even in dimension one. In fact the latter scheme works well for small maturities and with a starting point close to the solution.

It is always interesting to start by testing schemes in dimension one as one can easily compare graphically the numerical results to the theoretical solution. Then we take some examples in higher dimensions and show that our method seems to work well when the dimension increases higher.

7.5.1.1 An example in 1D

We take the following parameters for the BSDE problem defined by (7.1.2) and (7.3.1):

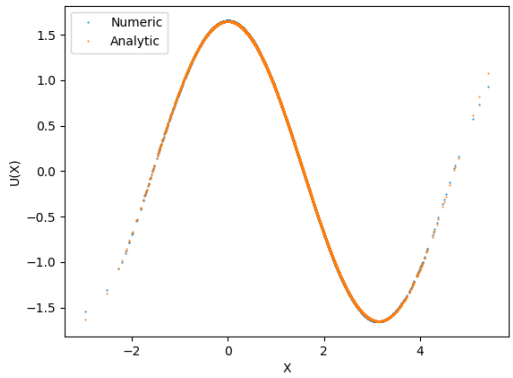
$$\sigma = 1,\ \mu = 0.2,\ T = 2,\ d = 1, \tag{7.5.1}$$

$$\begin{aligned} f(t,x,y,z) &= (\cos(x)(e^{\frac{T-t}{2}} + \tfrac{\sigma^2}{2}) + \mu\sin(x))e^{\frac{T-t}{2}} - \tfrac{1}{2}\left(\sin(x)\cos(x)e^{T-t}\right)^2 + \tfrac{1}{2}(yz)^2 \\ g(x) &= \cos(x). \end{aligned}$$

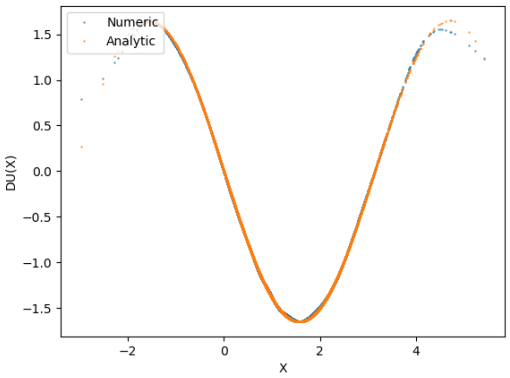
for which, the explicit analytic solution is equal to $u(t,x) = e^{\frac{T-t}{2}}\cos(x)$.

We want to estimate the solution u and its gradient $D_x u$ from our schemes. This example is interesting, because with $T = 1$, the method proposed in [HJE17], initializing $u(0,.)$ as the solution of the associated linear problem associated ($f = 0$) and randomly initializing $D_x u(0,.)$ works very well. However, for $T = 2$, the method in [HJE17] always fails on our test whatever the choice of the initialization: the algorithm is either trapped in a local minimum when the initial learning rate associated to the gradient method is too small or explodes when the learning rate is taken higher. This numerical failure is not dependent on the considered network: using some LSTM networks as in [CWNMW18] gives the same result.

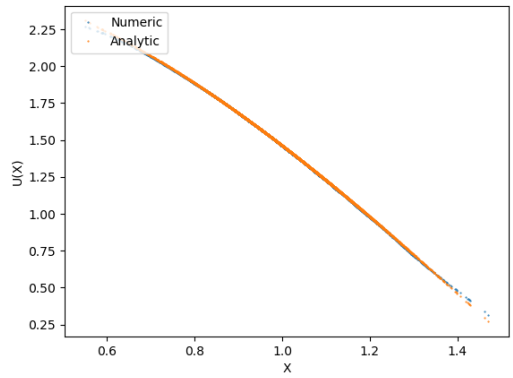
Because of the high non-linearity, we discretize the BSDE using $N = 240$ time steps, and implemented hidden layers with $d + 10 = 11$ neurons. Figure 7.1 (resp. Figure 7.2) depicts the estimated functions $u(t,.)$ and $D_x u(t,.)$ estimated from DBDP1 (resp. DBDP2) scheme.



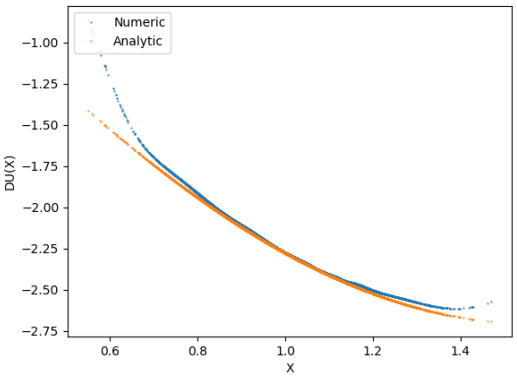
$u(t,.)$ and its estimate at time $t = 1$.



Z and its estimate at time $t = 1$.

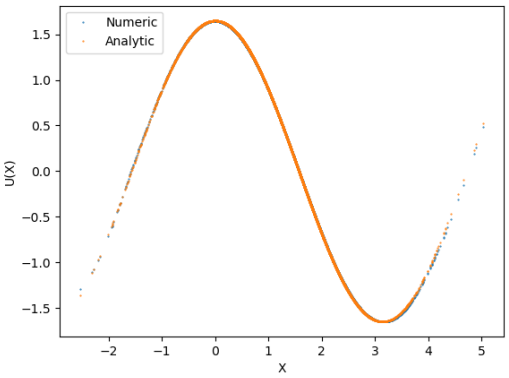


$u(t,.)$ and its estimate at time $t = 0.0091$.

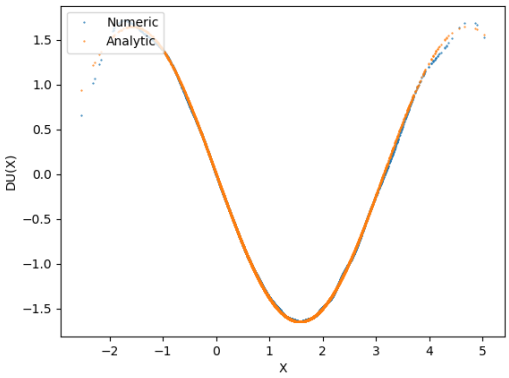


Z and its estimate at time $t = 0.0091$.

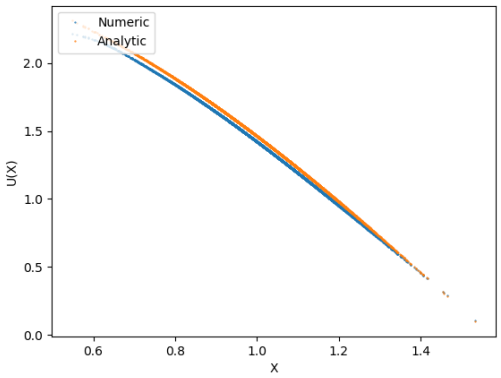
Figure 7.1 – Estimates of u and Z using DBDP1. We took the parameters defined in (7.5.1) and set $x_0 = 1$.



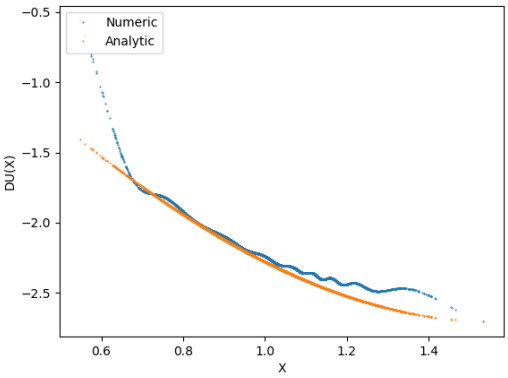
$u(t,.)$ and its estimate at time $t = 1$.



Z and its estimate at time $t = 1$.



$u(t,.)$ and its estimate at time $t = 0.0091$.



Z and its estimate at time $t = 0.0091$.

Figure 7.2 – Estimates of u and Z using DBDP2. We took the parameters defined in (7.5.1) and set $x_0 = 1$.

	Averaged value	Standard deviation
DBDP1	0.4637038	0.004253
DBDP2	0.46335	0.00137
Scheme [HJE17]	0.46562	0.0035

Table 7.2 – Estimate of $u(0, x_0)$ where $d = 5$ and $x_0 = \mathbb{1}_5$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 0.46768.

	Averaged value	Standard deviation
DBDP1	- 1.3895	0.00148
DBDP2	-1.3913	0.000583
Scheme [HJE17]	-1.3880	0.00155

Table 7.3 – Estimate of $u(0, x_0)$ where $d = 10$ and $x_0 = \mathbb{1}_{10}$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is -1.383395.

7.5.1.2 Increasing the dimension

We extend the example from the previous section to the following d-dimensional problem:

$$d \geq 1, \quad \sigma = \frac{1}{\sqrt{d}} \mathbf{I}_d, \quad \mu = \frac{0.2}{d} \mathbb{1}_d, \quad T = 1,$$

$$\begin{aligned} f(t, x, y, z) &= (\cos(\bar{x})(e^{\frac{T-t}{2}} + \tfrac{1}{2}) + 0.2\sin(\bar{x}))e^{\frac{T-t}{2}} - \tfrac{1}{2}\left(\sin(\bar{x})\cos(\bar{x})e^{T-t}\right)^2 + \tfrac{1}{2d}(u(\mathbb{1}_d.z))^2, \\ g(x) &= \cos(\bar{x}), \end{aligned}$$

with $\bar{x} = \sum_{i=1}^{d} x_i$.

We take $N = 120$ in the Euler scheme, and $d + 10$ neurons for each hidden layer. We take 1000 trajectories in mini batch, use data renormalization, and check the loss convergence every 50 iterations. For this small maturity, the scheme [HJE17] generally converges, and we give the results obtained with the same network and initializing the scheme with the linear solution of the problem. Results in dimension 5 to 50 are given in Tables 7.2, 7.3, 7.4 and 7.5. Both schemes (7.3.6) and (7.3.7) work well with results very close to the solution and close to the results calculated by the scheme [HJE17]. As the dimension increases, scheme (7.3.6) seems to be the most accurate.

Remark 7.5.1. In dimension 50, the initial learning rate in scheme [HJE17] is taken small in order to avoid a divergence of the method. In fact, running the test 3 times (with 10 runs each time), we observed convergence of the algorithm two times, and in the last test: one of the ten run exploded, and another one clearly converged to a wrong solution. □

Table 7.4 – Estimate of $u(0, x_0)$ where $d = 20$ and $x_0 = \mathbb{1}_{20}$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 0.6728135.

	Averaged value	Standard deviation
DBDP1	0.6760	0.00274
DBDP2	0.67102	0.00559
Scheme [HJE17]	0.68686	0.002402

Table 7.5 – Estimate of $u(0, x_0)$ where $d = 50$ and $x_0 = \mathbb{1}_{50}$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 1.5909.

	Averaged value	Standard deviation
DBDP1	1.5903	0.006276
DBDP2	1.58762	0.00679
Scheme [HJE17]	1.583023	0.0361

	Averaged value	Standard deviation
DBDP1	1.3720	0.00301
DBDP2	1.37357	0.0022
Scheme [HJE17]	1.37238	0.00045

Table 7.6 – Estimate of $u(0, x_0)$, where $d = 1$ and $x_0 = 0.5$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 1.37758.

7.5.2 PDEs with unbounded solution and more complex structure

In this section with take the following parameters

$$\begin{aligned} \sigma = \frac{1}{\sqrt{d}} \mathbf{I}_d, \quad \mu = 0, \quad T = 1, \\ f(x, y, z) = k(x) + \frac{1}{2\sqrt{d}} y(\mathbb{1}_d.z) + \frac{y^2}{2} \end{aligned} \tag{7.5.2}$$

where the function k is chosen such that the solution to the PDE is equal to

$$u(t, x) = \frac{T - t}{d} \sum_{i=1}^{d} (\sin(x_i) 1_{x_i < 0} + x_i 1_{x_1 \geq 0}) + \cos\left(\sum_{i=1}^{d} i x_i \right).$$

Notice that the structure of the solution is more complex than in the first example. We aim at evaluating the solution at $x = 0.5\mathbb{1}_d$. We take 120 time steps for the Euler time discretization and $d + 10$ neurons in each hidden layers. As shown in Figures 7.3 and 7.4 as well as in Table 7.6, the three schemes provide accurate and stable results in dimension $d = 1$.

In dimension 2, the three schemes provide very accurate and stable results, as shown in Figures 7.5 and 7.6, as well as in Table 7.7.

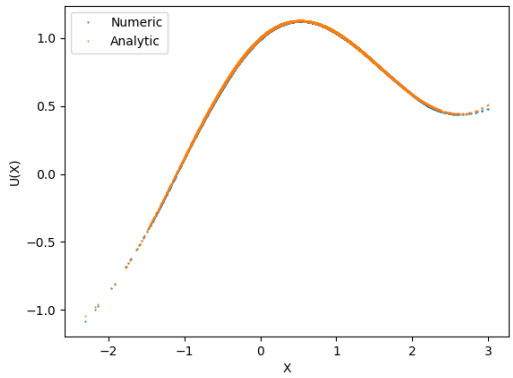
Above dimension 3, the scheme [HJE17] always explodes no matter the chosen initial learning rate and the activation function for the hidden layers (among the tanh, ELU, ReLu and sigmoid ones). Besides, taking 3 or 4 hidden layers does not improve the results.

We reported the results obtained in dimension $d = 5$ and 8 in Table 7.8 and 7.9. Scheme (7.3.6) seems to work better than scheme (7.3.7) as the dimension increases. Note that the standard deviation increases with the dimension of the problem.

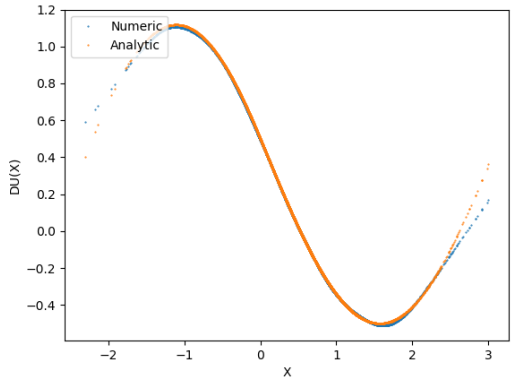
When $d \geq 10$, schemes (7.3.6) and (7.3.7) both fail at providing correct estimates of the solution, as shown in Table 7.10. Increasing the number of layers or neurons does not improve the result.

	Averaged value	Standard deviation
DBDP1	0.5715359	0.0038
DBDP2	0.5707974	0.00235
Scheme [HJE17]	0.57145	0.0006

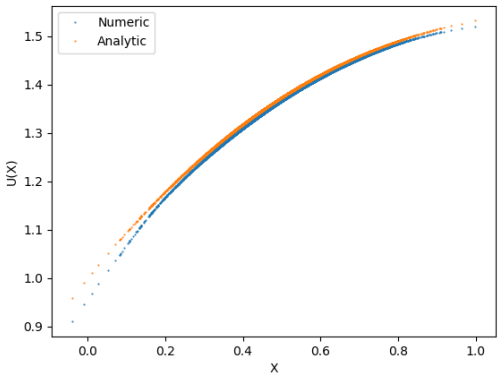
Table 7.7 – Estimate of $u(0, x_0)$, where $d = 2$ and $x_0 = 0.5\mathbb{1}_2$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 0.570737.



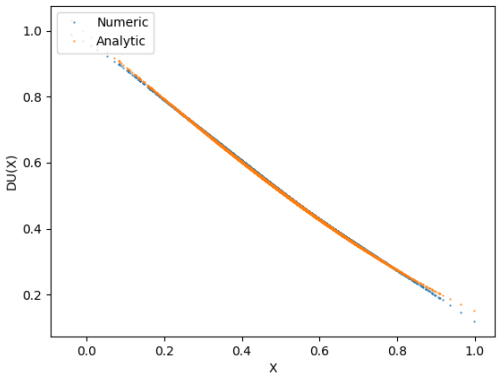
$u(t,.)$ and its estimate at time $t = 0.5$.



Z and its estimate at time $t = 0.5$



$u(t,.)$ and its estimate at time $t = 0.0085$.



Z and its estimate at time $t = 0.0085$.

Figure 7.3 – Estimates of u and Z using DBDP1. We took the parameters defined in (7.5.2), with $d = 1$, and set $x_0 = 0,5$.

7.5.3 Application to American options

Consider the stock price $X_t = (X_t^1, \dots, X_t^d)$ of d assets with the following dynamics under the risk neutral probability measure:

$$dX_t^i = rX_t^i dt + \sigma_i X_t^i dW_t^i,$$

where $W_. = (W_.^1, \dots, W_.^d)$ is a d-dimensional Brownian Motion, $\sigma = (\sigma_1, \dots, \sigma_d) \in \mathbb{R}^d$, and r is the risk-free rate.

The value at time t of an American option with payoff g and maturity T is given by:

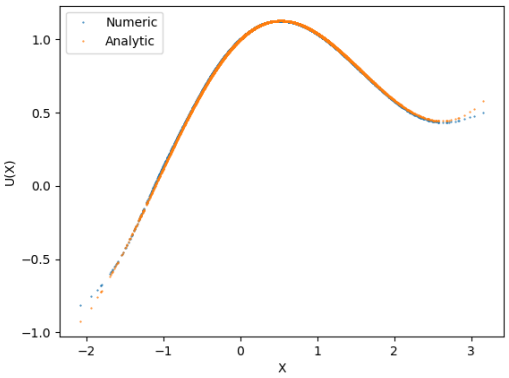
$$u(t,x) = \sup_{\tau \in \mathcal{T}_{t,T}} \mathbb{E}[e^{-r\tau} g(X_\tau)],$$

where $\mathcal{T}_{t,T}$ is the set of stopping time with values in $[t,T]$, and is solution of the variational inequality

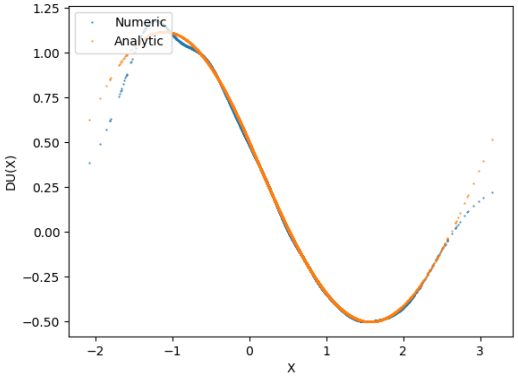
$$\begin{cases} \min\big[-\partial_t u - \hat{\mathcal{L}}u, u - g\big] &= 0, \quad \text{on } [0,T) \times (0,\infty)^d \\ u(T,.) &= g, \quad \text{on } (0,\infty)^d, \end{cases}$$

with

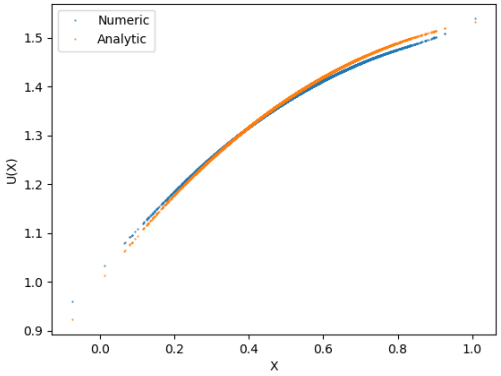
$$\hat{\mathcal{L}}u(t,x) = \frac{1}{2}\sum_{i=1}^d \sigma_i^2 x_i^2 D_{x_i}^2 u(t,x) + r\sum_{i=1}^d x_i D_{x_i} u(t,x) - ru(t,x),$$



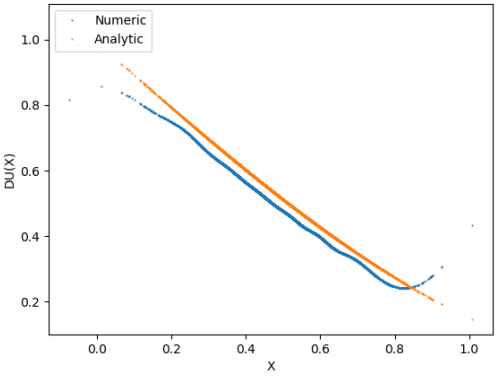
$u(t,.)$ and its estimate at time $t = 0.5$.



Z and its estimate at time $t = 0.5$

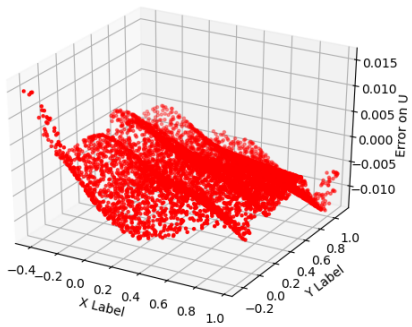


$u(t,.)$ and its estimate at time $t = 0.0085$.

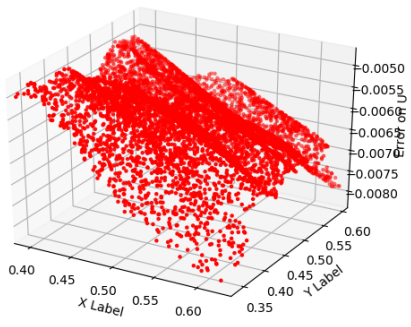


Z and its estimate at time $t = 0.0085$.

Figure 7.4 – Estimates of u and Z using DBDP2. We took the parameters defined in (7.5.2), with $d = 1$, and set $x_0 = 0.5$.

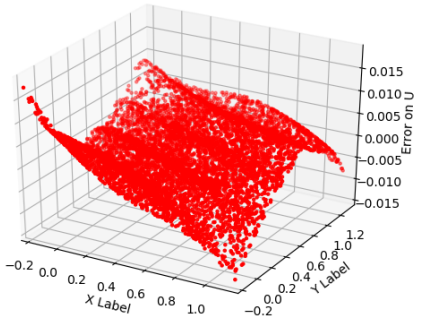


Error on solution at date $t = 0.5$.

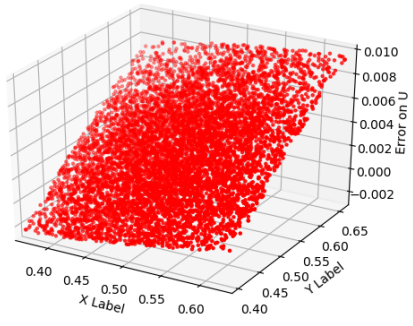


Error on solution at date $t = 0.0085$.

Figure 7.5 – Algebric error of the estimate of u using DBDP1. We took the parameters in (7.5.2) and set $d = 2$ and $x_0 = 0.5\mathbb{1}_d$.



Error on solution at date $t = 0.5$.



Error on solution at date $t = 0.0085$.

Figure 7.6 – Algebric error of the estimate of u using scheme (7.3.7). We took the parameters in (7.5.2) and set $d = 2$ and $x_0 = 0.5\mathbb{1}_d$.

	Averaged value	Standard deviation
DBDP1	0.8666	0.013
DBDP2	0.83646	0.00453
Scheme [HJE17]	NC	NC

Table 7.8 – Estimate of $u(0, x_0)$, where $d = 5$ and $x_0 = 0.5\mathbb{1}_5$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 0.87715.

as proved e.g. in [JLL90].

Let us define the change of function v by: $u(t, x) = e^{rt}v(t, \log(x))$, which is solution of the following variational inequality

$$\begin{cases} \min\left(-\partial_t v - \mathcal{L}v, v - \hat{g}\right) &= 0, \quad \text{on } [0,T) \times \mathbb{R}^d \\ v(T,.) &= \hat{g}, \quad \text{on } \mathbb{R}^d, \end{cases} \tag{7.5.3}$$

where

$$\hat{g}(t,x) = e^{-rt}g(e^x),$$

$$\mathcal{L}v = \frac{1}{2}\sum_{i=1}^{d} \sigma_i^2 D^2_{x_i} v_{ii} + \sum_{i=1}^{d}(r - \frac{1}{2}\sigma_i^2) D_{x_i} v_i.$$

In this section, we test the scheme described in section 7.3.3 on (7.5.3) in the special case of a geometrical put with strike $K = 1$, $T = 1$, $r = 0.05$, $X_0^i = 1$, $\sigma_i = 0.2$ for $i = 1$ to d, and payoff $(K - \prod_{i=1}^d X_t^i)_+$, as considered previously in [BW12]. In dimension d, the case boils down to the resolution of an American option in dimension $d = 1$ so that it can be very accurately estimated e.g. with a tree-based method. Results given in Table 7.11 show that scheme (7.3.10) is very accurate for the pricing of American options.

	Averaged value	Standard deviation
DBDP1	1.169441	0.02537
DBDP2	1.0758344	0.00780
Scheme [HJE17]	NC	NC

Table 7.9 – Estimate of $u(0, x_0)$, where $d = 8$ and $x_0 = 0.5\mathbb{1}_8$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is 1.1603167.

	Averaged value	Standard deviation
DBDP1	-0.3105	0.02296
DBDP2	-0.3961	0.0139
Scheme [HJE17]	NC	NC

Table 7.10 – Estimate of $u(0, x_0)$, where $d = 10$ and $x_0 = 0.5\mathbb{1}_{10}$. Average and standard deviation observed over 10 independent runs are reported. The theoretical solution is -0.2148861.

Dimension	nb step	value	std	reference
1	10	0.06047	0.00023	0.060903
1	20	0.060789	0.00021	0.060903
1	40	0.061122	0.00015	0.060903
1	80	0.0613818	0.00019	0.060903
5	10	0.10537	0.00014	0.10738
5	20	0.10657	0.00011	0.10738
5	40	0.10725	0.00012	0.10738
5	80	0.107650	0.00016	0.10738
10	10	0.12637	0.00014	0.12996
10	20	0.128292	0.00011	0.12996
10	40	0.12937	0.00014	0.12996
10	80	0.129923	0.00016	0.12996
20	10	0.1443	0.00014	0.1510
20	20	0.147781	0.00012	0.1510
20	40	0.149560	0.00012	0.1510
20	80	0.15050	0.00010	0.1510
40	10	0.15512	0.00018	0.1680
40	20	0.16167	0.00015	0.1680
40	40	0.16487	0.00011	0.1680
40	80	0.16665	0.00013	0.1680
40	160	0.16758	0.00016	0.1680

Table 7.11 – Estimates of the American option using RDBDP. Average and standard deviation over 40 independent runs for different numbers of time steps are reported.

Bibliography

[Abe+16] F. Abergel, A. Anane., A. Chakraborti, A. Jedidi, and I. Muni Toke. *Limit Order Books*. Cambridge University Press, 2016.

[AC01] R. Almgren and N. Chriss. "Optimal execution of portfolio transactions". In: *Journal of Risk* 3 (2001), pp. 5–40.

[AHP17] F. Abergel, C. Huré, and H. Pham. "Algorithmic trading in a microstructural limit order book model". In: *arXiv:1705.01446* (2017).

[Ala+19] C. Alasseur, A. Balata, S. Ben Aziza, A. Maheshwari, P. Tankov, and X. Warin. "Regression Monte Carlo for microgrid management". In: *ESAIM Proceedings and surveys* 65 (2019). to appear.

[AS07] M. Avellaneda and S. Stoikov. "High-frequency trading in a limit order book". In: *Quantitative Finance* 8 (2007), pp. 217–224.

[Avi09] R. Avikainen. "On irregular functionals of SDEs and the Euler scheme". In: *Finance and Stochastics* 13 (2009), pp. 381–401.

[Bac17] F. Bach. "Breaking the curse of dimensionality with convex neural networks". In: *Journal of Machine Learning Research* 18.19 (2017), pp. 1–53.

[Bal+19] A. Balata, C. Huré, M. Laurière, H. Pham, and I. Pimentel. "A class of finite-dimensional numerically solvable McKean-Vlasov control problems". In: *ESAIM Proceedings and surveys* 65 (2019).

[BBEM18] N. Baradel, B. Bouchard, D. Evangelista, and O. Mounjid. "Optimal inventory management and order book modeling". In: *arXiv:1802.08135* (2018).

[BC08] B. Bouchard and J.F. Chassagneux. "Discrete-time approximation for continuously and discretely reflected BSDEs". In: *Stochastic Processes and their Applications* 118 (2008), pp. 2269–2293.

[BGP16] A. Bismuth, O. Guéant, and J. Pu. "Portfolio choice, portfolio liquidation, and portfolio transition under drift uncertainty". In: *arXiv preprint arXiv:1611.07843* (2016).

[BGS18] C. Bender, C. Gärtner, and N. Schweizer. "Pathwise Dynamic Programming". In: *Mathematics of Operations Research* (2018).

[BHLP18] A. Bachouch, C. Huré, N. Langrené, and H. Pham. "Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications". In: *arXiv:1812.05916* (2018).

[BKL01] D. Bertsimas, L. Kogan, and A. W. Lo. "Hedging derivative securities and incomplete markets: an ε-arbitrage approach". In: *Operations Research* 49.3 (2001), pp. 372–397.

[BKS10] D. Belomestny, A. Kolodko, and J. Schoenmakers. "Regression methods for stochastic control problems and their convergence analysis". In: *SIAM Journal on Control and Optimization* 48.5 (2010), pp. 3562–3588.

[BLPR17] R. Buckdahn, J. Li, S. Peng, and C. Rainer. "Mean-field stochastic differential equations and associated PDEs". In: *The Annals of Probability* 45.2 (2017), pp. 824–878.

[BP03] V. Bally and G. Pagès. "Error analysis of the quantization algorithm for obstacle problems". In: *Stochastic Processes and their Applications* 106 (2003), pp. 1–40.

[BP17] A. Balata and J. Palczewski. "Regress-Later Monte Carlo for optimal control of Markov processes". In: *eprint arXiv:1712.09705* (2017).

[BP18] A. Balata and J. Palczewski. "Regress-Later Monte-Carlo for optimal inventory control with applications in energy". In: *arXiv:1703.06461* (2018).

[BR11] N. Bäuerle and U. Rieder. *Markov Decision Processes with Applications to Finance*. Springer, 2011.

[BRS18] C. Bayer, M. Redmann, and J. Schoenmakers. "Dynamic programming for optimal stopping via pseudo-regression". In: *arXiv:1808.04725v1* (2018).

[Bré81] P. Brémaud. *Point Processes and Queues : Martingale Dynamics*. Springer, 1981.

[BS12] C. Bender and J. Steiner. "Least-squares Monte Carlo for backward SDEs". In: *Numerical methods in finance*. Ed. by René Carmona, Pierre Del Moral, Peng Hu, and Nadia Oudjane. Springer, 2012, pp. 257–289.

[BSST18] D. Belomestny, J. Schoenmakers, V. Spokoiny, and Y. Tavyrikov. "Optimal stopping via reinforced regression". In: *arXiv:1808.02341* (2018).

[BT04] B. Bouchard and N. Touzi. "Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations". In: *Stochastic Processes and their applications* 111.2 (2004), pp. 175–206.

[BW12] B. Bouchard and X. Warin. "Monte-Carlo valuation of American options: facts and new algorithms to improve existing methods". In: *Numerical methods in finance*. Springer, 2012, pp. 215–255.

[Car10] P. Cardaliaguet. *Notes on mean field games*. Tech. rep. from P.-L. Lions' lectures at Collège de France, 2010.

[CCD14] J-F. Chassagneux, D. Crisan, and F. Delarue. "A probabilistic approach to classical solutions of the master equation for large population equilibria". In: *arXiv preprint arXiv:1411.3009* (2014).

[CFS15] R. Carmona, J-P. Fouque, and L-H. Sun. "Mean field games and systemic risk". In: *Communications in Mathematical Sciences* 13.4 (2015), pp. 911–933.

[CJ10] A. Cartea and S. Jaimungal. "Modeling Asset Prices for Algorithmic and High Frequency Trading". In: *Applied Mathematical Finance* 20(6) (2010), pp. 512–547.

[CJ13] A. Cartea and S. Jaimungal. "Risk metrics and fine tuning of high-frequency trading strategies." In: *Mathematical Finance* 25(3) (2013), pp. 576–611.

[CJR14] A. Cartea, S. Jaimungal, and J. Ricci. "Buy Low Sell High: a High Frequency Trading Perspective". In: *SIAM Journal on Financial Mathematics* (2014), pp. 415–444.

[CL10] R. Carmona and M. Ludkovski. "Valuation of energy storage: an optimal switching approach". In: *Quantitative Finance* 10.4 (2010), pp. 359–374.

[CMW18] Q. Chan-Wai-Nam, J. Mikael, and X. Warin. "Machine Learning for semi linear PDEs". In: (2018). arXiv: 1809.07609. URL: http://arxiv.org/abs/1809.07609.

[CPJ15] A. Cartea, J. Penalva, and S. Jaimungal. *Algorithmic and High-frequency trading*. Cambridge University Press, 2015.

[CR16] J-F. Chassagneux and A. Richou. "Numerical simulation of quadratic BSDEs". In: *The Annals of Applied Probabilities* 26.1 (2016), pp. 262–304.

[CST07] R. Cont, S. Stoikov, and R. Talreja. "A stochastic model for order book dynamics". In: *Operations Research* 58 (2007), pp. 549–563.

[CWNMW18] Q. Chan-Wai-Nam, J. Mikael, and X. Warin. "Machine Learning for semi linear PDEs". In: *arXiv preprint arXiv:1809.07609* (2018).

[Cyb89] G. Cybenko. "Approximations by superpositions of sigmoidal functions". In: *Mathematics of Control, Signals, and Systems* 2.4 (1989), pp. 303–314.

[EAA15] S. El Aoud and F. Abergel. "A stochastic control approach for options market making." In: *World scientific publishing company* 1(1) (2015).

[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 (2017), pp. 349–380.

[EK+97] N. El Karoui, C. Kapoudjian, E. Pardoux, S. Peng, and M.C. Quenez. "Reflected Solutions of Backward SDEs, and related obstacle problems for PDEs". In: *Annals of Probability* 25.2 (1997), pp. 702–737.

[FP15a] P. Fodra and H. Pham. "High Frequency trading and asymptotics for small risk aversion in a Markov renewal model". In: *SIAM Journal of Financial Mathematics* 6(1) (2015), pp. 656–684.

[FP15b] P. Fodra and H. Pham. "Semi Markov model for market microstructure". In: *Applied Mathematical Finance* 22(3) (2015), pp. 261–295.

[FPS18] L. Fiorin, G. Pagès, and A. Sagna. "Product Markovian quantization of a diffusion process with applications to finance". In: *Methodology and Computing in Applied Probability* (2018), pp. 1–32.

[GÍ7] A. Géron. *Deep Learning avec TensorFlow*. O'Reilly Media, 2017.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016.

[GHL10] J. Guyon and P. Henry-Labordère. "Uncertain volatility model: a Monte-Carlo approach". In: *SSRN* (2010).

[GKKW02] L. Gyorfi, M. Kohler, A. Krzyzak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer, 2002.

[GL00] S. Graf and H. Luschgy. *Foundations of quantization for probability distributions*. Vol. 1730. Springer-Verlag Berlin Heidelberg, 2000.

[GLFT12] O. Guéant, C.-A. Lahalle, and J. Fernandez-Tapia. "Dealing with the Inventory Risk". In: *Mathematics and Financial Economics* 7 (2012), pp. 477–507.

[GLW05] E. Gobet, J-P. Lemor, and X. Warin. "A regression-based Monte Carlo method to solve backward stochastic differential equations". In: *The Annals of Applied Probability* 15.3 (2005), pp. 2172–2202.

[Gob16] E. Gobet. *Monte-Carlo methods and stochastic processes: from linear to non-linear*. Chapman and Hall/CRC, 2016.

[GP13] F. Guilbaud and H. Pham. "Optimal High Frequency Trading with limit and market orders". In: *Quantitative Finance* 13(1) (2013), pp. 79–94.

[GS13] J. Gatheral and A. Schied. "Dynamical models of market impact and algorithms for order execution". In: *Handbook on Systemic Risk*. Cambridge University Press, 2013, pp. 579–602.

[Gu**1**6] O. Guéant. *The Financial Mathematics of Market Liquidity: From optimal execution to market making*. Chapman and Hall/CRC, 2016.

[GY04] P. Glasserman and B. Yu. "Simulation for American options: Regression Now or Regression Later?" In: *Monte Carlo and Quasi-Monte Carlo Methods 2002*. Ed. by Harald Niederreiter. Springer, 2004, pp. 213–226.

[HE16] J. Han and W. E. "Deep learning approximation for stochastic control problems". In: *arXiv:1611.07422* (2016).

[Hey+18] B. Heymann, J. F. Bonnans, P. Martinon, F. J. Silva, F. Lanas, and G. Jiménez-Estévez. "Continuous optimal control approaches to microgrid energy management". In: *Energy Systems* 9.1 (2018), pp. 59–77.

[HJE17] J. Han, A. Jentzen, and W. E. "Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning". In: *arXiv:1707.02568* (2017).

[HL+16] P. Henry-Labordere, N. Oudjane, X. Tan, N. Touzi, and X. Warin. "Branching diffusion representation of semilinear PDEs and Monte Carlo approximation". In: *Annales de l'Institut Henri Poincaré (B) Probabilités et Statistiques* (2016). to appear.

[HL17] P. Henry-Labordère. "Deep primal-dual algorithm for BSDEs: applications of machine learning to CVA and IM". In: *SSRN:3071506* (2017).

[HL18] J. Han and J. Long. "Convergence of the deep BSDE method for coupled FBSDEs". In: *arXiv:1811.01165v1* (2018).

[HLR15] W. Huang, C-A. Lehalle, and M. Rosenbaum. "Simulating and analyzing order book data: The queue-reactive model". In: *Journal of the American Statistical Association* 110(509) (2015), pp. 107–122.

[Hor91] K. Hornick. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks,* 4 (1991), pp. 251–257.

[HPBL18] C. Huré, H. Pham, A. Bachouch, and N. Langrené. "Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis". In: *arXiv:1812.04300* (2018).

[HPW19] C. Huré, H. Pham, and X. Warin. "Some machine learning schemes for high-dimensional nonlinear PDEs". In: *arXiv:1902.01599* (2019).

[HS79] T. Ho and H. Stoll. "Optimal dealer pricing under transactions and return uncertainty". In: *Journal of Financial Economics* 9 (1979), pp. 47–73.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[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.

[HSZD16] R. Hildebrand, J. Schoenmakers, J. Zhang, and F. Dickmann. "Regression based duality approach to optimal control with application to hydro electricity storage". In: *(preprint)* (2016).

[JL18] A. Jacquier and H. Liu. "Optimal Liquidation in a level-I Limit order book for large tick stocks". In: *SIFIN* 9(1) (2018), pp. 875–906.

[JLL90] P. Jaillet, D. Lamberton, and B. Lapeyre. "Variational Inequalities and the Pricing of American Options". In: *Acta Applicandae Mathematicae* 21(3) (1990), pp. 263–289.

[JP15] D. R. Jiang and W. B. Powell. "An approximate dynamic programming algorithm for monotone value functions". In: *Operations Research* 63.6 (2015), pp. 1489–1511.

[KKT10] M. Kohler, A. Krzyżak, and N. Todorovic. "Pricing of high-dimensional American options by neural networks". In: *Mathematical Finance* 20.3 (2010), pp. 383–410.

[KLP14] I. Kharroubi, N. Langrené, and H. Pham. "A numerical algorithm for fully nonlinear HJB equations: an approach by control randomization". In: *Monte Carlo Methods and Applications* 20.2 (2014), pp. 145–165.

[Koh06] M. Kohler. "Nonparametric regression with additional measurement errors in the dependent variable". In: *Journal of Statistical Planning and Inference* 136.10 (2006), pp. 3339–3361.

[Kol91] A. N. Kolmogorov. "On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables". In: *Mathematics and Its Applications (Soviet Series)* 25 (1991).

[KPX16] S. Kou, X. Peng, and X. Xu. "EM algorithm and stochastic control". Available at SSRN: https://ssrn.com/abstract=2865124. 2016.

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521 (2015), pp. 436–444.

[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.

[Li17] Y. Li. "Deep reinforcement learning: an overview". In: *arXiv 1701.07274v3* (2017).

[Lio12] P-L. Lions. "Théorie des jeux de champ moyen et applications (mean field games)". In: *Cours du College de France* (2012). URL: http://www.college-de-france.fr/default/EN/all/equder/audiovideo.jsp.

[LM18] M. Ludkovski and A. Maheshwari. "Simulation methods for stochastic storage problems: a statistical learning perspective". In: *arXiv:1803.11309* (2018).

[LOR18] C-A. Lehalle, M. Othmane, and M. Rosenbaum. "Optimal liquidity-based trading tactics". In: *arXiv:1803.05690* (2018).

[LS01] F. A. Longstaff and E. S. Schwartz. "Valuing American options by simulation: a simple least-squares approach". In: *The Review of Financial Studies* 14.1 (2001), pp. 113–147.

[Mas98] L. Massoulié. "Stability results for a general class of interacting point processes dynamics, and applications". In: *Stochastic Processes and their Applications* 75(1) (1998), pp. 1–30.

[MKSR15] V. Mnih, K. Kavukcuoglu, D. Silver, and A. A. Rusu. "Human-level control through deep reinforcement learning". In: *Nature* 518 (2015), pp. 529–533.

[ML09] M. Muja and D. Lowe. "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration". In: *International Conference on Computer Vision Theory and Applications (VISAPP)* (2009).

[Nie] M. Nielsen. *Neural networks and deep learning.*

[NMS17] S. Nadarajah, F. Margot, and N. Secomandi. "Comparison of least squares Monte Carlo methods with applications to energy real options". In: *European Journal of Operational Research* 256.1 (2017), pp. 196–204.

[PBT96] D. P. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[Pha16] H. Pham. "Linear quadratic optimal control of conditional McKean-Vlasov equation with random coefficients and applications". In: *Probability, Uncertainty and Quantitative Risk* 1.1 (2016), p. 7.

[Pow11] W. B. Powell. *Approximate dynamic programming: solving the curses of dimensionality.* Wiley & Sons, 2011.

[PP90] E. Pardoux and S. Peng. "Adapted solution of a backward stochastic differential equation". In: *Systems & Control Letters* 14.1 (1990), pp. 55–61.

[PPP04a] G. Pagès, H. Pham, and J. Printems. "An optimal Markovian quantization algorithm for multi-dimensional stochastic control problems". In: *Stochastics and Dynamics* 4.04 (2004), pp. 501–545.

[PPP04b] G. Pagès, H. Pham, and J. Printems. "Optimal quantization methods and applications to numerical problems in finance". In: *Handbook of computational and numerical methods in finance* (2004), pp. 253–297.

[PPP04c] G. Pagès, H. Pham, and J. Printems. "Optimal quantization methods and applications to numerical problems in finance". In: *Handbook on Numerical Methods in Finance*. Ed. by Svetlozar T. Rachev and George A. Anastassiou. Boston: Birkhäuser, 2004. Chap. 7, pp. 253–298.

[PW17] H. Pham and X. Wei. "Dynamic programming for optimal control of stochastic McKean-Vlasov dynamics". In: *SIAM Journal on Control and Optimization* 55.2 (2017), pp. 1069–1101.

[Ric10] A. Richou. "Etude théorique et numérique des équations différentielles stochastiques rétrogrades". PhD thesis. Université de Rennes 1, 2010.

[Ric11] A. Richou. "Numerical simulation of BSDEs with drivers of quadratic growth". In: *The Annals of Applied Probability* 21.5 (2011), pp. 1933–1964.

[Rog02] L. C. G. Rogers. "Monte Carlo Valuation of American Options". In: *Mathematical Finance* 12 (2002), pp. 271–286.

[Ros08] I. Rosu. "A dynamic model of the limit order book". In: *Review of Financial Studies* 22 (2008), pp. 4601–4641.

[SB98] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[Sil+16] D. Silver, A. Huang, Maddison C. J., A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529 (2016).

[Sil+17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. "Mastering the game of Go without human knowledge". In: *Nature* 550 (2017).

[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.

[War18a] X. Warin. "Monte Carlo for high-dimensional degenerated Semi Linear and Full Non Linear PDEs". In: *arXiv preprint arXiv:1805.05078* (2018).

[War18b] X. Warin. "Nesting Monte Carlo for high-dimensional Non Linear PDEs". In: *Monte Carlo Methods and Applications* (2018). to appear.

[Wil+17] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. "The marginal value of adaptive gradient methods in machine learning". In: 31st Conference on Neural Information Processing Systems (NIPS 2017). 2017.

[Zha04] J. Zhang. "A numerical scheme for BSDE's". In: *The Annals of Applied Probability* 14.1 (2004), pp. 459–488.

Méthodes Numériques et Réseaux de Neurones pour le Contrôle Stochastique et les Équations aux Dérivées Partielles

The present thesis deals with numerical schemes to solve Markov Decision Problems (MDPs), partial differential equations (PDEs), quasi-variational inequalities (QVIs), backward stochastic differential equations (BSDEs) and reflected backward stochastic differential equations (RBSDEs). The thesis is divided into three parts.

The first part focuses on methods based on quantization, local regression and global regression to solve MDPs. Firstly, we present a new algorithm, named *Qknn*, and study its consistency. A time-continuous control problem of market-making is then presented, which is theoretically solved by reducing the problem to a MDP, and whose optimal control is accurately approximated by *Qknn*. Then, a method based on Markovian embedding is presented to reduce McKean-Vlasov control problem with partial information to standard MDP. This method is applied to three different McKean-Vlasov control problems with partial information. The method and high accuracy of Qknn is validated by comparing the performance of the latter with some finite difference-based algorithms and some global regression-based algorithm such as regress-now and regress-later. In the second part of the thesis, we propose new algorithms to solve MDPs in high-dimension. Neural networks, combined with gradient-descent methods, have been empirically proved to be the best at learning complex functions in high-dimension, thus, leading us to base our new algorithms on them. We derived the theoretical rates of convergence of the proposed new algorithms, and tested them on several relevant applications. In the third part of the thesis, we propose a numerical scheme for PDEs, QVIs, BSDEs, and RBSDEs. We analyze the performance of our new algorithms, and compare them to other ones available in the literature on several tests, which illustrates the efficiency of our methods to estimate complex solutions in high-dimension.

Keywords: Deep Learning, MDPs, PDEs, optimal stopping time, BSDEs, RBSDEs, McKean-Vlasov, global regression, local regression, quantization, limit order book, algorithmic-trading, high-dimension.

Numerical Methods and Deep Learning for Stochastic Control Problems and Partial Differential Equations

La thèse porte sur les schémas numériques pour les problèmes de décisions Markoviennes (MDPs), les équations aux dérivées partielles (EDPs), les équation différentielles stochastiques rétrogrades (EDSRs), ainsi que les équations différentielles stochastiques rétrogrades réfléchies (EDSRs réfléchies). La thèse est divisée en trois parties.

La première partie porte sur des méthodes numériques pour résoudre les MDPs, à base de quantification et de régression locale ou globale. Un problème de market-making est proposé: il est résolu théoriquement en le réécrivant comme un MDP; et numériquement en utilisant le nouvel algorithme. Dans un second temps, une méthode de Markovian embedding est proposée pour réduire des problèmes de type McKean-Vlasov avec information partielle à des MDPs. Cette méthode est mise en œuvre sur trois différents problèmes de type McKean-Vlasov avec information partielle, qui sont par la suite numériquement résolus en utilisant des méthodes numériques à base de régression et de quantification. Dans la seconde partie de la thèse, on propose de nouveaux algorithmes pour résoudre les MDPs en grande dimension. Ces derniers reposent sur les réseaux de neurones, qui ont prouvé en pratique être les meilleurs pour apprendre des fonctions en grande dimension. La consistance des algorithmes proposés est prouvée, et ces derniers sont testés sur de nombreux problèmes de contrôle stochastique, ce qui permet d'illustrer leurs performances. Dans la troisième partie de la thèse, on s'intéresse à des méthodes basées sur les réseaux de neurones pour résoudre les EDPs, EDSRs et EDSRs réfléchies. La convergence des algorithmes proposés est prouvée; et ces derniers sont comparés à d'autres algorithmes récents de la littérature sur quelques exemples, ce qui permet d'illustrer leurs très bonnes performances.

Mots-clés: Réseaux de neurones, MDPs, EDPs, Temps d'arrêt optimal, EDPRs, EDPRs réfléchies, McKean-Vlasov, régression globale, regression locale, quantification, carnet d'ordres, algorithmic-trading, grande-dimension.