HAL open science

Stochastic control methods applied to portfolio construction, control with delay and PDE solving

William Lefebvre

▶ To cite this version:

William Lefebvre. Stochastic control methods applied to portfolio construction, control with delay and PDE solving. Optimization and Control [math.OC]. Université Paris Cité, 2022. English. NNT : 2022UNIP7099 . tel-04232758

HAL Id: tel-04232758

https://theses.hal.science/tel-04232758

Submitted on 9 Oct 2023

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

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

Université Paris Cité

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

Thèse de doctorat *

en Mathématiques Appliquées

Présentée par

William Lefebvre

Méthodes de contrôle stochastique appliquées à la construction de portefeuille, au contrôle avec retard et à la résolution d'EDPs

---

Stochastic control methods applied to portfolio construction, control with delay and PDE solving

---

Sous la direction de **Grégoire Loeper** et **Huyên Pham**

Soutenue le 09/12/2022 devant le jury composé de :

Carmine De Franco	Directeur R&D, OSSIAM	Examinateur
Julien Guyon	Professeur, Ecole des Ponts et Chaussées	Examinateur
Ruimeng Hu	Professeur, UC Santa Barbara	Rapporteur
Grégoire Loeper	Professeur, Monash University	Co-directeur de thèse
Huyên Pham	Professeur, Université Paris Cité	Directeur de thèse
Agnès Sulem	Directrice de recherche, INRIA Paris	Examinateur
Xiaolu Tan	Professeur, Chinese University of Hong Kong	Rapporteur

*Ce document est mis à disposition selon les termes de la Licence Creative Commons.

Remerciements

Je tiens avant tout à remercier mes deux directeurs de thèse: Huyên Pham et Grégoire Loeper, qui ont su me conseiller et me guider pendant ces trois (quatre) années de thèse. Vous avez toujours été disponibles pour discuter de nos projets de recherche et pour suggérer de nouvelles pistes et idées. Vous avez su être disponibles par Zoom pendant les confinements, qui ont duré pendant une partie non négligeable de ma thèse.

Je tiens également à remercier Ruimeng Hu et Xiaolu Tan pour avoir accepté d'être rapporteurs de cette thèse et pour le temps qu'ils ont pris pour relire attentivement ce manuscrit. Je remercie également Carmine de Franco, Julien Guyon et Agnès Sulem pour avoir accepté de participer à mon jury de soutenance.
Parmi les doctorants, je souhaite en particulier remercier Enzo Miller, pour le projet de recherche que nous avons partagé, et les multiples sessions Zoom durant lesquelles nous avons dessiné sur tablettes afin de comprendre comment le contrôle avec retard fonctionne, j'ai pris beaucoup de plaisir à mener ce projet de recherche avec toi. Je remercie également Maximilien Germain pour m'avoir initié à l'utilisation de Tensorflow, pour nos multiples discussions sur la résolution numérique d'EDP et de problèmes de contrôle, et le projet de recherche que nous avions entamé vers la fin de ma thèse!

Je remercie également les doctorants, en particulier ceux du bureau Serpentard (la meilleure maison), ayant soutenu ou encore en thèse, pour tous les cafés, déjeuners et discussions que nous avons partagé et qui ont contribué à faire régner une bonne ambiance au sein du laboratoire (et je m'excuse d'avance si certains noms ont été oubliés): Adrien, Antoine, Assaf, Barbara, Benjamin, Bogdan, Clément, Clément, Côme, Cyril, Fabio, Guillaume, Hiroshi, Houzhi, Johann, Junchao, Laure, Lucas, Mamadou, Marc, Médéric, Mi-Song, Mohan, Sothea, Sylvain, Yann, Yiyang, Ziad. Bon courage également aux nouveaux: Hoang Dung, Mohamed, Nathan et Nisrine. J'espère que les années de thèse qu'il vous reste seront passionnantes et riches en belles découvertes.

Je remercie également Nathalie, Valérie et Amina, qui m'ont beaucoup aidé pour mes démarches administratives.

Je voudrais également remercier mes collègues de BNP Paribas, qui ont contribué à rendre mon expérience en entreprise agréable: Jérôme (avec qui j'ai partagé bien des heures de débuggage et de tests statistiques), Fatma, Kyle, Alexandre et Guillaume.

Je remercie également les professeurs qui m'ont accompagné au cours de ma scolarité, et en particulier Mr. Lepilier, qui m'a donné envie de poursuivre des études scientifiques.

Je remercie enfin mes parents pour m'avoir soutenu pendant mes (longues) études, pour m'avoir fait réciter mes lecons le soir pendant de nombreuses années et m'avoir appris qu'il est important de bien travailler à l'école; et Diane pour m'avoir accompagné et supporté pendant ces années de thèse, avec leurs hauts et leurs bas, et pour m'avoir aidé à traduire dans l'urgence une partie de l'introduction de ce manuscrit.

Mes derniers remerciements vont enfin aux courageux qui assisteront à ma soutenance, ayant lieu inhabituellement tôt, ou qui liront ce manuscrit.

Méthodes de contrôle stochastique appliquées à la construction de portefeuille, au contrôle avec retard et à la résolution d'EDPs

Résumé : La présente thèse porte sur les méthodes de contrôle stochastique appliquées à la résolution de problèmes survenant dans le domaine de la finance quantitative, tels que la sélection de portefeuille et la résolution d'EDP non linéaires associées à la construction de stratégies d'investissement et à l'évaluation de produits dérivés. Elle est divisée en trois parties.

Dans la première partie, nous résolvons un problème de sélection de portefeuille de type moyenne-variance où le portefeuille est pénalisé par une distance entre la richesse investie dans chacun de ses actifs et la composition d'un portefeuille de référence à poids fixe. Les formules analytiques du contrôle optimal et de la fonction valeur sont obtenues et un analogue de la formule de frontière efficiente est obtenu dans la limite où la pénalisation tend vers zéro. La robustesse de cette allocation est testée sur des prix de marché simulés dans le cas où les paramètres de diffusion sont mal spécifiés.

La deuxième partie traite du contrôle d'équations différentielles stochastiques avec retard. Nous résolvons un problème de contrôle stochastique linéaire quadratique simple où le contrôle apparaît à la fois dans la partie "drift" et dans la partie diffusion de l'EDS de l'état et est affecté par un retard. Les expressions analytiques du contrôle optimal et de la fonction valeur sont obtenues en termes de solution d'un système d'EDP de Riccati couplées pour lesquelles l'existence et l'unicité d'une solution sont prouvées, à condition qu'une hypothèse, combinant l'horizon temporel, le retard, le "drift" et la volatilité de l'EDS de l'état, soit satisfaite. Une méthode d'apprentissage profond est utilisée pour résoudre le système d'EDP de Riccati dans le contexte de la sélection de portefeuille de Markovitz avec délai d'exécution.

Dans la troisième partie, trois méthodes basées sur l'apprentissage profond sont définies afin de résoudre des EDP entièrement non linéaires avec un hamiltonien convexe. Ces méthodes utilisent la représentation stochastique de l'EDP, dont le contrôle optimal est approximé numériquement, afin d'obtenir trois estimateurs différents de la solution de l'EDP basés sur les versions régressive ou trajectorielle de la représentation martingale et de sa dérivée. La solution et ses dérivées sont ensuite calculées simultanément. Nous exploitons ensuite nos méthodes pour concevoir des algorithmes permettant de résoudre des familles d'EDP avec une condition terminale paramétrique au moyen de réseaux de neurones DeepOnet.

Mots-clés: Problème de portefeuille moyenne-variance en temps continu, erreur de suivi, allocation robuste, paramètres mal spécifiés, contrôle stochastique linéaire quadratique, retard, EDP de Riccati, portefeuille de Markowitz, EDP non linéaires, apprentissage profond, apprentissage différentiel, valorisation d'option avec impact sur le marché

Articles inclus dans ce manuscrit:

- William Lefebvre, Gregoire Loeper, and Huyên Pham. "Mean-variance portfolio selection with tracking error penalization". In: *Mathematics* 8.11 (2020), p. 1915.
- William Lefebvre and Enzo Miller. "Linear-quadratic stochastic delayed control and deep learning resolution". In: *Journal of Optimization Theory and Applications* 191.1 (2021), pp. 134–168.
- William Lefebvre, Grégoire Loeper, and Huyên Pham. "Differential learning methods for solving fully nonlinear PDEs". In: *arXiv preprint arXiv:2205.09815* (2022).

Stochastic control methods applied to portfolio construction, control with delay and PDE solving

Abstract : The present thesis deals with stochastic control methods applied to the resolution of problems arising in the field of quantitative finance, such as portfolio selection and resolution of non linear PDEs associated to the construction of investment strategies and the pricing of derivative products. It is divided into three parts.

In the first part, we solve a mean variance portfolio selection problem where the portfolio is penalized by a distance between the wealth invested in each of its assets and the composition of a reference portfolio with fixed weights. The optimal control and value function are obtained in closed form and an analogue of the efficient frontier formula is obtained in the limit where the penalisation tends to zero. The robustness of this allocation is tested on simulated market prices with parameter misspecification.

The second part deals with the delayed control of stochastic differential equations. We solve a simple linear quadratic stochastic control problem where the control appears both in the drift and diffusion part of the state SDE and is affected by a delay. The expressions of the optimal control and value function are obtained in terms of the solution of a system of coupled Riccati PDEs for which the existence and uniqueness of a solution is proven, provided that a condition, combining the time horizon, the delay, the drift and the volatility of the state SDE is satisfied. A deep learning method is used to solve the system Riccati PDEs in the context of Markovitz portfolio selection with execution delay.

In the third part, three methods based on deep learning are defined in order to solve fully non linear PDEs with convex Hamiltonian. These methods use the stochastic representation form of the PDE, whose optimal control is approximated numerically, in order to obtain three different estimators of the PDE solution based on regression or pathwise versions of the martingale representation and its differential relation. The solution and its derivatives are then computed simultaneously. We further leverage our methods to design algorithms for solving families of PDEs with parametric terminal condition by means of DeepOnet neural networks.

Keywords: Continuous-time mean-variance problem, tracking error, robustified allocation, parameter misspecification, Linear-quadratic stochastic control, delay, Riccati PDEs, Markowitz portfolio allocation, Fully nonlinear PDEs, deep learning, differential learning, option pricing with market impact.

List of papers being part of this thesis:

- William Lefebvre, Gregoire Loeper, and Huyên Pham. "Mean-variance portfolio selection with tracking error penalization". In: *Mathematics* 8.11 (2020), p. 1915.
- William Lefebvre and Enzo Miller. "Linear-quadratic stochastic delayed control and deep learning resolution". In: *Journal of Optimization Theory and Applications* 191.1 (2021), pp. 134–168.
- William Lefebvre, Grégoire Loeper, and Huyên Pham. "Differential learning methods for solving fully nonlinear PDEs". In: *arXiv preprint arXiv:2205.09815* (2022).

Contents

1 Introduction (en francais) 1
1.1 Portefeuille moyenne-variance avec erreur de suivi . . . 2
1.1.1 Solution d'allocation avec erreur de suivi . . . 3
1.2 Contrôle avec retard . . . 8
1.2.1 Résultats de vérification et d'existence . . . 9
1.2.2 Schéma d'apprentissage profond pour les EDPs . . . 12
1.2.3 Application à la construction de portefeuille de Markowitz avec délai d'exécution . . . 14
1.3 Algorithmes d'apprentissage profond pour les EDPs complètement non linéaires . . . 17
1.3.1 Méthodes d'apprentissage différentiel . . . 19

2 Introduction 31
2.1 Mean-variance allocation with tracking error . . . 32
2.1.1 Solution allocation with tracking error . . . 33
2.2 Control with delay . . . 37
2.2.1 Verification and existence results . . . 38
2.2.2 Deep learning scheme for PDEs . . . 42
2.2.3 Application to Markowitz portfolio allocation with execution delay 43
2.3 Deep learning algorithms for fully nonlinear PDEs . . . 46
2.3.1 Differential learning methods . . . 48

I Mean-Variance with tracking error 59

3 Mean-variance portfolio selection with tracking error penalization 61
3.1 Introduction . . . 61
3.2 Results . . . 63
3.3 Solution allocation with tracking error . . . 65
3.4 Applications and numerical results . . . 70
3.4.1 Performance comparison with Monte Carlo simulations . . . 72
3.4.2 Performance comparison on a backtest . . . 76
3.5 Conclusions . . . 79
3.A Proof of Theorem 3.3.1 . . . 81
3.B Computation linear expansion of α^γ for $\Gamma = \gamma \mathbb{I}_d \to \mathbf{0}$ . . . 83
3.C Computation linear expansion of $Var(X_T)$ for $\Gamma = \gamma \mathbb{I}_d \to 0$ . . . 83
3.D Proof that K_t and Λ_t are bounded in γ . . . 84

II Stochastic delayed systems 87

4 Linear-quadratic stochastic delayed control and deep learning resolution 89

4.1 Introduction . . . 89
4.2 Formulation of the problem and heuristic approach . . . 91
4.3 Verification and existence results . . . 94
4.4 Deep learning scheme . . . 99
4.4.1 A quick reminder of PINNs and Deep Galerkin method for PDEs . 99
4.4.2 A tailor-made algorithm for Riccati partial differential equations . 100
4.5 Applications to mean-variance portfolio selection with execution delay . . 102
4.5.1 One asset with delay . . . 102
4.5.2 One asset with delay and one without . . . 105
4.A Proof of Proposition 4.3.2 . . . 112
4.A.1 Slice $t \in [T-d, T]$, initialization . . . 112
4.A.2 Slice $[T-2d, T-d]$ . . . 112
4.A.3 From slice $[T-nd, T]$ to $[T-(n+1)d, T]$ . . . 116

III Differential learning for non-linear PDEs 119

5 Differential learning methods for solving fully nonlinear PDEs 121

5.1 Introduction . . . 121
5.2 Dual stochastic control representation of fully nonlinear PDE . . . 124
5.3 Differential regression learning . . . 126
5.4 Pathwise learning . . . 129
5.4.1 Pathwise martingale learning . . . 129
5.4.2 Pathwise differential learning . . . 129
5.5 Numerical results . . . 131
5.5.1 Approximation of the optimal control . . . 131
5.5.2 Differential regression learning algorithm . . . 132
5.5.3 Pathwise learning algorithms . . . 134
5.5.4 Validation tests . . . 137
5.5.5 Example of Merton portfolio selection . . . 138
5.5.6 Example of the Black-Scholes model with linear market impact . . 139
5.6 Further step: resolution for parametric terminal functions . . . 148
5.6.1 Theory and network structure . . . 148
5.6.2 Application to the Black-Scholes model with linear market impact 152
5.A Alternative algorithms using multiple neural networks . . . 157

Bibliography 165

Chapitre 1

Introduction (en francais)

Les premiers mois de mon travail de doctorat ont été consacrés à un projet de recherche, mené en collaboration avec l'équipe de structuration QIS de BNP Paribas, qui a abouti à la publication de l'article [GLT20]. Ce travail, qui ne sera pas présenté dans le corps de ce manuscrit, a consisté à identifier les facteurs (facteurs macroéconomiques et techniques calculés à partir de la courbe des taux des obligations d'État) qui déterminent la valeur des contrats à terme (ou *futures*) sur obligations d'État. L'objectif est de choisir un petit ensemble de variables les plus pertinentes dans un grand ensemble de facteurs corrélés, une méthode de sélection de variables robuste, non affectée par les corrélations, doit être utilisée. Dans cette étude, le Lasso adaptatif, défini dans [Zou06], a été utilisé pour sélectionner ces facteurs, avec un paramètre de pénalisation L_1 choisi en minimisant un critère d'information bayésien modifié défini dans [Cha12].

Cette thèse est divisée en trois parties pouvant être lues indépendamment. Dans la première partie, nous utilisons l'approche dite de martingale faible (*weak martingale*) pour les problèmes de contrôle stochastique linéaire-quadratique de McKean-Vlasov pour résoudre un problème d'allocation de portefeuille moyenne-variance en temps continu avec une pénalisation de l'erreur de suivi (*tracking error*) par rapport à un portefeuille de référence. Dans ce problème, les contrôles correspondent à la richesse investie dans chaque actif risqué considéré et la pénalisation de la *tracking error* consiste en une distance entre ces contrôles et la composition d'un portefeuille de référence ayant la même richesse totale et des poids fixes. Nous dérivons le contrôle optimal en fonction des solutions d'un système d'ODEs de Riccati, ainsi qu'un théorème de vérification pour ce problème. Un développement asymptotique de l'expression de la variance de la richesse du portefeuille terminal en fonction de son espérance est obtenue dans le cas où le facteur de pénalisation tend vers zéro, analogue de la *formule de frontière efficiente* de la théorie classique du portefeuille moyenne-variance. La robustesse de cette allocation de portefeuille pénalisée dans le cas d'une mauvaise spécification des paramètres de diffusion des actifs est illustrée par des simulations de Monte Carlo et un backtest de l'allocation sur des données historiques de prix est réalisé.

Dans la deuxième partie, nous considérons une classe de problèmes de contrôle stochastique où le contrôle est présent à la fois dans la partie *drift* et dans la partie diffusion de l'EDS du système, et où l'état est contrôlé avec un retard (ou latence). Le contrôle optimal et la valeur du problème sont obtenus en termes de solutions d'un ensemble d'EDP de transport couplées par leurs termes sources. L'existence et l'unicité d'une solution de ce système d'EDP sont prouvées. Une condition suffisante d'existence, émergeant directement de la présence du retard, et fonction des coefficients de diffusion et du retard,

est fournie. Parallèlement, un théorème de vérification de l'optimalité de la fonction de contrôle et de valeur est prouvé. Un schéma d'apprentissage profond est proposé pour résoudre le système d'EDP et le contrôle optimal est calculé pour le problème d'allocation de portefeuille moyenne-variance avec retard avec un actif retardé et, de manière heuristique, dans le cas où un actif est contrôlé avec retard et l'autre instantanément. L'effet du retard sur cette allocation de portefeuille est ainsi illustré.

La troisième partie de cette thèse est consacrée à la définition d'algorithmes d'apprentissage profond pour résoudre des EDPs entièrement non linéaires avec hamiltonien convexe sur un l'ensemble d'un domaine. A partir de la convexité de l'hamiltonien de l'EDP, l'EDP est réécrite sous sa forme duale conjuguée. A partir de là, nous étudions le problème de contrôle stochastique associé et dérivons trois estimateurs de la solution de l'EDP et de sa dérivée première en fonction du contrôle optimal de ce problème. L'estimation de la solution de l'EDP est ensuite réalisée en utilisant un réseau neuronal et en entraînant sa valeur et sa dérivée première pour toutes nos méthodes, et en entraînant en plus sa dérivée seconde pour une de nos méthodes. La précision de ces méthodes est ensuite illustrée en résolvant numériquement les EDP associées au problème de Merton et une EDP correspondant à un problème de *pricing* sous un modèle Black Scholes avec impact de marché linéaire. La précision de nos solutions est évaluée en les comparant à la solution calculée sur un ensemble de points par Monte Carlo, et en calculant une perte associée au résidu de l'EDP et à la condition aux limites. Enfin, une méthode similaire basée sur les réseaux de neurones DeepOnet est définie afin de résoudre une EDP non linéaire avec une condition limite paramétrique pour chaque valeur de son paramètre se trouvant dans un ensemble compact. La précision et la capacité de généralisation de cette méthode sont testées en résolvant la même EDP de *pricing* avec une condition terminale correspondant au *payoff* d'une option d'achat (*call*).

1.1 Portefeuille moyenne-variance avec erreur de suivi

La Partie I est consacrée à l'étude d'un problème de sélection de portefeuille moyenne-variance où une erreur de suivi par rapport à un portefeuille de même richesse totale avec des poids fixes est ajoutée au critère de moyenne-variance. Cette optimisation peut être interprétée de deux facons. C'est un moyen de stabiliser l'allocation moyenne-variance en l'ancrant à une allocation de référence qui peut être indépendante de l'estimation des paramètres du marché, comme c'est le cas avec le portefeuille à poids égaux ou le portefeuille à poids nuls, qui correspond à une pénalisation de la norme L_2 des contrôles. L'autre facon d'interpréter ce paradigme d'allocation est de le voir comme un portefeuille avec des poids désirés fixes qui est optimisé davantage selon un critère de moyenne-variance sur sa richesse terminale. D'autres approches visant à rendre robuste l'allocation de portefeuille par optimisation sous l'incertitude du modèle ont été considérées, voir [NN18] et [PWZ22] pour des exemples.

Le problème de sélection de portefeuille moyenne-variance, initialement considéré par Markowitz dans [Mar52] dans un cadre à une période, est un problème de contrôle stochastique de McKean-Vlasov (MKV) lorsqu'il est formulé en temps continu. Ce type de problème de contrôle a été largement étudié dans la littérature récente, voir [DPT22]. En effet, le problème peut être formulé comme la minimisation de la fonction de coût suivante

$$J(\alpha) = \mu \mathrm{Var}(X_T) - \mathbb{E}[X_T], \tag{1.1.1}$$

qui dépend de la loi du processus X_T au temps final T. Ce processus représente la

richesse totale du portefeuille et évolue selon une dynamique donnée par l'EDS

$$dX_t = \alpha_t^\top b\mathrm{d}t + \alpha_t \sigma \mathrm{d}W_t,$$

où le contrôle $\alpha \in \mathbb{R}^d$ représente la richesse investie dans chacun des d actifs risqués considérés. Le marché financier sous-jacent est composé d'un actif sans risque dont le prix est considéré comme constant et égal à un, $P^0 = 1$, et d actifs risqués dont les prix $P := (P_t)_{t\in[0,T]}$ évoluent selon l'EDS

$$\begin{aligned} \mathrm{d}P_t &= P_t^\top \Big(b\ \mathrm{d}t + \sigma \mathrm{d}W_t \Big), \quad t \in [0,T], \\ P_0^i &> 0, \quad i = 1, ..., d. \end{aligned} \tag{1.1.2}$$

Une première approche pour résoudre ce problème, développée dans [ZL00], consiste à le considérer comme un cas particulier d'un problème de contrôle auxiliaire standard pouvant être résolu à l'aide de la théorie du controle stochastique linéaire quadratique. La construction du portefeuille moyenne-variance est alors résolue en traitant ce problème auxiliaire. Des approches récentes permettent de résoudre directement ce problème de construction de portefeuille comme un problème de contrôle de McKean-Vlasov, voir [PW17], [AD11], [FL16], pour quelques exemples. Dans cette Partie, nous considérons une version modifiée du problème (1.1.1) où un coût courant est ajouté

$$J(\alpha) = \mu \mathrm{Var}(X_T) - \mathbb{E}[X_T] + \mathbb{E}\Big[\int_0^T (\alpha_t - w_r X_t)^\top \Gamma (\alpha_t - w_r X_t) \mathrm{d}t \Big]. \tag{1.1.3}$$

Nous traitons ce problème de contrôle linéaire-quadratique de type McKean-Vlasov à l'aide de l'approche utilisant le principe d'optimalité martingale faible, développée dans citebasei2019weak. Le coût courant ajouté dans l'équation (1.1.3), que nous appelons pénalisation de *tracking error* est introduit de manière à assurer que le portefeuille de l'investisseur ne s'éloigne pas trop de la composition d'un portefeuille de référence $w_r X_t$, ayant la même richesse totale X_t et des poids constants w_r, au sens de la distance $|M| = M^\top \Gamma M$, où la matrice $\Gamma \in \mathbb{R}^{d\times d}$ est symétrique et définie positive. Nous ne considérons que des poids de référence constants, cependant nos résultats peuvent facilement être étendus au cas où les poids de références sont adaptés à la filtration du marché.

Le problème de construction de portefeuille moyenne-variance avec *tracking-error* est alors formulé comme

$$V_0 := \inf_{\alpha \in \mathcal{A}} J(\alpha), \tag{1.1.4}$$

où $\mathcal{A}$ désigne l'espace des contrôles admissibles et le coût J correspond au coût présent dans l'équation (1.1.3). Une allocation optimale pour le coût $J(\alpha)$ est définie comme

$$\alpha_t^* \in \arg\min_{\alpha \in \mathcal{A}} J(\alpha).$$

1.1.1 Solution d'allocation avec erreur de suivi

Comme mentionné précédemment, nous utilisons le principe d'optimalité faible pour résoudre le problème (1.1.4). L'idée est de définir une famille de processus à valeur réelle $\{V_t^\alpha,\ t \in [0,T], \alpha \in \mathcal{A}\}$ de la forme

$$V_t^\alpha = v_t(X_t^\alpha, \mathbb{E}[X_t^\alpha]) + \int_0^t (\alpha_s - w_r X_s^\alpha)^\top \Gamma (\alpha_s - w_r X_s^\alpha) \mathrm{d}s,$$

où $v : \mathbb{R}_+ \times \mathbb{R} \times \mathbb{R}$ est une fonction mesurable telle que $v_T(x, \overline{x}) = \mu(x - \overline{x})^2 - x$ pour tout x, $\overline{x} \in \mathbb{R}$, et la fonction $t \in [0, T] \mapsto \mathbb{E}[V_t^\alpha]$ est non décroissante $\alpha \in \mathcal{A}$. A partir du lemme d'optimalité martingale faible développé dans [BP19], la stratégie optimale est alors exprimée comme le contrôle $\alpha^* \in \mathcal{A}$ telle que la fonction $t \in [0, T] \mapsto \mathbb{E}[V_t^{\alpha^*}]$ soit constante.

Pour notre problème, nous cherchons une fonction v_t mesurable de la forme

$$v_t(x, \overline{x}) = K_t(x - \overline{x})^2 + \gamma_t \overline{x}^2 + 2Y_t x + R_t,$$

pour des processus déterministes $(K_t, \Lambda_t, Y_t, R_t)$. En calculant la dynamique du processus $\mathbb{E}[V_t^\alpha]$ obtenu, nous déterminons que (K_t, Λ_t) doit être solution des EDOs suivantes

$$\begin{cases} \mathrm{d}K_t &= \Big\{(K_t b - \Gamma w_r)^\top S_t^{-1}(K_t b - \Gamma w_r) - w_r^\top \Gamma w_r\Big\}\mathrm{d}t, \quad K_T = \mu, \\ \\ \mathrm{d}\Lambda_t &= \Big\{(\Lambda_t b - \Gamma w_r)^\top S_t^{-1}(\Lambda_t b - \Gamma w_r) - w_r^\top \Gamma w_r\Big\}\mathrm{d}t, \quad \Lambda_T = 0, \end{cases} \tag{1.1.5}$$

et que les processus (Y_t, R_t) doivent avoir la forme analytique suivante

$$\begin{cases} Y_t = -\frac{1}{2} e^{-\int_t^T b^\top S_s^{-1}(\Lambda_s b - \Gamma w_r)\mathrm{d}s}, \quad \forall t \in [0, T], \\ \\ R_t = \frac{1}{2}\int_t^T b^\top S_s^{-1} b \; e^{-2\int_s^T b^\top S_u^{-1}(\Lambda_u b - \Gamma w_r)\mathrm{d}u}\mathrm{d}s, \quad \forall t \in [0, T], \end{cases} \tag{1.1.6}$$

où $S_t := K_t \Sigma + \Lambda$, avec $\Sigma = \sigma\sigma^\top$.

Result 1: Théorème de vérification

Il existe une unique paire $(K, \Lambda) \in C([0, T], \mathbb{R}_+^*) \times C([0, T], \mathbb{R}_+)$ solution du système d'EDOs (1.1.5). Le contrôle optimale du problème (1.1.4) est alors donné par

$$\alpha_t^\Gamma = S_t^{-1}\Gamma w_r X_t - S_t^{-1} b\Big[K_t X_t + Y_t - (K_t - \Lambda_t)\mathbb{E}[X_t]\Big], \tag{1.1.7}$$

et $X = X^{\alpha^\Gamma}$ est le processus de richesse associé à α^Γ. De plus, on a

$$V_0 = J(\alpha^\Gamma) = \Lambda_0 X_0^2 + 2Y_0 X_0 + R_0.$$

Nous voyons que le contrôle optimal est constitué de deux composantes, une première determinée par le vecteur $S_t^{-1}\Gamma w_r = (K_t\Sigma + \Gamma)^{-1}\Gamma w_r$ et entièrement liée à la présence de la pénalisation de *tracking error*, et une seconde déterminée par le vecteur $S_t^{-1} b = (K_t\Sigma + \Gamma)^{-1} b$.

Dans le cas où Γ est la matrice nulle, la première composante disparaît, et nous vérifions que la seconde composante est égale au contrôle optimal obtenu dans le cas du problème classique de construction du portefeuille moyenne-variance. En effet, quand $\Gamma = \mathbf{0}$,

$$Y_t = -\frac{1}{2}, \quad R_t = \frac{1}{4\mu}\left(1 - e^{b^\top \Sigma^{-1} b\ (T-t)}\right),$$

et le système d'EDOs (1.1.6) devient

$$\begin{cases} \mathrm{d}K_t &= K_t b^\top \Sigma^{-1} b\ \mathrm{d}t, \quad K_T = \mu, \\ \mathrm{d}\Lambda_t &= \frac{\Lambda_t^2}{K_t} b^\top \Sigma^{-1} b\ \mathrm{d}t, \quad \Lambda_T = 0, \end{cases}$$

ce qui donne les solutions explicites suivantes

$$K_t = \mu e^{-b^\top \Sigma^{-1} b\ (T-t)}, \quad \Lambda_t = 0.$$

En placant des expressions dans l'équation du contrôle optimal (1.1.7) et en calculant l'expression de la richesse moyenne $\mathbb{E}[X_t]$, nous pouvons exprimer le contrôle optimal α^Γ comme une fonction de la richesse initiale x_0 et de la richesse actuelle X_t de l'investisseur, et nous obtenons l'expression

$$\alpha_t^{\Gamma=0} = \Sigma^{-1} b \left[\frac{1}{2\mu} e^{b^\top \Sigma^{-1} b\ T} + x_0 - X_t \right],$$

qui correspond à la solution du problème classique de construction de portefeuille moyenne-variance.

Dans le cas où $\Gamma = \gamma 1_d$, avec $\gamma \in \mathbb{R}_+^*$ avec 1_d la matrice identité de $\mathbb{R}^{d\times d}$, le contrôle optimal peut être réécrit de la manière suivante

$$\alpha_t^\gamma = \left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma \right)^{-1} w_r X_t - \frac{1}{\gamma} \left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma \right)^{-1} b \left[K_t X_t + Y_t - (K_t - \Lambda_t)\overline{X}_t \right].$$

Nous montrons que la solution K_t et Λ_t sont des fonctions bornées du paramètre de pénalisation γ, permettant d'obtenir les limites $\frac{K_t}{\gamma}$, $\frac{\Lambda_t}{\gamma} \underset{\gamma\to\infty}{\longrightarrow} 0$. Nous obtenons alors la limite suivante pour le contrôle optimal dans le cas où $\gamma \to \infty$

$$\alpha_t^\gamma \underset{\gamma\to\infty}{\longrightarrow} w_r X_t,$$

ce qui montre que dans la limite ou la pénalisation de tracking-error tend vers l'infini, l'allocation converge vers la composition du portefeuille de référence.

En fixant à nouveau $\Gamma = \gamma 1_d$, nous pouvons obtenir le développement du contrôle optimal α^γ dans le cas où le paramètre de pénalisation $\gamma \to 0$. Dans ce cas, en calculant le développement asymptotique de S_t^{-1}, K_t et Λ_t jusqu'à l'ordre linéaire en γ, nous obtenons le résultat suivant.

Result 2: Développement asymptotique pour $\Gamma = \gamma 1_d \to 0$

Quand le paramètre de pénalisation de *tracking error* $\gamma \to 0$, nous pouvons écrire le développement asymptique du contrôle optimal α^γ jusqu'au terme linéaire en γ comme

$$\alpha_t^\gamma = \Sigma^{-1} b\ \alpha_t^0 + \gamma \left(\Sigma^{-1} w_r\ \alpha_t^{1,3} - \Sigma^{-2} b\ \alpha_t^{1,2} - \Sigma^{-1} b\ \alpha_t^{1,1} \right) + O(\gamma^2),$$

où $\Sigma^{-2} := (\Sigma^{-1})^2$ et avec

$$\alpha_t^0 = \frac{1}{2\mu} e^{\rho T} + X_0 - X_t,$$

$$\alpha_t^{1,1} = \frac{\|w_r\|^2}{K_t^0}(T-t)\left(X_0 + \frac{e^{\rho T}}{\mu}\left(1 - e^{-\rho t}\right) \right) + X_0 C_{0,t}^1 + \frac{H_t^1}{2} + \frac{K_t^1}{2\left(K_t^0\right)^2} + \frac{C_{t,T}}{2K_t^0},$$

$$\alpha_t^{1,2} = \frac{e^{\rho T}}{2K_t^0 \mu}\left(1 - e^{-\rho t}\right) + \frac{1}{2\left(K_t^0\right)^2},$$

$$\alpha_t^{1,3} = \frac{X_t}{K_t^0}.$$

Dans ce développement, nous voyons clairement que pour $\gamma = 0$, nous retrouvons la solution du problème classique de portefeuille moyenne-variance. Pour des valeurs non nulles de γ, le poids associé à l'allocation donnée par le vecteur $\Sigma^{-1}b$ est modifié et deux allocations données par $\Sigma^{-2}b$ et $\Sigma^{-1}w_r$ apparaissent, avec les poids $\gamma\alpha_t^{1,2}$ et $\gamma\alpha_t^{1,3}$ respectivement.

A partir de ce développement du contrôle, nous pouvons aussi calculer de développement asymptotique au premier ordre en γ de l'équation donnant la relation entre la variance de la richesse terminale du portefeuille et son espérance, une relation appelée "formule de la frontière efficiente" dans le cas du portefeuille moyenne-variance classique.

Result 3: Développment de la variance de la richesse terminale pour $\Gamma = \gamma 1_d \to 0$

Le développement asymptotique à l'ordre linéaire de la variance de la richesse terminale du portefeuille X_T est donné par

$$\begin{aligned} Var(X_T) =& \frac{e^{-\rho T}}{1-e^{-\rho T}}\left(\overline{X_T}^0 - X_0\right)^2 \\ &+ \gamma\left\{\frac{b^\top\Sigma^{-1}w_r}{\mu^2}\left[X_0 T - \frac{1}{2\mu}e^{\rho T}\left(T - \frac{1-e^{-\rho T}}{\rho}\right)\right]\right. \\ &\left. - \int_0^T \left(\frac{\rho}{\mu}\alpha_s^{1,1} + \frac{b^\top\Sigma^{-2}b}{\mu}\alpha_s^{1,2}\right)e^{-\rho(T-s)}\mathrm{d}s\right\} + O(\gamma^2). \end{aligned}$$

Le terme d'ordre supérieur correspond à l'équation de frontière efficiente de l'allocation classique moyenne-variance calculée dans [ZL00], et pour $\gamma = 0$, nous retrouvons ce résultat classique. Nous voyons que ce résultat est cohérent avec le développement asymptotique obtenu pour le contrôle optimal α^γ, avec un terme linéaire en γ contenant les trois mêmes allocations perturbatives $\Sigma^{-1}b$, $\Sigma^{-2}b$ et $\Sigma^{-1}w_r$.

1.1.1.1 Résultats numériques

Nous appliquons ensuite ces résultats au calcul de l'allocation optimale de portefeuille pour quatre portefeuilles de référence statiques différents. Nous exécutons ces allocations sur des scénarios de marché simulés dans le cas de paramètres mal spécifiés, ce qui signifie que la diffusion utilisée pour simuler les prix du marché n'est pas la même que celle utilisée pour calculer l'allocation optimale. Il est connu que l'allocation classique moyenne-variance est très sensible à l'estimation des paramètres de diffusion du marché, et donc à une mauvaise spécification des paramètres. Ces simulations permettent donc de tester la robustesse de l'allocation moyenne-variance avec pénalisation de *tracking error* et de voir si elle est plus robuste à la mauvaise spécification des paramètres que l'allocation moyenne-variance classique. Nous supposons toujours que la matrice de pénalisation peut être écrite comme $\Gamma = \gamma 1_d$ et nous résolvons numériquement les EDO des processus K_t et Λ_t afin d'obtenir le contrôle optimal α^γ. Nous calculons l'allocation pénalisée avec quatre portefeuilles de référence différents

- Portefeuille à poids égaux

$$w_r^{\text{ew}} = \frac{1}{d}e,$$

- Portfefeuille à variance minimale

$$w_r^{\text{min-var}} = \frac{\Sigma^{-1}e}{e^\top\Sigma^{-1}e},$$

- Portefeuille à poids nuls (pénalisation des contrôles)

$$w_r^{\text{shrinking}} = \mathbf{0},$$

- Portefeuille à contributions égales au risque

$$w_r^{\text{erc}} = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{d} \sum_{j=1}^{d} \left(w^i \left(\Sigma w\right)^i - w^j \left(\Sigma w\right)^j \right)^2$$
$$\text{s.t } e^\top w = 1 \text{ and } 0 \leq w^i \leq 1, \ \forall i \in [\![1, d]\!].$$

Afin de comparer la performance de ces quatre allocations moyenne-variance pénalisées dans le cas de paramètres mal estimés, nous utilisons une simulation Monte Carlo où les prix des actifs évoluent selon l'EDS (1.1.2) et où les vrais rendements moyens b_{real} et covariances σ_{real} sont égaux au rendements moyens b_0 et covariances σ_0 auxquels s'ajoute un bruit :

$$b_{\text{real}} = b_0 + \epsilon \times \text{noise}, \quad \sigma_{\text{real}} = \sigma_0 + \epsilon \times \text{noise}.$$

Le bruit suit une distribution normale $\mathcal{N}(0,1)$ multipliée par un facteur $\epsilon \in \mathbb{R}_+^*$. Nous utilisons les simulations de Monte Carlo pour estimer l'espérance du ratio de Sharpe de chaque portefeuille, égal à la moyenne des rendements quotidiens du portefeuille R divisée par l'écart type de ces rendements : $\frac{\mathbb{E}[R]}{\text{Stdev}(R)}$.

La richesse initiale x_0 de l'investisseur est choisie égale à 1 et nous choisissons le paramètre de pénalisation $\gamma = \mu/100$. En effet, comme la valeur de μ dépend de la valeur de l'espérance de rendement, de la matrice de covariance des actions et du rendement visé, et peut être très grande, nous exprimons γ en fonction de ce μ afin que la pénalisation soit pertinente et non négligeable. Pour chaque portefeuille de référence, nous comparons le portefeuille de référence, l'allocation classique moyenne-variance et l'allocation pénalisée pour des valeurs de l'amplitude du bruit ϵ allant de 0 à 1. À titre d'exemple, nous montrons dans la figure 1.1 et 1.2 une comparaison du ratio de Sharpe du portefeuille classique moyenne-variance avec le portefeuille pénalisé par un portefeuille de référence à poids égaux et par un portefeuille de référence à poids nuls respectivement. Ces deux portefeuilles ne reposent pas sur des estimations des paramètres du marché et nous pouvons voir sur ces graphiques que si le portefeuille à moyenne variance atteint une meilleure performance lorsque la mauvaise spécification des paramètres est petite, pour des valeurs plus grandes de l'erreur d'estimation, le portefeuille pénalisé atteint de meilleurs ratios de Sharpe.

Pour compléter ces exemples, nous comparons les différentes allocations sur un *backtest* basé sur les cours de clôture quotidiens disponibles sur Quandl entre 2013-09-03 et 2017-12-28 pour quatre actions : Apple, Microsoft, Boeing et Nike. Nous considérons deux valeurs du facteur de pénalisation $\gamma = \mu$ et $\gamma = \mu/100$. Les figures 1.3 et 1.4 montrent la richesse totale en fonction du temps des portefeuilles moyenne-variance classique et pénalisés par un portefeuille de référence à poids égaux et par un portefeuille de référence à poids nuls respectivement. On constate que les portefeuilles pénalisés se situent entre les portefeuilles de moyenne-variance et de référence et atteignent des ratios de Sharpe supérieurs à ceux du portefeuille de moyenne-variance non pénalisé.

Dans le cas du portefeuille pénalisé par le portefeuille de référence à poids nuls, présenté dans la Figure 1.4, l'allocation pénalisée correspond à une pénalisation de la richesse investie dans les actifs risqués. Sur ce graphique, la richesse des portefeuilles est normalisée par l'écart-type de leurs rendements quotidiens et nous voyons que la richesse normalisée du portefeuille pénalisé est supérieure à la richesse du portefeuille non pénalisé.

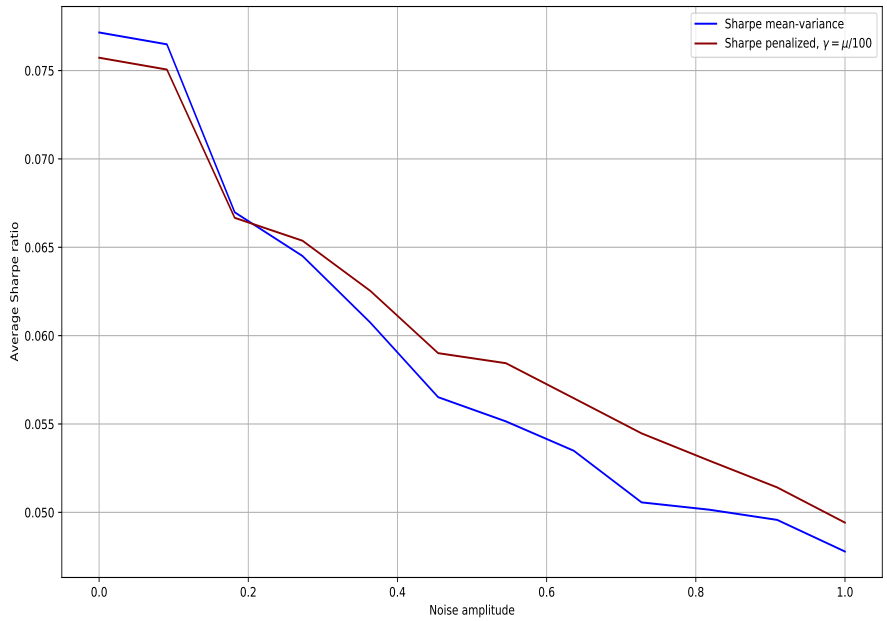


FIGURE 1.1 : Le ratio de Sharpe le plus élevé atteint par le portefeuille à poids égaux est égal à 0.047 pour $\epsilon = 0$.

1.2 Contrôle avec retard

La Partie II est consacrée au contrôle avec retard de systèmes stochastiques. Le temps nécessaire pour acquérir l'information, calculer la décision et l'exécuter, avec des systèmes souvent affectés par une latence entre le moment où une commande est envoyée et le moment où elle est exécutée, rend les retards omniprésents dans les systèmes de contrôle. Divers effets des retards sur la modélisation des flux de trafic, les processus chimiques, la dynamique des populations, la chaîne d'approvisionnement et la publicité ont été étudiés dans la littérature.

Dans un système contrôlé avec retard, l'état X et le contrôle α sont les deux principaux termes qui peuvent présenter un retard. Lorsque le retard est uniquement présent dans la variable d'état, le problème est maintenant bien compris et peut être traité en plongeant la variable d'état dans un espace de dimension infinie $(X_t, s \in [-d, 0] \mapsto X_{t+s})$, voir [DM72], [FGG10] pour n'en citer que quelques-uns. Une situation beaucoup moins bien comprise est celle où le retard se situe dans la variable de contrôle. Dans cette situation, deux approches principales ont émergé : la méthode dite de *stuctural state* et la méthode dite d'*extended state*, nous nous référons à [Ben+07, Partie II, Chapitre 3] pour l'étude de ces dernières dans le cas déterministe et [FF14] pour l'approche de *structural state* dans le cas stochastique. Pour une liste complète de références, voir également [FF14].

Dans ce travail, l'objectif a été d'étudier la situation où la partie *drift* et la partie diffusive de l'état sont contrôlée avec un retard. Nous considérons la classe de problème

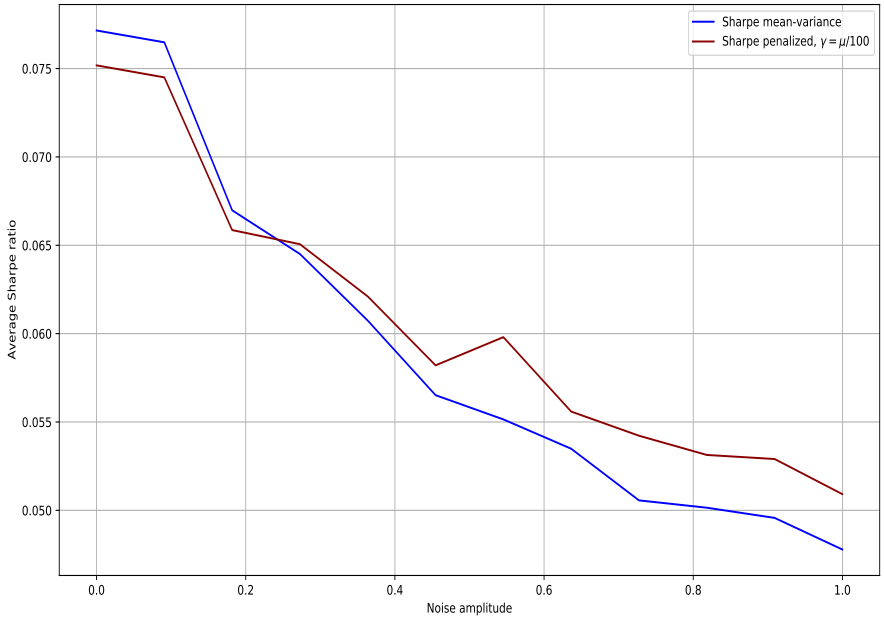


FIGURE 1.2 : Dans le cas où les poids de référence sont égaux à zéro, aucun ratio de Sharpe n'est calculé pour le portefeuille de référence.

de contrôle stochastique retadé suivante

$$\begin{cases} dX_t^\alpha = \alpha_{t-d}\left(bdt + \sigma dW_t\right), \qquad 0 \le t \le T, \\ X_0 = x, \quad \alpha_s = \gamma(s), \qquad s \in [-d, 0], \quad x \in \mathbb{R} \\ J(\alpha) = \mathbb{E}[(X_T^\alpha)^2]. \end{cases} \tag{1.2.1}$$

Sauf dans [FF14], cette situation n'est pas traitée théoriquement ni numériquement dans les références ci-dessus. La principale difficulté vient du fait que les problèmes d'optimisation avec un contrôle retardé appartiennent naturellement à la classe des *boundary control problems*.

1.2.1 Résultats de vérification et d'existence

Notre approche est inspirée de l'approche *extended state* initiée par Ichikawa, voir [Ich82], où l'idée clée est de projeter l'espace initiale de l'état, à savoir $\mathbb{R}$, dans un espace de Hilbert de dimension infinie $H = \mathbb{R} \times L^2([-d, 0], \mathbb{R})$, doté du produit interne

$$\langle x, y \rangle_H = x_0 y_0 + \int_{-d}^{0} x_1(s) y_1(s) ds \qquad x, y \in H.$$

Pour chaque élément $(x, u) \in H$, le premier élément x peut être interprété comme la position ou l'état du système, tandis que le second élément $s \in [-d, 0] \mapsto u(s)$ peut être vu comme la mémoire du contrôle appliqué, contenant à chaque temps t les valeurs passées du contrôle aux instants $[t - d, t]$. Notre prochain résultat étend au cas stochastique

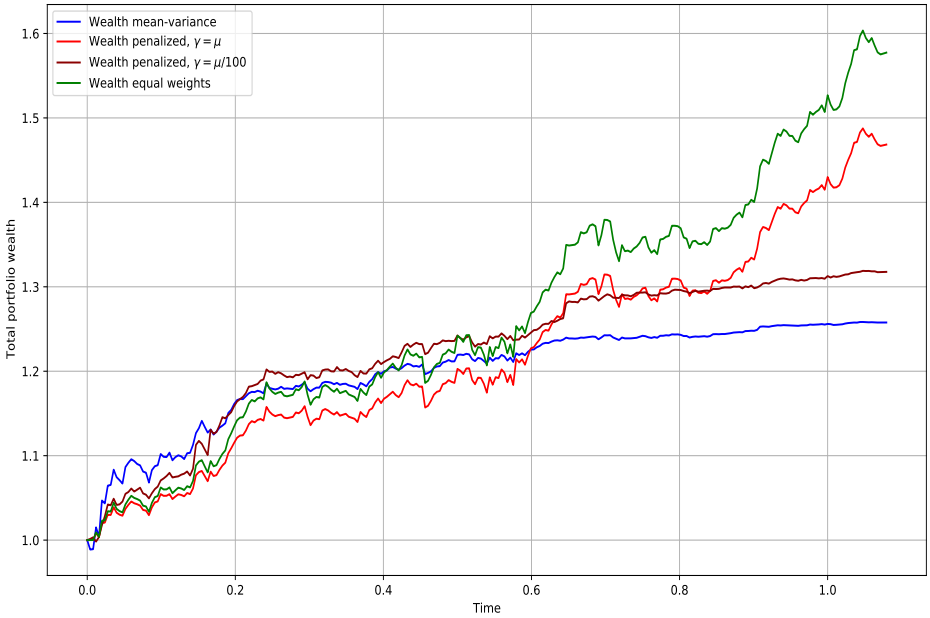


FIGURE 1.3 : Ratios de Sharpe :
Moyenne-variance : 0.183
Poids égaux : 0.258
Pénalisé pour $\gamma = \mu$: 0.260
Pénalisé pour $\gamma = \mu/100$: 0.226

les résultats développées dans [Ale+71] et [Ich82] dans le cas où un système déterministe est contrôlé avec du retard. La fontion valeur et le contrôle optimal du problème (1.2.1) sont exprimés en terme d'un opérateur auto-adjoint borné à valeurs positives $P \in C([0,T], \mathcal{L}(H,H))$ de la forme

$$P_t : (x, \gamma(\cdot)) \mapsto \begin{pmatrix} P_{11}(t)x + \int_{-d}^{0} P_{12}(t,s)\gamma(s)ds \\ P_{12}(t,\cdot)x + P_{\hat{2}\hat{2}}(t,\cdot)\gamma(\cdot) + \int_{-d}^{0} P_{22}(t,\cdot,s)\gamma(s)ds \end{pmatrix}.$$

qui satisfait le système suivant d'équations aux dérivées partielle de transport, couplées par leurs termes sources

$$\begin{aligned} \dot{P}_{11}(t) &= \frac{P_{12}(t,0)^2}{P_{\hat{2}\hat{2}}(t,0)}, & (\partial_t - \partial_s)(P_{12})(t,s) &= \frac{P_{12}(t,0)P_{22}(t,s,0)}{P_{\hat{2}\hat{2}}(t,0)}, \\ (\partial_t - \partial_s)(P_{\hat{2}\hat{2}})(t,s) &= 0, & (\partial_t - \partial_s - \partial_r)(P_{22})(t,s,r) &= \frac{P_{22}(t,s,0)P_{22}(t,0,r)}{P_{\hat{2}\hat{2}}(t,0)}, \end{aligned} \tag{1.2.2}$$

avec les conditions aux bords

$$\begin{aligned} P_{12}(t,-d) &= bP_{11}(t), & P_{\hat{2}\hat{2}}(t,-d) &= \sigma^2 P_{11}(t), \\ P_{22}(t,s,-d) &= bP_{12}(t,s), & P_{22}(t,-d,r) &= bP_{12}(t,r), \end{aligned} \tag{1.2.3}$$

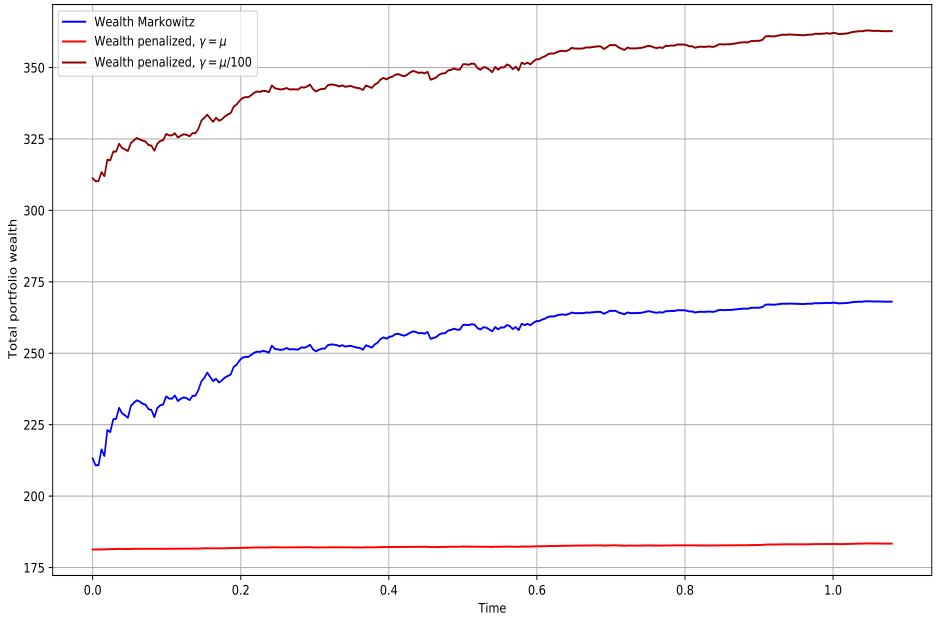


FIGURE 1.4 : Richesse totale pour le portefeuille moyenne-variance et les portefeuilles pénalisés pour $\gamma = \mu$ et $\gamma = \mu/100$, en fonction du temps, normalisée par l'écart-type des rendements quotidiens.
Ratios de Sharpe :
Moyenne-variance : 0.183
Pénalisé pour $\gamma = \mu$: 0.252
Pénalisé pour $\gamma = \mu/100$: 0.221

et les conditions terminales

$$P_{11}(T) = 1, \qquad P_{12}(T,s) = P_{\hat{22}}(T,s) = P_{22}(T,s,r) = 0, \tag{1.2.4}$$

pour presque tous $s, r \in [-d, 0]$.

Result 4: Théorème de vérification

Supposons qu'il existe une solution P à (1.2.2)-(1.2.3)-(1.2.4). Alors le contrôle optimal du problème d'optimisation (1.2.1) est exprimé comme suit

$$\alpha_t^* = \frac{-1_{t \leq T-d}}{P_{\hat{22}}(t,0)} \left\{ X_t^{\alpha^*} P_{11}(t,0) + \int_{t-d}^{t} P_{22}(t,0,s-t)\alpha_s^* ds \right\},$$

et la fonction valeur est donnée par

$$V(z) = \langle P_0 z, z \rangle_H,$$

où $z = (x, \gamma) \in H$ désigne l'état initial du système contrôlé.

La prochaine étape est de démontrer l'existence d'une solution $P = (P_{11}, P_{12}, P_{\hat{22}}, P_{22})$ aux système d'EDPs (1.2.2)-(1.2.3)-(1.2.4). Dans le cas où le système est contrôlé sans retard, $d = 0$, bien que le contrôle ne soit pas pénalisé dans le coût J du problème (1.2.1), le problème d'optimisation (1.2.1) admet un optimiseur à condition que $\sigma \neq 0$.

En effet, plus α est agressif pour amener la valeur terminale X_T^2 à zéro, plus la variation quadratique de X augmente à cause du terme de diffusion. Comme cette variation quadratique contient le terme de contrôle, celui-ci est implicitement pénalisé et le problème reste donc bien posé. Cela est facilement observable dans le problème classique d'optimisation stochastique linéaire quadratique où la partie *drift* et la partie diffusion sont contrôlées, tel que

$$\begin{aligned} dX_t^\alpha &= \alpha_t(bdt + \sigma dW_t), \qquad t \leq T, \\ X_0 &= x, \\ J(\alpha) &= \mathbb{E}[(X_T^\alpha)^2], \end{aligned}$$

où le contrôle optimal et exprimé comme $\alpha_t^* = -\frac{b}{\sigma^2} X_t^{\alpha^*}$ et la fonction valeur $V_t = e^{(t-T)\frac{b^2}{\sigma^2}}$. Un résultat surprenant dans notre travail est la nécessité d'une contrainte plus restrictive sur le coefficient de diffusion lorsque le retard n'est pas nul, $d > 0$. En effet, nous constatons qu'une solution au système (1.2.2)-(1.2.3)-(1.2.4) existe et est unique lorsqu'une certaine relation entre l'horizon temporel T, le retard d et les paramètres de *drift* et de volatilité b et σ est satisfaite.

Soit $a = (a_n)_{n\geq 1}$ la suite suivante

$$\begin{cases} a_0 &= 1, \\ a_{n+1} &= a_n - \frac{d}{a_n}\left(\frac{b}{\sigma}\right)^2, \qquad n \geq 0, \end{cases}$$

et soit $\mathcal{N} : (d, b, \sigma) \mapsto \inf\{n \geq 1 : a_n > 0 \text{ and } a_{n+1} \leq 0\}$. Nous exprimons le résultat d'existence de P en terme de la séquence a.

Result 5: Existence de $t \in [0, T] \mapsto P_t$

Supposons que $T < \mathcal{N}(d, b, \sigma)d$. Alors (1.2.2)-(1.2.3)-(1.2.4) a une solution unique P sur $[0, T]$ avec $0 < P_{11}(0) < 1$.

1.2.2 Schéma d'apprentissage profond pour les EDPs

Nous proposons maintenant un algorithme d'approximation de la solution $t \mapsto P_t$. Nous décidons de nous appuyer sur des techniques de *machine learning* pour résoudre le système d'EDP couplées, dans l'esprit de la littérature émergente sur les *Physics Informed Neural Network* (PINN) et de *deep Galerkin*, voir [SS18] et [RPK19] pour n'en citer que quelques-uns.

Nous rappelons ici certaines des idées principales. Supposons que nous voulions résoudre numériquement l'équation différentielle partielle suivante

$$\begin{aligned} \partial_t u + \mathcal{N}(u) &= 0, \qquad \text{on } \Omega, \\ u &= g, \qquad \text{on } \partial\Omega, \end{aligned} \tag{1.2.5}$$

où $\mathcal{N}$ est un opératuer non linéaire, Ω est un sous-ensemble ouvert borné et g est une fonction de la frontière du domaine. where $\mathcal{N}$ is a nonlinear operator, Ω a bounded open subset and g a function on the boundary of the domain. L'idée clé est d'utiliser un réseau neuronal comme substitut (ou fonction de test) à la solution u de (1.2.5). Notons

$t \mapsto u(t, \Theta)$ un tel réseau, où Θ désigne ses paramètres et t un élément générique de $\Omega \cup \partial\Omega$. L'algorithme repose sur la minimisation de la fonction de perte suivante sur des échantillons de $\Omega \cup \partial\Omega$ tirés aléatoirement :

$$\mathcal{L}(\Theta, \mathcal{T}) = \mathcal{L}_u(\Theta, \mathcal{T}) + \mathcal{L}_f(\Theta, \mathcal{T}), \tag{1.2.6}$$

où $\mathcal{L}_u$ et $\mathcal{L}_f$ sont définies comme

$$\mathcal{L}_r(\Theta, \mathcal{T}) = \frac{1}{|\mathcal{T}_r|} \sum_{t \in \mathcal{T}_r} |(\partial_t + \mathcal{N})u(t, \Theta)|^2, \quad \mathcal{L}_f(\Theta, \mathcal{T}) \quad = \frac{1}{|\mathcal{T}_f|} \sum_{t \in \mathcal{T}_f} |u(t, \Theta) - g(t)|^2,$$

et servent respectivement à entraîner les dérivées partielles du réseau à satisfaire l'EDP et ses valeurs à satisfaire la condition terminale. Les sous-ensembles $\mathcal{T}_r = \mathcal{T} \cap \Omega$ et $\mathcal{T}_f = \mathcal{T} \cap \partial\Omega$ sont respectivement des sous-ensembles aléatoires de Ω and $\partial\Omega$. Nous résumons la procédure numérique dans l'Algorithme 1.

Algorithm 1: Schéma d'apprentissage profond pour résoudre des EDPs

Initialiser :
la vitesse d'apprentissage η et le réseau de neurones $u(\cdot, \Theta)$; ;

Pour chaque *batch* :
Echantillonner aléatoirement $\mathcal{T} \subset \partial\Omega \cup \Omega$;
Calculer le gradient de la fonction de coût (1.2.6) :
$\nabla_\Theta \mathcal{L}(\Theta, \mathcal{T}) = \nabla_\Theta (\mathcal{L}_r + \mathcal{L}_f)(\Theta, \mathcal{T})$;
Actualiser $\Theta \leftarrow \Theta - \eta \nabla_\Theta \mathcal{L}(\Theta, \mathcal{T})$;

Return : L'ensemble des paramètres optimisés Θ^*.

La structure précise de (1.2.2)-(1.2.3)-(1.2.4) nous a conduit vers un algorithme spécialement adapté.

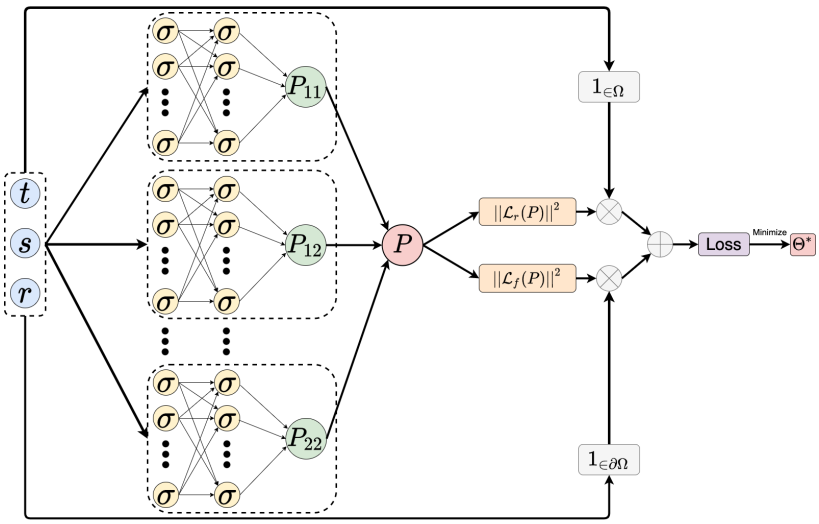


FIGURE 1.5 : Structure du réseau de neurones utilisé pour résoudre (1.2.2)-(1.2.3)-(1.2.4).

Commentaires sur la méthode : La méthode *Deep Galerkin* offre une procédure facile à mettre en oeuvre pour approximer les solutions de toute équation différentielle sur l'ensemble du domaine $\Omega \cup \partial\Omega$. En théorie, elles permettent de résoudre des problèmes en grande dimension, bien que d'autres recherches soient attendues sur ce sujet. Dans la Partie II, nous adaptons la méthode Deep Galerkin pour résoudre le système d'EDP (1.2.2)-(1.2.3)-(1.2.4) dans le contexte de l'allocation de portefeuille de Markowitz avec délai d'exécution. Dans notre cas, la dimension de sortie est de 4.

Néanmoins, la méthode *Deep Galerkin* ne s'étend pas bien au cas où l'espace d'entrée est de haute dimension, de l'ordre de 100 par exemple. Ce cas est important car il est courant en finance, par exemple dans le cas de problèmes d'allocation de portefeuille, où un grand nombre d'actifs risqués sont considérés, et dans d'autres domaines tels que la recherche opérationnelle, la physique, etc. Dans ces cas, les méthodes basées sur les BSDE, telles que celles présentées dans [HJW18] sont bien adaptées.

1.2.3 Application à la construction de portefeuille de Markowitz avec délai d'exécution

Nous appliquons maintenant notre schéma numérique au problème de sélection de portefeuille moyenne-variance, voir [Mar52], où les actifs risqués sont traités avec un délai, dans l'esprit du problème de couverture d'options européennes avec délai d'exécution présenté dans [FF14]. Le problème de sélection de portefeuille moyenne-variance en temps continu consiste à résoudre le problème contraint suivant

$$\begin{cases} \min_{\alpha \in \mathcal{A}} \mathrm{Var}(X_T^\alpha) \\ \text{t.q. } \mathbb{E}[X_T^\alpha] = c. \end{cases} \tag{1.2.7}$$

où X^α dénote la richesse de l'investisseur contrôlée par une stratégie d'investissement α. Dans notre travail, nous étudions d'abord le cas où un seul actif risqué est considéré, et est traité avec un délai,

$$\begin{cases} dX_t^\alpha = \alpha_{t-d}\left((\sigma\lambda)\,dt + \sigma dW_t\right), \qquad t \in [0,T], \\ X_0 = x_0, \quad \alpha_s = \gamma_s, \qquad \forall s \in [-d, 0], \end{cases} \tag{1.2.8}$$

où

où α désigne la richesse investie dans l'actif risqué, et λ et σ sont des constantes représentant respectivement la prime de risque et la volatilité de l'actif risqué. Notez que le cas d'un actif retardé permet une application directe des résultats 4 et 5 :

Result 6: Cas d'un actif avec retard

Supposons que $T < d\mathcal{N}(d, (\sigma\lambda), \sigma)$, fixons $\xi^* = c - \eta^*$ et

$$\eta^* = \frac{K(\gamma) + P_{11}(0)(x_0 - c)}{1 - P_{11}(0)},$$
$$K(\gamma) = \int_{-d}^{0} \gamma_s P_{12}(0, s) ds.$$

La stratégie d'investissement $\alpha^*(\xi)$ est définie comme

$$\alpha_t^*(\xi^*) = \frac{-1_{t \leq T-d}}{P_{\hat{22}}(t,0)} \left\{ (X_t^{\alpha^*} - \xi^*) P_{12}(t,0) + \int_{t-d}^{t} \alpha_s^*(\xi) P_{22}(t, 0, s-t) ds \right\},$$

où P désigne la solution de (1.2.2)-(1.2.3)-(1.2.4). Alors, le problème d'optimisation (1.2.7)-(1.2.8) admet $\alpha^*(\xi)$ comme stratégie *feedback* optimale admissible et la valeur optimale, correspondant à la variance de la richesse terminale du portefeuille, est donnée par

$$\begin{aligned}\mathrm{Var}(X_T^{\alpha^*}) = &\frac{P_{11}(0)}{1 - P_{11}(0)} (x_0 - c + K(\gamma))^2 \\ &+ \int_{-d}^{0} \gamma_s^2 P_{\hat{22}}(0, s) ds + \int_{[-d,0]^2} \gamma_s \gamma_u P_{22}(0, s, r) ds dr.\end{aligned}$$

Résultats numériques : Du résultat précédent émerge une application directe pour le schéma numérique présenté dans la section précédente. Voir les figures 1.6 et 1.7 pour quelques exemples.

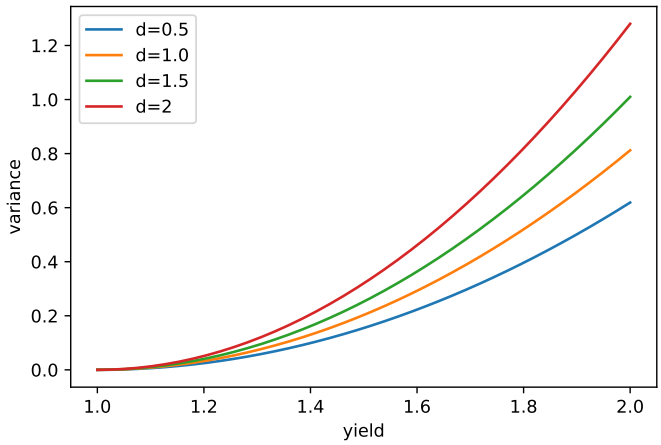
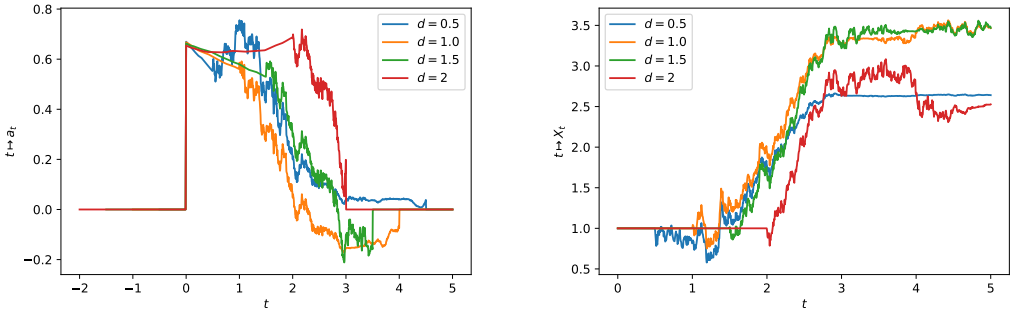


FIGURE 1.6 : Frontière efficiente avec $\sigma = 1$, $\lambda = 0.5$, $T = 5$ et $\gamma \equiv 0$.

Afin d'étudier davantage l'effet du retard sur le problème de contrôle, nous étudions également, dans la Partie II, le problème de sélection de portefeuille moyenne-variance, dans le cas où deux actifs corrélés risqués sont considérés, l'un étant traité avec un délai



(a)

FIGURE 1.7 : Stratégies optimales et portefeuilles optimaux avec $c = 1,6$, $\sigma = 1$, $\lambda = 0,5$, et $T = 5$. Gauche : $t \mapsto \alpha^*$, droite : $t \mapsto X_t^*$. Notez l'effet de déstabilisation et l'augmentation de volatilité provoqué par la présence du retard. Notez également la tendance à investir de manière plus agressive de l'investisseur retardé, car il a moins de temps pour obtenir le rendement voulu. $\xi^*(d = 0.5) = 2.57$, $\xi^*(d = 1) = 2.68$, $\xi^*(d = 1.5) = 2.80$, $\xi^*(d = 2) = 2.97$.

et l'autre instantanément :

$$\begin{cases} dX_t^{(\alpha,\beta)} = \alpha_t \left\{(\sigma_1\lambda_1)dt + \sigma_1 dW_t^1\right\} + \beta_{t-d}\left\{(\sigma_2\lambda_2)dt + \sigma_2 dW_t^2\right\}, \quad t \in [0,T], \\ X_0 = x_0, \quad \beta_s = \gamma_s, \quad s \in [-d,0], \\ \langle W^1, W^2\rangle_t = \rho t, \end{cases} \tag{1.2.9}$$

où α désigne la richesse investie dans l'actif risqué sans retard, β la richesse investie dans l'actif risqué avec retard, et λ_i et σ_i sont des constantes représentant respectivement les primes de risque et les volatilités des actifs risqués.

En suivant une approche heuristique, nous définissons l'ensemble suivant d'EDP de Riccati sur le domaine $[0,T] \times [-d,0]^2$, qui apparaîtra dans l'expression du contrôle optimal et de la fonction valeur de ce problème,

$$\begin{aligned} &\dot{P}_{11}(t) = \lambda_1^2 P_{11}(t) + \frac{P_{12}(t,0)^2}{P_{\hat{2}\hat{2}}(t,0)}, \\ &(\partial_t - \partial_s)(P_{12})(t,s) = \lambda_1^2 P_{12}(t,s) + \frac{P_{12}(t,0)P_{22}(t,s,0)}{P_{\hat{2}\hat{2}}(t,0)}, \\ &(\partial_t - \partial_s)(P_{\hat{2}\hat{2}})(t,s) = 0, \\ &(\partial_t - \partial_s - \partial_r)(P_{22})(t,s,r) = \lambda_1^2 \frac{P_{12}(t,s)P_{12}(t,r)}{P_{11}(t)} + \frac{P_{22}(t,s,0)P_{22}(t,0,r)}{P_{\hat{2}\hat{2}}(t,0)}, \end{aligned} \tag{1.2.10}$$

accompagnées des conditions aux bords, pour presque tout $t, s \in [0,T] \times [-d,0]$

$$\begin{aligned} &P_{12}(t,-d) = \lambda_2\sigma_2\left(1 - \rho\frac{\lambda_1}{\lambda_2}\right)P_{11}(t), \qquad P_{\hat{2}\hat{2}}(t,-d) = \sigma_2^2\left(1-\rho^2\right)P_{11}(t), \\ &P_{22}(t,s,-d) = \lambda_2\sigma_2\left(1 - \rho\frac{\lambda_1}{\lambda_2}\right)P_{12}(t,s), \quad P_{22}(t,-d,s) = \lambda_2\sigma_2\left(1 - \rho\frac{\lambda_1}{\lambda_2}\right)P_{12}(t,s), \end{aligned} \tag{1.2.11}$$

et les conditions terminales

$$P_{11}(T) = 1, \qquad P_{12}(T,s) = P_{\hat{2}\hat{2}}(T,s) = P_{22}(T,s,r) = 0, \tag{1.2.12}$$

pour presque tout $s, r \in [-d, 0]$.

Nous avons maintenant tous les éléments pour présenter la stratégie et la valeur optimales du problème (1.2.1) dans le cas de 2 actifs.

Result 7: Cas d'un actif avec retard et un actif sans

Supposons qu'il existe une solution P au système (1.2.10)-(1.2.11)-(1.2.12). Définissons $\xi^* = c - \eta^*$ et

$$\eta^* = \frac{K(\gamma) + P_{11}(0)(x_0 - c)}{1 - P_{11}(0)},$$
$$K(\gamma) = \int_{-d}^{0} \gamma_s P_{12}(0, s) ds.$$

Définissons $(\alpha^*(\xi^*), \beta^*(\xi^*))$ la stratégie d'investissement

$$\alpha_t^*(\xi) = -\left\{ \frac{\lambda_1}{\sigma_1} (X_t^* - \xi) + \rho \frac{\sigma_2}{\sigma_1} \beta_{t-d}^* + \frac{\lambda_1}{\sigma_1 P_{11}(t)} \int_{t-d}^{t} \beta_s^*(\xi) P_{12}(t, s-t) ds \right\}.$$
$$\beta_t^*(\xi) = \frac{-1_{t \leq T-d}}{P_{\hat{2}\hat{2}}(t, 0)} \left\{ P_{12}(t, 0) (X_t^* - \xi) + \int_{t-d}^{t} \beta_s^*(\xi) P_{22}(t, 0, r-t) dr \right\},$$

Alors, le problème d'optimisation (1.2.7)-(1.2.9) admet $(\alpha^*(\xi^*), \beta^*(\xi^*))$ comme contrôle optimal *feedback*, et la fonction valeur est donnée par

$$\mathrm{Var}(X_T^{\alpha^*}) = \frac{P_{11}(0)}{1 - P_{11}(0)} (x_0 - c + K(\gamma))^2 + \int_{-d}^{0} \gamma_s^2 P_{\hat{2}\hat{2}}(0, s) ds + \int_{[-d,0]^2} \gamma_s \gamma_u P_{22}(0, s, r) ds dr.$$

Simulations numériques : La flexibilité du schéma d'apprentissage profond 1 permet une adaptation facile au nouvel ensemble d'EDP de Riccati (1.2.10)-(1.2.11)-(1.2.12). La Figure 1.8 permet de voir l'influence du retard et de la corrélation ρ. Une présentation plus approfondie sera faite dans la Partie II.

1.3 Algorithmes d'apprentissage profond pour les EDPs complètement non linéaires

La Partie III est dédiée à la résolution d'équations aux dérivées partielles complètement non linéaires de la forme

$$\begin{cases} \partial_t u + H(x, D_x u, D_x^2 u) &= 0, \quad (t, x) \in [0, T) \times \mathbb{R}^d, \\ u(T, x) &= g(x), \quad x \in \mathbb{R}^d, \end{cases} \tag{1.3.1}$$

où l'Hamiltonien $H : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^{d \times d} \to \mathbb{R} \cup \{\infty\}$ est une fonction convexe semi-continue inférieurement par rapport à ses deux derniers arguments (z, γ), et g est une fonction mesurable sur $\mathbb{R}^d$. La résolution de cette classe d'équations aux dérivées partielles et la bonne approximation de la seconde dérivée spatiale $D_x^2 u$ de la solution est une problème d'une difficulté notoire. Nous proposons des algorithmes pour résoudre ce problème, fondés sur des techniques d'apprentissage profond, dans le but d'estimer la valeur de

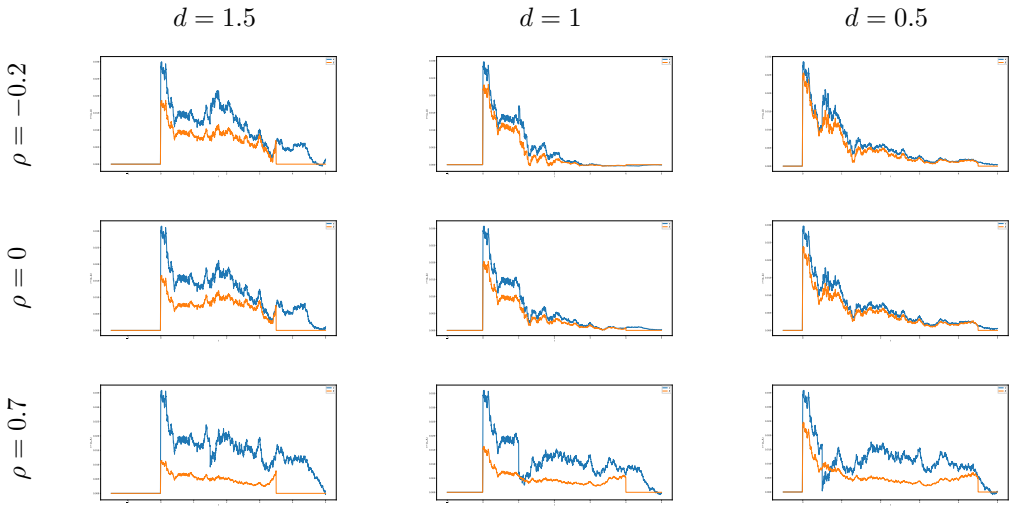


FIGURE 1.8 : $t \mapsto (\alpha_t^*, \beta_t^*)$, avec $\sigma_1 = \sigma_2 = 1$, $\lambda_1 = \lambda_2 = 0.5$ et $T = 5$. Bleu : α^*, orange : β^*. Les mêmes réalisations de W et B ont été utilisées pour toutes les expériences. Notez que plus les actifs sont positivement corrélés, plus l'actif non retardé est favorisé.

la solution EDP et de ses deux premières dérivées spatiales $D_x u$ and $D_x^2 u$ sur l'intégralité d'un domaine. Les méthodes d'apprentissage profond appliquées à la résolution d'équations aux dérivées partielles ou à des problèmes de contrôle stochastique ont fait l'objet d'une littérature récente et abondante, et ont été appliquées avec succès dans des situations où des méthodes numériques traditionnelles échouent (voir [HL22] pour une revue des avancées récentes de ces méthodes). Ces méthodes sont d'un grand intérêt pour plusieurs domaines appliqués, y compris la finance, où les EDP non linéaires en grande dimension apparaissent dans des problèmes de *pricing* (voir [BT22] pour illustration), et où nombre de problèmes de contrôle stochastique apparaissent dans des problèmes de construction de stratégie et de couverture (voir [HLLR16], [GM19] et [GMP20] pour des exemples d'applications à des problèmes de calculs de CVA, de *market making*, et de rachat de titre).

Les algorithmes que nous proposons procèdent en deux étapes. Tout d'abord, l'EPD est réécrite sous forme duale correspondant à un problème de contrôle stochastique, en exprimant l'hamiltonien H en tant que son conjugué concave par rapport à ses deux dernières variables

$$H(x, z, \gamma) = \sup_{(b,c)\in D_f} \big[b \cdot z + \frac{1}{2} c : \gamma + \mathrm{f}(x, b, c)\big], \qquad \text{pour } x \in \mathbb{R}^d, z \in \mathbb{R}^d, \gamma \in \mathbb{S}^d,$$

où $A : B = \mathrm{Tr}(A^\top B)$ et $D_f := \{(b, c) \in \mathbb{R}^d \times \mathbb{S}^d : \mathrm{f}(x, b, c) > -\infty\} \subset \mathbb{R}^d \times \mathbb{S}^d_+$. A partir de cette expression, il est connu que la solution de cette EDP admet une représentation stochastique

$$u(t, x) = \sup_{\alpha\in\mathcal{A}} \mathbb{E}\Big[g(X_T^{t,x,\alpha}) + \int_t^T \mathrm{f}(X_s^{t,x,\alpha}, \alpha_s)\mathrm{d}s\Big], \quad (t, x) \in [0, T] \times \mathbb{R}^d,$$

où $X = X^{t,x,\alpha}$ est solution de l'équation différentielle stochastique

$$\mathrm{d}X_s \;=\; \mathrm{b}(X_s, \alpha_s)\mathrm{d}s + \sigma(X_s, \alpha_s)\mathrm{d}W_s, \quad t \le s \le T, \quad X_t = x.$$

A partir d'une méthode déjà développée dans la littérature, nous utilisons un réseau de neurones $\mathrm{a}_\theta : [0, T] \times \mathbb{R}^d \to A \subset \mathbb{R}^q$, afin d'approximer le contrôle optimal *feedback*, en

maximisant la fonction de coût suivante par rapport à θ

$$J(\theta) = \mathbb{E}\Big[g(X_T^\theta) + \int_0^T f(X_t^\theta, \mathrm{a}_\theta(t, X_t^\theta))\mathrm{d}t\Big], \tag{1.3.2}$$

où l'état X^θ est contrôlé par le réseau de neurones et a un état initial X_0^θ distribué selon a loi μ_0 sur $\mathbb{R}^d$.

Une fois une approximation du contrôle *feeback* optimal obtenue, que l'on note $\mathrm{a}^* = \mathrm{a}_{\theta^*}$, où θ^* est l'ensemble des paramètre "optimaux" qui maximisent (1.3.2), il est possible d'obtenir une approximation de la solution de l'EDP sur des points isolés du domaine en calculant l'espérance conditionnelle

$$u(t,x) = \mathbb{E}\Big[g(X_T^*) + \int_t^T f(X_s^*, \mathrm{a}^*(s, X_s^*))\mathrm{d}s \Big| X_t^* = x\Big], \tag{1.3.3}$$

par Monte-Carlo, où X^* est le processus stochastic contrôlé par l'approximation de contrôle optimal a^*. Notre objectif étant de résoudre l'EPD sur l'intégralité du domaine, nous concevons à la place trois méthodes de Machine Learning pour obtenir cette solution. Nous nous réfèrerons à ces deux méthodes comme à des méthodes de *Differential Learning* dans la mesure où elles consistent à entraîner simultanément la valeur et les dérivées du réseau de neurones. Nous les présentons dans la section suivante.

1.3.1 Méthodes d'apprentissage différentiel

La première méthode, appelée *apprentissage par régression différentielle*, est basée sur la représentation sous forme d'espérance conditionnelle (1.3.3) de la solution de l'EDP, et sa caractérisation fondamentale comme une régression L^2

$$u(t, \hat{X}_t) = \arg\min_{v_t} \mathbb{E}\big|\hat{Y}_T^t - v_t(\hat{X}_t)\big|^2(\hat{X}_t), \qquad \text{pour tout } t \in [0,T],$$

où le *payoff* cible est

$$\hat{Y}_T^t = g(\hat{X}_T) + \int_t^T f(\hat{X}_s, \hat{\mathrm{a}}(s, \hat{X}_s))\mathrm{d}s, \quad t \in [0,T]. \tag{1.3.4}$$

Cette formulation suggère d'utiliser une classe de fonction de réseau de neurones ϑ^η sur $[0,T] \times \mathbb{R}^d$, avec les paramètres η, pour approximer la fonction valeur u, en minimisant une fonction de coût

$$\begin{aligned} \hat{L}_{val}(\eta) &= \mathbb{E}\Big[\int_0^T \big|\hat{Y}_T^t - \vartheta^\eta(t, \hat{X}_t)\big|^2 \mathrm{d}t\Big] \\ &\simeq \mathbb{E}\Big[\int_0^T \big|Y_T^{*,t} - \vartheta^\eta(t, X_t^*)\big|^2 \mathrm{d}t\Big] =: L_{val}^*(\eta), \end{aligned} \tag{1.3.5}$$

où $Y_T^{*,t}$ est le *payoff* (1.3.4) calculé sur des trajectoires contrôlée par une approximation de contrôle optimal a^*.

De facon similaire, un estimateur non-biaisé de la dérivée de la solution de l'EDP peut être obtenu comme l'espérance conditionnelle de la dérivée trajectorielle du payoff $\hat{Y}_T^t$

$$D_x u(t, \hat{X}_t) = \mathbb{E}\Big[\hat{Z}_T^t \big| \mathcal{F}_t\Big], \quad t \in [0,T],$$

où $\hat{Z}_T^t = D_{\hat{X}_t}\hat{Y}_T^t$. Cela suggère de compléter l'apprentissage de la fonction valeur par l'apprentissage de sa dérivée. En pratique, cela peut être fait en entraînant la même fonction de réseau de neurone ϑ^η pour minimiser la fonction de coût

$$L^*_{der}(\eta) \;=:\; \mathbb{E}\Big[\int_0^T \big|Z_T^{*,t} - D_x\vartheta^\eta(t, X_t^*)\big|^2 \mathrm{d}t\Big],$$

où $Z_T^{*,t} = D_{X_t^*}Y_T^{*,t}$ à valeurs dans $\mathbb{R}^d$, est obtenue par différentiation automatique tel que

$$Z_T^{*,t} = \big(D_{X_t}X_T\big)^\top D_x g(X_T) + \int_t^T \big(D_{X_t}X_s\big)^\top D_x f^{\mathrm{a}^*}(s, X_s)\mathrm{d}s, \quad t \in [0,T], \tag{1.3.6}$$

où nous abandonnons l'exposant * pour l'état $X = X^*$, notons $f^{\mathrm{a}^*}(t,x) = f(x, \mathrm{a}^*(t,x))$, et faisons l'hypothèse que g et f are dérivable continûment.

Result 8: ***Apprentissage par régression différentielle***

La méthode d'*apprentissage par régression différentielle* consiste à entraîner un réseau de neurones ϑ^η sur $[0,T]\times\mathbb{R}^d$ pour minimiser simultanément les fonctions de coût L^*_{val} et L^*_{der} définies précédemment.

Dans l'expression (1.3.6), la dérivée $D_x f^{\mathrm{a}^*} = D_x f + (D_x \mathrm{a}^*)^\top D_a f$ peut être calculée facilement par différentiation automatique du réseau de neurones a_{θ^*}. De plus, une approximation de la dérivée de *flow* du processus optimalement contrôlé $D_{X_t}X_s$ peut aussi être calculée en pratique par une différentiation automatique du schéma de diffusion d'Euler de ce processus. La dérivée du *payoff* est également bien définie lorsque la fonction g est dérivable presque partout car la mesure du processus d'état au temps terminal X_T est absolument continue par rapport à la mesure de Lebesgue. Un argument formel pour cette affirmation peut être donné, fondé sur l'utilisation de la dérivée de Malliavin.

Les deux autres méthodes développées sont similaires mais reposent sur une approximation trajectorielle du *payoff* sur un ensemble de trajectoires aléatoires *optimalement contrôlés* plutôt que sur l'estimation d'une espérance conditionnelle.

À partir de la représentation martingale liée à l'espérance conditionnelle (1.3.3), nous pouvons réécrire, en utilisant la formule d'Itô

$$\hat{Y}_T^t = u(t, \hat{X}_t) + \int_t^T \big(D_x u(s, \hat{X}_s)\big)^\top \sigma^{\hat{\mathrm{a}}}(s, \hat{X}_s)\mathrm{d}W_s, \quad t \in [0,T], \tag{1.3.7}$$

où $\hat{Y}_T^t = g(\hat{X}_T) + \int_t^T f(\hat{X}_s, \hat{\mathrm{a}}(s, \hat{X}_s))\mathrm{d}s$.

Result 9: ***Apprentissage martingale trajectoriel***

Nous utilisons à nouveau une classe de réseaux de neurones ϑ^η définis sur $[0,T]\times\mathbb{R}^d$, à paramètres η. La fonction valeur u est approchée en entraînant le réseau de neurones à minimuser la fonction de coût

$$L^*_{mar}(\eta) := \mathbb{E}\Big[\int_0^T \Big|Y_T^{*,t} - \vartheta^\eta(t, X_t^*) - \int_t^T \big(D_x\vartheta^\eta(s, X_s^*)\big)^\top \sigma^{\mathrm{a}^*}(s, X_s^*)\mathrm{d}W_s\Big|^2 \mathrm{d}t\Big].$$

Nous appellons cette méthode d'entraînement la méthode d'*apprentissage martingale trajectoriel*.

La troisième méthode, appelée *apprentissage trajectoriel différentiel*, est similaire à celle-ci, avec l'ajout d'une fonction de coût d'apprentissage basée sur la dérivée trajectorielle de la représentation martingale (1.3.7) produisant un second estimateur reliant les dérivées première et seconde de $u(t,x)$.

$$D_{\hat{X}_t}\hat{Y}_T^t = D_x u(t,\hat{X}_t) + \int_t^T \Big(\big[D_x\sigma^{\hat{a}}(s,\hat{X}_s) \bullet_3 D_{\hat{X}_t}\hat{X}_s\big] \bullet_1 D_x u(s,\hat{X}_s) + \sigma^{\hat{a}}(s,\hat{X}_s)^\top D_x^2 u(s,\hat{X}_s) D_{\hat{X}_t}\hat{X}_s \Big)^\top \mathrm{d}W_s, \quad t\in[0,T],$$

où $(\boldsymbol{M} \bullet_3 B)_{i_1 i_2 \ell} = \sum_{i_3=1}^{d_3} M_{i_1 i_2 i_3} B_{i_3 \ell}$ pour $M \in \mathbb{R}^{d_1\times d_2\times d_3}$ et $B \in \mathbb{R}^{d_p\times d}$ et $(\boldsymbol{M} \bullet_1 b)_{i_2 i_3} = \sum_{i_1=1}^{d_1} M_{i_1 i_2 i_3} b_{i_1}$ pour $M \in \mathbb{R}^{d_1\times d_2\times d_3}$ et $b \in \mathbb{R}^{d_p}$.

Result 10: *Apprentissage trajectoriel différentiel*

La méthode d'*apprentissage trajectoriel différentiel* consiste à entraîner le réseau de neurone à minimiser la fonction de coût suivante

$$L^*_{dermar}(\eta) = \mathbb{E}\Big[\int_0^T \Big| Z_T^{*,t} - D_x\vartheta^\eta(t,X_t) - \int_t^T \Big(\big[D_x\sigma^{a^*}(s,X_s) \bullet_3 D_{X_t}X_s\big] \bullet_1 D_x\vartheta^\eta(s,X_s) + \sigma^{a^*}(s,X_s)^\top D_x^2\vartheta^\eta(s,X_s) D_{X_t}X_s \Big)^\top \mathrm{d}W_s \Big|^2 \mathrm{d}t\Big],$$

en plus de la foncton de coût L^*_{mar} définie précédemment.

L'intérêt d'ajouter une fonction de coût liée à la dérivée du *payoff* que nous voulons approximer est illustré dans la Figure 1.3.5 où nous montrons l'approximation obtenue en entraînant le même réseau à résoudre l'EDP non linéaire avec un hamiltonien donné sur $(0,\infty)$ par

$$H(x,\gamma) = \begin{cases} \frac{1}{2}\sigma^2 \frac{x^2\gamma}{1-\lambda x^2\gamma}, & \text{si } \lambda x^2\gamma < 1 \\ \infty, & \text{sinon.} \end{cases} \tag{1.3.8}$$

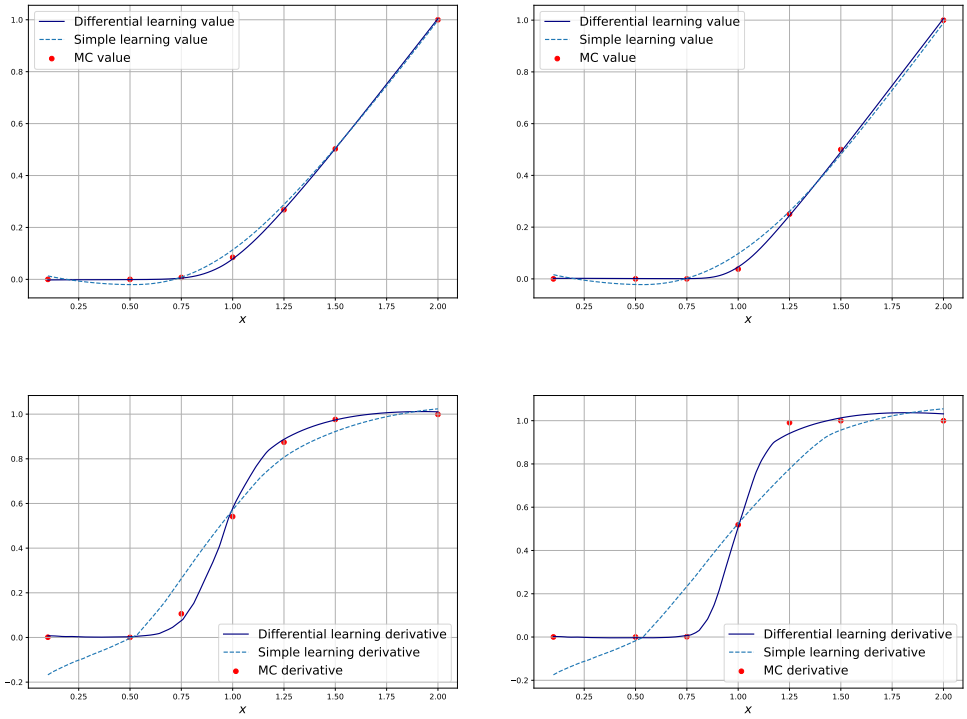
apparaissant dans le problème de valorisation d'option dans un modèle de Black-Scholes avec impact de marché linéaire, par la méthode d'*apprentissage par régression différentielle* et par la minimisation de la seule fonction de coût L^*_{val} définie dans l'Equation (1.3.5) (que nous appelons *apprentissage simple*).

1.3.1.1 Résultats numériques

Le réseau de neurone utilisé pour approcher le contrôle optimal possède la même structure que celui utilisé pour approcher la fonction valeur. Cette structure est représentée dans la Figure 1.10.

Ce réseau est composé de deux sous-réseaux denses, prenant respectivement la variable de temps et la variable d'espace en entrée, dont les sorties sont concaténées et envoyées dans de nouvelle couches denses.

Nos algorithmes sont des version discrétisées en temps des algorithmes décrits dans les Sections précédentes.



(a) $t = 0.5$ (b) $t = 0.9$

FIGURE 1.9 : Fonction valeur (première ligne) et sa dérivée (seconde ligne) obtenue par *apprentissage par régression différentielle* (courbe bleue foncée), *apprentissage simple* (courbe en pointillés bleus clairs) et Monte Carlo (points rouges) tracées en fonction de x, pour des valeurs constantes de t.

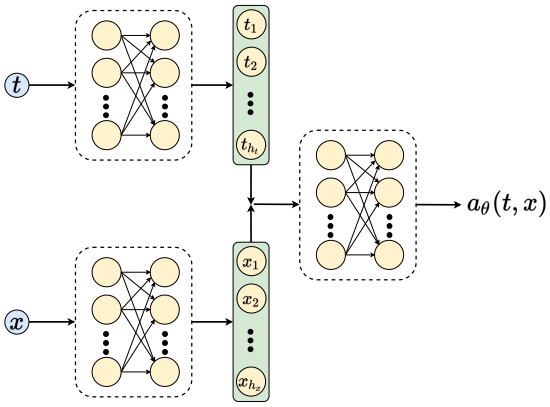


FIGURE 1.10 : Structure du réseau de neurones utilisé pour approcher le contrôle optimal et la fonction valeur.

Dans un premier temps, nous considérons un réseau de neurones a_θ de $[0, T] \times \mathbb{R}^d$ dans $A \subset \mathbb{R}^q$ pour l'approximation du contrôle *feedback*, et le processus d'état discrétisé

associé

$$X^\theta_{t_{n+1}} = X^\theta_{t_n} + \mathrm{b}(X^\theta_{t_n}, \mathrm{a}_\theta(t_n, X^\theta_{t_n}))\Delta t + \sigma(X^\theta_{t_n}, \mathrm{a}_\theta(t_n, X^\theta_{t_n}))\Delta W_{t_n}, \quad n = 0, \dots, N-1,$$

ayant pour valeur intiale $X_0 \sim \mu_0$ (distribution de probabilité définie sur $\mathbb{R}^d$), et où $\Delta W_{t_n} = W_{t_{n+1}} - W_{t_n}$.

Pour entraîner ce réseau de neurones, nous utilisons un *batch* de M trajectoires indépendantes $\{x^{m,\theta}_{t_n}, t_n \in \mathcal{T}_N\}$, $m = 1, \dots, M$, de $\{X^\theta_{t_n}, t_n \in \mathcal{T}_N\}$, et nous appliquons la méthode de descente de gradient à la fonction de coût :

$$J_M(\theta) = \frac{1}{M}\sum_{m=1}^{M}\Big[g(x^{m,\theta}_T) + \sum_{n=0}^{N-1} f\big(x^{m,\theta}_{t_n}, \mathrm{a}_\theta(t_n, x^{m,\theta}_{t_n})\big)\Delta t_n\Big].$$

Nous obtenons une approximation $\mathrm{a}^* = \mathrm{a}_{\theta^*}$ du contrôle optimal et du processus d'état optimal associé avec $X^* = X^{\theta^*}$

Nous considérons ensuite un réseau de neurones ϑ^η de $[0,T] \times \mathbb{R}^d$ dans $\mathbb{R}$ pour l'approximation de la fonction de valeur. Les dérivées $D_x g$, $D_x f$, $D_x \mathrm{b}$, $D_x \sigma$, $D_x \mathrm{a}$, et $D_x \vartheta^\eta$ qui apparaissent dans les méthodes d'*apprentissage par régression différentielle* sont calculées directement par différenciation automatique. En ce qui concerne la dérivée de *flow* du processus d'état optimal, elle peut être obtenue efficacement en stockant les dérivées de chaque pas du schéma de diffusion :

$$D_{X_{t_n}} X_{t_{n+1}} = I_d + D_x b^{\mathrm{a}^*}(t_n, X_{t_n})\Delta t_n + \sum_{j=1}^{d} D_x \sigma_j^{\mathrm{a}^*}(t_n, X_{t_n})\Delta W^j_{t_n}, \quad n = 0, \dots, N-1,$$

et en utilisant la formule de dérivation des fonctions composées

$$D_{X_{t_n}} X_{t_p} = D_{X_{t_n}} X_{t_{n+1}} \cdots D_{X_{t_{p-1}}} X_{t_p}, \quad \text{for } n < p \in [\![0, N]\!].$$

A partir de l'approximation du contrôle optimal, la solution de l'EDP peut être approchée en entraînant un réseau de neurones ϑ^η selon les trois méthodes présentées précédemment, en minimisant des version discrétisées en temps des fonctions de coût que nous avons définies.

La qualité des approximations obtenues est ensuite évaluée en calculant la fonction de coût associée au résidu et aux conditions aux limites de l'EDP,

$$\mathcal{L}_{res} := \frac{1}{|\mathcal{T}||\chi|}\sum_{t\in\mathcal{T}, x\in\chi}\Big|\partial_t \vartheta^\eta + H(x, D_x\vartheta^\eta, D_x^2\vartheta^\eta)\Big|^2, \tag{1.3.9}$$

$$\mathcal{L}_{term} := \frac{1}{|\chi|}\sum_{x\in\chi}\Big|\vartheta^\eta(T,x) - g(x)\Big|^2, \tag{1.3.10}$$

deux fonctions de coût qui seraient nulles pour une approximation parfaire de la solution de l'EDP.

Une autre méthode consiste à calculer la valeur de la solution EDP en calculant l'espérance conditionnelle (1.3.3) et la dérivée de l'espérance conditionnelle représentant la solution EDP pour un ensemble de points (t, x). Pour l'exemple présenté ici, les valeurs de coût associées au résidu et aux conditions terminales obtenues avec les trois méthodes sont présentées dans un tableau et les points de Monte Carlo sont utilisés comme une vérification graphique de la qualité de l'approximation, et sont tracés à côté de la solution obtenue avec nos trois méthodes.

Pour une condition terminale $g(x) = \ln(x)$, l'EDP (1.3.8) présentée précédemment a une solution analytique donnée par

$$u(t,x) = \ln(x) - \frac{\sigma^2}{2(1+\lambda)}(T-t).$$

Dans la Table 1.1, nous indiquons les valeurs des fonctions de coût de validation (1.3.9) Et (1.3.10) et le temps d'entraînement du réseau de neurones pour résoudre l'EDP (1.3.8) avec un *payoff* logarithmique terminal en utilisant les trois méthodes. Lors de ces entraînements, nous avons utilisé 8192 points de départ x_0 et trajectoires browniennes et entraîné le réseau pendant 500 *epoch*.

Nous tracons ci-dessous la fonction valeur $\vartheta^\eta(t,x)$ et ses dérivées $\partial_x \vartheta^\eta(t,x)$ et $\partial_{xx} v_\eta(t,x)$ pour des valeurs fixes $t = 0$, $t = 0.5$, $t = 0.9$, un paramètre $\sigma = 0.2$ et le paramètre de l'EDP $\lambda = 5e^{-3}$, et nous les comparons avec la solution analytique du problème et les points de Monte Carlo. La Figure 1.11 correspond à la méthode d'*apprentissage par régression différentielle*, la Figure 1.12 correspond à la méthode d'*apprentissage martingale trajectoriel* tandis que la figure 1.13 correspond à la méthode d'*apprentissage trajectoriel différentiel*. Graphiquement, les résultats des méthodes d'*apprentissage par régression différentielle* et d'*apprentissage trajectoriel différentiel* sont très proches des points obtenus par estimation Monte Carlo pour la valeur et la dérivée première. La méthode d'*apprentissage martingale trajectoriel* ne donne pas une bonne approximation de la valeur et des dérivées pour les valeurs de x inférieures à 1. La différence de performance entre les méthodes *trajectorielle* et *trajectorielle différentielle* est analogue à celle observée entre la méthode d'*apprentissage par régression différentielle* et l'apprentissage par régression "simple" dans la figure 1.9, démontrant l'intérêt d'ajouter un terme de régression pour la dérivée du réseau de neurones.

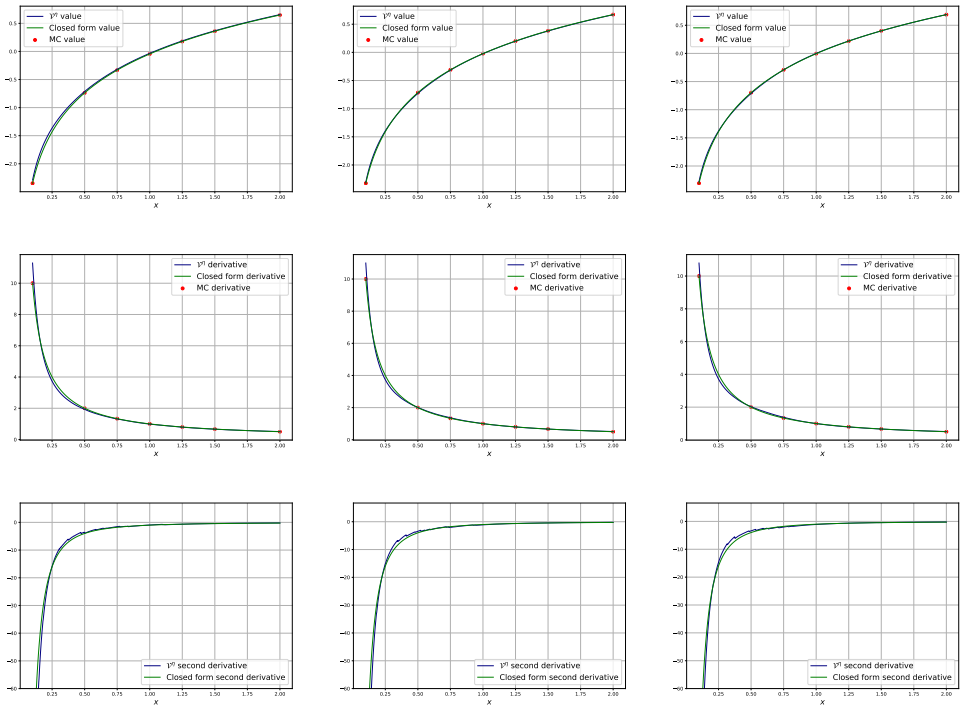
	Diff. regr. learning	Path. 1NN	Path. diff. 1NN
Coût résidu	$2.046e^{-3}$	$3.484e^{-4}$	$6.644e^{-4}$
Coût résidu + coût terminal	$2.179e^{-3}$	$3.864e^{-4}$	$6.758e^{-4}$
Temps d'entraînement (500 epochs)	163s	130s	525s

Table 1.1 : Coût résiduel et coût terminal calculés sur une grille 1002x1002 avec $t \in [0, 0.9]$ et $x \in [0.1, 2]$ pour un *payoff* terminal logarithmique, avec un paramètre $\sigma = 0.2$ et un paramètre d'impact de marché linéaire $\lambda = 5e^{-3}$.

1.3.1.2 DeepOnet pour des fonctions terminales paramétriques

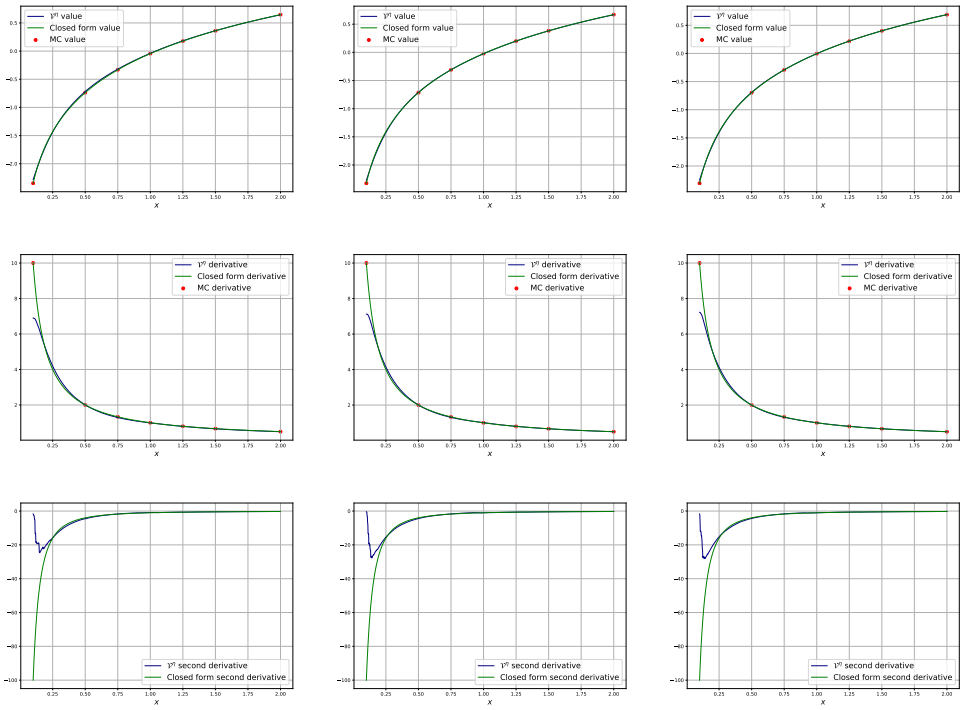
Enfin, notre objectif est de concevoir une méthode de Machine Learning permettant d'obtenir directement une solution à une EDP de type (1.3.1) avec condition terminale paramétrique g_K, pour toute valeur du paramètre $K \in \mathbb{R}^p$ dans un ensemble compact. Pour ce faire, nous nous appuyons sur une classe de réseaux de neurones appelée DeepONet, visant à approximer des opérateurs fonctionnels. Ces réseaux neuronaux sont basés sur le théorème d'approximation universelle des opérateurs suivant.

Theorem 1.3.1. *Supposons que σ soit une fonction non polynomiale continue, que X soit un espace de Banach, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ deux ensembles compacts dans X et $\mathbb{R}^d$, respectivement, V est un ensemble compact de $C(K_1)$, G est un opérateur non linéaire continu, de V dans $C(K_2)$. Alors, pour tout $\epsilon > 0$, il existe des entiers positifs n, p, m, des constantes c_i^k, ξ_{ij}^k, θ_i^k, $\zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, ..., n$, $k = 1, ..., p$,*



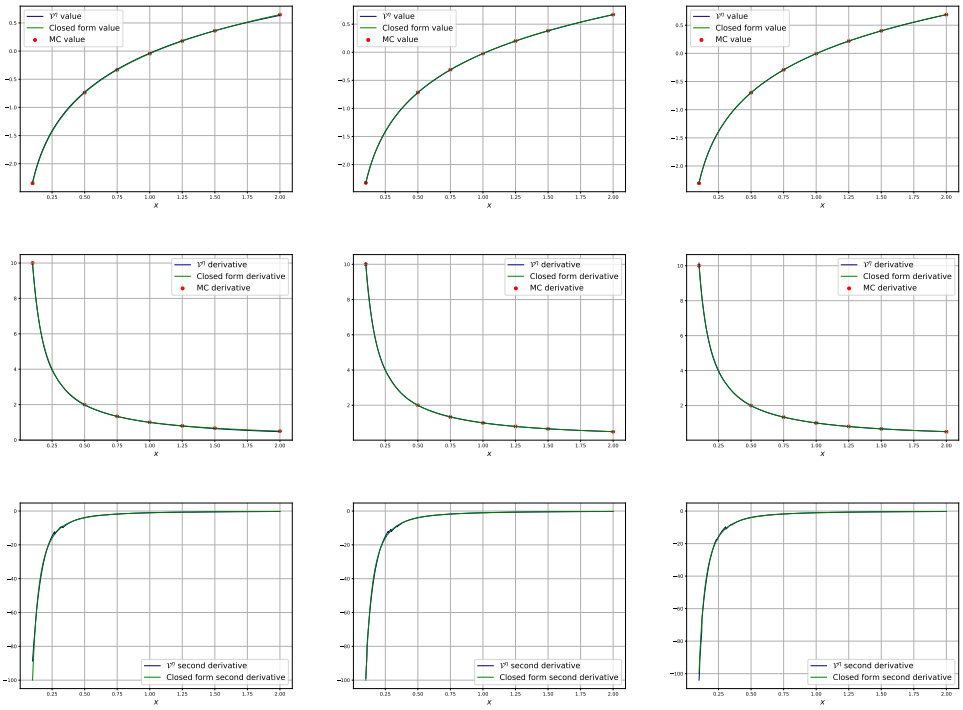
(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

FIGURE 1.11 : Fonction valeur ϑ^η et ses dérivées première et seconde obtenues par la méthode d'*apprentissage par régression différentielle* pour un *payoff* terminal logarithmique, avec un paramètre $\sigma = 0.2$ et un paramètre d'impact de marché linéaire $\lambda = 5e^{-3}$, tracées en fonction de x, pour des valeurs fixes de t.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

FIGURE 1.12 : Fonction valeur ϑ^η et ses dérivées première et seconde obtenues par la méthode de *Pathwise Learning* pour un *payoff* terminal logarithmique, avec un paramètre $\sigma = 0.2$ et un paramètre d'impact de marché linéaire $\lambda = 5e^{-3}$, tracées en fonction de x, pour des valeurs fixes de t.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

FIGURE 1.13 : Fonction valeur ϑ^η et ses dérivées première et seconde obtenues par la méthode d'*apprentissage trajectoriel différentiel* pour un *payoff* terminal logarithmique, avec un paramètre $\sigma = 0.2$ et un paramètre d'impact de marché linéaire $\lambda = 5e^{-3}$, tracées en fonction de x, pour des valeurs fixes de t.

$j = 1, ..., m$, tels que

$$\left\| G(u)(y) - \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{branch\ net} \underbrace{\sigma\left(w_k y + \zeta_k\right)}_{trunk\ net} \right\| < \epsilon,$$

est valable pour tout $u \in V$ dans $y \in K_2$.

Le réseau utilisé, dont la structure est représentée dans la Figure 1.14, est une combinaison entre le DeepONet *standard* et le réseau utilisé dans la partie précédente.

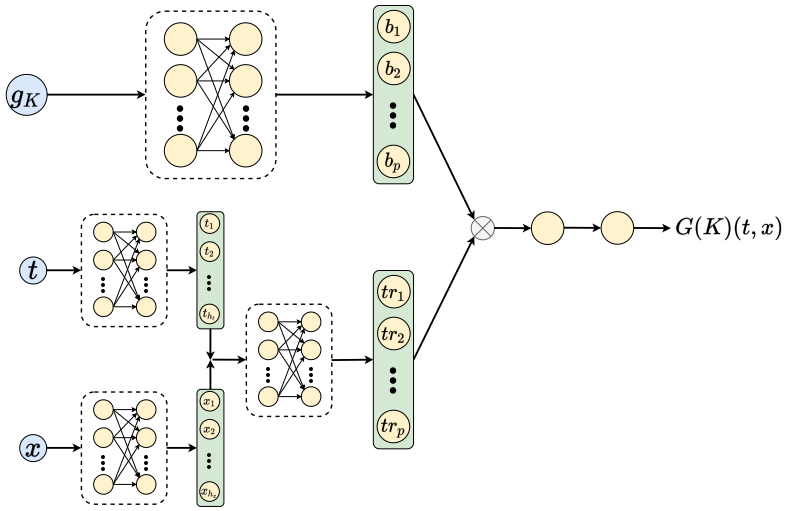


FIGURE 1.14 : Structure du réseau DeepONet.

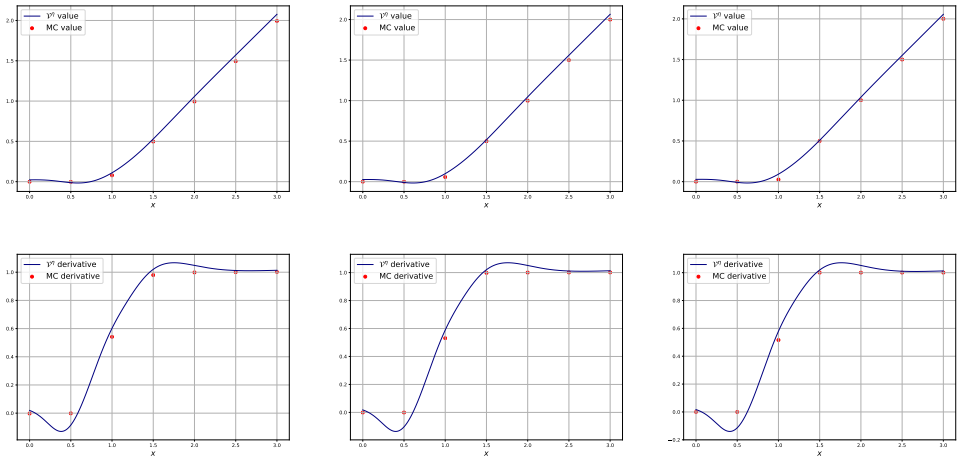
Nous entraînons ce réseau, comme précédemment, de manière non supervisée (aucun autre solveur n'est nécessaire pour calculer une première solution que ce réseau essaierait d'approcher). Notre algorithme d'entraînement est très proche de ceux décrits précédemment, avec l'ajout que, pour chaque trajectoire d'état que nous générons, un paramètre aléatoire de condition terminale K est également généré de manière aléatoire et uniforme dans un ensemble compact. Nous entraînons le réseau à résoudre l'EDP de *payoff* (1.3.8) définie précédemment avec une condition terminale paramétrique correspondant au *payoff* d'une option d'achat

$$g_K(x) = \max(x - K, 0),$$

avec un *strike* K. Pour tester la capacité de généralisation de cette méthode, nous entraînons ce réseau sur 8192 trajectoires et *strikes* aléatoires, avec des *strikes* échantillonés aléatoirement dans $\mathcal{U}([0.25, 0.75] \cup [1.5, 2])$ le testons sur des *strikes* choisis dans $[0.2, 2.1]$.

Dans le Tableau 1.2, nous calculons le coût associés au résidu pour le réseau DeepONet ϑ^η entraîné par la méthode d'*apprentissage par régression différentielle*. Nous calculons le coût associé au résidu sur une grille de points 102x102 linéairement espacés avec $t \in [0, 0.9]$ et $x \in [0, 3]$ pour un *payoff* terminal de *call* avec des *strikes* $K \in \{0.2,\ 0.5,\ 1,\ 1.75,\ 2,\ 2.1\}$, un paramètre $\sigma = 0.3$ et un paramètre d'EDP $\lambda = 5e^{-3}$.

Nous tracons ci-dessous la fonction valeur $\vartheta^\eta(t, x)$ et sa dérivée $\partial_x \vartheta^\eta(t, x)$ obtenues par *apprentissage par régression différentielle* avec le réseau DeepONet, pour des valeurs fixes $t = 0$, $t = 0.5$, $t = 0.9$, un paramètre $\sigma = 0.3$ et le paramètre d'EDP $\lambda = 5e^{-3}$, et les comparons avec l'estimation Monte-Carlo obtenue. Nous tracons ces fonctions valeur pour des valeurs de *strike* $K \in \{0.5,\ ;2\}$ à l'intérieur du domaine d'entraînement et $K \in \{0.2,\ 1,\ 2.1\}$ à l'extérieur du domaine d'entraînement.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

FIGURE 1.15 : Fonction valeur ϑ^η (première ligne) et sa dérivée (deuxième ligne) obtenue par *apprentissage par régression différentielle* (Algorithme 9) pour un *payoff* terminal de *call* avec un *strike* $K = 1$, un paramètre $\sigma = 0.3$ et un paramètre d'EDP $\lambda = 5e^{-3}$, tracées en fonction de x, pour des valeurs fixes de t.

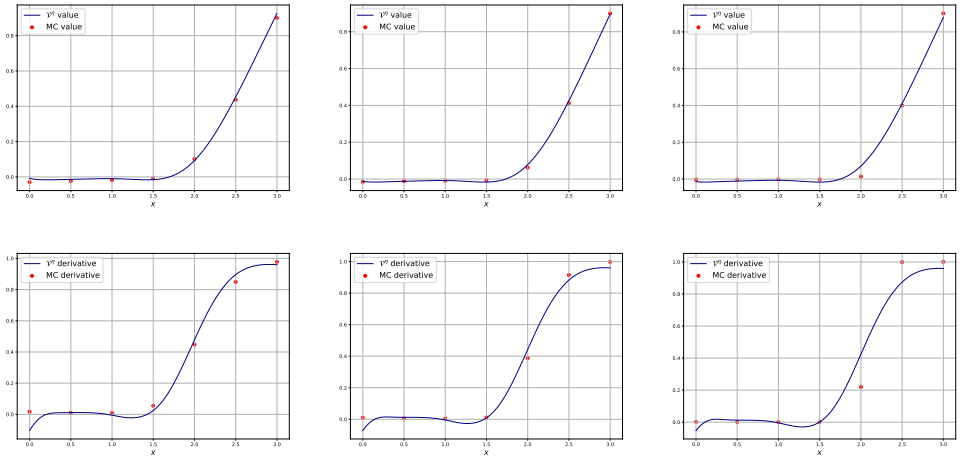


FIGURE 1.16 : Fonction valeur ϑ^η (première ligne) et sa dérivée (deuxième ligne) obtenue par *apprentissage par régression différentielle* (Algorithme 9) pour un *payoff* terminal de *call* avec un *strike* $K = 2.1$, un paramètre $\sigma = 0.3$ et un paramètre d'EDP $\lambda = 5e^{-3}$, tracées en fonction de x, pour des valeurs fixes de t.

	$K = 0.2$	$K = 0.5$	$K = 1$	$K = 1.75$	$K = 2$	$K = 2.1$
Coût résidu + coût terminal	$1.918e^{-3}$	$1.461e^{-4}$	$1.700e^{-3}$	$1.162e^{-3}$	$2.002e^{-3}$	$2.175e^{-3}$

TABLE 1.2 : Coût résiduel et coût terminal calculés sur une grille 102x102 avec $t \in [0, 0.9]$ et $x \in [0, 3]$ pour un *payoff* terminal de *call* $g(x) = \max(x - K, 0)$ pour $K \in \{0.2,\ 0.5,\ 1,\ 1.75,\ 2,\ 2.1\}$, avec un paramètre $\sigma = 0.3$ et un paramètre d'EDP $\lambda = 5e^{-3}$.

Chapter 2

Introduction

The first months of my PhD work were dedicated to a research project, carried out in collaboration with a structuring team of BNP Paribas, which led to the publication of the article [GLT20]. This work, which will not be presented in the body of this manuscript, consisted in the identification of the factors (macroeconomic and technical factors computed from the government bond yield curve) which drive the value of government bond futures. As the aim is to choose a small set of most relevant features in a large set of correlated factors, a robust feature selection method, not impaired by correlations, must be used. In this study the adaptive Lasso, defined in [Zou06], was used to select the factors, with an L_1 penalization parameter chosen by minimizing a modified Bayesian Information Criterion defined in [Cha12].

The thesis is divided into three parts that can be read independently. In the first part, we make use of the weak martingale approach to linear-quadratic McKean–Vlasov stochastic control problems to solve a problem of mean-variance portfolio allocation in continuous time with a tracking error penalisation. In this problem, the controls correspond to the wealth invested in each risky asset under consideration and the tracking error penalisation consists in a distance between these controls and the composition of a reference portfolio with same total wealth and fixed weights. We derive the optimal control as a function of the solutions of a system of Riccati ODEs and a verification theorem for this problem. An asymptotic expansion of the expression of the terminal portfolio wealth variance as a function of its expectation is obtained in the case where the penalisation factor is small, analogous to the *efficient frontier formula* of the classical mean-variance portfolio theory. The robustness of this penalised portfolio allocation in the case of parameter misspecification is illustrated with Monte Carlo simulations and a backtest of the allocation on historical price data is performed.

In the second part, we consider a class of stochastic control problems where the control is present both in the drift and in the diffusion part of the state SDE and where the state is controlled with a delay (or latency). The optimal control and value of the problem are obtained in term of the solutions of a set of transport PDEs coupled through their source terms. The existence and uniqueness of a solution of this system of PDEs is proven. A sufficient condition of existence, directly emerging from the delayed structure, and function of the diffusion coefficients and delay, is provided. Along this, a verification theorem of optimality of the control and value function is proven. A deep learning scheme is proposed to solve the PDE system and the optimal control is computed for the problem of mean-variance portfolio allocation with delay with one delayed asset and, in a heuristic way, in the case where one asset is controlled with delay

and the other one instantaneously. The effect of the delay on this portfolio allocation is thus illustrated.

The third part of this thesis is devoted to the definition of deep learning algorithms to solve fully nonlinear PDEs with convex Hamiltonian on a full space-time domain of interest. From the convexity of the PDE Hamiltonian, the PDE is rewritten in its dual conjugate form. From here, we study the associated stochastic control problem and derive three estimators of the PDE solution and its first derivative as a function of the optimal control of this problem. The estimation of the PDE solution is then realized by using a neural network and training its value and its first derivative for all our methods, and training additionally its second derivative for one of our methods. The accuracy of these methods is then illustrated by solving numerically the PDEs associated to the Merton problem and a PDE corresponding to a problem of pricing under a Black Scholes model with linear market impact. The precision of our solutions is assessed by comparing them to the solution computed on a set of points by Monte Carlo, and by computing a loss associated to the PDE residual and boundary condition. Finally, a similar method based on DeepOnet neural networks is defined in order to solve a non linear PDE with a parametric terminal condition for every value of its parameter lying in a compact set. The accuracy and generalization capacity of this method are tested by solving the same pricing PDE with a terminal condition corresponding to the payoff of a call option.

2.1 Mean-variance allocation with tracking error

Part I is devoted to the study of a mean-variance portfolio selection problem where a tracking error relative to a portfolio of same total wealth with fixed weights is added to the mean-variance criterion. This optimization can be interpreted in two ways. It is a way to stabilise the mean variance allocation by "anchoring it" to a reference allocation which can be independent of the estimation of market parameters, as it is the case with the equal weights portfolio or the portfolio with zeros weights, which corresponds to a shrinking of the controls by a penalization of their L_2 norm. The other way to interpret this allocation paradigm is to see it as a portfolio with fixed desired weights that is being further optimized according to a mean variance criterion on its terminal wealth. Other approaches aiming at robustifying the portfolio allocation by optimization under model uncertainty have been considered, see [NN18] and [PWZ22] for examples.

The mean-variance portfolio selection problem, initially considered by Markowitz in [Mar52] in a single period setting, is a McKean-Vlasov (MKV) stochastic control problem when formulated in continuous time. This type of control problem have been extensively studied in recent literature, see [DPT22]. Indeed, the problem can be formulated as the minimisation of the following cost function

$$J(\alpha) = \mu \mathrm{Var}(X_T) - \mathbb{E}[X_T], \tag{2.1.1}$$

which depends on the law of the state process X_T at the final time T. This state process represents the total wealth of the investor's portfolio and evolves according to the SDE

$$dX_t = \alpha_t^\top b \mathrm{d}t + \alpha_t \sigma \mathrm{d}W_t,$$

where the control $\alpha \in \mathbb{R}^d$ represents the wealth invested in each of the d assets considered. The underlying financial market is composed of one risk-free asset assumed to have constant price equal to one, $P^0 = 1$, and d risky assets whose prices $P := (P_t)_{t\in[0,T]}$

satisfy the following SDE

$$\begin{aligned} \mathrm{d}P_t &= P_t^\top \Big(b \, \mathrm{d}t + \sigma \mathrm{d}W_t \Big), \quad t \in [0,T], \\ P_0^i &> 0, \quad i = 1, ..., d. \end{aligned} \tag{2.1.2}$$

A first approach to solve this problem, from [ZL00], consists in embedding the mean-variance problem into an auxiliary standard control problem that can be solved using stochastic linear-quadratic theory. The mean-variance portfolio selection problem is then solved as a special case of this auxiliary problem. Some more recent approaches allow to directly solve this problem as a MKV control problem, see [PW17], [AD11], [FL16] to name just a few. In this Part we consider a modified version of (2.1.1) where a running cost is added

$$J(\alpha) = \mu \mathrm{Var}(X_T) - \mathbb{E}[X_T] + \mathbb{E}\Big[\int_0^T (\alpha_t - w_r X_t)^\top \Gamma (\alpha_t - w_r X_t) \mathrm{d}t \Big]. \tag{2.1.3}$$

We tackle this MKV linear-quadratic control problem by the weak martingale optimality principle approach developed in [BP19]. The running cost added in (2.1.3), which we call *tracking error penalization* is introduced in order to ensure that the portfolio of the investor does not move away too much of the reference composition $w_r X_t$ of a portfolio with same total wealth X_t and constant weights w_r with respect to the distance $|M| = M^\top \Gamma M$, where the matrix $\Gamma \in \mathbb{R}^{d \times d}$ is symmetric positive definite. We only consider constant reference weights although our results can easily be extended to the case where the reference weights are adapted to the market filtration.

The mean-variance portfolio selection problem with tracking error is then formulated as

$$V_0 := \inf_{\alpha \in \mathcal{A}} J(\alpha), \tag{2.1.4}$$

where $\mathcal{A}$ denotes the space of admissible controls and the cost J corresponds to the cost of equation (2.1.3). An optimal allocation given the cost $J(\alpha)$ will be given by

$$\alpha_t^* \in \arg \min_{\alpha \in \mathcal{A}} J(\alpha).$$

2.1.1 Solution allocation with tracking error

As mentioned, we use the weak optimality principe to solve problem (2.1.4). The idea is to build a family of real-valued processes $\{V_t^\alpha, \ t \in [0,T], \alpha \in \mathcal{A}\}$ of the form

$$V_t^\alpha = v_t(X_t^\alpha, \mathbb{E}[X_t^\alpha]) + \int_0^t (\alpha_s - w_r X_s^\alpha)^\top \Gamma (\alpha_s - w_r X_s^\alpha) \mathrm{d}s,$$

with $v : \mathbb{R}_+ \times \mathbb{R} \times \mathbb{R}$ a measurable function such that $v_T(x, \overline{x}) = \mu(x - \overline{x})^2 - x$ for all x, $\overline{x} \in \mathbb{R}$, and the function $t \in [0,T] \mapsto \mathbb{E}[V_t^\alpha]$ is non decreasing for all $\alpha \in \mathcal{A}$. From the weak optimality principle lemma stated in [BP19], the optimal portfolio strategy is then obtained as the control $\alpha^* \in \mathcal{A}$ such that the map $t \in [0,T] \mapsto \mathbb{E}[V_t^{\alpha^*}]$ is constant.

For our problem, we look for a measurable function v_t of the form

$$v_t(x, \overline{x}) = K_t (x - \overline{x})^2 + \gamma_t \overline{x}^2 + 2 Y_t x + R_t,$$

for some deterministic processes $(K_t, \Lambda_t, Y_t, R_t)$. By computing the dynamics of the process $\mathbb{E}[V_t^\alpha]$ obtained, we find that the processes (K_t, Λ_t) must be solutions of the following ODEs

$$\begin{cases} \mathrm{d}K_t &= \Big\{(K_t b - \Gamma w_r)^\top S_t^{-1}(K_t b - \Gamma w_r) - w_r^\top \Gamma w_r\Big\}\mathrm{d}t, \quad K_T = \mu, \\ \\ \mathrm{d}\Lambda_t &= \Big\{(\Lambda_t b - \Gamma w_r)^\top S_t^{-1}(\Lambda_t b - \Gamma w_r) - w_r^\top \Gamma w_r\Big\}\mathrm{d}t, \quad \Lambda_T = 0, \end{cases} \tag{2.1.5}$$

and that the processes (Y_t, R_t) must have the following closed form

$$\begin{cases} Y_t = -\frac{1}{2} e^{-\int_t^T b^\top S_s^{-1}(\Lambda_s b - \Gamma w_r)\mathrm{d}s}, \quad \forall t \in [0,T], \\ \\ R_t = \frac{1}{2}\int_t^T b^\top S_s^{-1} b \; e^{-2\int_s^T b^\top S_u^{-1}(\Lambda_u b - \Gamma w_r)\mathrm{d}u}\mathrm{d}s, \quad \forall t \in [0,T], \end{cases} \tag{2.1.6}$$

where $S_t := K_t\Sigma + \Lambda$, with $\Sigma = \sigma\sigma^\top$.

Result 11: Verification theorem

There exist a unique pair $(K, \Lambda) \in C([0,T], \mathbb{R}_+^*) \times C([0,T], \mathbb{R}_+)$ solution to the system of ODEs (2.1.5). The optimal control for problem (2.1.4) is then given by

$$\alpha_t^\Gamma = S_t^{-1}\Gamma w_r X_t - S_t^{-1} b\Big[K_t X_t + Y_t - (K_t - \Lambda_t)\mathbb{E}[X_t]\Big], \tag{2.1.7}$$

and $X = X^{\alpha^\Gamma}$ is the wealth process associated to α^Γ. Moreover, we have

$$V_0 = J(\alpha^\Gamma) = \Lambda_0 X_0^2 + 2Y_0 X_0 + R_0.$$

We see that the optimal control is composed of two components, one determined by the vector $S_t^{-1}\Gamma w_r = (K_t\Sigma + \Gamma)^{-1}\Gamma w_r$ and completely linked to the presence of the tracking error penalization, and a second one determined by the vector $S_t^{-1} b = (K_t\Sigma + \Gamma)^{-1} b$.

In the case where Γ is the null matrix, the first component vanishes, and we check that the second one is equal to the optimal control obtained for the classical mean-variance allocation problem. Indeed, when $\Gamma = \mathbf{0}$,

$$Y_t = -\frac{1}{2}, \quad R_t = \frac{1}{4\mu}\left(1 - e^{b^\top \Sigma^{-1} b\ (T-t)}\right),$$

and the system of ODEs (2.1.6) becomes

$$\begin{cases} \mathrm{d}K_t &= K_t b^\top \Sigma^{-1} b\ \mathrm{d}t, \quad K_T = \mu, \\ \mathrm{d}\Lambda_t &= \frac{\Lambda_t^2}{K_t} b^\top \Sigma^{-1} b\ \mathrm{d}t, \quad \Lambda_T = 0, \end{cases}$$

which yields the explicit solutions

$$K_t = \mu e^{-b^\top \Sigma^{-1} b\ (T-t)}, \quad \Lambda_t = 0.$$

Plugging these expressions in the equation of the optimal control (2.1.7) and computing the expression of the average wealth $\mathbb{E}[X_t]$, we can express the optimal control α^Γ as a function of the initial wealth x_0 and the current wealth X_t of the investor and we obtain the expression

$$\alpha_t^{\Gamma=0} = \Sigma^{-1} b\left[\frac{1}{2\mu} e^{b^\top \Sigma^{-1} b\ T} + x_0 - X_t\right],$$

which corresponds to the solution of the classical mean-variance portfolio construction problem.

In the case where $\Gamma = \gamma 1_d$, with $\gamma \in \mathbb{R}_+^*$ and 1_d the $\mathbb{R}^{d\times d}$ identity matrix, the optimal control can be rewritten as

$$\alpha_t^\gamma = \left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma\right)^{-1} w_r X_t - \frac{1}{\gamma}\left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma\right)^{-1} b\left[K_t X_t + Y_t - (K_t - \Lambda_t)\overline{X}_t\right].$$

We show that the ODE solutions K_t and Λ_t are bounded functions of the penalization parameter γ, thus giving the limits $\frac{K_t}{\gamma}, \frac{\Lambda_t}{\gamma} \underset{\gamma\to\infty}{\longrightarrow} 0$. We then get the following limit for the optimal control in the case where $\gamma \to \infty$

$$\alpha_t^\gamma \underset{\gamma\to\infty}{\longrightarrow} w_r X_t,$$

showing that in the limit where the tracking-error penalization tends to infinity, the portfolio allocation converges to the reference portfolio allocation. Setting again $\Gamma = \gamma 1_d$, we can compute the expansion of the optimal control α^γ in the case where the penalisation parameter $\gamma \to 0$. In that case, computing the expansion of S_t^{-1}, K_t and Λ_t up to the linear order in γ, we get the following result.

Result 12: Expansion for $\Gamma = \gamma 1_d \to 0$

When the tracking-error penalisation parameter $\gamma \to 0$, we can write the expansion of the optimal control α^γ up to the linear term in γ as

$$\alpha_t^\gamma = \Sigma^{-1} b\ \alpha_t^0 + \gamma\left(\Sigma^{-1} w_r\ \alpha_t^{1,3} - \Sigma^{-2} b\ \alpha_t^{1,2} - \Sigma^{-1} b\ \alpha_t^{1,1}\right) + O(\gamma^2),$$

where $\Sigma^{-2} := (\Sigma^{-1})^2$ and with

$$\begin{aligned}
\alpha_t^0 &= \frac{1}{2\mu} e^{\rho T} + X_0 - X_t,\\
\alpha_t^{1,1} &= \frac{\|w_r\|^2}{K_t^0}(T-t)\left(X_0 + \frac{e^{\rho T}}{\mu}\left(1 - e^{-\rho t}\right)\right) + X_0 C_{0,t}^1 + \frac{H_t^1}{2} + \frac{K_t^1}{2\left(K_t^0\right)^2} + \frac{C_{t,T}}{2K_t^0},\\
\alpha_t^{1,2} &= \frac{e^{\rho T}}{2K_t^0\mu}\left(1 - e^{-\rho t}\right) + \frac{1}{2\left(K_t^0\right)^2},\\
\alpha_t^{1,3} &= \frac{X_t}{K_t^0}.
\end{aligned}$$

On this expansion, we clearly see that for $\gamma = 0$, the classical mean-variance optimal control is recovered. For non-zero values of γ, the weight of the allocation given by the vector $\Sigma^{-1}b$ is modified and two allocations $\Sigma^{-2}b$ and $\Sigma^{-1}w_r$ appear with weights $\gamma\alpha_t^{1,2}$ and $\gamma\alpha_t^{1,3}$.

From this expansion of the control we can also compute the first order asymptotic expansion in γ of the equation giving the relation between the variance of the terminal wealth of the portfolio and its expectation, a relation called the *efficient frontier formula* in the classical mean-variance case.

Result 13: Expansion of the variance of terminal wealth for $\Gamma = \gamma 1_d \to 0$

The linear order asymptotic expansion of the variance of the terminal wealth of the portfolio X_T is given by

$$\begin{aligned} Var(X_T) =& \frac{e^{-\rho T}}{1-e^{-\rho T}}\left(\overline{X_T}^0 - X_0\right)^2 \\ &+ \gamma\left\{\frac{b^\top \Sigma^{-1} w_r}{\mu^2}\left[X_0 T - \frac{1}{2\mu}e^{\rho T}\left(T - \frac{1-e^{-\rho T}}{\rho}\right)\right]\right. \\ &\left. - \int_0^T \left(\frac{\rho}{\mu}\alpha_s^{1,1} + \frac{b^\top \Sigma^{-2} b}{\mu}\alpha_s^{1,2}\right) e^{-\rho(T-s)}\mathrm{d}s\right\} + O(\gamma^2). \end{aligned}$$

The leading order term corresponds to the efficient frontier equation of the classical mean-variance allocation computed in [ZL00], and thus for $\gamma = 0$, we recover this classical result. We see that this result is coherent with the expansion obtained for the optimal control α^γ, with a linear term in γ containing the same three perturbative allocations $\Sigma^{-1}b$, $\Sigma^{-2}b$ and $\Sigma^{-1}w_r$.

2.1.1.1 Numerical results

We then apply these results to compute the optimal portfolio allocation obtained with four different static reference portfolios. We run these allocations on simulated market scenarios in the case of misspecified parameters, meaning that the diffusion used to simulate the market prices are not the same as the ones used to compute the optimal allocation. It is known that the classical mean-variance allocation is very sensitive to the estimation of market diffusion parameters, and thus to parameter misspecification. These simulations thus allow to test the robustness of the mean-variance allocation with tracking-error penalization and to see if it is more robust to parameter misspecification than the classical mean-variance allocation. We still assume that the penalization matrix can be written as $\Gamma = \gamma 1_d$ and we numerically solve the ODEs of the processes K_t and Λ_t in order to obtain the optimal control α^γ. We compute the penalized allocation with four different reference portfolios

- Equal-weights portfolio

$$w_r^{\text{ew}} = \frac{1}{d}e,$$

- Minimum variance portfolio

$$w_r^{\text{min-var}} = \frac{\Sigma^{-1}e}{e^\top \Sigma^{-1} e},$$

- Portfolio with zero weights (control shrinking)

$$w_r^{\text{shrinking}} = \mathbf{0},$$

- Equal risk contributions (ERC) portfolio

$$\begin{aligned} w_r^{\text{erc}} = \underset{w\in\mathbb{R}^d}{\text{argmin}} \sum_{i=1}^d \sum_{j=1}^d \left(w^i\left(\Sigma w\right)^i - w^j\left(\Sigma w\right)^j\right)^2 \\ \text{s.t } e^\top w = 1 \text{ and } 0 \le w^i \le 1,\ \forall i \in [\![1,d]\!]. \end{aligned}$$

To compare the performance of the four penalized mean-variance allocations in the case of misspecified parameters, we run Monte Carlo simulations assuming that the asset's prices evolve according to the SDE (2.1.2) and that the real-world expected returns b_{real} and covariances σ_{real} are equal to reference expected returns b_0 and covariances σ_0 plus some noise:

$$b_{\text{real}} = b_0 + \epsilon \times \text{noise}, \quad \sigma_{\text{real}} = \sigma_0 + \epsilon \times \text{noise}.$$

The noise follows a standard $\mathcal{N}(0,1)$ distribution and is scaled by $\epsilon \in \mathbb{R}_+^*$. We use the Monte Carlo simulations to estimate the expected Sharpe ratio of each portfolio, equal to the average of the portfolio daily returns R divided by the standard deviation of those returns: $\frac{\mathbb{E}[R]}{\text{Stdev}(R)}$.

The initial wealth of the investor x_0 is chosen equal to 1 and we choose the penalization parameter $\gamma = \mu/100$. Indeed, as the value of μ depends on the value of the stocks expected return and covariance matrix and on the targeted return, and can be very big, we express γ as a function of this μ in order for the penalization to be relevant and non-negligible. For each reference portfolio, we compare the reference portfolio, the classical mean-variance allocation and the penalized one for values of noise amplitude ϵ ranging from 0 to 1. As an example, we show in Figure 2.1 and 2.2 a comparison of the Sharpe ratio of the classical mean-variance portfolio with the penalized portfolio with equal-weights reference portfolio and zero reference portfolio respectively. These two portfolios do not rely on estimations of market parameters and we can see on these graphs that while the mean-variance portfolio attains better performance when the parameter misspecification is small, for bigger values of estimation error, the penalization portfolio attains better Sharpe ratios.

To complete these examples, we compare the different allocations on a backtest based on daily close prices available on Quandl between 2013-09-03 and 2017-12-28 for four stocks: Apple, Microsoft, Boeing and Nike. We consider two values of the penalization factor $\gamma = \mu$ and $\gamma = \mu/100$. The Figures 2.3 and 2.4 show the total wealth as a function of time of the mean-variance and penalized portfolios with equal-weights reference and zero reference respectively. We can see that the penalized portfolios interpolate between the mean-variance and the reference portfolios and attain Sharpe ratios greater than the unpenalized mean-variance portfolio.

In the case of the penalized portfolio with zero reference, showed in Figure 2.4, the penalized allocation corresponds to a shrinking of the wealth invested in risky assets. On this graph, the wealth of the portfolios is normalized by the standard deviation of their daily returns and we see that the normalized wealth of the penalized portfolio is greater than the wealth of the unpenalized one.

2.2 Control with delay

Part II is devoted to the control of stochastic systems with delays. The time needed to acquire information, compute decision and execute, with systems often affected by latency between the time a command is sent and the time it is executed, make delays in control systems ubiquitous. Various effects of delays on traffic flow modelling, chemical processes, population dynamics, supply chain and advertising have been studied in the literature.

In a delayed controlled system, the state X and the control α are the two main components which can present a delay feature. When the delay is only present in the state variable, the problem is now well understood and can be treated by lifting the state variable to the infinite dimensional space $(X_t, s \in [-d, 0] \mapsto X_{t+s})$, see [DM72], [FGG10] to name just a few. A much less understood situation is the one where the

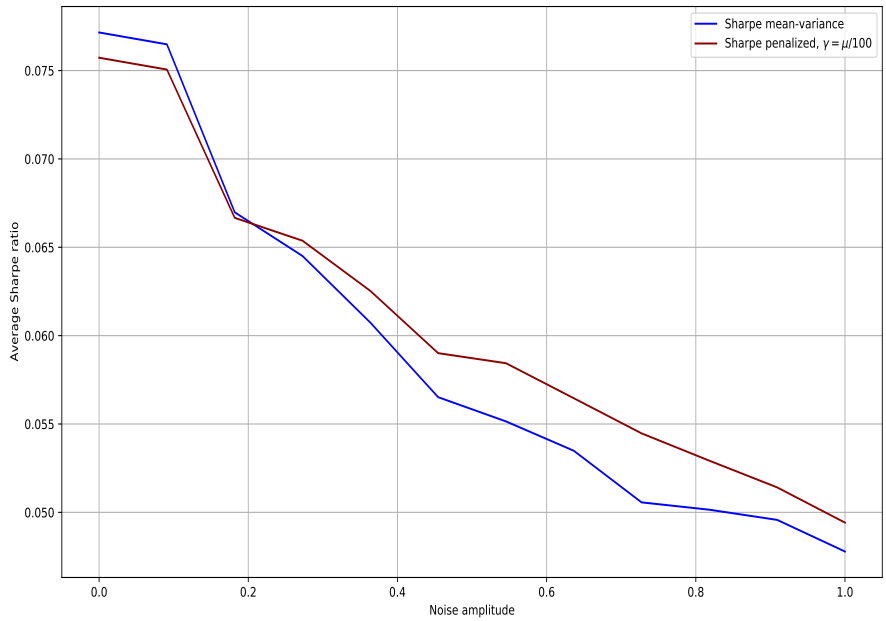


Figure 2.1: The highest average Sharpe ratio attained by the equal-weight portfolio is equal to 0.047 for $\epsilon = 0$.

delay enters the control variable. In this situation, two main approaches have emerged: the *structural state method* and the *extended state method*, we refer to [Ben+07, Part II, Chapter 3] for the study of these latter in the deterministic case and [FF14] for the structural state approach in the stochastic case. For a complete list of references see also [FF14].

In our work, we aim at shedding some lights on the case where the system is controlled both in its drift and diffusion parts by a control affected by a delay. We consider the following class of stochastic delayed linear-quadratic control problem

$$\begin{cases} dX_t^\alpha = \alpha_{t-d}\left(bdt + \sigma dW_t\right), \qquad 0 \le t \le T, \\ X_0 = x, \quad \alpha_s = \gamma(s), \qquad s \in [-d, 0], \quad x \in \mathbb{R} \\ J(\alpha) = \mathbb{E}[(X_T^\alpha)^2]. \end{cases} \tag{2.2.1}$$

Except in [FF14], this situation is not treated theoretically nor numerically in the references above. The main difficulty comes from the fact that optimization problems with a delayed control naturally belong to the class of *boundary control problems*.

2.2.1 Verification and existence results

Our approach is inspired from the *extended state* approach initiated by Ichikawa, see [Ich82], where the key idea is to lift the initial state space, namely $\mathbb{R}$, to the infinite dimensional Hilbert space $H = \mathbb{R} \times L^2([-d, 0], \mathbb{R})$, endowed with the inner product

$$\langle x, y\rangle_H = x_0 y_0 + \int_{-d}^{0} x_1(s) y_1(s) ds \qquad x, y \in H.$$

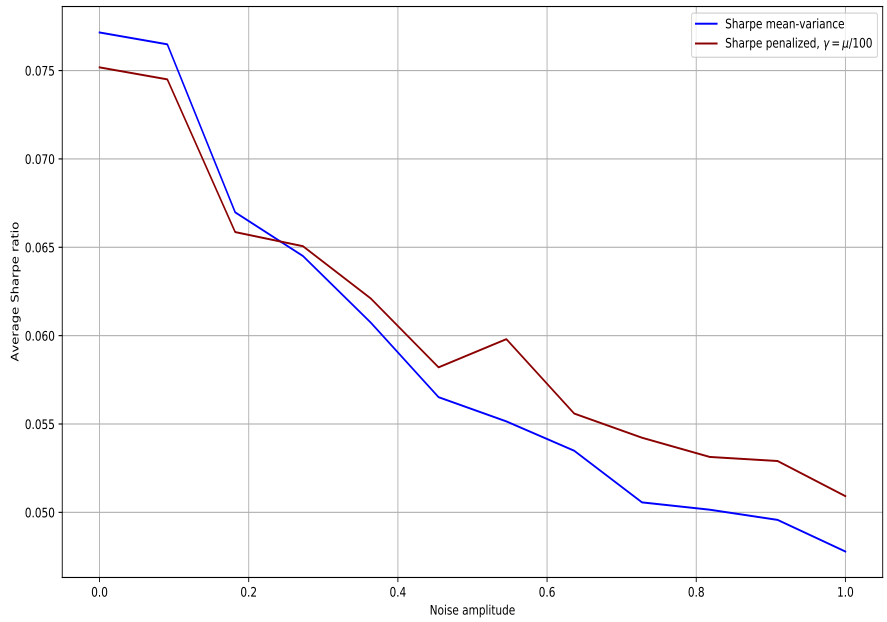


Figure 2.2: In this case the reference weights are equal to zero, and no Sharpe ratio is computed for the reference portfolio.

For each element $(x, u) \in H$, the first element x is to be interpreted as the *position* or state of the controlled system, and the second one $s \in [-d, 0] \mapsto u(s)$ as the *memory* of the control, containing at each time t the past values of the control for times in $[t-d, t]$. Our next result extends to the stochastic case the results obtained in [Ale+71] and [Ich82] in the case where a deterministic system is controlled with a delay. The value function and optimal control of (2.2.1) are expressed in terms of a self-adjoint bounded positive operator valued function $P \in C([0,T], \mathcal{L}(H,H))$ of the form

$$P_t : (x, \gamma(\cdot)) \mapsto \begin{pmatrix} P_{11}(t)x + \int_{-d}^{0} P_{12}(t,s)\gamma(s)ds \\ P_{12}(t,\cdot)x + P_{\hat{22}}(t,\cdot)\gamma(\cdot) + \int_{-d}^{0} P_{22}(t,\cdot,s)\gamma(s)ds \end{pmatrix}.$$

which satisfies a set of transport partial differential equations coupled through their source terms

$$\dot{P}_{11}(t) = \frac{P_{12}(t,0)^2}{P_{\hat{22}}(t,0)}, \qquad (\partial_t - \partial_s)(P_{12})(t,s) = \frac{P_{12}(t,0)P_{22}(t,s,0)}{P_{\hat{22}}(t,0)}, \tag{2.2.2}$$
$$(\partial_t - \partial_s)(P_{\hat{22}})(t,s) = 0, \qquad (\partial_t - \partial_s - \partial_r)(P_{22})(t,s,r) = \frac{P_{22}(t,s,0)P_{22}(t,0,r)}{P_{\hat{22}}(t,0)},$$

together with boundary conditions

$$P_{12}(t,-d) = bP_{11}(t), \qquad P_{\hat{22}}(t,-d) = \sigma^2 P_{11}(t), \tag{2.2.3}$$
$$P_{22}(t,s,-d) = bP_{12}(t,s), \qquad P_{22}(t,-d,r) = bP_{12}(t,r),$$

and terminal conditions

$$P_{11}(T) = 1, \qquad P_{12}(T,s) = P_{\hat{22}}(T,s) = P_{22}(T,s,r) = 0, \tag{2.2.4}$$

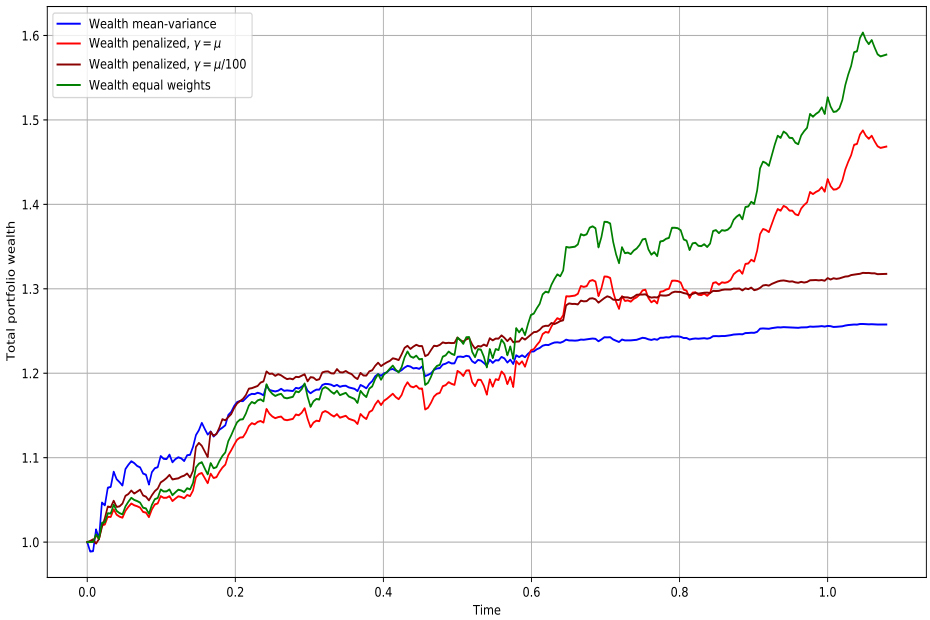


Figure 2.3: Sharpe ratios:
Mean-variance : 0.183
Equal weights : 0.258
Penalized $\gamma = \mu$: 0.260
Penalized $\gamma = \mu/100$: 0.226

for almost every $s, r \in [-d, 0]$.

Result 14: Verification theorem

Assume that there exists a solution P to (2.2.2)-(2.2.3)-(2.2.4). Then, the optimal control of optimization problem (2.2.1) is expressed as

$$\alpha_t^* = \frac{-1_{t \leq T-d}}{P_{\hat{22}}(t,0)} \left\{ X_t^{\alpha^*} P_{11}(t,0) + \int_{t-d}^{t} P_{22}(t,0,s-t)\alpha_s^* ds \right\},$$

and the optimal value is given by

$$V(z) = \langle P_0 z, z \rangle_H,$$

where $z = (x, \gamma) \in H$ denotes the initial state of the controlled system.

The next step is to give an existence result of a solution $P = (P_{11}, P_{12}, P_{\hat{22}}, P_{22})$ to the system of PDEs (2.2.2)-(2.2.3)-(2.2.4). In the case without delay, $d = 0$, although the control is not *penalized* in the optimized cost J of (2.2.1), the optimization problem (2.2.1) admits an optimizer provided $\sigma \neq 0$. Indeed, the more α is aggressive in bringing the terminal value X_T^2 to zero, the more the quadratic variation of X increases due to the diffusion term. As this quadratic variation contains the control term, it is implicitly

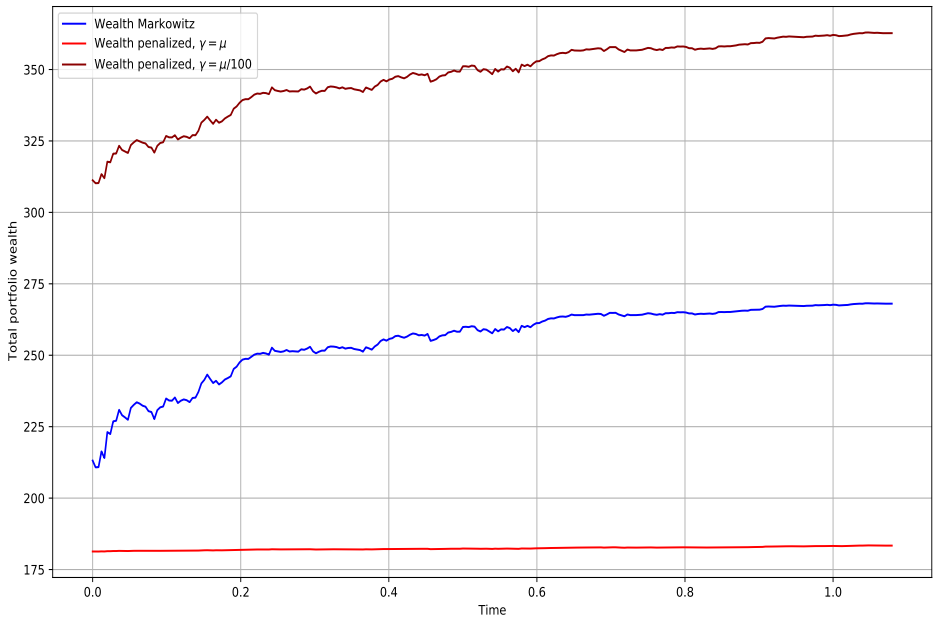


Figure 2.4: Total wealth of the mean-variance and penalized portfolios for $\gamma = \mu$ and $\gamma = \mu/100$, normalized by the standard deviation of daily returns, as a function of time.
Sharpe ratios:
mean-variance : 0.183
Penalized $\gamma = \mu$: 0.252
Penalized $\gamma = \mu/100$: 0.221

penalised and thus the problem remains well-posed. It is easily observable in the classical LQ stochastic optimization problem with controlled drift and diffusion, such as

$$\begin{aligned} dX_t^\alpha &= \alpha_t(bdt + \sigma dW_t), \qquad t \leq T, \\ X_0 &= x, \\ J(\alpha) &= \mathbb{E}[(X_T^\alpha)^2], \end{aligned}$$

where the optimal control is expressed as $\alpha_t^* = -\frac{b}{\sigma^2} X_t^{\alpha^*}$ and the value function $V_t = e^{(t-T)\frac{b^2}{\sigma^2}}$. A surprising finding in our work is the necessity for a more restricting constraint on the diffusion coefficient when the delay is not null, $d > 0$. Indeed, we find that a solution to the system (2.2.2)-(2.2.3)-(2.2.4) exists and is unique when a certain relation between the time horizon T, the delay d and the drift and volatility parameters b and σ is satisfied.

Let $a = (a_n)_{n\geq 1}$ denote the following sequence

$$\begin{cases} a_0 &= 1, \\ a_{n+1} &= a_n - \frac{d}{a_n}\left(\frac{b}{\sigma}\right)^2, \qquad n \geq 0, \end{cases}$$

and define $\mathcal{N} : (d, b, \sigma) \mapsto \inf\{n \geq 1 : a_n > 0 \text{ and } a_{n+1} \leq 0\}$. We express the existence result on P in term of the sequence a.

Result 15: Existence of $t \in [0,T] \mapsto P_t$

Assume $T < \mathcal{N}(d,b,\sigma)d$. Then (2.2.2)-(2.2.3)-(2.2.4) has a unique solution P on $[0,T]$ with $0 < P_{11}(0) < 1$.

2.2.2 Deep learning scheme for PDEs

We now propose an algorithm to approximate the solution $t \mapsto P_t$. We decide to rely on machine learning techniques to solve the system of couples PDEs, in the spirit of the emerging Physics Informed Neural Networks (PINNs) and Deep Galerkin literatures, see [SS18] and [RPK19] to name just a few.

We recall here some of the main ideas. Assume we want to numerically solve the following partial differential equation

$$\begin{aligned} \partial_t u + \mathcal{N}(u) &= 0, \qquad \text{on } \Omega, \\ u &= g, \qquad \text{on } \partial\Omega, \end{aligned} \tag{2.2.5}$$

where $\mathcal{N}$ is a nonlinear operator, Ω a bounded open subset and g a function on the boundary of the domain. The key idea is to use a neural network as a surrogate (or test function) to the solution u to (2.2.5). Thus, let us call $t \mapsto u(t,\Theta)$ such network, where Θ denotes its parameters and t a generic element of $\Omega \cup \partial\Omega$. The algorithm relies on minimizing the following loss functional over samples of $\Omega \cup \partial\Omega$ drawn randomly:

$$\mathcal{L}(\Theta,\mathcal{T}) = \mathcal{L}_u(\Theta,\mathcal{T}) + \mathcal{L}_f(\Theta,\mathcal{T}), \tag{2.2.6}$$

where $\mathcal{L}_u$ and $\mathcal{L}_f$ are defined as

$$\mathcal{L}_r(\Theta,\mathcal{T}) = \frac{1}{|\mathcal{T}_r|}\sum_{t\in\mathcal{T}_r} |(\partial_t + \mathcal{N})u(t,\Theta)|^2, \quad \mathcal{L}_f(\Theta,\mathcal{T}) \quad = \frac{1}{|\mathcal{T}_f|}\sum_{t\in\mathcal{T}_f} |u(t,\Theta) - g(t)|^2,$$

and serve respectively to train the partial derivatives of the network to satisfy the PDE and its value to satisfy the terminal condition. The subsets $\mathcal{T}_r = \mathcal{T} \cap \Omega$ and $\mathcal{T}_f = \mathcal{T} \cap \partial\Omega$ are respectively random subsets of Ω and $\partial\Omega$. We summarize the numerical procedure in Algorithm 2.

Algorithm 2: Deep learning scheme to solve PDEs

Initialize:
the learning rate η and the neural network $u(\cdot,\Theta)$; ;

For each batch:
Randomly sample $\mathcal{T} \subset \partial\Omega \cup \Omega$;
Compute the gradient's loss (2.2.6): $\nabla_\Theta \mathcal{L}(\Theta,\mathcal{T}) = \nabla_\Theta(\mathcal{L}_r + \mathcal{L}_f)(\Theta,\mathcal{T})$;
Update $\Theta \leftarrow \Theta - \eta\nabla_\Theta \mathcal{L}(\Theta,\mathcal{T})$;

Return: The set of optimized parameters Θ^*.

The precise structure of (2.2.2)-(2.2.3)-(2.2.4) led us toward a tailored-made algorithm.

Comments on the method: The Deep Galerkin method offers an easy to implement procedure to approximate solutions of any differential equations on the whole domain $\Omega\cup$

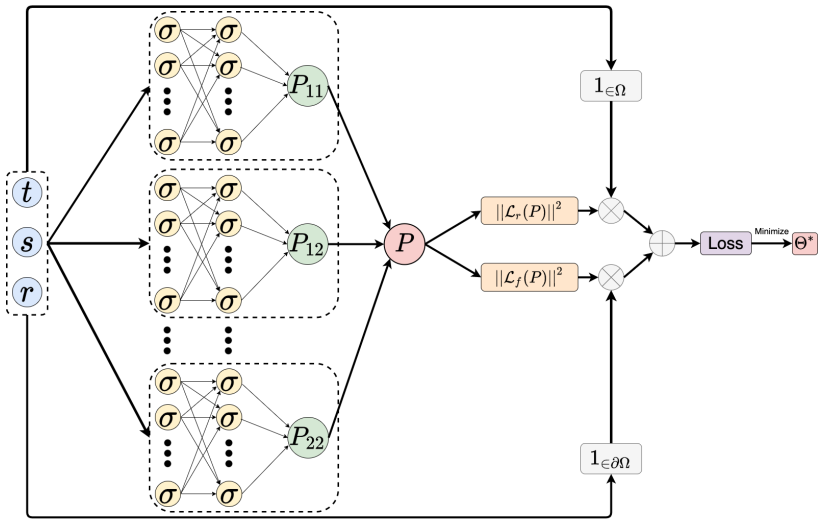


Figure 2.5: Structure of the neural network used to solve (2.2.2)-(2.2.3)-(2.2.4).

$\partial\Omega$. In theory they allow a high dimensional output space, although more investigations are expected on this topic. In Part II, we adapt the Deep Galerkin method to solve the system of PDEs (2.2.2)-(2.2.3)-(2.2.4) in the context of Markowitz portfolio allocation with execution delay. Note that in this case the output dimension is 4.

Nonetheless, the Deep Galerkin method does not extend well to the case where the input space is high dimensional, say 100 for instance. Such case is important as it is common in finance, for example in the case of portfolio allocations problems, where a large numbers of risky assets are considered, and in other fields such as operational research, physics, etc. In these case, BSDE based methods, such as the one presented in [HJW18] are well suited.

2.2.3 Application to Markowitz portfolio allocation with execution delay

We now apply our numerical scheme to the mean-variance portfolio selection, see [Mar52], where the risky assets are traded with a delay, in the spirit of the problem of hedging of European options with execution delay presented in [FF14]. The Mean-Variance portfolio selection problem in continuous-time consists in solving the following constrained problem

$$\begin{cases} \min_{\alpha\in\mathcal{A}} \mathrm{Var}(X_T^\alpha) \\ \text{s.t. } \mathbb{E}[X_T^\alpha] = c. \end{cases} \tag{2.2.7}$$

where X^α denotes the wealth of the investor controlled by an investment strategy α. In our work, we first study the case where only one risky asset is considered, and is traded with a delay,

$$\begin{cases} dX_t^\alpha = \alpha_{t-d}\left((\sigma\lambda)\,dt + \sigma dW_t\right), \qquad t\in[0,T], \\ X_0 = x_0, \quad \alpha_s = \gamma_s, \qquad \forall s\in[-d,0], \end{cases} \tag{2.2.8}$$

where α denotes the wealth invested in the risky asset, and λ and σ are constants representing respectively the risk premium and the volatility of the risky asset. Note that the one delayed asset case allows for a direct application of Results 14 and 15:

Result 16: One delayed asset case

Assume $T < d\mathcal{N}(d, (\sigma\lambda), \sigma)$, set $\xi^* = c - \eta^*$ and

$$\eta^* = \frac{K(\gamma) + P_{11}(0)(x_0 - c)}{1 - P_{11}(0)},$$
$$K(\gamma) = \int_{-d}^{0} \gamma_s P_{12}(0, s) ds.$$

Define $\alpha^*(\xi)$ as the investment strategy

$$\alpha_t^*(\xi^*) = \frac{-1_{t \leq T-d}}{P_{\hat{22}}(t, 0)} \left\{ (X_t^{\alpha^*} - \xi^*) P_{12}(t, 0) + \int_{t-d}^{t} \alpha_s^*(\xi) P_{22}(t, 0, s-t) ds \right\},$$

where P denotes the solution to (2.2.2)-(2.2.3)-(2.2.4). Then, the optimization problem (2.2.7)-(2.2.8) admits $\alpha^*(\xi)$ as an admissible optimal feedback strategy and the optimal value, corresponding to the variance of the terminal wealth of the portfolio, is given by

$$\mathrm{Var}(X_T^{\alpha^*}) = \frac{P_{11}(0)}{1 - P_{11}(0)} (x_0 - c + K(\gamma))^2 + \int_{-d}^{0} \gamma_s^2 P_{\hat{22}}(0, s) ds + \int_{[-d,0]^2} \gamma_s \gamma_u P_{22}(0, s, r) ds dr.$$

Numerical resutls: From the aforementioned result emerges a direct application for the numerical scheme presented in the previous section. See Figures 2.6 and 2.7 for some examples of interests.

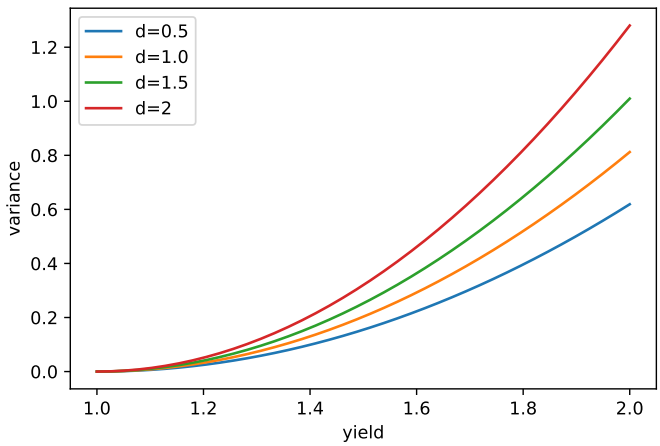
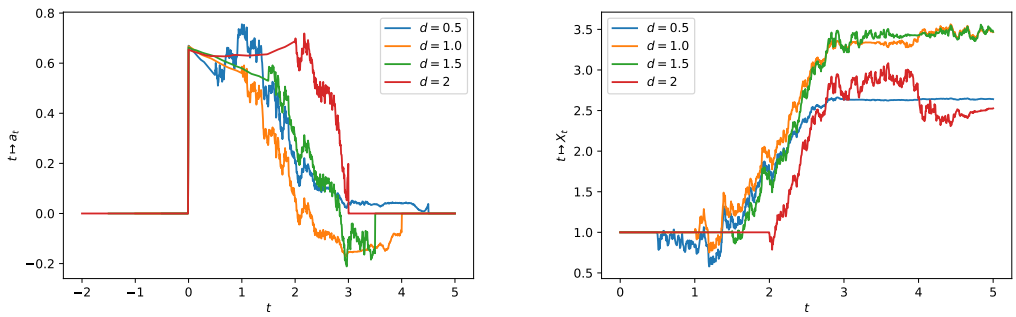


Figure 2.6: Efficient frontier with $\sigma = 1$, $\lambda = 0.5$, $T = 5$ and $\gamma \equiv 0$.

To further explore the effect of the delay on the control problem we also study, in Chapter II, the Mean-Variance portfolio selection problem, in the case where two risky correlated assets are considered, one being traded with a delay and the other



(a)

Figure 2.7: Optimal strategies and optimal portfolios with $c = 1.6$, $\sigma = 1$, $\lambda = 0.5$, and $T = 5$. Left: $t \mapsto \alpha^*$, right: $t \mapsto X_t^*$. Note the destabilization effect and the supplement of volatility induced by the delay feature. Note also the tendency to invest more aggressively the delayed investor has, as she has less time to ensure the promised yield. $\xi^*(d = 0.5) = 2.57$, $\xi^*(d = 1) = 2.68$, $\xi^*(d = 1.5) = 2.80$, $\xi^*(d = 2) = 2.97$.

instantaneously:

$$\begin{cases} dX_t^{(\alpha,\beta)} = \alpha_t \left\{(\sigma_1\lambda_1)dt + \sigma_1 dW_t^1\right\} + \beta_{t-d}\left\{(\sigma_2\lambda_2)dt + \sigma_2 dW_t^2\right\}, \quad t \in [0,T], \\ X_0 = x_0, \quad \beta_s = \gamma_s, \quad s \in [-d, 0], \\ \langle W^1, W^2\rangle_t = \rho t, \end{cases} \tag{2.2.9}$$

where α denotes the amount invested in the risky undelayed asset, β the amount invested in the risky delayed asset, and λ_i and σ_i are constants representing respectively the risk premiums and the volatilities of the risky assets.

Following an heuristic approach, we define the following set of Riccati PDEs on the domain $[0,T] \times [-d,0]^2$, which will appear in the expression of the optimal control and value function of this problem,

$$\begin{aligned} &\dot{P}_{11}(t) = \lambda_1^2 P_{11}(t) + \frac{P_{12}(t,0)^2}{P_{\hat{2}\hat{2}}(t,0)}, \\ &(\partial_t - \partial_s)(P_{12})(t,s) = \lambda_1^2 P_{12}(t,s) + \frac{P_{12}(t,0)P_{22}(t,s,0)}{P_{\hat{2}\hat{2}}(t,0)}, \\ &(\partial_t - \partial_s)(P_{\hat{2}\hat{2}})(t,s) = 0, \\ &(\partial_t - \partial_s - \partial_r)(P_{22})(t,s,r) = \lambda_1^2 \frac{P_{12}(t,s)P_{12}(t,r)}{P_{11}(t)} + \frac{P_{22}(t,s,0)P_{22}(t,0,r)}{P_{\hat{2}\hat{2}}(t,0)}, \end{aligned} \tag{2.2.10}$$

accompanied by the boundary conditions, for almost any $t, s \in [0,T] \times [-d,0]$

$$\begin{aligned} &P_{12}(t,-d) = \lambda_2\sigma_2\left(1 - \rho\frac{\lambda_1}{\lambda_2}\right)P_{11}(t), \qquad P_{\hat{2}\hat{2}}(t,-d) = \sigma_2^2\left(1-\rho^2\right)P_{11}(t), \\ &P_{22}(t,s,-d) = \lambda_2\sigma_2\left(1 - \rho\frac{\lambda_1}{\lambda_2}\right)P_{12}(t,s), \quad P_{22}(t,-d,s) = \lambda_2\sigma_2\left(1 - \rho\frac{\lambda_1}{\lambda_2}\right)P_{12}(t,s), \end{aligned} \tag{2.2.11}$$

and the terminal constraints

$$P_{11}(T) = 1, \qquad P_{12}(T,s) = P_{\hat{2}\hat{2}}(T,s) = P_{22}(T,s,r) = 0, \tag{2.2.12}$$

for almost every $s, r \in [-d, 0]$.

We are now equipped to present the optimal strategy and value of problem (2.2.1) in the 2-assets case

Result 17: One delayed asset and one undelayed asset case

Assume there exists a solution P to (2.2.10)-(2.2.11)-(2.2.12). Set $\xi^* = c - \eta^*$ and

$$\eta^* = \frac{K(\gamma) + P_{11}(0)(x_0 - c)}{1 - P_{11}(0)},$$
$$K(\gamma) = \int_{-d}^{0} \gamma_s P_{12}(0, s) ds.$$

Define $(\alpha^*(\xi^*), \beta^*(\xi^*))$ as the investment strategy

$$\alpha_t^*(\xi) = -\left\{ \frac{\lambda_1}{\sigma_1} (X_t^* - \xi) + \rho \frac{\sigma_2}{\sigma_1} \beta_{t-d}^* + \frac{\lambda_1}{\sigma_1 P_{11}(t)} \int_{t-d}^{t} \beta_s^*(\xi) P_{12}(t, s-t) ds \right\}.$$
$$\beta_t^*(\xi) = \frac{-1_{t \leq T-d}}{P_{\hat{2}\hat{2}}(t, 0)} \left\{ P_{12}(t, 0) (X_t^* - \xi) + \int_{t-d}^{t} \beta_s^*(\xi) P_{22}(t, 0, r-t) dr \right\},$$

Then, the optimization problem (2.2.7)-(2.2.9) admits $(\alpha^*(\xi^*), \beta^*(\xi^*))$ as an admissible optimal feedback strategy and the optimal value is

$$\mathrm{Var}(X_T^{\alpha^*}) = \frac{P_{11}(0)}{1 - P_{11}(0)} (x_0 - c + K(\gamma))^2 + \int_{-d}^{0} \gamma_s^2 P_{\hat{2}\hat{2}}(0, s) ds + \int_{[-d,0]^2} \gamma_s \gamma_u P_{22}(0, s, r) ds dr.$$

Numerical simulations: The flexibility of the deep learning scheme 2 allows for an easy adaptation to the new set of Riccati PDEs (2.2.10)-(2.2.11)-(2.2.12). See Figure 2.8 to see the interaction between the delay feature and the correlation ρ, and Chapter II for a more in depth analysis.

2.3 Deep learning algorithms for fully nonlinear PDEs

Part III is devoted to the resolution of fully nonlinear partial diferential equations (PDEs) of the form

$$\begin{cases} \partial_t u + H(x, D_x u, D_x^2 u) &= 0, \quad (t, x) \in [0, T) \times \mathbb{R}^d, \\ u(T, x) &= g(x), \quad x \in \mathbb{R}^d, \end{cases} \tag{2.3.1}$$

where the Hamiltonian $H : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^{d \times d} \to \mathbb{R} \cup \{\infty\}$ is a lower semi-continuous convex function with respect to the two last arguments (z, γ), and g is a measurable function on $\mathbb{R}^d$. The resolution of this class of PDE and the good approximation of the second spatial derivative $D_x^2 u$ of the solution is a notorious challenging problem. We propose algorithms to solve this problem based on Machine Learning techniques with the aim of approximating the value of the PDE solution and its two first spatial derivatives $D_x u$ and $D_x^2 u$ on the entire space-time domain of interest. Deep learning methods applied to solve partial differential equations or stochastic control problems

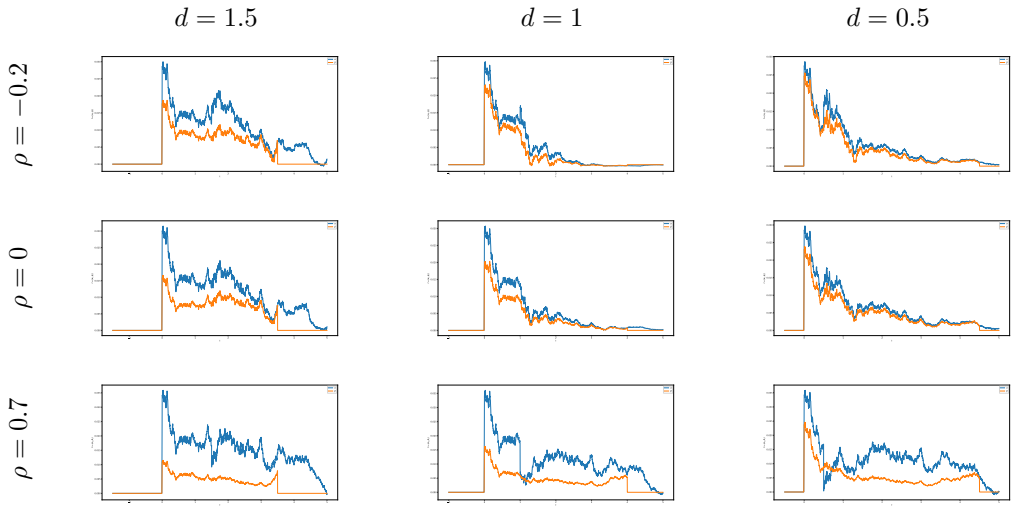


Figure 2.8: $t \mapsto (\alpha_t^*, \beta_t^*)$, with $\sigma_1 = \sigma_2 = 1$, $\lambda_1 = \lambda_2 = 0.5$ and $T = 5$. Blue : α^*, orange : β^*. The same realizations of W and B were used for all experiments. Note that the more positively correlated the assets are, the more favored the undelayed asset is.

have been the object of a lot of recent literature and have been applied with success in situations where traditional numerical methods fail, see [HL22] for a survey of the recent development of these methods. These methods are of great interests in several applied fields, including finance where nonlinear PDEs in high dimension appear from pricing problems, see [BT22] for an example, and where a lot of stochastic control problems appear in hedging and strategy design problems, see [HLLR16], [GM19] and [GMP20] for examples of applications to CVA computation, market making. and share repurchase problems.

The algorithms we propose are conducted in two steps. First the PDE is rewritten in its dual stochastic control representation form by expressing the Hamiltonian H in terms of its concave conjugate w.r.t the last two variables

$$H(x, z, \gamma) = \sup_{(b,c)\in D_f} \big[b \cdot z + \frac{1}{2} c : \gamma + \mathrm{f}(x, b, c)\big], \qquad \text{for } x \in \mathbb{R}^d, z \in \mathbb{R}^d, \gamma \in \mathbb{S}^d,$$

where $A : B = \mathrm{Tr}(A^\top B)$ and $D_f := \{(b, c) \in \mathbb{R}^d \times \mathbb{S}^d : \mathrm{f}(x, b, c) > -\infty\} \subset \mathbb{R}^d \times \mathbb{S}^d_+$. Using this expression, it is well known that the solution to this PDE admits the stochastic representation

$$u(t, x) = \sup_{\alpha\in\mathcal{A}} \mathbb{E}\Big[g(X_T^{t,x,\alpha}) + \int_t^T \mathrm{f}(X_s^{t,x,\alpha}, \alpha_s)\mathrm{d}s\Big], \quad (t, x) \in [0, T] \times \mathbb{R}^d,$$

where $X = X^{t,x,\alpha}$ is solution to the stochastic differential equation

$$\mathrm{d}X_s = \mathrm{b}(X_s, \alpha_s)\mathrm{d}s + \sigma(X_s, \alpha_s)\mathrm{d}W_s, \quad t \le s \le T, \quad X_t = x.$$

Following a method already developed in the literature, we use neural networks functions $\mathrm{a}_\theta : [0, T] \times \mathbb{R}^d \to A \subset \mathbb{R}^q$, to approximate the optimal feedback control, by maximizing over parameters θ the objective function

$$J(\theta) = \mathbb{E}\Big[g(X_T^\theta) + \int_0^T f(X_t^\theta, \mathrm{a}_\theta(t, X_t^\theta))\mathrm{d}t\Big], \tag{2.3.2}$$

where the state X^θ is driven by the neural network control and has an initial state X_0^θ distributed according to some law μ_0 on $\mathbb{R}^d$.

Once we get an approximation of the optimal feedback control, denoted as $\mathrm{a}^* = \mathrm{a}_{\theta^*}$, where θ^* is the set of "optimal parameters" which maximizes (2.3.2), it is possible to obtain an approximation of the PDE solution on isolated points (t, x) of the domain by computing the conditional expectation

$$u(t,x) = \mathbb{E}\Big[g(X_T^*) + \int_t^T f(X_s^*, \mathrm{a}^*(s, X_s^*))\mathrm{d}s \Big| X_t^* = x\Big], \tag{2.3.3}$$

by Monte-Carlo, where X^* is the stochastic process controlled by the optimal control approximation a^*. Since our purpose is to solve the PDE on the entire domain, we instead devise three Machine Learning methods to obtain this solution. We will refer to these methods as *differential learning* methods as they consist in training the value and the derivatives of the neural network simultaneously. We will present them in the following section.

2.3.1 Differential learning methods

The first method, called Differential regression learning, is based on the conditional expectation representation (2.3.3) of the PDE solution, and its fundamental characterization property as an L^2-regression

$$u(t, \hat{X}_t) = \arg\min_{v_t} \mathbb{E}\big|\hat{Y}_T^t - v_t(\hat{X}_t)\big|^2(\hat{X}_t), \qquad \text{for all } t \in [0,T],$$

where the target payoff is

$$\hat{Y}_T^t = g(\hat{X}_T) + \int_t^T f(\hat{X}_s, \hat{\mathrm{a}}(s, \hat{X}_s))\mathrm{d}s, \quad t \in [0,T]. \tag{2.3.4}$$

This formulation suggests to use a class of neural network functions ϑ^η on $[0,T] \times \mathbb{R}^d$, with parameters η, for approximating the value function u, by minimizing a loss function

$$\begin{aligned} \hat{L}_{val}(\eta) &= \mathbb{E}\Big[\int_0^T \big|\hat{Y}_T^t - \vartheta^\eta(t, \hat{X}_t)\big|^2 \mathrm{d}t\Big] \\ &\simeq \mathbb{E}\Big[\int_0^T \big|Y_T^{*,t} - \vartheta^\eta(t, X_t^*)\big|^2 \mathrm{d}t\Big] \;=: L_{val}^*(\eta), \end{aligned} \tag{2.3.5}$$

where $Y_T^{*,t}$ is the payoff (2.3.4) computed on trajectories controlled by optimal control approximation a^*.

Similarly, an unbiased estimator of the derivative of the PDE solution can be obtained as the conditional expectation of the pathwise derivative of the payoff $\hat{Y}_T^t$

$$D_x u(t, \hat{X}_t) = \mathbb{E}\Big[\hat{Z}_T^t \big| \mathcal{F}_t\Big], \quad t \in [0,T],$$

where $\hat{Z}_T^t = D_{\hat{X}_t}\hat{Y}_T^t$. This suggests to complete the learning of the value function by the learning of its derivative. In practice, this can be done by training the same neural network function ϑ^η to minimize the loss function

$$L_{der}^*(\eta) \;=:\; \mathbb{E}\Big[\int_0^T \big|Z_T^{*,t} - D_x\vartheta^\eta(t, X_t^*)\big|^2 \mathrm{d}t\Big],$$

where $Z_T^{*,t} = D_{X_t^*} Y_T^{*,t}$ valued in $\mathbb{R}^d$, is obtained by automatic differentiation as

$$Z_T^{*,t} = \big(D_{X_t} X_T\big)^\top D_x g(X_T) + \int_t^T \big(D_{X_t} X_s\big)^\top D_x f^{\mathrm{a}^*}(s, X_s)\mathrm{d}s, \quad t \in [0,T], \tag{2.3.6}$$

where we dropped the superscript * for the state $X = X^*$, denoted by $f^{\mathrm{a}^*}(t,x) = f(x, \mathrm{a}^*(t,x))$, and assumed that g and f are continuously differentiable.

Result 18: Differential regression learning

The *differenial regression learning* method thus consists in training a neural network function ϑ^η on $[0,T] \times \mathbb{R}^d$ to simultaneously minimize the loss functions L^*_{val} and L^*_{der} defined above.

In the expression (2.3.6), the derivative $D_x f^{\mathrm{a}^*} = D_x f + (D_x \mathrm{a}^*)^\top D_a f$ can be computed easily by automatic differentiation of the neural network a_{θ^*}. Furthermore, an approximation of the flow derivative of the optimal state process $D_{X_t} X_s$ can also be computed in practice by automatic differentiation of the Euler diffusion scheme of this state. This payoff derivative is also well defined when the function g is differentiable almost everywhere as the measure of the state process at terminal time X_T is absolutely continuous w.r.t the Lebesgue measure. A formal argument for this assertion can be given, based on the use of the Malliavin derivative.

The two other methods developped are similar but rely on a pathwise approximation of the payoff on a set of random *optimally controlled* state paths rather than on the estimation of a conditional expectation.

From the martingale representation related to the conditional expectation (2.3.3), we can rewrite, using Itô's formula

$$\hat{Y}_T^t = u(t, \hat{X}_t) + \int_t^T \big(D_x u(s, \hat{X}_s)\big)^\top \sigma^{\hat{\mathrm{a}}}(s, \hat{X}_s)\mathrm{d}W_s, \quad t \in [0,T], \tag{2.3.7}$$

where $\hat{Y}_T^t = g(\hat{X}_T) + \int_t^T f(\hat{X}_s, \hat{\mathrm{a}}(s, \hat{X}_s))\mathrm{d}s$.

Result 19: Pathwise martingale learning

This suggests again to use a class of NN functions ϑ^η on $[0,T] \times \mathbb{R}^d$, with parameters η, for approximating the value function u by training the NN to minimize the loss

$$L^*_{mar}(\eta) := \mathbb{E}\Big[\int_0^T \Big|Y_T^{*,t} - \vartheta^\eta(t, X_t^*) - \int_t^T \big(D_x \vartheta^\eta(s, X_s^*)\big)^\top \sigma^{\mathrm{a}^*}(s, X_s^*)\mathrm{d}W_s\Big|^2 \mathrm{d}t\Big].$$

We call this training method the *Pathwise martingale learning*.

The third method, called *Pathwise differential learning* is similar to this one, with the addition of another training loss based on the pathwise derivative of the martingale representation (2.3.7) yielding a second estimator linking the first and second derivatives of $u(t,x)$

$$D_{\hat{X}_t}\hat{Y}_T^t = D_x u(t,\hat{X}_t) + \int_t^T \Big(\big[D_x\sigma^{\hat{\mathrm{a}}}(s,\hat{X}_s) \bullet_3 D_{\hat{X}_t}\hat{X}_s\big] \bullet_1 D_x u(s,\hat{X}_s) + \sigma^{\hat{\mathrm{a}}}(s,\hat{X}_s)^\top D_x^2 u(s,\hat{X}_s) D_{\hat{X}_t}\hat{X}_s \Big)^\top \mathrm{d}W_s, \quad t\in[0,T],$$

where $(\boldsymbol{M} \bullet_3 B)_{i_1 i_2 \ell} = \sum_{i_3=1}^{d_3} M_{i_1 i_2 i_3} B_{i_3 \ell}$ for $M \in \mathbb{R}^{d_1\times d_2\times d_3}$ and $B \in \mathbb{R}^{d_p\times d}$ and $(\boldsymbol{M} \bullet_1 b)_{i_2 i_3} = \sum_{i_1=1}^{d_1} M_{i_1 i_2 i_3} b_{i_1}$ for $M \in \mathbb{R}^{d_1\times d_2\times d_3}$ and $b \in \mathbb{R}^{d_p}$.

Result 20: Pathwise differential learning

The *pathwise diffential learning* method consists in training a neural network to minimise the loss

$$L^*_{dermar}(\eta) = \mathbb{E}\Big[\int_0^T \Big| Z_T^{*,t} - D_x\vartheta^\eta(t,X_t) - \int_t^T \Big(\big[D_x\sigma^{\mathrm{a}^*}(s,X_s) \bullet_3 D_{X_t}X_s\big] \bullet_1 D_x\vartheta^\eta(s,X_s) + \sigma^{\mathrm{a}^*}(s,X_s)^\top D_x^2\vartheta^\eta(s,X_s) D_{X_t}X_s \Big)^\top \mathrm{d}W_s \Big|^2 \mathrm{d}t\Big],$$

in addition to the loss L^*_{mar} previously defined above.

The interest of adding a loss related to the derivative of the payoff we want to approximate is illustrated in the Figure 2.3.5 where we show the approximation obtained by training the same network to solve the non-linear PDE with an Hamiltonian given on $(0,\infty)$ by

$$H(x,\gamma) = \begin{cases} \frac{1}{2}\sigma^2 \frac{x^2\gamma}{1-\lambda x^2\gamma}, & \text{if } \lambda x^2\gamma < 1 \\ \infty, & \text{otherwise.} \end{cases} \tag{2.3.8}$$

related to the pricing of options in a Black-Scholes model with linear market impact, by the *Differential regression learning* method and by minimizing only the loss L^*_{val} defined in Equation (2.3.5) (denoted *simple learning*).

2.3.1.1 Numerical results

The neural network used to approximate the optimal control has the same structure as the one used to approximate the value function, represented in Figure 2.10.

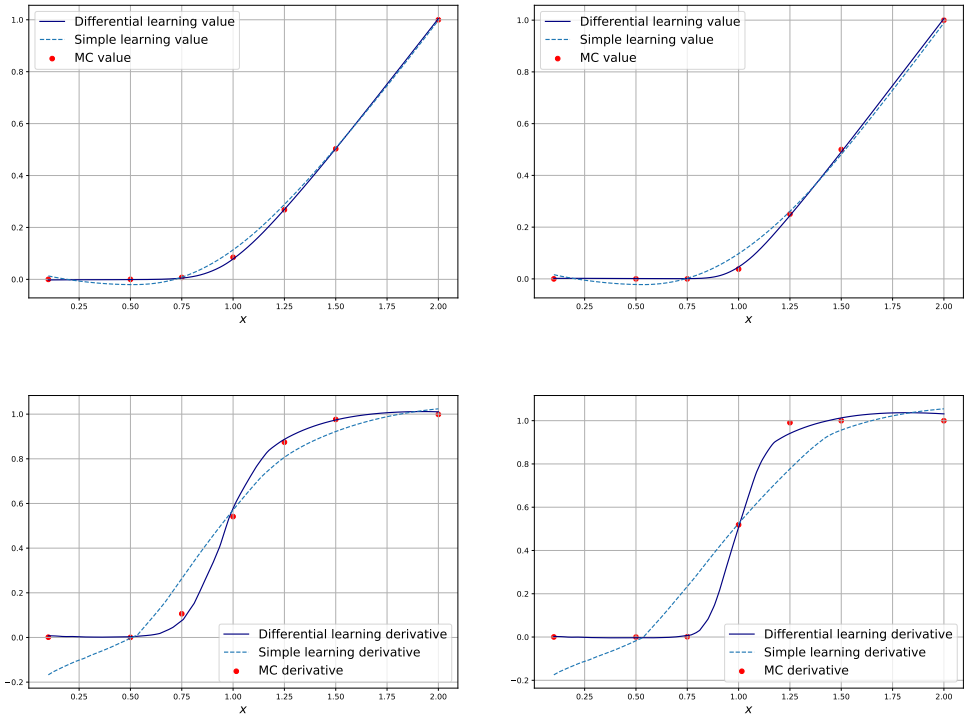
It is composed of two dense sub-networks, taking respectively the time and the space variable as input, whose outputs are concatenated and passed through additional dense layers.

Our algorithms are time discretized versions of the algorithms described in the previous Section.

As a first step, we consider a neural network a_θ from $[0,T]\times\mathbb{R}^d$ into $A\subset\mathbb{R}^q$ for the approximation of the feedback control, and the associated discretised state process

$$X^\theta_{t_{n+1}} = X^\theta_{t_n} + \mathrm{b}(X^\theta_{t_n}, \mathrm{a}_\theta(t_n, X^\theta_{t_n}))\Delta t + \sigma(X^\theta_{t_n}, \mathrm{a}_\theta(t_n, X^\theta_{t_n}))\Delta W_{t_n}, \quad n = 0,\dots,N-1,$$

starting from $X_0 \sim \mu_0$ (probability distribution on $\mathbb{R}^d$), and where $\Delta W_{t_n} = W_{t_{n+1}} - W_{t_n}$. For the training of this neural network, we use a batch of M independent trajectories



(a) $t = 0.5$ (b) $t = 0.9$

Figure 2.9: Value function values (first line) and derivatives (second line) obtained by *Differential Learning* (navy curve), *Simple learning* (blue dashed curve) and Monte Carlo (red dots) plotted as functions of x, for fixed values of t.

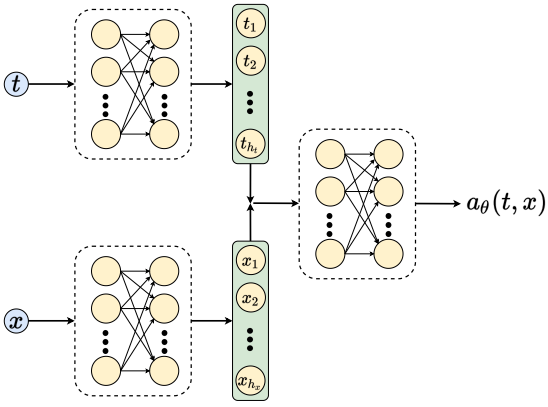


Figure 2.10: Structure of the neural network used to approximate the optimal control and value function.

$\{x_{t_n}^{m,\theta}, t_n \in \mathcal{T}_N\}$, $m = 1, \ldots, M$, of $\{X_{t_n}^{\theta}, t_n \in \mathcal{T}_N\}$, and apply a stochastic gradient ascent method to the empirical gain function:

$$J_M(\theta) = \frac{1}{M} \sum_{m=1}^{M} \Big[g(x_T^{m,\theta}) + \sum_{n=0}^{N-1} f\big(x_{t_n}^{m,\theta}, \mathrm{a}_\theta(t_n, x_{t_n}^{m,\theta})\big) \Delta t_n \Big].$$

We obtain an approximation $\mathrm{a}^* = \mathrm{a}_{\theta^*}$ of the optimal control and of the associated optimal state process with $X^* = X^{\theta^*}$.

We then consider a neural network ϑ^η from $[0,T] \times \mathbb{R}^d$ into $\mathbb{R}$ for the approximation of the value function. The derivatives $D_x g$, $D_x f$, $D_x \mathrm{b}$, $D_x \sigma$, $D_x \mathrm{a}$, and $D_x \vartheta^\eta$ that appear in the differential regression learning methods are computed straightforwardly by auto-differentiation. Concerning the flow derivative of the approximate optimal state process, it can be efficiently obtained by storing the one-step derivatives:

$$D_{X_{t_n}} X_{t_{n+1}} \quad = \quad I_d + D_x b^{\mathrm{a}^*}(t_n, X_{t_n})\Delta t_n + \sum_{j=1}^{d} D_x \sigma_j^{\mathrm{a}^*}(t_n, X_{t_n})\Delta W_{t_n}^j, \quad n = 0, \dots, N-1,$$

and then using the chain rule

$$D_{X_{t_n}} X_{t_p} \quad = \quad D_{X_{t_n}} X_{t_{n+1}} \cdots D_{X_{t_{p-1}}} X_{t_p}, \qquad \text{for } n < p \in [\![0, N]\!].$$

From the approximation of the optimal control, the solution of the PDE can be approximated by training the neural network ϑ^η according to the three methods presented above, by minimizing discretized versions of the losses we defined.

The quality of the approximations obtained is then evaluated by computing the loss associated to the residue and boundary conditions of the PDE,

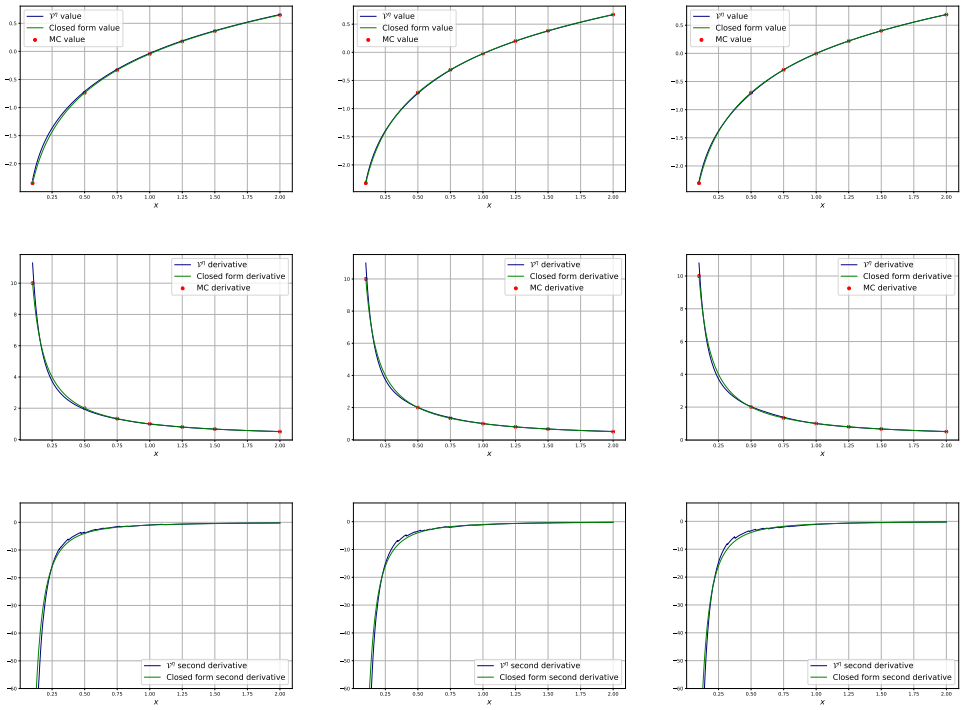
$$\mathcal{L}_{res} := \frac{1}{|\mathcal{T}||\chi|} \sum_{t \in \mathcal{T}, x \in \chi} \left| \partial_t \vartheta^\eta + H(x, D_x \vartheta^\eta, D_x^2 \vartheta^\eta) \right|^2, \tag{2.3.9}$$

$$\mathcal{L}_{term} := \frac{1}{|\chi|} \sum_{x \in \chi} \left| \vartheta^\eta(T, x) - g(x) \right|^2, \tag{2.3.10}$$

two losses that would be equal to zero for a perfect fit. Another method is to compute the value of the PDE solution by computing the conditional expectation (2.3.3) and the conditional expectation representation of the PDE solution's derivative for a set of points (t, x). For the example presented here, the residue and terminal losses obtained with the three methods are presented in a table and the Monte Carlo points are used as a graphical verification of the approximation quality and are plotted alongside the solution obtained with our three methods. For a terminal condition $g(x) = \ln(x)$, the PDE (2.3.8) mentioned above has a closed form solution given by

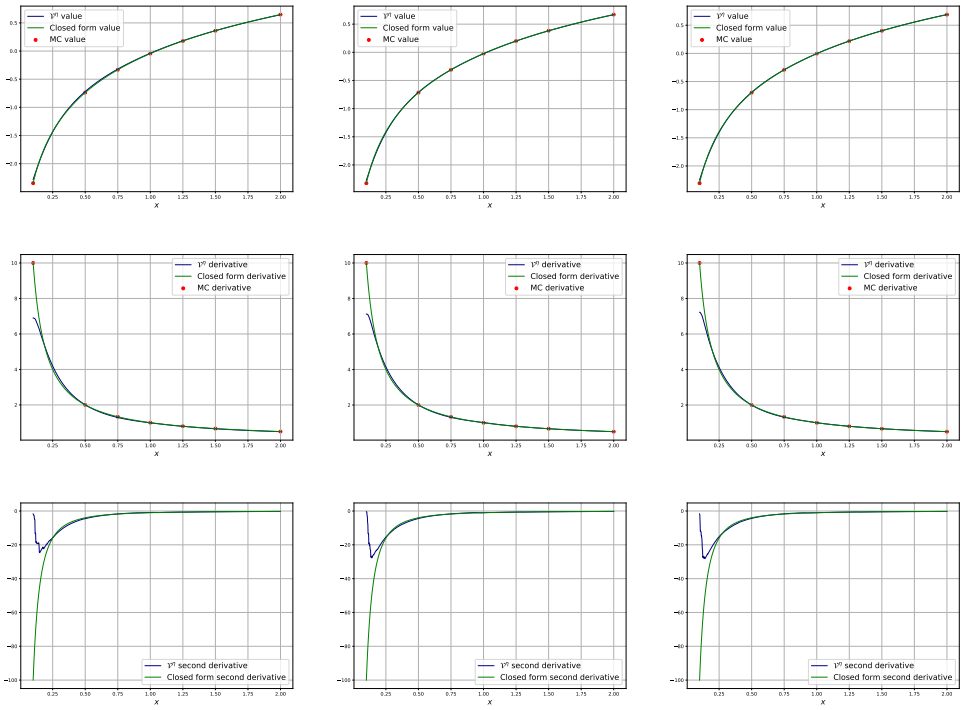
$$u(t, x) = \ln(x) - \frac{\sigma^2}{2(1+\lambda)}(T - t).$$

In Table 2.1, we give the validation losses (2.3.9) and (2.3.10) and the training time obtained by training a neural network to solve PDE (2.3.8) with terminal logarithmic payoff using the three methods. During these training, we used 8192 starting points x_0 and Brownian trajectories and trained the network for 500 epochs. We plot below the value function $\vartheta^\eta(t, x)$ and its derivatives $\partial_x \vartheta^\eta(t, x)$ and $\partial_{xx} v_\eta(t, x)$ for fixed values $t = 0$, $t = 0.5$, $t = 0.9$, parameter $\sigma = 0.2$ and PDE parameter $\lambda = 5e^{-3}$, and compare it with the closed-form solution of the problem and Monte Carlo points. Figure 2.11 corresponds to the Differential regression learning method, Figure 2.12 corresponds to the pathwise martingale learning while Figure 2.13 corresponds to the pathwise differential learning method. Graphically, the results of the Differential regression learning and the Pathwise differential learning methods are very close to the points obtained by Monte Carlo estimation for the value and the first derivative. The Pathwise methods does not give a good approximation of the value and the derivatives for values of x smaller than 1. The difference of performance between the Pathwise and the Differential pathwise methods is analogous to the one observed between Differential regression learning and "simple" regression learning in Figure 2.9, demonstrating the interest of adding a regression term for the derivative of the neural network.



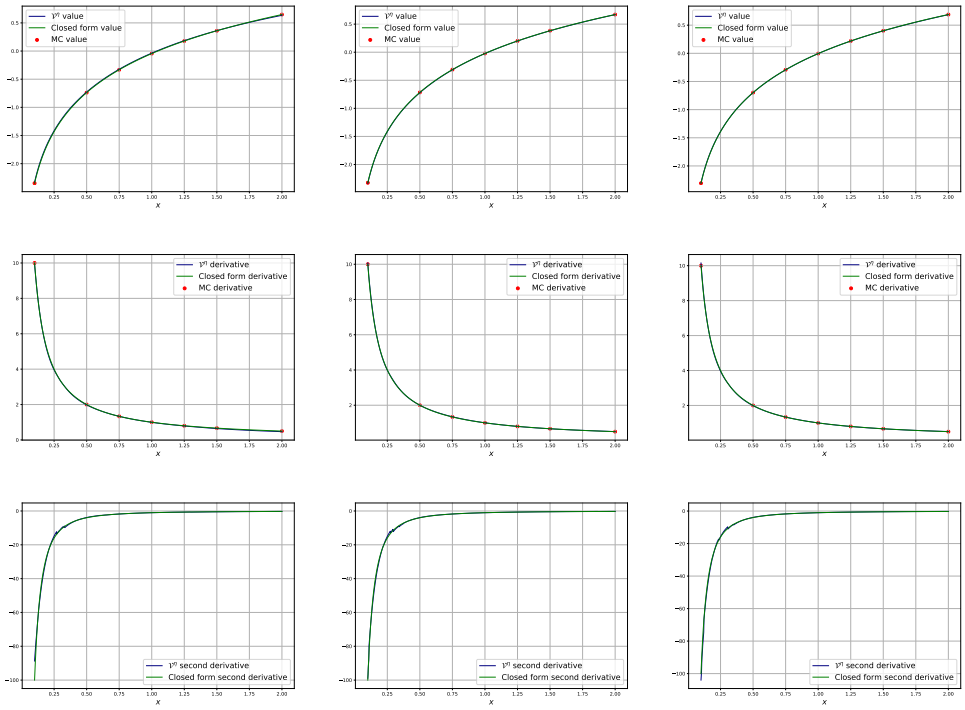
(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 2.11: Value function ϑ^η and its first and second derivative obtained by Differential Regression Learning for a logarithmic option payoff, with parameter $\sigma = 0.2$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 2.12: Value function ϑ^η and its first and second derivative obtained by Pathwise Learning for a logarithmic option payoff, with parameter $\sigma = 0.2$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 2.13: Value function ϑ^η and its first and second derivative obtained by Pathwise Differential Learning for a logarithmic option payoff, with parameter $\sigma = 0.2$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.

	Diff. regr. learning	Path. 1NN	Path. diff. 1NN
Residual loss	$2.046e^{-3}$	$3.484e^{-4}$	$6.644e^{-4}$
Residual loss + terminal loss	$2.179e^{-3}$	$3.864e^{-4}$	$6.758e^{-4}$
Training time (500 epochs)	163s	130s	525s

Table 2.1: Residual and boundary losses computed on a 1002x1002 time and space grid with $t \in [0, 0.9]$ and $x \in [0.1, 2]$ for a terminal logarithmic payoff, with parameter $\sigma = 0.2$ and linear market impact factor $\lambda = 5e^{-3}$.

2.3.1.2 DeepOnet for parametric terminal functions

Last, our goal is to design a Machine Learning method allowing us to directly obtain a solution to a PDE of type (2.3.1) with parametric terminal condition g_K, for every value of the parameter $K \in \mathbb{R}^p$ in a compact set. To do this, we rely on a class of neural networks called DeepONet, aiming to approximate functional operators. These neural networks are based on the following universal approximation theorem for operators.

Theorem 2.3.1. *Suppose that σ is a continuous non-polynomial function, X is a Banach space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$, G is a nonlinear continuous operator, which maps V into $C(K_2)$. Then for any $\epsilon > 0$, there are positive integers n, p, m, constants c_i^k, ξ_{ij}^k, θ_i^k, $\zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, ..., n$, $k = 1, ..., p$, $j = 1, ..., m$, such that*

$$\left\| G(u)(y) - \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{branch\ net} \underbrace{\sigma\left(w_k y + \zeta_k\right)}_{trunk\ net} \right\| < \epsilon,$$

holds for all $u \in V$ and $y \in K_2$.

The network we use, whose structure is presented in Figure 2.14, is a mix of the *standard* deepONet and of the network we used previously.

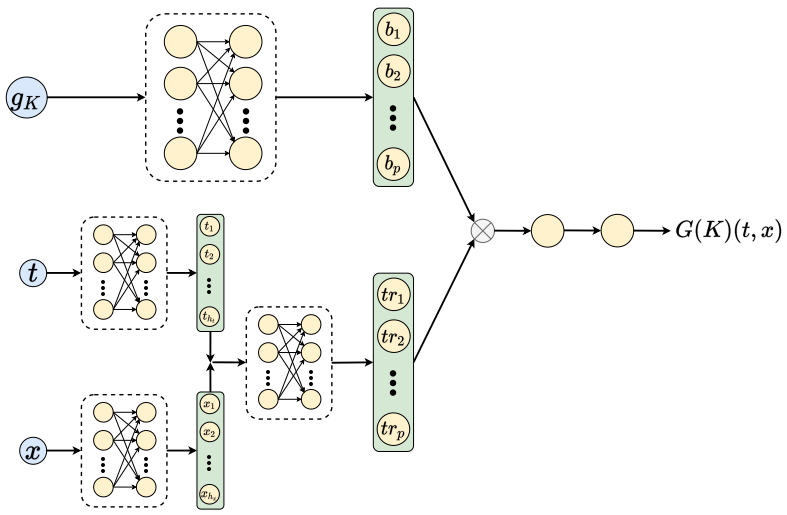


Figure 2.14: Structure of the DeepONet network.

We train this network, as before, in an unsupervised way (no other solver is needed in order to compute a first solution that this network would try to fit). Our training algorithm is very close to the ones described previously, with the addition that, for each state trajectory we generate, a random terminal condition parameter K is also randomly generated uniformly in a compact set. We train the network to solve the pricing PDE (2.3.8) defined previously with a parametric terminal condition corresponding to the payoff of a call option

$$g_K(x) = \max(x - K, 0),$$

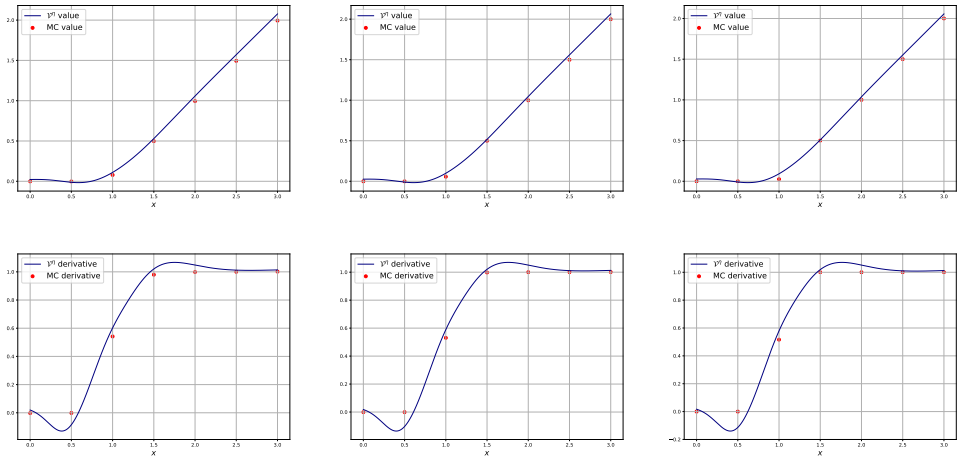
with *strike* K. To test the generalization capacity of this method, we train this network on 8192 random trajectories and strike values, with strikes randomly sampled from $\mathcal{U}([0.25, 0.75] \cup [1.5, 2])$ and test the network on strikes chosen in $[0.2, 2.1]$.

We compute in Table 2.2, the residual losses for the DeepONet ϑ^η trained by the Differential Regression Learning scheme. We compute the residual losses on a 102x102 linearly spaced time and space grid with $t \in [0, 0.9]$ and $x \in [0, 3]$ for a terminal call option payoff for strikes $K \in \{0.2,\ 0.5,\ 1,\ 1.75,\ 2,\ 2.1\}$, with parameter $\sigma = 0.3$ and PDE parameter $\lambda = 5e^{-3}$.

	$K = 0.2$	$K = 0.5$	$K = 1$	$K = 1.75$	$K = 2$	$K = 2.1$
Residual loss + terminal loss	$1.918e^{-3}$	$1.461e^{-4}$	$1.700e^{-3}$	$1.162e^{-3}$	$2.002e^{-3}$	$2.175e^{-3}$

Table 2.2: Residual and boundary losses computed on a 102x102 time and space grid with $t \in [0, 0.9]$ and $x \in [0, 3]$ for a terminal call option payoff $g(x) = \max(x - K, 0)$ for $K \in \{0.2,\ 0.5,\ 1,\ 1.75,\ 2,\ 2.1\}$, with parameter $\sigma = 0.3$ and PDE parameter $\lambda = 5e^{-3}$.

We plot below the value function $\vartheta^\eta(t, x)$ and its derivative $\partial_x \vartheta^\eta(t, x)$ obtained by Differential Regression Learning with DeepONet networks, for fixed values $t = 0$, $t = 0.5$, $t = 0.9$, parameter $\sigma = 0.3$ and PDE parameter $\lambda = 5e^{-3}$, and compare it with the Monte-Carlo estimation obtained. We plot these value functions for strike values $K \in \{0.5,\ 2\}$ inside the training domain and $K \in \{0.2,\ 1,\ 2.1\}$ outside the training domain.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 2.15: Value function ϑ^η (first line) and its derivative (second line) obtained by Differential Regression Learning (Algorithm 9) for a terminal call option payoff with strike $K = 1$, with parameter $\sigma = 0.3$ and PDE ârameter $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.

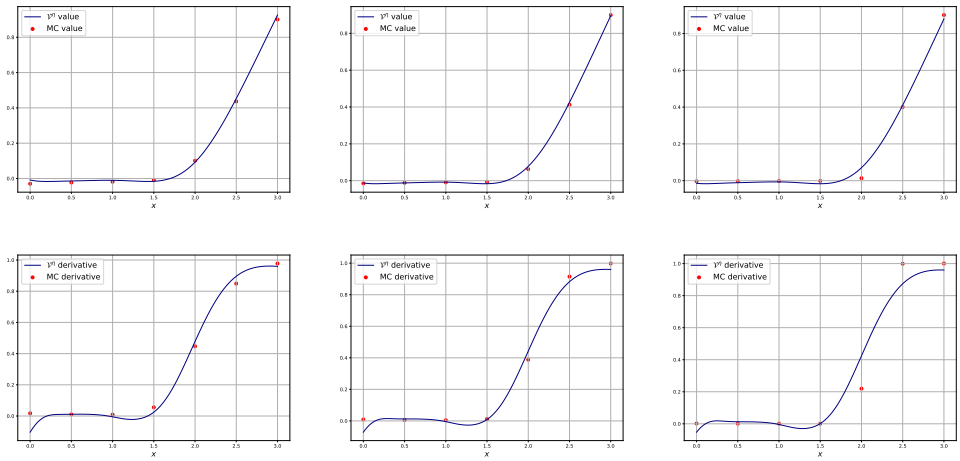


Figure 2.16: Value function ϑ^η (first line) and its derivative (second line) obtained by Differential Regression Learning (Algorithm 9) for a terminal call option payoff with strike $K = 2.1$, with parameter $\sigma = 0.3$ and PDE parameter $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.

Part I

Mean-Variance with tracking error

Chapter 3

Mean-variance portfolio selection with tracking error penalization

Abstract This section studies a variation of the continuous-time mean-variance portfolio selection where a tracking-error penalization is added to the mean-variance criterion. The tracking error term penalizes the distance between the allocation controls and a reference portfolio with same wealth and fixed weights. Such consideration is motivated as follows: (i) On the one hand, it is a way to robustify the mean-variance allocation in case of misspecified parameters, by "fitting" it to a reference portfolio that can be agnostic to market parameters; (ii) On the other hand, it is a procedure to track a benchmark and improve the Sharpe ratio of the resulting portfolio by considering a mean-variance criterion in the objective function. This problem is formulated as a McKean-Vlasov control problem. We provide explicit solutions for the optimal portfolio strategy and asymptotic expansions of the portfolio strategy and efficient frontier for small values of the tracking error parameter. Finally, we compare the Sharpe ratios obtained by the standard mean-variance allocation and the penalized one for four different reference portfolios: equal-weights, minimum-variance, equal risk contributions and shrinking portfolio. This comparison is done on a simulated misspecified model, and on a backtest performed with historical data. Our results show that in most cases, the penalized portfolio outperforms in terms of Sharpe ratio both the standard mean-variance and the reference portfolio.

Keywords: Continuous-time mean-variance problem; tracking error; robustified allocation; parameter misspecification.

3.1 Introduction

The Markowitz mean-variance portfolio selection problem has been initially considered in [Mar52] in a single-period model. In this framework, investment decision rules are made according to the objective of maximizing the expected return of the portfolio for a given financial risk quantified by its variance. The Markowitz portfolio is widely used in the financial industry due to its intuitive formulation and the fact that it produces, by construction, portfolios with high Sharpe ratios (defined as the ratio of the average of portfolio returns over their volatility), which is a key metric used to compare investment strategies.

The mean-variance criterion involves the expected terminal wealth in a nonlinear way due to the presence of the variance term. In a continuous-time dynamic setting, this induces the so-called time inconsistency problem and prevents the direct use of the dynamic programming technique. A first approach, from [ZL00], consists in embedding the mean-variance problem into an auxiliary standard control problem that can be solved by using stochastic linear-quadratic theory. Some more recent approaches rely on the development of stochastic control techniques for control problems of McKean-Vlasov (MKV) type. MKV control problems are problems in which the equation of the state process and the cost function involve the law of this process and/or the law of the control, possibly in a non-linear way. The mean-variance portfolio problem in continuous-time is a McKean-Vlasov control problem of the linear-quadratic type. The state diffusion, which represents the wealth of the portfolio, involves the state process and the control in a linear way while the cost involves the terminal value of the state and the square of its expectation due to the variance criterion. In [AD11], the authors solved the mean-variance problem as a McKean-Vlasov control problem by deriving a version of the Pontryagin maximum principle. More recently, [PW17] have developed a general dynamic programming approach for the control of MKV dynamics and applied it for the resolution of the mean-variance portfolio selection problem. In [FL16], the mean-variance problem is viewed as the MKV limit of a family of controlled many-component weakly interacting systems. These prelimit problems are solved by standard dynamic programming, and the solution to the original problem is obtained by passage to the limit.

A frequent criticism addressed to the mean-variance allocation is its sensitivity to the estimation of expected returns and covariance of the stocks and the risk of a poor out-of-sample performance. Several solutions to these issues have been considered. An approach consists in using a more sophisticated model than the Black-Scholes model, in which the parameters are stochastic or ambiguous and to take decisions under the worst-case scenario over all conceivable models. Robust mean-variance problems have thus been considered in the economic and engineering literature, mostly on single-period or multi- period models; see, e.g., [FHZ10], [Pin16], and [LZ16]. In a continuous-time setting, [IP19] have developed a robust approach by studying the mean-variance allocation with a market model where the model uncertainty affects the covariance matrix of multiple risky assets. In [Guo+22], the authors study the problem of utility maximization under uncertain parameters in a model where the parameters of the model do not evolve freely within a given range, but are constrained via a penalty function. Let us also mention uncertain volatility models in [MPZ15] and [LR14] for robust portfolio optimization with expected utility criterion. Another approach is to rely on the shrinking of the portfolio weights or of the wealth invested in each risky asset in order to obtain a more sparse or more stable portfolio. In [DeM+09], the authors find single-period portfolios that perform well out-of-sample in the presence of estimation error. Their framework deals with the resolution of the traditional minimum-variance problem with the additional constraint that the norm of the portfolio-weight vector must be smaller than a given threshold. In [HSX15], the authors study a one-period mean-variance problem in which the mean-variance objective function is regularized with a weighted elastic net penalty. They show that the use of this penalty can be justified by a robust reformulation of the mean-variance criterion that directly accounts for parameter uncertainty. In the same spirit, in [Che+13], l_p-norm regularized models are used to seek near-optimal sparse portfolios.

In this section, we investigate the mean-variance portfolio selection in continuous time with a tracking error penalization. This penalization represents the distance between the optimized portfolio composition and the composition of a reference portfolio with the same wealth but fixed weights that have been chosen in advance. Typical reference

portfolios widely used in the financial industry are the equal weights, the minimum variance and the equal risk contribution (ERC) portfolios. The equal weights portfolio studied, e.g. in [DL09], is a portfolio where all the wealth of the investor is invested in risky assets and divided equally between the different assets. The minimum variance portfolio is a portfolio where all the wealth is invested in risky assets and portfolio weights are optimized in order to attain the minimal portfolio volatility. The ERC portfolio, presented in [MRT10] and in the monography [Ron13], is totally invested in risky assets and optimized such that the contributions of each asset to the total volatility of the portfolio are equal. The mix of the mean-variance and of this tracking error criterion can be interpreted in two different ways: (i) From a first viewpoint, it is a procedure to regularize and robustify the mean-variance allocation. By choosing reference portfolio weights which are not based on the estimation of market parameters, or which are less sensible to estimation error, the allocation obtained is more robust to parameters estimation error than the standard mean-variance one. (ii) From a second viewpoint, this optimization permits to mimic an allocation corresponding to the reference portfolio weights while improving its Sharpe ratio via the consideration of the mean-variance criterion.

We tackle this problem as a McKean-Vlasov linear-quadratic control problem and adopt the approach developed in [BP19], where the authors give a general method to solve this type of problems by means of a weak martingale optimality principle. We obtain explicit solutions for the optimal portfolio strategy and value function, and provide asymptotic expansions of the portfolio strategy and efficient frontier for small values of the portfolio tracking error penalization parameter. We then compare the Sharpe ratios obtained by the standard mean-variance portfolio, the penalized one and the reference portfolio in two different ways. First, we compare these performances on simulated market data with misspecified market parameters. Different magnitudes of parameter misspecifications are used to illustrate the impact of the parameter estimation error on the performance of the different portfolios. In a second time, we compare the performances of these portfolios on a backtest based on historical market data. In these tests, we shall consider three reference portfolios cited above: the equal weights, the minimum variance and the equal risk contribution (ERC) portfolios. Finally, we will also consider the case where the reference portfolio weights are all equal to zero. This case corresponds to a shrinking of the wealth invested in the different risky assets along the investment horizon.

Outline of this section. The rest of the section is organized as follows. Section 3.2 formulates the mean-variance problem with tracking error. In Section 3.3 we derive explicit solutions for this control problem and provide expansion of this solution for small values of the tracking error penalization parameter. Section 3.4 is devoted to the applications of those results and to the comparison of the mean-variance, penalized and reference portfolio for the different reference portfolios presented above. We show the benefit of the penalized portfolio compared to the standard mean-variance portfolio and the different reference portfolios on simulated and historical data in terms of Sharpe ratio and the lower sensitivity of the penalized portfolio to parameter estimation error.

3.2 Results

Throughout this section, we fix a finite horizon $T \in (0, \infty)$, and a complete probability space $\big(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{F} = \{\mathcal{F}_t\}_{0 \le t \le T}\big)$ on which a standard $\mathbb{F}$-adapted d-dimensional Brownian motion $W = (W^1, ..., W^d)$ is defined. We denote by $L^2_{\mathbb{F}}(0, T; \mathbb{R}^d)$ the set of all $\mathbb{R}^d$-valued, measurable stochastic processes $(f_t)_{t \in [0,T]}$ adapted to $\mathbb{F}$ such that $\mathbb{E}\big[\int_0^T |f_t|^2 dt\big] < \infty$. We consider a financial market with price process $P := (P_t)_{t \in [0,T]}$, composed of one

risk-free asset, assumed to be constant equal to one, i.e., $P^0 \equiv 1$, and d risky assets on a finite investment horizon $[0,T]$. These assets price processes $P_t^i,\ i=1,...,d$ satisfy the following stochastic differential equation:

$$\begin{cases} dP_t^i \ = \ P_t^i \left(b_i\ dt + \sum_{j=1}^n \sigma_{ij} dW_t^j\right), \quad t \in [0,T] \\ P_0^i \ > \ 0 \end{cases}$$

where $b_i > 0$ is the appreciation rate, and $\sigma := (\sigma_{ij})_{i,j=1,...,d} \in \mathbb{R}^{d\times d}$ is the volatility matrix of the d stocks. We denote by $\Sigma := \sigma\sigma^\top$ the covariance matrix. Throughout this section, we will assume that the following nondegeneracy condition holds

$$\Sigma \ \geq \ \delta \mathbb{I}_d,$$

for some $\delta > 0$, where $\mathbb{I}_d$ is the $d \times d$ identity matrix.

Let us consider an investor with total wealth at time $t \geq 0$ denoted by X_t, starting from some initial capital $x_0 > 0$. It is assumed that the trading of shares takes place continuously and transaction cost and consumptions are not considered. We define the set of admissible portfolio strategies $\alpha = (\alpha^1, \ldots, \alpha^d)$ as

$$\mathcal{A} := \left\{ \alpha : \Omega \times [0,T] \to \mathbb{R}^d \text{ s.t } \alpha \text{ is } \mathbb{F}-\text{adapted and } \int_0^T \mathbb{E}[|\alpha_t|^2] dt < \infty \right\},$$

where $\alpha_t^i,\ i=1,...,d$ represents the total market value of the investor's wealth invested in the ith asset at time t. The dynamics of the self-financed wealth process $X = X^\alpha$ associated to a portfolio strategy $\alpha \in \mathcal{A}$ is then driven by

$$dX_t \ = \ \alpha_t^\top b\ dt + \alpha_t^\top \sigma dW_t. \tag{3.2.1}$$

Given a risk aversion parameter $\mu > 0$, and a reference weight $w_r \in \mathbb{R}^d$, the objective of the investor is to minimize over admissible portfolio strategies a mean-variance functional to which is added a running cost:

$$J(\alpha) = \mu \mathrm{Var}(X_T) - \mathbb{E}[X_T] + \mathbb{E}\Big[\int_0^T (\alpha_t - w_r X_t)^\top \Gamma (\alpha_t - w_r X_t)\, dt\Big]. \tag{3.2.2}$$

This running cost represents a running *tracking error* between the portfolio composition α_t of the investor and the reference composition $w_r X_t$ of a portfolio of same wealth X_t and constant weights w_r. The matrix $\Gamma \in \mathbb{R}^{d\times d}$ is symmetric positive definite and is used to introduce an anisotropy in the portfolio composition penalization. The penalization $\int_0^T (\alpha_t - w_r X_t)^\top \Gamma (\alpha_t - w_r X_t)$, which we will call "tracking error penalization", is introduced in order to ensure that the portfolio of the investor does not move away too much from this reference portfolio with respect to the distance $|M| := M^\top \Gamma M,\ M \in \mathbb{R}^d$.

The mean-variance portfolio selection with tracking error is then formulated as

$$V_0 := \inf_{\alpha \in \mathcal{A}} J(\alpha), \tag{3.2.3}$$

and an optimal allocation given the cost $J(\alpha)$ will be given by

$$\alpha_t^* \ \in \ \arg\min_{\alpha \in \mathcal{A}} J(\alpha).$$

We complete this section by recalling the solution to the mean-variance problem when there is no tracking error running cost, and which will serve later as benchmark for comparison when studying the effect of the tracking error with several reference portfolios.

Remark 3.2.1 (Case of no tracking error). *When $\Gamma = 0$, it is known, see e.g. [ZL00] that the optimal mean-variance strategy is given by*

$$\alpha_t^* = \Sigma^{-1} b \left[\frac{1}{2\mu} e^{b^\top \Sigma^{-1} b \; T} + x_0 - X_t^* \right], \quad 0 \le t \le T, \tag{3.2.4}$$

where X_t^* is the wealth process associated to α^*. The vector $\Sigma^{-1} b$, which depends only on the model parameters of the risky assets, determines the allocation in the risky assets.

In the sequel, we study the quantitative impact of the tracking error running cost on the optimal mean-variance strategy.

3.3 Solution allocation with tracking error

Our main theoretical result provides an analytic characterization of the optimal control to the mean-variance problem with tracking error.

Theorem 3.3.1. *There exist a unique pair $(K, \Lambda) \in C\left([0,T], \mathbb{R}_+^*\right) \times C\left([0,T], \mathbb{R}_+\right)$ solution to the system of ODEs*

$$\begin{cases} dK_t = \left\{ \left(K_t b - \Gamma w_r\right)^\top S_t^{-1} \left(K_t b - \Gamma w_r\right) - w_r^\top \Gamma w_r \right\} dt, \quad K_T = \mu \\ d\Lambda_t = \left\{ \left(\Lambda_t b - \Gamma w_r\right)^\top S_t^{-1} \left(\Lambda_t b - \Gamma w_r\right) - w_r^\top \Gamma w_r \right\} dt, \quad \Lambda_T = 0 \end{cases} \tag{3.3.1}$$

where $S_t := K_t \Sigma + \Gamma$. The optimal control for problem (3.2.3) is then given by

$$\alpha_t^\Gamma = S_t^{-1} \Gamma w_r X_t - S_t^{-1} b \big[K_t X_t + Y_t - (K_t - \Lambda_t) \mathbb{E}[X_t] \big], \tag{3.3.2}$$

with

$$Y_t = -\frac{1}{2} e^{-\int_t^T b^\top S_s^{-1} (\Lambda_s b - \Gamma w_r) ds}$$
$$R_t = \frac{1}{2} \int_t^T b^\top S_s^{-1} b \; e^{-2 \int_s^T b^\top S_u^{-1} (\Lambda_u b - \Gamma w_r) du} \; ds,$$

and $X = X^{\alpha^\Gamma}$ is the wealth process associated to α^Γ. Moreover, we have

$$V_0 \;=\; J(\alpha^\Gamma) \;=\; \Lambda_0 X_0^2 + 2 Y_0 X_0 + R_0.$$

Proof. Given the existence of a pair $(K, \Lambda) \in C\left([0,T], \mathbb{R}_+^*\right) \times C\left([0,T], \mathbb{R}_+\right)$ solution to (3.3.1), the optimality of the control process in (3.3.2) follows by the weak version of the martingale optimality principle as developed in [BP19]. The arguments are recalled in appendix 3.A.

Here, let us verify the existence and uniqueness of a solution to the system (3.3.1).

1. We first consider the equation for K, which is a scalar Riccati equation. The equation for K is associated to the standard linear-quadratic stochastic control problem:

$$\tilde{v}(t,x) := \inf_{\alpha \in \mathcal{A}} \mathbb{E} \left[\int_t^T \left(w_r^\top \Gamma w_r (\tilde{X}_s^{t,x,\alpha})^2 - 2 \alpha_s^\top \Gamma w_r \tilde{X}_s^{t,x,\alpha} + \alpha_s^\top \Gamma \alpha_s \right) ds \right]$$

where $\tilde{X}_s^{t,x,\alpha}$ is the controlled linear dynamics solution to

$$d\tilde{X}_s = \alpha_s^\top b \; ds + \alpha_s^\top \sigma dW_s, \quad t \le s \le T, \; \tilde{X}_t = x.$$

By a standard result in control theory [YZ99, Ch. 6, Thm. 6.1, 7.1, 7.2], there exists a unique solution $K \in C([0,T],\mathbb{R}_+)$ to the first equation of system (3.3.1) (more, $K \in C([0,T],\mathbb{R}_+^*)$ if w_r is nonzero). In this case, we have $\tilde{v}(t,x) = x^\top K_t x$.

2. Given K, we consider the equation for Λ. This is also a scalar Riccati equation. By the same arguments as for the K equation, there exists a unique solution $\Lambda \in C([0,T],\mathbb{R}_+)$ to the second equation of (3.3.1), provided that

$$\Lambda_T \geq 0, \qquad w_r^\top \Gamma w_r - w_r^\top \Gamma \left(K_t\Sigma + \Gamma\right)^{-1} \Gamma w_r \geq 0, \qquad K_t\Sigma + \Gamma \geq \delta \mathbb{I}_d, \qquad 0 \leq t \leq T$$

for some $\delta > 0$. We already have that $\Lambda_T = 0$. From the fact that $K > 0$, together with the nondegeneracy condition on the matrix Σ, we have that $K_t\Sigma + \Gamma \geq \Gamma \geq \delta \mathbb{I}_d$. Since $\Gamma > 0$, and under the nondegeneracy condition of matrix Σ, we can use the Woodbury matrix identity to obtain

$$\left(K_t\Sigma + \Gamma\right)^{-1} = \Gamma^{-1} - \Gamma^{-1}\left(\Gamma^{-1} + \frac{\Sigma^{-1}}{K_t}\right)^{-1}\Gamma^{-1}.$$

We then get

$$w_r^\top \Gamma w_r - w_r^\top \Gamma \left(K_t\Sigma + \Gamma\right)^{-1} \Gamma w_r = w_r^\top \left(\Gamma^{-1} + \frac{\Sigma^{-1}}{K_t}\right)^{-1} w_r \geq 0.$$

3. Given (K,Λ), the equation for Y is a linear ODE, whose unique continuous solution is explicitly given by

$$Y_t = -\frac{1}{2} e^{-\int_t^T b^\top S_s^{-1}(\Lambda_s b - \Gamma w_r) ds}.$$

4. Given (K,Λ,Y), R can be directly integrated into

$$R_t = \frac{1}{2}\int_t^T b^\top S_s^{-1} b \; e^{-2\int_s^T b^\top S_u^{-1}(\Lambda_u b - \Gamma w_r) du} \; ds.$$

□

We can see from the expression of the optimal control (3.3.2) that the allocation in the risky assets has two components. One component is determined by the vector $S_t^{-1}\Gamma w_r = \left(K_t\Sigma + \Gamma\right)^{-1}\Gamma w_r$ with leverage X_t, and the second one by the vector $S_t^{-1} b = \left(K_t\Sigma + \Gamma\right)^{-1} b$ with leverage $[K_t X_t + Y_t - (K_t - \Lambda_t)\mathbb{E}[X_t]]$. Computing the average wealth $\overline{X} = \mathbb{E}[X]$ associated to α^Γ, we can express the control α^Γ as a function of the initial wealth of the investor x_0 and the current wealth X_t

$$\begin{aligned} \alpha_t^\Gamma = {} & S_t^{-1}\Gamma w_r X_t - \Lambda_t S_t^{-1} b \left(X_0 C_{0,t} + \frac{1}{2} H_t\right) \\ & + S_t^{-1} b \left[K_t \left(X_0 C_{0,t} + \frac{1}{2} H_t - X_t\right) - Y_t\right] \end{aligned} \tag{3.3.3}$$

where we set $C_{s,t} := e^{-\int_s^t b^\top S_u^{-1}(\Lambda_u b - \Gamma w_r) du}$ and $H_t := C_{t,T} \int_0^t C_{s,t}^2 \, b^\top S_s^{-1} b \, ds$.

Remark 3.3.1. *In the case when Γ is the null matrix, $\Gamma = \mathbf{0}$, we see that the first component of the optimal control (3.3.3) vanishes,*

$$Y_t = -\frac{1}{2}, \quad R_t = \frac{1}{4\mu}\left(1 - e^{b^\top \Sigma^{-1} b\ (T-t)}\right),$$

and the system of ODES (3.3.1) of (K, Λ) becomes

$$\begin{cases} dK_t = K_t b^\top \Sigma^{-1} b\ dt, & K_T = \mu \\ \\ d\Lambda_t = \frac{\Lambda_t^2}{K_t} b^\top \Sigma^{-1} b\ dt, & \Lambda_T = 0, \end{cases}$$

which yields the explicit forms

$$K_t = \mu e^{-b^\top \Sigma^{-1} b\ (T-t)}, \quad \Lambda_t = 0.$$

We get $S_t^{-1} = \frac{\Sigma^{-1}}{K_t} = \frac{\Sigma^{-1} e^{b^\top \Sigma^{-1} b\ (T-t)}}{\mu}$, $C_{\cdot,\cdot} = 1$ and $H_t = \frac{1}{\mu}\int_0^t b^\top \Sigma^{-1} b\ e^{b^\top \Sigma^{-1} b\ (T-s)} ds$. The first line of the optimal control α^Γ equation vanishes and the second line can be rewritten as

$$\alpha_t^\Gamma = \Sigma^{-1} b \left[\frac{1}{2\mu}\left(e^{b^\top \Sigma^{-1} b\ (T-t)} + \int_0^t b^\top \Sigma^{-1} b\ e^{b^\top \Sigma^{-1} b\ (T-s)} ds\right) + X_0 - X_t\right].$$

Computing the integral in this expression, we recover the optimal control of the classical mean-variance problem (3.2.4).

Remark 3.3.2 (Limit of α_t^γ for $\Gamma = \gamma \mathbb{I}_d \to \infty$). *If we consider Γ in the form $\Gamma = \gamma \mathbb{I}_d$, the optimal control can be rewritten as*

$$\alpha_t^\gamma = \left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma\right)^{-1} w_r X_t - \frac{1}{\gamma}\left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma\right)^{-1} b\left[K_t X_t + Y_t - (K_t - \Lambda_t)\overline{X}_t\right]. \quad (3.3.4)$$

We show in appendix 3.D that K_t and Λ_t are bounded functions of the penalization parameter γ, thus $\frac{K_t}{\gamma}, \frac{\Lambda_t}{\gamma} \underset{\gamma\to\infty}{\longrightarrow} 0$.
We rewrite Y_t as

$$Y_t = -\frac{1}{2} e^{b^\top w_r (T-t)} e^{-\int_t^T \frac{1}{\gamma} b^\top \left(\mathbb{I}_d + \frac{K_s}{\gamma}\Sigma\right)^{-1} (\Lambda_s b + K_s \Sigma w_r) ds}$$

and we get that $Y_t \underset{\gamma\to\infty}{\longrightarrow} -\frac{1}{2} e^{b^\top w_r (T-t)}$. Thus the second term of (3.3.4) vanishes and we get

$$\alpha_t^\gamma \underset{\gamma\to\infty}{\longrightarrow} w_r X_t$$

which corresponds to the reference portfolio.

Remark 3.3.3 (Expansion for $\Gamma = \gamma \mathbb{I}_d \to 0$). *We take $\Gamma = \gamma \mathbb{I}_d$. Since the covariance matrix Σ is symmetric, there exists an invertible matrix $Q \in \mathbb{R}^{d\times d}$ and a diagonal matrix $D \in \mathbb{R}^{d\times d}$ such that $\Sigma = Q \cdot D \cdot Q^{-1}$. We can then rewrite the matrix $S_t^{-1} := (K_t \Sigma + \gamma \mathbb{I}_d)^{-1}$ as*

$$S_t^{-1} = Q \cdot (K_t D + \gamma \mathbb{I}_d)^{-1} Q^{-1}$$

with

$$\left((K_t D + \gamma \mathbb{I}_d)^{-1}\right)_{ij} = \begin{cases} \frac{1}{K_t d_i + \gamma} & if\ i = j \\ 0 & if\ i \neq j \end{cases}$$

where d_i is the i-th diagonal value of the diagonal matrix D. From the nondegeneracy condition of the covariance matrix, we have $d_i > 0$, $\forall i \in [\![1, n]\!]$. As $\gamma \longrightarrow 0$, we want to write the Taylor expansion of the diagonal elements of the inverse matrix $(D + \gamma \mathbb{I}_d)^{-1}$ equal to $\frac{1}{K_t d_i}\left(1 + \frac{\gamma}{K_t d_i}\right)^{-1}$. We have that $K_t \underset{\gamma \to 0}{\longrightarrow} \mu e^{-\rho(T-t)}$, thus $\frac{\gamma}{K_t} \underset{\gamma \to 0}{\longrightarrow} 0$. We can then write the Taylor expansion of the matrix S_t^{-1} as

$$S_t^{-1} = \frac{\Sigma^{-1}}{K_t} - \gamma \frac{\left(\Sigma^{-1}\right)^2}{K_t^2} + O(\gamma^2)$$

keeping only the terms up to the linear term in γ.
Putting this expression in the differential equation of K, and keeping only the terms up to the linear term in γ, we get the differential equation

$$\frac{dK_t}{dt} = K_t \rho - \gamma \|w_r + \Sigma^{-1} b\|^2 + O(\gamma^2), \tag{3.3.5}$$

where we set $\rho := b^\top \Sigma^{-1} b$. We look for a solution to this equation of the form

$$K_t^\gamma = K_t^0 + \gamma K_t^1 + O(\gamma^2).$$

Putting this expression in the differential equation (3.3.5), we get two differential equations, for the leading order and the linear order in γ respectively

$$\begin{cases} \frac{dK_t^0}{dt} = K_t^0 \rho, & K_T^0 = \mu \\ \frac{dK_t^1}{dt} = K_t^1 \rho - \|w_r + \Sigma^{-1} b\|^2, & K_T^1 = 0 \end{cases}$$

which yield the explicit solution

$$K_t^\gamma = K_t^0 + \gamma \|w_r + \Sigma^{-1} b\|^2 \, \frac{1 - e^{-\rho(T-t)}}{\rho} + O(\gamma^2)$$

where $K_t^0 = \mu e^{-\rho(T-t)}$ is the solution to the differential equation in the unpenalized case. From the expansion for K, we can write the expansion of the differential equation for Λ up to the linear term in γ. We use the expansion

$$\frac{1}{K_t^\gamma} = \frac{1}{K_t^0}\left(1 - \gamma \|w_r + \Sigma^{-1} b\|^2 \, \frac{1 - e^{-\rho(T-t)}}{K_t^0 \rho}\right) + O(\gamma^2)$$

and we get the following expansion of the differential equation of Λ

$$\begin{aligned} \frac{d\Lambda_t}{dt} =& \frac{\Lambda_t^2}{K_t^0} \rho \left(1 - \gamma \|w_r + \Sigma^{-1} b\|^2 \, \frac{1 - e^{-\rho(T-t)}}{K_t^0 \rho}\right) \\ & - \gamma \left(2 \frac{\Lambda_t}{K_t^0} b^\top \Sigma^{-1} w_r - \left(\frac{\Lambda_t}{K_t^0}\right)^2 b^\top \Sigma^{-2} b - \|w_r\|^2\right) + O(\gamma^2). \end{aligned} \tag{3.3.6}$$

As before, we look for a solution of this differential equation of the form

$$\Lambda_t^\gamma = \Lambda_t^0 + \gamma \Lambda_t^1 + O(\gamma^2).$$

Plugging this expression into the equation (3.3.6), we get the two following differential equations

$$\begin{cases} \frac{d\Lambda_t^0}{dt} = \frac{\left(\Lambda_t^0\right)^2}{K_t^0} \rho, & \Lambda_T^0 = 0 \\ \\ \frac{d\Lambda_t^1}{dt} = 2 \frac{\Lambda_t^0 \Lambda_t^1}{K_t^0} \rho - \left(\frac{\Lambda_t^0}{K_t^0}\right)^2 \rho \|w_r + \Sigma^{-1} b\|^2 \, \frac{1 - e^{-\rho(T-t)}}{\rho} & \\ - \left(2 \frac{\Lambda_t^0}{K_t^0} b^\top \Sigma^{-1} w_r + \left(\frac{\Lambda_t^0}{K_t^0}\right)^2 b^\top \Sigma^{-2} b + \|w_r\|^2\right), & \Lambda_T^1 = 0. \end{cases}$$

The first differential equation yields the solution $\Lambda_t^0 = 0,\ \forall t \in [0,T]$. *Replacing* Λ_t^0 *by this value in the second differential equation, we get the equation*

$$\frac{d\Lambda_t^1}{dt} = -\|w_r\|^2$$

and obtain the solution

$$\Lambda_t^\gamma = \gamma\|w_r\|^2(T-t) + O(\gamma^2).$$

We can also compute the first order expansion of $C_{\cdot,\cdot}$

$$\begin{aligned} C_{s,t}^\gamma =& 1 - \gamma \int_s^t \frac{\rho}{K_u^0}\left(\|w_r\|^2(T-u) - \frac{b^\top\Sigma^{-1}w_r}{\rho}\right) du + O(\gamma^2) \\ =& 1 - \gamma C_{s,t}^1 + O(\gamma^2) \end{aligned}$$

where we set

$$C_{s,t}^1 := \frac{e^{\rho(T-t)}}{\mu\rho}\left\{\rho\|w_r\|^2(t-s) + \left(e^{\rho(t-s)} - 1\right)\left(\|w_r\|^2(\rho T - 1) - b^\top\Sigma^{-1}w_r\right)\right\},$$

and we have

$$Y_t^\gamma = -\frac{1}{2} + \frac{\gamma}{2}C_{t,T}^1.$$

The last expansion we need to compute before rewritting the optimal control is the expansion of H_t. *We can rewrite*

$$H_t = \frac{e^{\rho T}}{\mu}\left(1 - e^{-\rho t}\right) - \gamma H_t^1 + O(\gamma^2)$$

with

$$H_t^1 := \int_0^t \left(2C_{s,t}^1 + C_{t,T}^1\right)\frac{b^\top\Sigma^{-1}b}{K_s^0}ds + \int_0^t b^\top \frac{\Sigma^{-1}}{(K_s^0)^2}\left(K_s^1\mathbb{I}_d + \Sigma^{-1}\right) b\, ds.$$

As shown in appendix 3.B, we can rewrite the optimal control

$$\alpha_t^\gamma = \Sigma^{-1}b\ \alpha_t^0 + \gamma\left(\Sigma^{-1}w_r\ \alpha_t^{1,3} - \Sigma^{-2}b\ \alpha_t^{1,2} - \Sigma^{-1}b\ \alpha_t^{1,1}\right) + O(\gamma^2) \tag{3.3.7}$$

where we set $\Sigma^{-2} := (\Sigma^{-1})^2$, *and with*

$$\begin{cases} \alpha_t^0 = \frac{1}{2\mu}e^{\rho T} + X_0 - X_t \\ \alpha_t^{1,1} = \frac{\|w_r\|^2}{K_t^0}(T-t)\left(X_0 + \frac{e^{\rho T}}{\mu}\left(1 - e^{-\rho t}\right)\right) + X_0 C_{0,t}^1 + \frac{H_t^1}{2} + \frac{K_t^1}{2(K_t^0)^2} + \frac{C_{t,T}}{2K_t^0} \\ \alpha_t^{1,2} = \frac{e^{\rho T}}{2K_t^0\mu}\left(1 - e^{-\rho t}\right) + \frac{1}{2(K_t^0)^2} \\ \alpha_t^{1,3} = \frac{X_t}{K_t^0}. \end{cases} \tag{3.3.8}$$

We see that for $\gamma = 0$, *we recover the classical mean-variance optimal control. For non-zero values of* γ, *we see that a mix of three different portfolio allocations is obtained. The weight of the allocation* $\Sigma^{-1}b$ *is modified and two allocations* $\Sigma^{-2}b$ *and* $\Sigma^{-1}w_r$ *appear with weights* $\gamma\alpha_t^{1,2}$ *and* $\gamma\alpha_t^{1,3}$.

From this expansion of the control α^γ, *we can compute the first order asymptotic expansion in* γ *of the equation giving the relation between the variance of the terminal wealth of the portfolio and its expectation. In the classical mean-variance case, this*

equation is called the efficient frontier formula. As shown in appendix 3.C, with the tracking error penalization, the first order asymptotic expansion in γ gives

$$\begin{aligned} Var(X_T) =& \frac{e^{-\rho T}}{1-e^{-\rho T}}\left(\overline{X_T}^0 - X_0\right)^2 \\ &+ \gamma\left\{\frac{b^\top\Sigma^{-1}w_r}{\mu^2}\left[X_0 T - \frac{1}{2\mu}e^{\rho T}\left(T - \frac{1-e^{-\rho T}}{\rho}\right)\right]\right. \\ &\left. - \int_0^T \left(\frac{\rho}{\mu}\alpha_s^{1,1} + \frac{b^\top\Sigma^{-2}b}{\mu}\alpha_s^{1,2}\right)e^{-\rho(T-s)}ds\right\} + O(\gamma^2). \end{aligned}$$

The leading order term corresponds to the efficient frontier equation of the classical mean-variance allocation computed in [ZL00], and thus for $\gamma = 0$, we recover this classical result. The linear term in γ contains contributions of the three perturbative allocations. A modification of "leverage" of the original mean-variance allocation $\Sigma^{-1}b$ and two different allocations $\Sigma^{-2}b$ and $\Sigma^{-1}w_r$.

3.4 Applications and numerical results

In this section, we apply the results of the previous section and study the allocation obtained by considering four different static portfolios as reference. First, we shall study these allocations on simulated data, in the case of misspecified parameters. The misspecification of parameters means that the market parameters used to compute the portfolio allocations are different from the ones driving the stocks prices. This study allows us to estimate the impact of the estimation error on the portfolio performance. In a second time, we perform a backtest and run the different portfolios on real market data. To simplify the presentation, we will assume now that the tracking error penalization matrix is in the form $\Gamma = \gamma\mathbb{I}_d$ with $\gamma \in \mathbb{R}_+^*$. With this simplification, we have $S_t^{-1} = (K_t\Sigma + \gamma\mathbb{I}_d)^{-1}$ and we can rewrite the system of ODEs (3.3.1) and the optimal control (3.3.2) as

$$\begin{cases} dK_t = \left\{(K_t b - \gamma w_r)^\top S_t^{-1}(K_t b - \gamma w_r) - \gamma (w_r)^\top w_r\right\}dt, & K_T = \mu \\ d\Lambda_t = \left\{(\Lambda_t b - \gamma w_r)^\top S_t^{-1}(\Lambda_t b - \gamma w_r) - \gamma (w_r)^\top w_r\right\}dt, & \Lambda_T = 0 \end{cases}$$

and

$$\begin{aligned} \alpha_t^\gamma =& \gamma S_t^{-1}w_r X_t - \Lambda_t S_t^{-1}b\left(X_0 C_{0,t} + \frac{1}{2}H_t\right) \\ &+ S_t^{-1}b\left[K_t\left(X_0 C_{0,t} + \frac{1}{2}H_t - X_t\right) - Y_t\right] \end{aligned}$$

where

$$S_t = K_t\Sigma + \gamma\mathbb{I}_d, \quad C_{s,t} := e^{-\int_s^t b^\top S_u^{-1}(\Lambda_u b - \gamma w_r)du}, \quad Y_t = -\frac{1}{2}C_{t,T}.$$

We will consider three different classical allocations as reference portfolio.

1. **Equal-weights portfolio**: in this classical equal-weights portfolio, the same capital is invested in each asset, thus

$$w_r^{\text{ew}} = \frac{1}{d}\,e$$

where d is the number of risky assets considered and $e \in \mathbb{R}^d$ is the vector of ones.

2. **Minimum variance portfolio**: the minimum variance portfolio is the portfolio which achieves the lowest variance while investing all its wealth in the risky assets. The weight vector of this portfolio is equal to

$$w_r^{\text{min-var}} = \frac{\Sigma^{-1}e}{e^\top \Sigma^{-1} e}.$$

These weights correspond to the one-period Markowitz portfolio when every asset expected return b_i is taken equal to 1. In that case, only the portfolio variance is relevant and is minimized during the optimization process.

3. **ERC portfolio**: the equal risk contributions (ERC) portfolio, presented in [MRT10] and in the monograph [Ron13] is constructed by choosing a risk measure and computing the risk contribution of each asset to the global risk of the portfolio. When the portfolio volatility is chosen as the risk measure, the principle of the ERC portfolio lays in the fact that the volatility function satisfies the hypothesis of Euler's theorem and can be reduced to the sum of its arguments multiplied by their first partial derivatives. The portfolio volatility $\sigma(w) = \sqrt{w^\top \Sigma w}$ of a portfolio with weights vector $w \in \mathbb{R}^d$ can then be rewritten as

$$\sigma(w) = \sum_{i=1}^{d} w^i \partial_i \sigma(w) = \sum_{i=1}^{d} \frac{w^i \left(\Sigma w\right)^i}{\sigma(w)}.$$

The term under the sum $\frac{w^i (\Sigma w)^i}{\sigma(w)}$, corresponding to the i-th asset, can be interpreted as the contribution of this risky asset to the total portfolio volatility. The equal risk contribution allocation is then defined as the allocation in which these contributions are equal for all the risky assets of the portfolio, $\frac{w^i (\Sigma w)^i}{\sigma(w)} = \frac{w^j (\Sigma w)^j}{\sigma(w)}$ for every $i, j \in [\![1, d]\!]$. The equal risk contribution allocation is thus obtained when the portfolio weights w^* are given by

$$w^* = \left\{ w \in [0,1]^d : \sum_{i=1}^{d} w^i = 1,\ w^i \left(\Sigma w\right)^i = w^j \left(\Sigma w\right)^j,\ \forall i, j \in [\![1, d]\!] \right\}.$$

With this risk measure, the ERC portfolio weights can be expressed in a closed-form only in the case where the correlations between every couple of stocks are equal, that is $\text{corr}(P_i, P_j) = c,\ \forall\ i, j \in [\![1, d]\!]$, with the additional assumption that $c \geq -\frac{1}{d-1}$. Under these assumptions, and with the constaint that $\sum_{i=1}^{d} \left(w_r^{\text{erc}}\right)_i = 1$, the weights of this portfolio are equal to

$$\left(w_r^{\text{erc}}\right)_i = \frac{\sigma_i^{-1}}{\sum_{j=1}^{d} \sigma_j^{-1}}$$

where σ_i is the volatility of the i-th asset.

In the general case, the weights of the ERC portfolio do not have a closed form and must be computed numerically by solving the following optimization problem

$$w_r^{\text{erc}} = \underset{w \in \mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{d} \sum_{j=1}^{d} \left(w^i \left(\Sigma w\right)^i - w^j \left(\Sigma w\right)^j \right)^2$$
$$\text{s.t } e^\top w = 1 \text{ and } 0 \leq w^i \leq 1,\ \forall i \in [\![1, d]\!].$$

4. **Control shrinking (zero portfolio)**: this is the portfolio where all weights are equal to zero, $w_r^i = 0$ for all i. This case corresponds to a shrinking of the controls of the penalized allocation, in the same spirit as the shrinking of regression coefficients in the Ridge regression (or Tikhonov regularization).

3.4.1 Performance comparison with Monte Carlo simulations

In this section we compare, for each reference portfolio, the classical dynamic mean-variance allocation, the reference portfolio and the "tracking error" penalized portfolio. In a real investment situation, expected return and covariance estimates are noisy and biased. Thus, in order to compare the three portfolios and observe the impact of adding a tracking error penalization in the mean-variance allocation, we will run Monte Carlo simulations, assuming that the real-world expected returns b_{real} and covariances σ_{real} are equal to reference expected returns b_0 and covariances σ_0 plus some noise:

$$b_0 = \begin{pmatrix} 0.12 \\ 0.14 \\ 0.16 \\ 0.10 \end{pmatrix}, \quad v_0 = \begin{pmatrix} 0.20 \\ 0.30 \\ 0.40 \\ 0.50 \end{pmatrix}, \quad C_0 = \begin{pmatrix} 1. & 0.05 & -0.05 & 0.10 \\ 0.05 & 1. & -0.03 & 0.12 \\ -0.05 & -0.03 & 1. & -0.13 \\ 0.10 & 0.12 & -0.13 & 1. \end{pmatrix},$$

with the volatilties v_0 and correlations C_0 and

$$b_{\text{real}} = b_0 + \epsilon \times \text{noise}, \quad \sigma_{\text{real}} = \sigma_0 + \epsilon \times \text{noise}$$

where the covariance matrix σ_0 is obtained from v_0 and C_0. The noise follows a standard normal distribution $\mathcal{N}(0,1)$ and ϵ is its magnitude. We use Monte Carlo simulations to estimate the expected Sharpe ratio of each portfolio, equal to the average of the portfolio daily returns R divided by the standard deviation of those returns: $\mathbb{E}\big[\frac{\mathbb{E}[R]}{\text{Stdev}(R)}\big]$.

We consider an investment horizon of one year, with 252 business days and a daily rebalancing of the portfolio. The risk aversion parameter μ is chosen so that the targeted annual return of the classical mean-variance allocation is equal to 20%, thus $\mu = \frac{e^{b^\top \Sigma^{-1} b}}{2x_0 * 1.20}$ according to [ZL00]. The initial wealth of the investor x_0 is chosen equal to 1 and we choose the penalization parameter $\gamma = \mu/100$. Indeed, as the value of μ depends on the value of the stocks expected return and covariance matrix and on the targeted return, and can be very big, we express γ a function of this μ in order for the penalization to be relevant and non-negligible.

For each reference portfolio, we compare the reference portfolio, the classical mean-variance allocation and the penalized one for values of noise amplitude ϵ ranging from 0 to 1. For each value of ϵ, we run 2000 scenarios and we plot the graphs of the average Sharpe ratio as a function of ϵ.

On the following graphs, we can see that in the four cases, the mean-variance and the penalized portfolios are superior to the reference. In the case where the equal weights portfolio is chosen as reference, the penalized portfolio's Sharpe ratio is lower than the mean-variance one for small values of ϵ. For ϵ greater than approximately 0.25, the penalized portfolio's Sharpe ratio becomes larger and the gap with the mean-variance's Sharpe tends to increase with ϵ. The same phenomenon occurs in the case where the ERC portfolio is chosen as reference, with a smaller gap between the mean-variance and penalized portfolios' Sharpe ratios. When the minimum variance portfolio is chosen as reference, the penalized portfolio's Sharpe ratio is lower than the one of the mean-variance portfolio for all ϵ in the interval $[0,1]$. This is certainly due to the sensitivity of the minimum variance portfolio to the estimator of the covariance matrix. Finally, in the case of the control shrinking, the Sharpe ratio of the penalized portfolio is significantly higher that the Sharpe ratio of the mean-variance portfolio, for every value of the noise amplitude ϵ in the interval $[0,1]$.

- Equal-weights reference portfolio

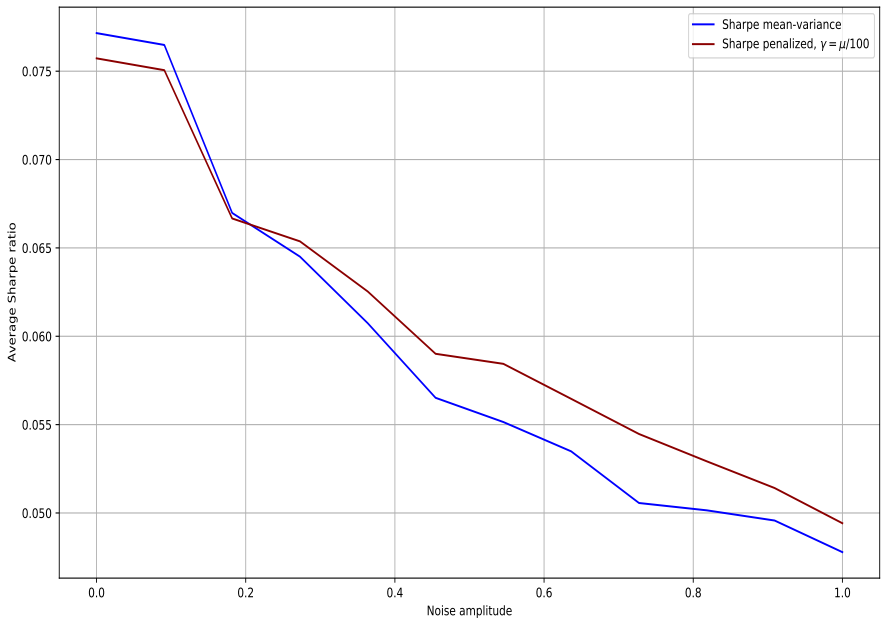


Figure 3.1: The highest average Sharpe ratio attained by the equal-weight portfolio is equal to 0.047 for $\epsilon = 0$.

- Minimum-variance reference portfolio

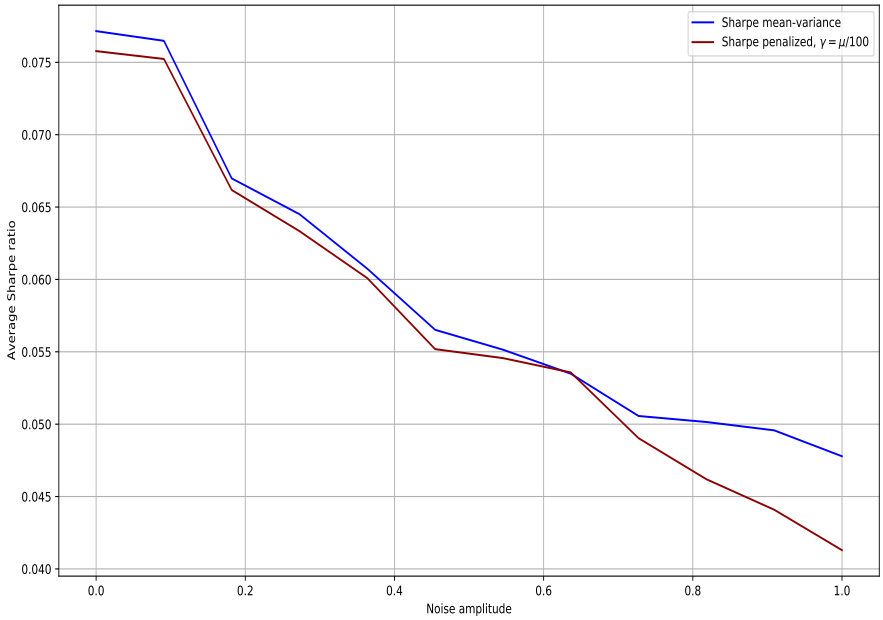


Figure 3.2: The highest average Sharpe ratio attained by the minimum-variance portfolio is equal to 0.057 for $\epsilon = 0$.

- ERC reference portfolio

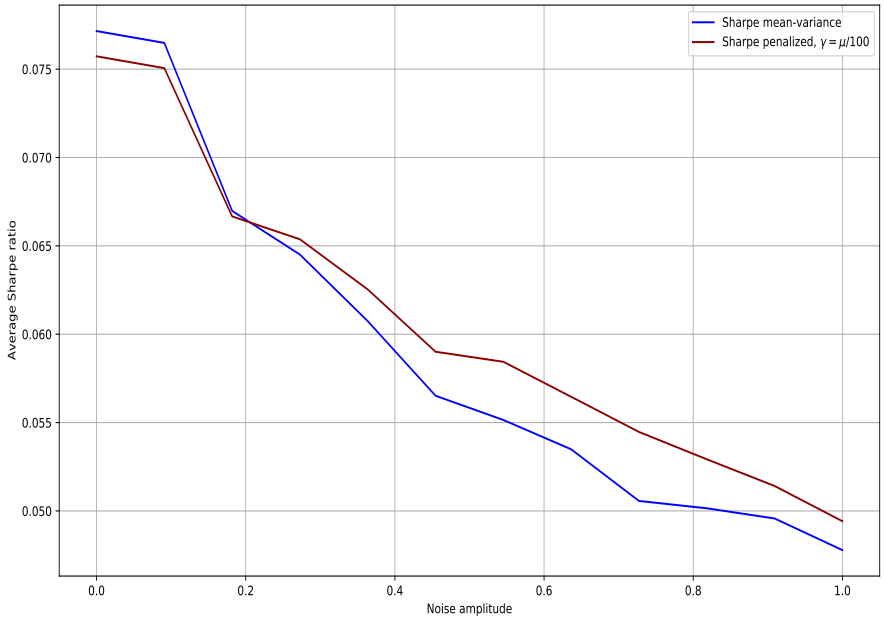


Figure 3.3: The highest average Sharpe ratio attained by the ERC portfolio is equal to 0.051 for $\epsilon = 0$.

- Control shrinking (zero reference)

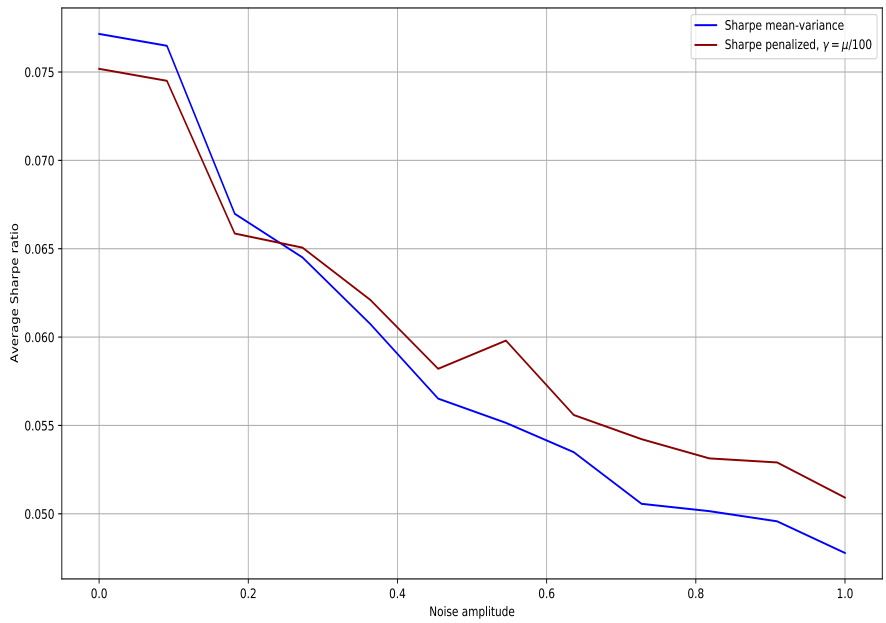


Figure 3.4: In this case the reference weights are equal to zero, and no Sharpe ratio is computed for the reference portfolio.

3.4.2 Performance comparison on a backtest

We now compare the different allocations on a backtest based on adjusted close daily prices available on Quandl between 2013-09-03 and 2017-12-28 for four stocks: Apple, Microsoft, Boeing and Nike. Here we chose a value of μ which corresponds to an annual expected return of 25%. In our example, we express again γ as a function of μ and we consider two different values, $\gamma = \mu$ and $\gamma = \mu/100$.

Figures 3.5, 3.6 and 3.7 show the total wealth of the four different portfolios, mean-variance, reference and the penalized portfolio with the big and the small penalization as a function of time. On these graphs we observe that, at the beginning of the investment horizon, the mean-variance allocation has the largest wealth increase, hence the largest leverage. As the wealth of this portfolio attains the target wealth, expressed as $\frac{1}{2\mu}e^{b^\top \Sigma^{-1} b\ T} + x_0$ in the mean-variance control equation (3.2.4), its leverage decreases and its wealth curve flattens. The same phenomenon occurs for the penalized allocation with large penalization parameter $\gamma = \mu$. In this case, the high value of the penalization parameter keeps the penalized portfolio controls close to the ones of the mean-variance portfolio. On the contrary, the reference portfolios have constant weights and no target wealth. We can see that in each case the reference portfolio's wealth keeps increasing over the entire horizon. The wealth of the penalized portfolio with penalization parameter $\gamma = \mu/100$ follows the wealth of these reference portfolio due to the small value of the tracking error penalization.

For these three reference portfolios, we observe that the penalized portfolio with penalization parameter $\gamma = \mu$ outperforms both the mean-variance and the reference portfolios in terms of Sharpe ratio whereas the penalized portfolio with penalization parameter $\gamma = \mu/100$ outperforms the mean-variance but underperforms the reference portfolio. This can be attributed to the larger weight of the mean-variance criterion with respect to the tracking error in the optimized cost (3.2.2) with penalization parameter $\gamma = \mu$.

Finally, Figure 3.8 corresponds to the case of a reference portfolio with weights all equal to zero. This corresponds to a shrinking of the optimal control of the penalized portfolio. In that case, for a better visualization, we plot the total wealth of the mean-variance and penalized portfolios for penalization parameters $\gamma = \mu$ and $\gamma = \mu/100$ normalized by the standard deviation of their daily returns. On this graph, we can see that the normalized wealth of the two penalized portfolio is higher than the one of the mean-variance allocation. Similarly to the three precedent reference portfolios, the two penalized portfolios outperform the mean-variance allocation in terms of Sharpe ratio. As previously, we observe that the Sharpe ratio of the penalized portfolio with penalization parameter $\gamma = \mu$ is greater than the one with $\gamma = \mu/100$, due to the larger weight of the mean-variance criterion in the functional cost.

- Equal-weights reference portfolio

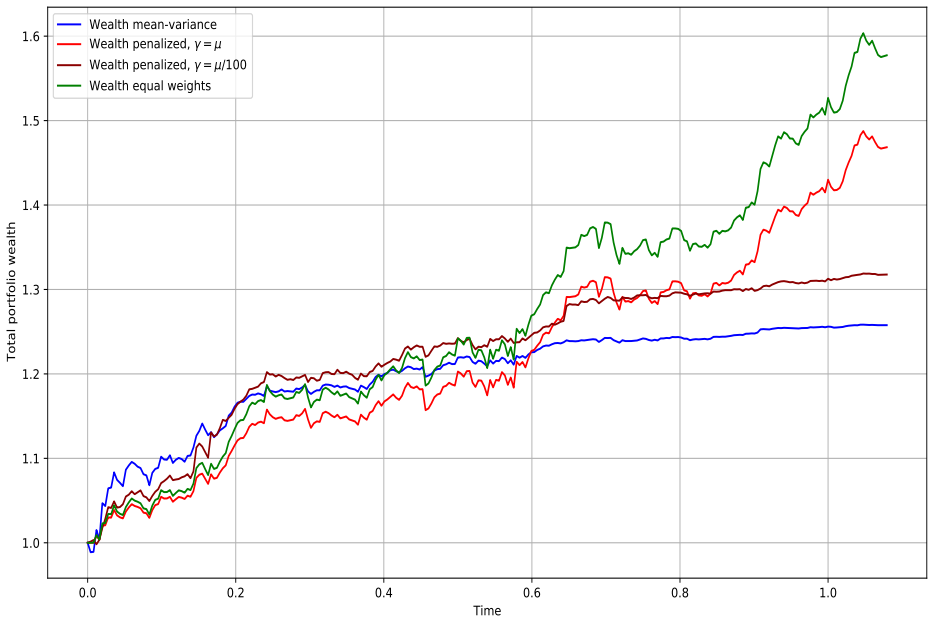


Figure 3.5: Sharpe ratios:
Mean-variance : 0.183
Equal weights : 0.258
Penalized $\gamma = \mu$: 0.260
Penalized $\gamma = \mu/100$: 0.226

- Minimum variance reference portfolio

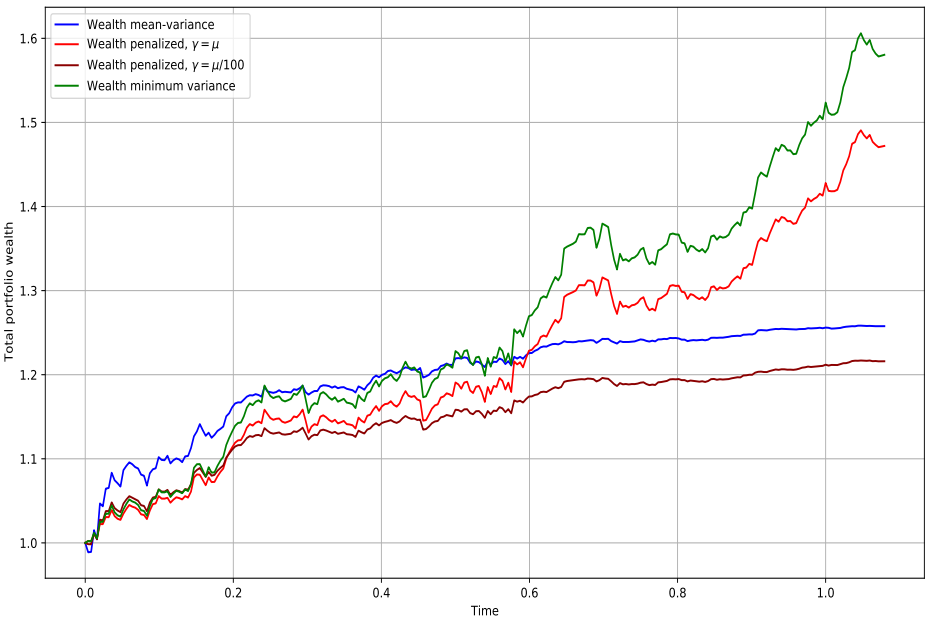


Figure 3.6: Sharpe ratios:
Mean-variance : 0.183
Minimum variance : 0.255
Penalized $\gamma = \mu$: 0.256
Penalized $\gamma = \mu/100$: 0.220

- ERC portfolio

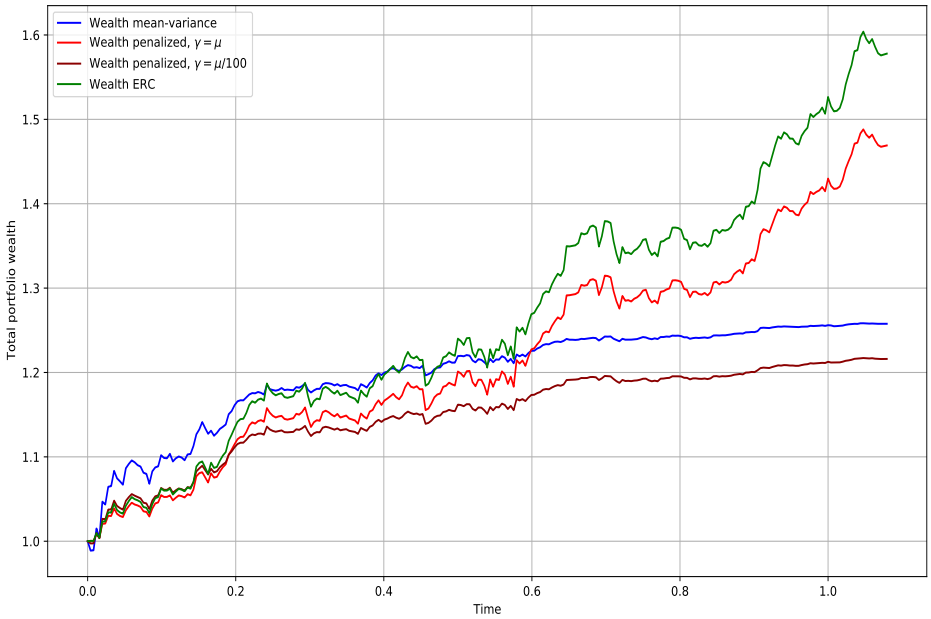


Figure 3.7: Sharpe ratios:
Mean-variance : 0.183
ERC : 0.258
Penalized $\gamma = \mu$: 0.260
Penalized $\gamma = \mu/100$: 0.225

- Zero portfolio (shrinking)

3.5 Conclusions

In this section, we propose an allocation method based on a mean-variance criterion plus a tracking error between the optimized portfolio and a reference portfolio of same wealth and fixed weights. We solve this problem as a linear-quadratic McKean-Vlasov stochastic control problem using a weak martingale approach. We then show using simulations that for a certain degree of market parameter misspecification and the right choice of reference portfolio, the mean-variance portfolio with tracking error penalization outperforms the standard mean-variance and the mean-variance allocations in terms of Sharpe ratio. Another backtest based on historical market data also shows that the mean-variance portfolio with tracking error outperforms the traditional mean-variance and the reference portfolios in terms of Sharpe ratio for the four reference portfolios considered.

Compared to the approaches of robust and bayesian optimization ([IP19], [DFNP19]), the regularization of the mean-variance allocation by a tracking-error penalization offers a more intuitive and simple approach from the financial point of view as the specification of the reference portfolio has a clear operationnal meaning. The benchmark tracking

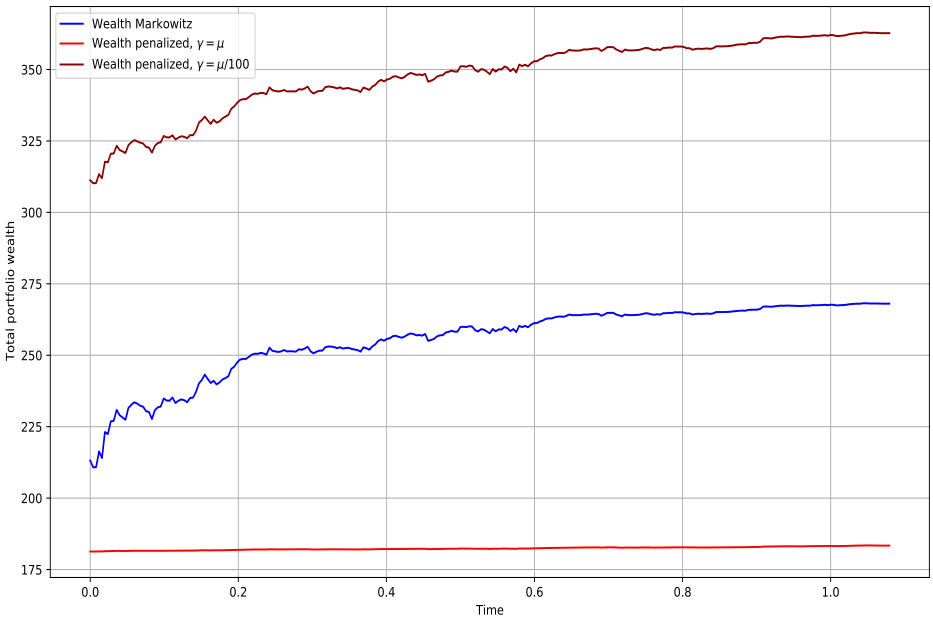


Figure 3.8: Total wealth of the mean-variance and penalized portfolios for $\gamma = \mu$ and $\gamma = \mu/100$, normalized by the standard deviation of daily returns, as a function of time.
Sharpe ratios:
mean-variance : 0.183
Penalized $\gamma = \mu$: 0.252
Penalized $\gamma = \mu/100$: 0.221

with improvement of the Sharpe ratio is intrinsically linked to the method described in this section and is, by definition, absent of the two other approaches.

Depending on the penalization parameter Γ, the choice of the reference portfolio plays an important role. If this reference portfolio is not agnostic and the computation of its weights is based on estimated market parameters, it will also be impacted by the parameter misspecification and the allocation will be more sensitive to estimation errors. The allocation would be more robust when an agnostic portfolio to market parameters, such as the equal-weights portfolio or the zero portfolio (control shrinking), is chosen as reference.

In our approach, the mean-variance criterion in the optimized cost function is based on estimated market parameters. Hence, while regularized by the tracking-error penalization, the allocation obtained still has a sensitivity to parameter misspecification. This constitutes a limitation compared to a robust optimization approach which optimizes the portfolio in the worst case scenario and is then unimpaired by parameter misspecification. Nevertheless, as the optimization of the mean-variance allocation with tracking error is based on estimators of the market parameters and not on a worst-case scenario, this allocation should outperform the robust approach for smaller values of parameter misspecification.

A potential direction for further studies would be to compare quantitatively the approach of the tracking error penalization with the robust optimization and the Bayesian

approach. It would also be interesting to compare the method presented in this section with Deep Learning based approaches such as the ones presented in [TLT20] and [Run+19].

3.A Proof of Theorem 3.3.1

The proof of Theorem 3.3.1 is based on the weak optimality principle lemma stated in [BP19], and formulated in the case of the mean-variance problem (3.2.3) as:

Lemma 3.A.1 (Weak optimality principle). *Let $\{V_t^\alpha, t \in [0,T], \alpha \in \mathcal{A}\}$ be a family of real-valued processes in the form*

$$V_t^\alpha = v_t(X_t^\alpha, \mathbb{E}[X_t^\alpha]) + \int_0^t (\alpha_s - w_r X_s^\alpha)^\top \Gamma (\alpha_s - w_r X_s^\alpha)\, ds,$$

for some measurable functions v_t on $\mathbb{R} \times \mathbb{R}$, $t \in [0,T]$, such that:

1. *$v_T(x, \bar{x}) = \mu(x - \bar{x})^2 - x$, for all $x, \bar{x} \in \mathbb{R}$,*
2. *the function $t \in [0,T] \to \mathbb{E}[V_t^\alpha]$ is nondecreasing for all $\alpha \in \mathcal{A}$*
3. *the map $t \in [0,T] \to \mathbb{E}\left[V_t^{\alpha^*}\right]$ is constant for some $\alpha^* \in \mathcal{A}$.*

Then, α^* is an optimal portfolio strategy for the mean-variance problem with tracking error (3.2.3), and

$$V_0 = J(\alpha^*).$$

We aim to construct a family of processes $\{V_t^\alpha, t \in [0,T], \alpha \in \mathcal{A}\}$ as in Lemma (3.A.1), and given the linear-quadratic structure of our optimization problem, we look for a measurable function v_t in the form:

$$v_t(x, \overline{x}) = K_t(x - \overline{x})^2 + \Lambda_t \overline{x}^2 + 2Y_t x + R_t \tag{3.A.1}$$

for some deterministic processes $(K_t, \Lambda_t, Y_t, R_t)$ to be determined. Condition (i) in Lemma (3.A.1) fixes the terminal condition

$$K_T = \mu,\ \Lambda_T = 0,\ Y_T = -1/2,\ R_T = 0.$$

For any $\alpha \in \mathcal{A}$, with associated wealth process $X := X^\alpha$, let us compute the derivative of the deterministic function $t \to \mathbb{E}[V_t^\alpha] = \mathbb{E}\left[v_t(X_t, \mathbb{E}[X_t]) + \int_0^t (\alpha_s - w_r X_s)^\top \Gamma (\alpha_s - w_r X_s)\, ds\right]$ with v_t as in (3.A.1). From the dynamics of $X = X_t^\alpha$ in (3.2.1) and by applying Itô's formula, we obtain

$$\begin{aligned}\frac{d\mathbb{E}[V_t^\alpha]}{dt} =& \mathrm{Var}(X_t)\left(\dot{K}_t + w_r^\top \Gamma w_r\right) + \overline{X}_t^2\left(\dot{\Lambda}_t + w_r^\top \Gamma w_r\right) + 2\overline{X}_t \dot{Y}_t + \dot{R}_t \\ &+ \mathbb{E}[G_t(\alpha)]\end{aligned} \tag{3.A.2}$$

where

$$G_t(\alpha) := \alpha_t^\top S_t \alpha_t + 2\left\{\left(K_t(X_t - \overline{X}_t) + Y_t + \Lambda_t \overline{X}_t\right) b^\top - X_t w_r^\top \Gamma\right\} \alpha_t.$$

By completing the square in α, and setting $S_t := K_t \Sigma + \Gamma$ and $\tilde{\rho}_t := b^\top S_t^{-1} b$, we rewrite $G_t(\alpha)$ as

$$\begin{aligned}G_t(\alpha) =& \mathbb{E}\left[\left(\alpha_t - \alpha_t^\Gamma\right)^\top S_t \left(\alpha_t - \alpha_t^\Gamma\right)\right] \\ &- \mathrm{Var}(X_t)\left\{K_t^2 \tilde{\rho}_t + w_r^\top \Gamma S_t^{-1} \Gamma w_r - 2K_t b^\top S_t^{-1} \Gamma w_r\right\} \\ &- \overline{X}_t^2\left\{\Lambda_t^2 \tilde{\rho}_t + w_r^\top \Gamma S_t^{-1} \Gamma w_r - 2\Lambda_t b^\top S_t^{-1} \Gamma w_r\right\} \\ &- 2\overline{X}_t\left\{\Lambda_t Y_t \tilde{\rho}_t - Y_t b^\top S_t^{-1} \Gamma w_r\right\} - Y_t^2 \rho_t\end{aligned}$$

with $\alpha_t^\Gamma := S_t^{-1}\Gamma w_r X_t - S_t^{-1} b\left[K_t X_t + Y_t - (K_t - \Lambda_t)\overline{X}_t\right]$. The expression in (3.A.2) is then rewritten as

$$\begin{aligned}\frac{d\mathbb{E}[V_t^\alpha]}{dt} =& \mathbb{E}\left[\left(\alpha_t - \alpha_t^\Gamma\right)^\top S_t \left(\alpha_t - \alpha_t^\Gamma\right)\right] \\ &+ \mathrm{Var}(X_t)\left\{\dot{K}_t - K_t^2\tilde{\rho}_t + w_r^\top \Gamma w_r + 2K_t b^\top S_t^{-1}\Gamma w_r - w_r^\top \Gamma S_t^{-1}\Gamma w_r\right\} \\ &+ \overline{X}_t^2\left\{\dot{\Lambda}_t - \Lambda_t^2\tilde{\rho}_t + w_r^\top \Gamma w_r + 2\Lambda_t b^\top S_t^{-1}\Gamma w_r - w_r^\top \Gamma S_t^{-1}\Gamma w_r\right\} \\ &+ 2\overline{X}_t\left(\dot{Y}_t + Y_t b^\top S_t^{-1}\Gamma w_r - \Lambda_t Y_t \tilde{\rho}_t\right) \\ &+ \dot{R}_t - Y_t^2\tilde{\rho}_t.\end{aligned}$$

Therefore, whenever

$$\begin{cases}\dot{K}_t - K_t^2\tilde{\rho}_t + w_r^\top \Gamma w_r + 2K_t b^\top S_t^{-1}\Gamma w_r - w_r^\top \Gamma S_t^{-1}\Gamma w_r &= 0 \\ \dot{\Lambda}_t - \Lambda_t^2\tilde{\rho}_t + w_r^\top \Gamma w_r + 2\Lambda_t b^\top S_t^{-1}\Gamma w_r - w_r^\top \Gamma S_t^{-1}\Gamma w_r &= 0 \\ \dot{Y}_t + Y_t b^\top S_t^{-1}\Gamma w_r - \Lambda_t Y_t \tilde{\rho}_t &= 0 \\ \dot{R}_t - Y_t^2\tilde{\rho}_t &= 0\end{cases}$$

holds for all $t \in [0,T]$, we have

$$\frac{d\mathbb{E}[V_t^\alpha]}{dt} = \mathbb{E}\left[\left(\alpha_t - \alpha_t^\Gamma\right)^\top S_t \left(\alpha_t - \alpha_t^\Gamma\right)\right]$$

which is nonnegative for all $\alpha \in \mathcal{A}$, i.e., the process V_t^α satisfies the condition (ii) of Lemma (3.A.1). Moreover, we see that $V_t^\alpha = 0,\ 0 \le t \le T$ if and only if $\alpha_t = \alpha_t^\Gamma$, $0 \le t \le T$.
$X^\Gamma := X^{\alpha^\Gamma}$ is solution to a linear McKean-Vlasov dynamics and, since $K \in C([0,T],\mathbb{R}_+^*)$, $\Lambda \in C([0,T],\mathbb{R}_+)$ and $Y \in C([0,T],\mathbb{R})$, X^Γ satisfies the square integrability condition $\mathbb{E}\left[\sup_{0\le t\le T}|X_t^\Gamma|^2\right] < \infty$, which implies that α^Γ is $\mathbb{F}$-progressively measurable and $\int_0^T \mathbb{E}[|\alpha_t^\Gamma|^2]dt < \infty$. Therefore, $\alpha^\Gamma \in \mathcal{A}$, and we conclude by the verification lemma 3.A.1 that it is the unique optimal control. □

3.B Computation linear expansion of α^γ for $\Gamma = \gamma \mathbb{I}_d \to \mathbf{0}$

$$
\begin{aligned}
\alpha_t^\gamma =& \Sigma^{-1} b \left(\frac{1}{2\mu} e^{\rho T} + X_0 - X_t \right) \\
&+ \gamma \left(\Sigma^{-1} w_r + \Sigma^{-2} b \right) \frac{X_t}{K_t^0} - \gamma \|w_r\|^2 (T-t) \frac{\Sigma^{-1}}{K_t^0} b \left(X_0 + \frac{e^{\rho T}}{\mu} \left(1 - e^{-\rho t}\right) \right) \\
&- \gamma \left(\Sigma^{-1} b X_0 C_{0,t}^1 + \Sigma^{-2} b \frac{X_0}{K_t^0} \right) \\
&- \gamma \left(\Sigma^{-1} b \frac{H_t^1}{2} + \Sigma^{-2} b \frac{e^{\rho T}}{2 K_t^0 \mu} \left(1 - e^{-\rho t}\right) \right) \\
&- \frac{\gamma}{2} \left\{ \frac{1}{\left(K_t^0\right)^2} \Sigma^{-1} \left(K_t^1 \mathbb{I}_d + \Sigma^{-1} \right) b + \Sigma^{-1} b \frac{C_{t,T}}{K_t^0} \right\} + O(\gamma^2) \\
=& \Sigma^{-1} b \left(\frac{1}{2\mu} e^{\rho T} + X_0 - X_t \right) \\
&+ \gamma \Sigma^{-1} w_r \frac{X_t}{K_t^0} \\
&- \gamma \Sigma^{-1} b \left\{ \frac{\|w_r\|^2}{K_t^0} (T-t) \left(X_0 + \frac{e^{\rho T}}{\mu} \left(1 - e^{-\rho t}\right) \right) + X_0 C_{0,t}^1 + \frac{H_t^1}{2} + \frac{K_t^1}{2 \left(K_t^0\right)^2} + \frac{C_{t,T}}{2 K_t^0} \right\} \\
&- \gamma \Sigma^{-2} b \left\{ \frac{e^{\rho T}}{2 K_t^0 \mu} \left(1 - e^{-\rho t}\right) + \frac{1}{2 \left(K_t^0\right)^2} \right\} + O(\gamma^2).
\end{aligned}
$$

3.C Computation linear expansion of $Var(X_T)$ for $\Gamma = \gamma \mathbb{I}_d \to 0$

We recall that the linear expansion of the optimal control can be written as

$$
\alpha_t^\gamma = \Sigma^{-1} b \; \alpha_t^0 + \gamma \left(\Sigma^{-1} w_r \; \alpha_t^{1,3} - \Sigma^{-2} b \; \alpha_t^{1,2} - \Sigma^{-1} b \; \alpha_t^{1,1} \right) + O(\gamma^2)
$$

where the coefficients $\alpha_t^{1,1}$, $\alpha_t^{1,2}$ and $\alpha_t^{1,3}$ are given by (3.3.8). The average total wealth of the portfolio constructed by the optimal control at time t is given by the ODE

$$
d\overline{X_t} = \rho \zeta - \gamma \left(\rho \alpha_t^{1,1} + b^\top \Sigma^{-2} b \alpha_t^{1,2} \right) + \left(\gamma \frac{b^\top \Sigma^{-1} w_r}{K_t^0} - \rho \right) \overline{X}_t + O(\gamma^2), \quad \overline{X_t} = X_0,
$$

where we set $\zeta := X_0 + \frac{1}{2\mu} e^{\rho T}$. We get the solution

$$
\begin{aligned}
\overline{X_T} =& X_0 e^{-\rho T} + \zeta \left(1 - e^{-\rho T}\right) \\
&+ \gamma \left\{ \frac{b^\top \Sigma^{-1} w_r}{\mu} \left(T\zeta - \frac{1}{2\mu} e^{\rho T} \frac{1 - e^{-\rho T}}{\rho} \right) - \int_0^T \left(\rho \alpha_s^{1,1} + b^\top \Sigma^{-2} b \alpha_s^{1,2} \right) e^{-\rho (T-s)} ds \right\} + O\left(\gamma^2\right) \\
=& \overline{X_T}^0 + \gamma \overline{X_T}^1 + O(\gamma^2)
\end{aligned}
$$

with

$$\begin{cases} \overline{X_T}^0 := X_0 e^{-\rho T} + \zeta\left(1 - e^{-\rho T}\right) \\ \overline{X_T}^1 := \frac{b^\top \Sigma^{-1} w_r}{\mu}\left(T\zeta - \frac{1}{2\mu} e^{\rho T} \frac{1-e^{-\rho T}}{\rho}\right) - \int_0^T \left(\rho \alpha_s^{1,1} + b^\top \Sigma^{-2} b \alpha_s^{1,2}\right) e^{-\rho(T-s)} ds \end{cases}$$

and

$$\overline{X_T}^2 = \left(\overline{X_T}^0\right)^2 + 2\gamma \overline{X_T}^0 \overline{X_T}^1 + O(\gamma^2)$$

The average of the square of the portfolio wealth at time t is given by the ODE

$$\begin{aligned} d\overline{X_t^2} = &\left(\zeta - \gamma \alpha_t^{1,1}\right)^2 \rho - 2\gamma b^\top \Sigma^{-2} b \alpha_t^{1,2}\left(\zeta - \gamma \alpha_t^{1,1}\right) \\ &+ 2\gamma \frac{w_r^\top \Sigma^{-1} b}{K_t^0}\left(\zeta - \gamma \alpha_t^{1,1}\right) \overline{X}_t \\ &- \rho \overline{X_t^2} + O(\gamma^2) \end{aligned}$$

which gives the solution

$$\begin{aligned} \overline{X_T^2} = &X_0^2 e^{-\rho T} + \zeta^2\left(1 - e^{-\rho T}\right) \\ &- 2\gamma\zeta \int_0^T \left\{\rho \alpha_s^{1,1} + b^\top \Sigma^{-2} b \alpha_s^{1,2}\right\} e^{-\rho(T-s)} ds \\ &+ 2\gamma \frac{w_r^\top \Sigma^{-1} b}{\mu} \zeta \int_0^T \overline{X_s}^0 ds + O(\gamma^2). \end{aligned}$$

We can then compute the variance of the terminal total wealth of the portfolio given by the control (3.3.7)

$$\begin{aligned} Var(X_T) = &\overline{X_T^2} - \overline{X_t}^2 \\ = &\frac{e^{-\rho T}}{1 - e^{-\rho T}}\left(\overline{X_T}^0 - X_0\right)^2 \\ &+ \gamma \frac{b^\top \Sigma^{-1} w_r}{\mu^2}\left(\zeta T - \frac{1}{2\mu} e^{\rho T} \frac{1 - e^{-\rho T}}{\rho}\right) \\ &- \gamma \int_0^T \left(\frac{\rho}{\mu} \alpha_s^{1,1} + \frac{b^\top \Sigma^{-2} b}{\mu} \alpha_s^{1,2}\right) e^{-\rho(T-s)} ds + O(\gamma^2). \end{aligned}$$

□

3.D Proof that K_t and Λ_t are bounded in γ

To prove this, we use a theorem from [Gro19] (also in [HNW93], Theorem 14.1, p93). We rewrite the differential equation of K as

$$\frac{dK_t}{dt} = f(t, K_t, \gamma), \quad K_T = \mu$$

with $f(t, K_t, \gamma) := (K_t b - \gamma w_r)^\top (K_t \Sigma + \gamma \mathbb{I}_d)^{-1} (K_t b - \gamma w_r) - \gamma \|w_r\|^2$, where $\|\cdot\|$ denotes the euclidean norm in $\mathbb{R}^d$.
For $t \in [0, T]$, the partial derivatives $\partial f / \partial K$ and $\partial f / \partial \gamma$ exist and are continuous in the neighbourhood of the solution K_t. Then the partial derivative

$$\frac{\partial K_t}{\partial \gamma} = \psi_t$$

exists, is continuous, and satisfies the differential equation

$$\psi_t^{'} = \frac{\partial f}{\partial K}(t, K_t, \gamma)\psi_t + \frac{\partial f}{\partial \gamma}(t, K_t, \gamma).$$

Recalling that the derivative of the inverse of a nonsingular matrix M whose elements are functions of a scalar parameter p w.r.t this parameter is equal to $\frac{\partial M^{-1}}{\partial_p} = -M^{-1}\frac{\partial M}{\partial_p}M^{-1}$, we can compute the partial derivatives $\partial f/\partial K$ and $\partial f/\partial \gamma$, and we obtain the following differential equation for ψ

$$\begin{cases} (\psi_t)^{'} = \left[-\|\sigma^\top S_t^{-1}\left(K_t b - \gamma w_r\right)\|^2 + 2b^\top S_t^{-1}\left(K_t b - \gamma w_r\right)\right]\psi_t - \|w_r + S_t^{-1}\left(K_t b - \gamma w_r\right)\|^2, \quad t \in [0,T] \\ \psi_T = 0. \end{cases}$$

This ODE has an explicit solution given by

$$\psi_t = \int_t^T A_s e^{-\int_t^s B_u du} ds$$

with $A_t \geq 0,\ \forall t \in [0,T]$ equal to

$$A_t := \frac{K_t^2}{\gamma^2}\|\left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma\right)^{-1}(b + \Sigma w_r)\|^2 \underset{\gamma\to\infty}{\longrightarrow} 0$$

and

$$\begin{aligned} B_t :=& 2\frac{K_t}{\gamma}\left(b + \Sigma w_r\right)^\top\left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma\right)^{-1}(b + \Sigma w_r) \\ &- \frac{K_t^2}{\gamma^2}\left(b + \Sigma w_r\right)^\top\left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma\right)^{-1}\Sigma\left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma\right)^{-1}(b + \Sigma w_r) \\ &- 2b^\top w_r - \|\sigma^\top w_r\|^2. \end{aligned}$$

We have $B_t \underset{\gamma\to\infty}{\to} -2b^\top w_r - \|\sigma^\top w_r\|^2$, thus $\psi_t \underset{\gamma\to\infty}{\longrightarrow} 0,\ \forall t \in [0,T]$ and K_t is bounded in γ for every $t \in [0,T]$.
In the same spirit, we rewrite the differential equation of Λ_t as

$$\frac{d\Lambda_t}{dt} = g(t, \Lambda_t, \gamma), \quad \Lambda_t = 0$$

with $g(t, \Lambda_t, \gamma) := \left(\Lambda_t b - \gamma w_r\right)^\top S_t^{-1}\left(\Lambda_t b - \gamma w_r\right) - \gamma\|w_r\|^2$. The partial derivative

$$\frac{\partial \Lambda_t}{\partial \gamma} = \phi_t$$

exists, is continuous and satisfies the differential equation

$$\begin{cases} \phi_t^{'} = 2b^\top S_t^{-1}\left(\Lambda_t b - \gamma w_r\right)\phi_t - \left[\|w_r + S_t^{-1}\left(\Lambda_t b - \gamma w_r\right)\|^2 + \psi_t\|\sigma^\top S_t^{-1}\left(\Lambda_t b - \gamma w_r\right)\|^2\right], \quad t \in [0,T] \\ \phi_0 = 0. \end{cases}$$

which gives the explicit solution

$$\phi_t = \int_t^T C_s e^{-\int_t^s D_u du} ds$$

with $C_t \geq 0,\ \forall t \in [0,T]$ equal to

$$C_t := \|\frac{1}{\gamma}\left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma\right)^{-1}\left(\Lambda_t b + K_t\Sigma w_r\right)\|^2 + \psi_t\|\frac{1}{\gamma}\sigma^\top\left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma\right)^{-1}\left(\Lambda_t b + K_t\Sigma w_r\right) - \sigma^\top w_r\|^2$$

and

$$D_t := 2\left[\frac{1}{\gamma}b^\top\left(\mathbb{I}_d + \frac{K_t}{\gamma}\Sigma\right)^{-1}(\Lambda_t b + K_t\Sigma w_r) - b^\top w_r\right].$$

We showed that $\frac{K_t}{\gamma}, \psi_t \underset{\gamma\to\infty}{\longrightarrow} 0$ for every $t \in [0,T]$. Thus $C_t \underset{\gamma\to\infty}{\longrightarrow} 0$, $D_t \underset{\gamma\to\infty}{\longrightarrow} -2b^\top w_r$ and $\phi_t \underset{\gamma\to\infty}{\longrightarrow} 0$, $\forall t \in [0,T]$. Λ_t is then bounded in γ for every $t \in [0,T]$. $\square$

Part II

Stochastic delayed systems

Chapter 4

Linear-quadratic stochastic delayed control and deep learning resolution

Abstract We consider a class of stochastic control problems with a delayed control, both in drift and diffusion, of the type $dX_t = \alpha_{t-d}(bdt + \sigma dW_t)$. We provide a new characterization of the solution in terms of a set of Riccati partial differential equations. Existence and uniqueness are obtained under a sufficient condition expressed directly as a relation between the horizon T and the quantity $d(b/\sigma)^2$. Furthermore, a deep learning scheme* is designed and used to illustrate the effect of delay on the Markowitz portfolio allocation problem with execution delay.

Keywords: Linear-quadratic stochastic control; delay; Riccati PDEs; Markowitz portfolio allocation.

4.1 Introduction

The control of systems whose dynamic contains delays on the state and/or control has attracted the attention of the optimization and engineering communities in the last decades due to its wide variety of applications, allowing to tackle problems where the past of a system influences its present or where an agent controls a system with a latency. As a non-exhaustive list of applications we may cite the following papers, classified by their applications domain: Engineering ([TG99], [Huz+02]); Advertising ([Set74], [Pau77], [GRM05], [GMS09]); Learning by doing with memory effect ([dAVV12]); Growth model with lags between investment decision and project completion ([AZ99], [Hal+77], [JD07], [Bam08], [BFG12]); Investment ([Tso11], [KP82]). More recently, the introduction of delayed control together with mean-field effects was studied ([Car+18], [FZ20]) and new machine learning methods have been designed to numerically solve stochastic control problems with delay ([HH21]). We also refer to the monograph [Sip+11] to find literature on the various effects of delays on traffic flow modelling, chemical processes, population dynamics, supply chain, etc.

In the optimal control community, two main approaches have emerged: the *structural state method* and the *extended state method*, and we refer to [Ben+07, Part II, Chapter 3] for the study of the latter in the deterministic case and [FF14] for the structural state

*The code is available in a IPython notebook.

approach in the stochastic case. Let us also mention the paper by [FF14] for an overview and exhaustive list of references.

In this section, we aim at studying the challenging case where there is a delayed control both in the drift and volatility. Except in [FF14], this situation is not treated theoretically nor numerically in the references above. The main difficulty comes from the fact that the natural formulation of a control problem with delayed control involves a *boundary control problem*. Indeed, assume for instance that X denotes a state variable following the simple dynamic $\dot{X}_t = \alpha_{t-d}$, where α denotes the control. For any time t and index $s \in [-d, 0]$, set $u_t(s) = \alpha_{t+s}$, the *memory* of the control α. Then, note that $\partial_t u_t(s) = \partial_s u_t(s)$ and $u_t(0) = \alpha_t$. Thus, the natural infinite dimensional formulation of the controlled system is

$$\begin{aligned} \text{State eq. on } (X,u) &\begin{cases} \dot{X}_t = \mathcal{M}u_t \\ (\partial_t - \partial_s)u_t(s) = 0, \end{cases} \\ \text{Boundary constraint } &\{ \ \mathcal{B}u_t = \alpha_t, \\ \text{Initial conditions } &\{ \ u_0(s) = \gamma_s, \quad X_0 = x, \end{aligned}$$

where $t, s \in [0,T] \times [-d, 0]$, $\mathcal{M}u := u(-d)$, $\mathcal{B}u := u(0)$ and γ is the initial value of the control over $[-d, 0]$. Consequently, any delayed controlled problem where the delay appears in the control variable can be recast as a boundary control problem whose geometry is parametrized by the delay d, see Figure 4.1.

Main contributions. Our goal is to shed some lights on the difficulty related to delayed control on the volatility and to provide a practical and simple tool for designing a numerical scheme practitioners can play with. In this section, we study the most simple linear-quadratic control problem with delayed control both in drift and volatility. The optimal feedback control and the value function are given in terms of Riccati partial differential equations and the extended state $(x, u) \in H = \mathbb{R} \times L^2([0,T], \mathbb{R})$, where x denotes the position and $s \in [-d, 0] \mapsto u(s)$ the memory of the control. The existence and uniqueness of these latter are proven under a condition, emerging from the delay feature, involving the drift b, the volatility σ, the delay d and the horizon T. Finally, we adopt a deep learning approach in the spirit of the papers by [RPK19] (Physics Informed Neural Network) and [SS18] (Deep Galerkin) to propose a numerical scheme. Our results are illustrated on the celebrated Markowitz portfolio allocation problem where we take into account execution delay. We believe the semi-explicit resolution of infinite dimensional control problem by means of deep learning method will open the door to several interesting applications such as quick simulations of richer models, precise benchmarking of *reinforcement learning* algorithms, etc.

Outline of this section. The rest of the section is organized as follows: In Section 4.2 we formulate the stochastic delayed control problem and derive an heuristic approach through a lifting in an infinite dimensional space, namely the *extended state space* in the spirit of [Ich82], but without the use of semi-group theory. We state in Section 4.3 a verification theorem and prove existence and uniqueness results for the Riccati PDEs. A deep learning based numerical scheme with two applications on Markowitz portfolio allocation is given in Section 4.4, with a detailed analysis of the effect of the delay feature on the allocation strategy.

Notations.
Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, a filtration $\mathbb{F} = (\mathcal{F}_t)_{t \geq 0}$ satisfying the usual condi-

tions and $a < b$ two real numbers, we denote by

$$L^2([a,b],\mathbb{R}) = \left\{ Y : [a,b] \mapsto \mathbb{R}, \text{ s.t. } \int_a^b |Y_t|^2 dt < \infty \right\},$$

$$L^2_{\mathbb{F}}([a,b],\mathbb{R}) = \left\{ Y : \Omega \times [a,b] \mapsto \mathbb{R}, \mathbb{F}-\text{prog. measurable s.t. } \mathbb{E}\left[\int_a^b |Y_t|^2 dt\right] < \infty \right\}.$$

Here $|\cdot|$ denotes the Euclidian norm on $\mathbb{R}$ or $\mathbb{R}^d$, and $H = \mathbb{R} \times L^2([0,T],\mathbb{R})$ denotes the *extended state space* endowed with the scalar product $\langle x, y\rangle_H = x_0 y_0 + \int_{-d}^0 x_1(s) y_1(s) ds$. For any $z = (x,u) \in H$, we use the notation $z_0 = x$ and $z_1 = u$.

4.2 Formulation of the problem and heuristic approach

Let $(\Omega, \mathcal{F}, \mathbb{F} := (\mathcal{F}_t)_{t\le 0}, \mathbb{P})$ be a complete filtered probability space on which a real-valued Brownian motion $(W_t)_{t\le 0}$ is defined and consider the simple system defined on $[0,T]$ by the following dynamics

$$\begin{cases} dX_t^\alpha = \alpha_{t-d}\left(bdt + \sigma dW_t\right), & 0 \le t \le T, \\ X_0 = x, \quad \alpha_s = \gamma(s), & s \in [-d, 0], \end{cases} \tag{4.2.1}$$

endowed with the cost functional

$$J(\alpha) = \mathbb{E}\left[(X_T^\alpha)^2\right], \tag{4.2.2}$$

where $\gamma \in L^2([-d,0],\mathbb{R})$ and α models the control chosen in the set of admissible strategies $\mathcal{A}$:

$$\mathcal{A} = \left\{ \alpha \in L^2_{\mathbb{F}}([0,T],\mathbb{R}) \text{ such that (4.2.1) has a solution satisfying } \mathbb{E}\left[\sup_{t\le T} |X_t^\alpha|^2\right] < \infty \right\}.$$

For any $0 \le a < b \le T$, we also define the set $\mathcal{A}_{a,b}$ as the restriction of $\mathcal{A}$ to $L^2_{\mathbb{F}}([a,b],\mathbb{R})$.

Remark 4.2.1. *At this point, we may expect a priori that the optimization problem (4.2.1)-(4.2.2) admits an optimizer provided $\sigma \neq 0$, even if the control is not directly penalized. The intuition behind this a priori belief is that, the more α is aggressive in bringing X to 0, the more the variance of X increases due to the diffusion term. It is the case in the classical LQ stochastic optimization problem with controlled volatility such as*

$$\begin{aligned} dX_t^\alpha &= \alpha_t(bdt + \sigma dW_t), \qquad t \le T, \\ X_0 &= x, \\ J(\alpha) &= \mathbb{E}[(X_T^\alpha)^2], \end{aligned}$$

where the optimal control reads $\alpha_t^* = -\frac{b}{\sigma^2} X_t^{\alpha^*}$ and the value function $V_t = e^{(t-T)\frac{b^2}{\sigma^2}}$. A surprising finding in our work is the necessity for a more restricting condition on the diffusion coefficient due to the delay feature, see Proposition 4.3.2.

Remark 4.2.2. *In the rest of this section we focus on the one dimensional case with delayed control both in drift and volatility which features the main difficulties related to the presence of the delay. Although Proposition 4.3.2 concerning the existence and uniqueness of a Riccati-PDE system does not directly extend to the multidimensional case, the verification Theorem can easily be adapted to the multidimensional case with delayed state and control.*

The first step consists in lifting the dynamics in the infinite dimensional Hilbert space $H = \mathbb{R} \times L^2([0,T],\mathbb{R})$, where the system is naturally Markovian. To do so, denote $u_t(s) = \alpha_{t+s}$ for any $t \leq T$ and $s \in [-d, 0]$, a transport of the control. The dynamics (4.2.1) then reads

$$\begin{cases} dZ_t^\alpha = AZ_t^\alpha dt + BZ_t^\alpha dW_t + Cd\alpha_t, & 0 \leq t \leq T, \\ Z_0 = (x, \gamma), \end{cases} \tag{4.2.3}$$

where Z^α is defined as the $H = \mathbb{R}\times L^2([0,T],\mathbb{R})$-valued random process $Z_t^\alpha = (X_t^\alpha, u_t(\cdot))$ and

$$A = \begin{pmatrix} 0 & b\delta_{-d} \\ 0 & \partial_s \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & \sigma\delta_{-d} \\ 0 & 0 \end{pmatrix}, \qquad C = \begin{pmatrix} 0 \\ 1_0(\cdot) \end{pmatrix}.$$

Let V be the value function

$$V(t,z) = V(t,(x,u)) = \inf_{\alpha\in\mathcal{A}_{t,T}} \mathbb{E}[(Z_T^\alpha)_0^2] = \inf_{\alpha\in\mathcal{A}_{t,T}} \mathbb{E}[(X_T^\alpha)^2], \qquad z \in H,$$

where Z^α denotes the solution to (4.2.3) starting from $z = (x,u)$ at time t. Then, assuming $V \in C^{1,2}\left([0,T] \times L^2\left([-d,0]\right),\mathbb{R}\right)$, the dynamic programming principle reads

$$\begin{aligned} V(t,z) &= \inf_{\alpha\in\mathcal{A}_{t,t+h}} \mathbb{E}[V(t+h, Z_{t+h}^\alpha)] \\ &= \inf_{\alpha\in\mathcal{A}_{t,t+h}} \mathbb{E}\Bigg[V(t,z) + \int_t^{t+h} \partial_t V(s, Z_s^\alpha)ds + \int_t^{t+h} \partial_z V(s, Z_s^\alpha)dZ_s^\alpha \\ &\quad + \frac{1}{2}\int_t^{t+h} \partial_z^2 V(s, Z_s^\alpha)d\langle Z^\alpha\rangle_s\Bigg]. \\ &= \inf_{\alpha\in\mathcal{A}_{t,t+h}} \mathbb{E}\Bigg[V(t,z) + \int_t^{t+h} \partial_t V(s, Z_s^\alpha)ds + \int_t^{t+h} \partial_z V(s, Z_s^\alpha)(AZ_s^\alpha ds + Cd\alpha_s) \\ &\quad + \frac{1}{2}\int_t^{t+h} \partial_z^2 V(s, Z_s^\alpha)d\langle Z^\alpha\rangle_s\Bigg], \end{aligned}$$

Note that $1_0(\cdot) = 0_{L^2}$. As a result, simplifying by $V(t,z)$, dividing by h and letting $h \to 0$ yields (informally) the Hamilton-Jacobi equation

$$\begin{gathered} \partial_t V + \inf_{\alpha\in\mathbb{R}}\{\partial_z V Az + \partial_z^2 V(Bz \otimes Bz)\} = 0, \qquad t \leq T, \quad z \in L^2([-d,0],\mathbb{R}), \\ V(T,z) = z_0^2. \end{gathered} \tag{4.2.4}$$

Recall that in equation (4.2.4), we have $z_1(0) = u(0) = \alpha$. Let us now assume that the value function V is of the following form

$$V(t,z) = \langle P_t z, z\rangle_H,$$

where $P \in C([0,T],\mathcal{L}(H,H))$ is a self-adjoint bounded positive operator valued function of the form

$$P_t : (x,\gamma(\cdot)) \mapsto \begin{pmatrix} P_{11}(t)x + \int_{-d}^0 P_{12}(t,s)\gamma(s)ds \\ P_{12}(t,\cdot)x + P_{\hat{22}}(t,\cdot)\gamma(\cdot) + \int_{-d}^0 P_{22}(t,\cdot,s)\gamma(s)ds \end{pmatrix}.$$

Thus, for any $z = (x, u) \in H$ such that $u(0) = \alpha$ and $t \leq T$, equation (4.2.4) reduces to

$$\langle \dot{P}_t z, z\rangle_H + \inf_{\alpha\in\mathbb{R}}\{\langle P_t Az, z\rangle_H + \langle P_t z, Az\rangle_H + \langle P_t Bz, Bz\rangle_H\} = 0. \tag{4.2.5}$$

Furthermore, using the boundary condition $u(0) = \alpha$ together with integration by part, we have

$$\begin{aligned}
\langle P_t z, Az\rangle_H &= (P_t z)_0 (Az)_0 + \int_{-d}^{0} (P_t z)_1(s)(Az)_1(s)ds \\
&= bu(-d)\left(P_{11}(t)x + \int_{-d}^{0} P_{12}(t,s)u(s)ds\right) + \alpha x P_{12}(t,0) - u(-d)xP_{12}(t,-d) \\
&\quad - x\int_{-d}^{0} \partial_s P_{12}(t,s)u(s)ds + \alpha\int_{-d}^{0} P_{22}(t,0,s)u(s)ds \\
&\quad - u(-d)\int_{-d}^{0} P_{22}(t,-d,s)u(s)ds - \int_{-d}^{0}\int_{-d}^{0} \partial_s P_{22}(t,s,r)u(s)u(r)dsdr \\
&\quad + \alpha^2 P_{\hat{2}\hat{2}}(t,0) - u(-d)^2 P_{\hat{2}\hat{2}}(t,-d) - \int_{-d}^{0} \partial_s P_{\hat{2}\hat{2}}(t,s)ds,
\end{aligned} \tag{4.2.6}$$

and

$$\langle P_t Bz, Bz\rangle_H = \sigma^2 P_{11}(t)u(-d)^2. \tag{4.2.7}$$

Remark 4.2.3. *In (4.2.6), along with the integration by part, formulas such as $2u\partial_s u = \partial_s u^2$ were (formally) used. However, as it appears in the verification Theorem 4.3.1, the feedback optimal control obtained is as regular as the controlled process X^α and thus as regular as the Brownian motion W. This is why our approach is only heuristic and justifies the need for the verification Theorem 4.3.1.*

As a consequence, the minimizer of the Hamiltonian in (4.2.5) reads

$$\alpha^*(t,z) = -\frac{1}{P_{\hat{2}\hat{2}}(t,0)}\left(xP_{12}(t,0) + \int_{-d}^{0} P_{22}(t,0,s)u(s)ds\right). \tag{4.2.8}$$

Remark 4.2.4. *Note that when $d \to 0$, then $\alpha^*(t,z) \to -\frac{b}{\sigma^2}x$ which agrees with the optimal strategy in the undelayed case.*

Combining (4.2.5), (4.2.6) and (4.2.7) yields the set of Riccati partial differential equations

$$\begin{aligned}
&\dot{P}_{11}(t) = \frac{P_{12}(t,0)^2}{P_{\hat{2}\hat{2}}(t,0)}, && (\partial_t - \partial_s)(P_{12})(t,s) = \frac{P_{12}(t,0)P_{22}(t,s,0)}{P_{\hat{2}\hat{2}}(t,0)}, \\
&(\partial_t - \partial_s)(P_{\hat{2}\hat{2}})(t,s) = 0, && (\partial_t - \partial_s - \partial_r)(P_{22})(t,s,r) = \frac{P_{22}(t,s,0)P_{22}(t,0,r)}{P_{\hat{2}\hat{2}}(t,0)},
\end{aligned} \tag{4.2.9}$$

accompanied by the boundary conditions

$$\begin{aligned}
&P_{12}(t,-d) = bP_{11}(t), && P_{\hat{2}\hat{2}}(t,-d) = \sigma^2 P_{11}(t), \\
&P_{22}(t,s,-d) = bP_{12}(t,s), && P_{22}(t,-d,r) = bP_{12}(t,r),
\end{aligned} \tag{4.2.10}$$

and the final conditions

$$P_{11}(T) = 1, \qquad P_{12}(T,s) = P_{\hat{2}\hat{2}}(T,s) = P_{22}(T,s,r) = 0, \tag{4.2.11}$$

for almost every $s, r \in [-d, 0]$.

Remark 4.2.5. *Looking at the expression (4.2.8), we can already guess some effects of the existence of a delay on the optimal strategy. Indeed, from (4.2.10) one notes that $P_{12} \approx b$, $P_{\hat{2}\hat{2}} \approx \sigma^2$, $P_{22} \approx b^2$, and we may write*

$$\alpha \approx \frac{-1}{\sigma^2}(bx + db^2\alpha) \approx \frac{-bx}{\sigma^2(1 + d(b/\sigma)^2)}.$$

In Section 4.4, we illustrate numerically the various effects of the delayed control through two examples of Markowitz portfolio allocation with execution delay.

Note that due to the existence of the delay, the value function is independent of the control chosen after $T - d$, so that $P_{12}(t, s) = P_{\hat{2}\hat{2}}(t, s) = P_{22}(t, s, r) = 0$ whenever $t + s \geq T - d$ or $t + r \geq T - d$. Similarly, the optimal control defined in (4.2.8) is ill defined on $[T - d, T]$ so we decide to set to zero the control after time $T - d$ and rewrite

$$\alpha^*(t, z) = -\frac{1_{t \leq T-d}}{P_{\hat{2}\hat{2}}(t, 0)} \left(xP_{12}(t, 0) + \int_{-d}^{0} P_{22}(t, 0, s)u(s)ds \right). \tag{4.2.12}$$

Thus, to make sense of the set of Ricatti partial differential equations (4.2.9)-(4.2.10)-(4.2.11) and the optimal control (4.2.12), we adopt the convention $0^2/0 = 0$ and define the concept of solution as follows

Definition 4.2.1. *A 4-uplets $P = (P_{11}, P_{12}, P_{\hat{2}\hat{2}}, P_{22})$ is said to be a solution to (4.2.9)-(4.2.10)-(4.2.11) if $P_{11} : [0, T] \mapsto \mathbb{R}$, $P_{22}, P_{\hat{2}\hat{2}} : [0, T] \times [-d, 0] \mapsto \mathbb{R}$ and $P_{22} : [0, T] \times [-d, 0]^2 \mapsto \mathbb{R}$ are piecewise absolutely continuous functions satisfying (4.2.9)-(4.2.10)-(4.2.11) with $P_{\hat{2}\hat{2}}(t) > 0$ for any $t < T - d$.*

The reason we chose *piecewise* absolutely continuous functions as our set of functions is because we expect the kernel P to be discontinuous. To illustrate this consideration, cut the domain $\mathcal{D}$ into three pieces $\mathcal{D} = [0, T] \times [-d, 0]^2 = \mathcal{D}_a \cup \mathcal{D}_b \cup \mathcal{D}_c$ as represented in Figure 4.1, with

$$\begin{aligned}
\mathcal{D}_a &= [0, T - d] \times [-d, 0]^2, \\
\mathcal{D}_b &= \{(t, s, r) \in \mathcal{D} \quad \text{s.t.} \quad t > T - d, \quad t + s \vee r < T - d\}, \\
\mathcal{D}_c &= \{(t, s, r) \in \mathcal{D} \quad \text{s.t.} \quad t > T - d, \quad t + s \vee r \geq T - d\}
\end{aligned}$$

and note that, necessarily, $P_{12}, P_{\hat{2}\hat{2}}$ and P_{22} are null on $\mathcal{D}_c$ but not on the remaining domain, see also the numerical simulations in Figure 4.5.

In the next section, we provide a proof of the existence and uniqueness of system (4.2.9)-(4.2.10)-(4.2.11), and a verification theorem yielding rigorously the optimal control and value of (4.2.1)-(4.2.2).

4.3 Verification and existence results

In this section, we establish a verification result for the optimization problem (4.2.1)-(4.2.2).

Theorem 4.3.1 (Verification Theorem)**.** *Assume that*

1. *There exists a solution P to (4.2.9)-(4.2.10)-(4.2.11) in the sense of Definition 4.2.1,*
2. *The control strategy defined as*

$$\alpha_t^* = \frac{-1_{t \leq T-d}}{P_{\hat{2}\hat{2}}(t, 0)} \left\{ X_t^{\alpha^*} P_{11}(t, 0) + \int_{t-d}^{t} P_{22}(t, 0, s - t)\alpha_s^* ds \right\}. \tag{4.3.1}$$

where X^{α^*} denotes the controlled state is an admissible control.

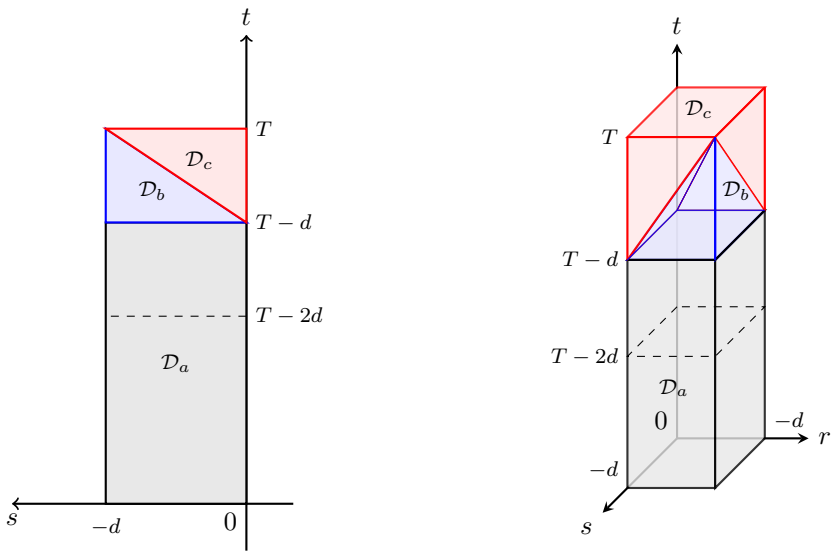


Figure 4.1: Left: Cross section of $\mathcal{D}$ along $r = 0$. Right: full domain $\mathcal{D} = \mathcal{D}_a \cup \mathcal{D}_b \cup \mathcal{D}_c$.

Then the optimization problem (4.2.1)-(4.2.2) *admits* (4.3.1) *as an optimal feedback control. Furthermore, for* $z = (x, \gamma) \in H$*, the value is given by*

$$\begin{aligned} V(z) &= P_{11}(0)x^2 + 2x\int_{-d}^{0} P_{12}(0,s)\gamma_s ds + \int_{-d}^{0} P_{\hat{22}}(0,s)\gamma_s^2 ds \\ &\quad + \int_{[-d,0]^2} \gamma_s\gamma_u P_{22}(0,s,r)dsdr \\ &= \langle P_0 z, z\rangle_H. \end{aligned} \tag{4.3.2}$$

Proof. The proof is a basic application of the martingale optimality principle, see [EK81]. Let $\alpha \in \mathcal{A}$ and define

$$\begin{aligned} V_t^\alpha &= P_{11}(t)(X_t^\alpha)^2 + 2x\int_{t-d}^{t} P_{12}(t,s-t)\alpha_s ds + \int_{t-d}^{t} P_{\hat{22}}(t,s-t)\alpha_s^2 ds \\ &\quad + \int_{[t-d,t]^2} P_{22}(t,s-t,r-t)\alpha_s\alpha_r dsdr \\ &= \langle P_t Z_t^\alpha, Z_t^\alpha\rangle_H. \end{aligned} \tag{4.3.3}$$

An application of Itô's formula to (4.3.3) combined with differentiation under the integral symbol, authorized by the assumed boundness of P and its derivatives, yield

$$\begin{aligned} dV_t^\alpha = &\Bigg\{ \mathbf{1}_t (X_t^\alpha)^2 + 2\Bigg(\mathbf{2}_t X_t^\alpha \alpha_t + \mathbf{3}_t X_t^\alpha \alpha_{t-d} + X_t^\alpha \int_{t-d}^{t} \mathbf{4}_t(s)\alpha_s ds\Bigg) \\ &+ \mathbf{5}_t \alpha_{t-d}^2 + \alpha_{t-d}\mathbf{6}_t(s)\int_{t-d}^{t}\alpha_s ds + \alpha_t \int_{t-d}^{t} \mathbf{7}_t(s)\alpha_s ds \\ &+ \int_{t-d}^{t} \mathbf{8}_t(s)\alpha_s^2 ds + \int_{[t-d,t]^2} \mathbf{9}_t(s,u)\alpha_s\alpha_u dsdu \Bigg\} dt + Z_t^\alpha dW_t, \end{aligned}$$

where we have set

$$\mathbf{1}_t = \dot{P}_{11}(t) \qquad \mathbf{2}_t = P_{12}(t,0)$$
$$\mathbf{3}_t = 2bP_{11}(t) - P_{12}(t,-d) \qquad \mathbf{4}_t(s) = (\partial_t - \partial_s)(E_2)(t,s-t)$$
$$\mathbf{5}_t = \sigma^2 P_{11}(t) - P_{\hat{2}\hat{2}}(t,-d) \qquad \mathbf{6}_t(s) = \left(bP_{12}(t,\cdot) - P_{22}(t,\cdot,-d)\right)(s-t)$$

$$\mathbf{7}_t(s) = P_{22}(t,0,s-t) \qquad \mathbf{8}_t(s) = (\partial_t - \partial_s)(P_{\hat{2}\hat{2}})(t,s-t)$$
$$\mathbf{9}_t(s,r) = (\partial_t - \partial_s - \partial_r)(P_{22})(t,s-t,r-t),$$

and

$$Z_t^\alpha = 2\sigma\alpha_{t-d}\left(X_t^\alpha P_{11}(t) + \int_{t-d}^t \alpha_s P_{12}(t,s-t)ds\right).$$

Then, using the set of constraints (4.2.9)-(4.2.10)-(4.2.11) together with (4.3.1) and a completion of the square in α yield

$$dV_t^\alpha = \left(P_{\hat{2}\hat{2}}(t,0)\left(\alpha_t - \mathcal{T}(\alpha)_t\right)^2\right)dt + Z_t^\alpha dW_t,$$

where $\mathcal{T}(\alpha)$ is defined as

$$\mathcal{T}(\alpha)_t = -\frac{1_{t\le T-d}}{P_{\hat{2}\hat{2}}(t,0)}\left\{X_t^\alpha P_{12}(t,0) + \int_{t-d}^t \alpha_s P_{22}(t,0,s-t)ds\right\}, \qquad t \le T.$$

Note that since the kernels P_i's are bounded and the control α^* is assumed to be admissible, $\alpha^* \in L^2_{\mathbb{F}}([0,T],\mathbb{R})$, X^α is continuous and the stochastic integral $\int_0^\cdot Z_s^\alpha dW_s$ is well posed. Furthermore it is a local martingale. Thus, there exists a localizing increasing sequence of stopping times $\{\tau_k\}_{k\ge1}$ converging to T such that $\int_0^{\cdot\wedge\tau_k} Z_s^\alpha dW_s$ is a martingale for every $k \ge 1$. Then, for any $k \ge 1$

$$\mathbb{E}\left[V_{T\wedge\tau_k}^\alpha\right] = V_0^\alpha + \mathbb{E}\left[\int_0^{T\wedge\tau_k} P_{\hat{2}\hat{2}}(s,0)(\alpha_s - \mathcal{T}(\alpha)_s)^2 ds\right].$$

Note that $t \mapsto V_t^\alpha$ is continuous since P is bounded and $\alpha \in L^2_{\mathbb{F}}([-d,T],\mathbb{R})$. Thus, an application of the dominated convergence theorem on the left term (recall that $\mathbb{E}\left[\sup_{t\le T}|X_t^\alpha|^2\right] < \infty$, $\alpha \in L^2_{\mathbb{F}}([-d,T],\mathbb{R})$ as $\alpha \in \mathcal{A}$) combined with the monotone convergence theorem on the right term yields, as $k \to \infty$

$$\mathbb{E}\left[V_T^\alpha\right] = \mathbb{E}[(X_T^\alpha)^2] = V_0^\alpha + \mathbb{E}\left[\int_0^T P_{\hat{2}\hat{2}}(s,0)(\alpha_s - \mathcal{T}(\alpha)_s)^2 ds\right].$$

Note that here we used the assumption $P_{\hat{2}\hat{2}}(t,0) \ge 0$ on [0,T]. Since $P_{\hat{2}\hat{2}}$ is non-negative, we obtain that the optimal strategy is given by α^* and that the optimal value equals (4.3.2). □

Proposition 4.3.1. *Assume that there exists a bounded 4-uplets P solution to* (4.2.9)-(4.2.10)-(4.2.11) *in the sense of Definition 4.2.1. Then* (4.3.1) *defines an admissible control.*

Proof. Let $\gamma \in L^2([-d,0],\mathbb{R})$. To prove the claim, note that it suffices to show that the equation

$$\begin{cases} \alpha_t &= \frac{-1_{t\leq T-d}}{P_{\hat{22}}(t,0)}\left\{P_{12}(t,0)\left(x+\int_0^t \alpha_{s-d}(bds+\sigma dW_s)\right)+\int_{t-d}^t \alpha_s P_{22}(t,0,s-t)ds\right\}, \\ \alpha_s &= \gamma_s, \qquad s\in[-d,0], \end{cases} \tag{4.3.4}$$

admits a solution in $L^2_{\mathbb{F}}([0,T],\mathbb{R})$ and that the process X, then defined as

$$X_t = x+\int_0^t \alpha_{s-d}(bds+\sigma dW_s), \qquad t\leq T, \tag{4.3.5}$$

satisfies $\mathbb{E}\left[\sup_{t\leq T}|X_t|^2\right]<\infty$. To prove the first point, consider the linear operator ϕ on $L^2_{\mathbb{F}}([0,T],\mathbb{R})$ defined as, for any $a\in L^2_{\mathbb{F}}([0,T],\mathbb{R})$

$$\phi(a)_t = \frac{-1}{P_{\hat{22}}(t,0)}\left\{P_{12}(t,0)\left(x+\int_0^t \hat{a}_{s-d}(bds+\sigma dW_s)\right)+\int_{t-d}^t \hat{a}_s P_{22}(t,0,s-t)ds\right\},$$

where $\hat{a}_t = 1_{t\leq 0}\gamma_t + 1_{t>0}a_t$. For $\lambda\leq 0$, we endow $L^2_{\mathbb{F}}([0,T],\mathbb{R})$ with the norm $\|a\|_{2,\lambda} = \sqrt{\int_0^T e^{-\lambda s}|a_s|^2 ds}$. Then, for any $a,a'\in L^2_{\mathbb{F}}([0,T],\mathbb{R})$, we have

$$\begin{aligned} \|\phi(a)-\phi(a')\|^2_{2,\lambda} =& \mathbb{E}\left[\int_0^T e^{-\lambda s}|\phi(a)_s-\phi(a')_s|^2 ds\right] \\ \leq& 2(\mathbf{I}+\mathbf{II}). \end{aligned} \tag{4.3.6}$$

An application of Jensen's inequality on the normalised measure $\frac{dr}{s}$ on $[0,s]$, combined with $\frac{1-e^{-\lambda(T-r)}}{\lambda}\leq(1\vee T)(1\wedge\lambda^{-1})$, and the Burkholder-Davis-Gundy inequality lead to

$$\begin{aligned} \mathbf{I} \leq& \mathbb{E}\left[\int_0^T e^{-\lambda s}\left|\frac{P_{12}(s,0)}{P_{\hat{22}}(s,0)}\int_0^s(\hat{a}_{r-d}-\hat{a}'_{r-d})(bdr+\sigma dW_r)\right|^2 ds\right] \\ \leq& \sup_{s\leq T}\left|\frac{P_{12}(s,0)}{P_{\hat{22}}(s,0)}\right|^2\left\{\mathbb{E}\left[\int_0^T e^{-\lambda s}\left(\int_0^s(\hat{a}_{r-d}-\hat{a}'_{r-d})bdr\right)^2 ds\right]\right. \\ &\left.+\mathbb{E}\left[\int_0^T e^{-\lambda s}\left(\int_0^s(\hat{a}_{r-d}-\hat{a}'_{r-d})\sigma dWr\right)^2 ds\right]\right\} \\ \leq& c(1\wedge\lambda^{-1})\mathbb{E}\left[\int_0^T e^{-\lambda r}(\hat{a}_{r-d}-\hat{a}'_{r-d})^2 dr\right], \end{aligned}$$

where $c>0$ depends only on b,σ,T and $\sup_{s\leq T}\left|\frac{P_{12}(s,0)}{P_{\hat{22}}(s,0)}\right|$. Furthermore, we have

$$\begin{aligned} \mathbf{II} \leq& \mathbb{E}\left[\int_0^T e^{-\lambda s}\left|\frac{1}{P_{\hat{22}}(s,0)}\int_{s-d}^s(\hat{a}_r-\hat{a}'_r)P_{22}(r,0,s-r)dr\right|^2 ds\right] \\ \leq& \sup_{\substack{s\leq T\\ r\in[-d,0]}}\left|\frac{P_{22}(s,0,r)}{P_{\hat{22}}(s,0)}\right|^2\mathbb{E}\left[\int_0^T e^{-\lambda s}\left|\int_0^s(\hat{a}_r-\hat{a}'_r)dr\right|^2 ds\right] \\ \leq& \hat{c}(1\wedge\lambda^{-1})\mathbb{E}\left[\int_0^T e^{-\lambda r}(a_r-\hat{a}'_r)^2 dr\right], \end{aligned}$$

where $\hat{c} > 0$ depends only on T and $\sup_{\substack{s\leq T\\ r\in[-d,0]}} \left|\frac{P_{22}(s,0,r)}{P_{\hat{22}}(s,0)}\right|$. Consequently, (4.3.6) reduces to

$$\|\phi(a) - \phi(a')\|_{2,\lambda}^2 \leq (c + \hat{c})(1 \wedge \lambda^{-1})\|a - a'\|_{2,\lambda}^2.$$

As a result, for λ large enough, ϕ is a contraction on the Banach space $\left(L^2_{\mathbb{F}}([0,T],\mathbb{R}), \|.\|_{2,\lambda}\right)$, thus proving the existence of $\alpha \in L^2_{\mathbb{F}}([0,T],\mathbb{R})$ solution to (4.3.4). Finally, an application of Burkholder-Davis-Gundy's inequality to (4.3.5) yields $\mathbb{E}\left[\sup_{t\leq T}|X_t|^2\right]$. The proof is thus complete. □

Next, we give a sufficient condition for the existence of $P = (P_{11}, P_{12}, P_{\hat{22}}, P_{22})$ in terms of b, σ, d and T. Let $a = (a_n)_{n\geq 1}$ be the sequence defined as

$$\begin{cases} a_0 & = 1, \\ a_{n+1} & = a_n - \frac{d}{a_n}\left(\frac{b}{\sigma}\right)^2, \qquad n \geq 0. \end{cases} \tag{4.3.7}$$

Let us denote $\mathcal{N} : (d, b, \sigma) \mapsto \inf\{n \geq 1 : a_n > 0 \text{ and } a_{n+1} \leq 0\}$. Clearly, $\mathcal{N}$ is a well defined finite valued function on $\mathbb{R}^3$ whose image is not restricted to $\{0\}$.

Proposition 4.3.2. *Assume $\mathcal{N}(d,b,\sigma) \geq 2$ and $T < \mathcal{N}(d,b,\sigma)d$. Then (4.2.9)-(4.2.10)-(4.2.11) has a unique solution in the sense of definition 4.2.1 on $[0,T]$ with $0 < a_{\mathcal{N}(d,b,\sigma)} \leq P_{11}(0) < 1$.*

Proof. See appendix 4.A. □

Remark 4.3.1. *Note that when $d = 0$, the sufficient condition above reduces to $\sigma \neq 0$.*

Let us give some intuition as of why the delay feature induces the condition on the coefficients described above to ensure existence. We focus on the first slice $[T - 2d, T - d] \times [-d, 0]$ of the domain, where the solution P is not trivial.

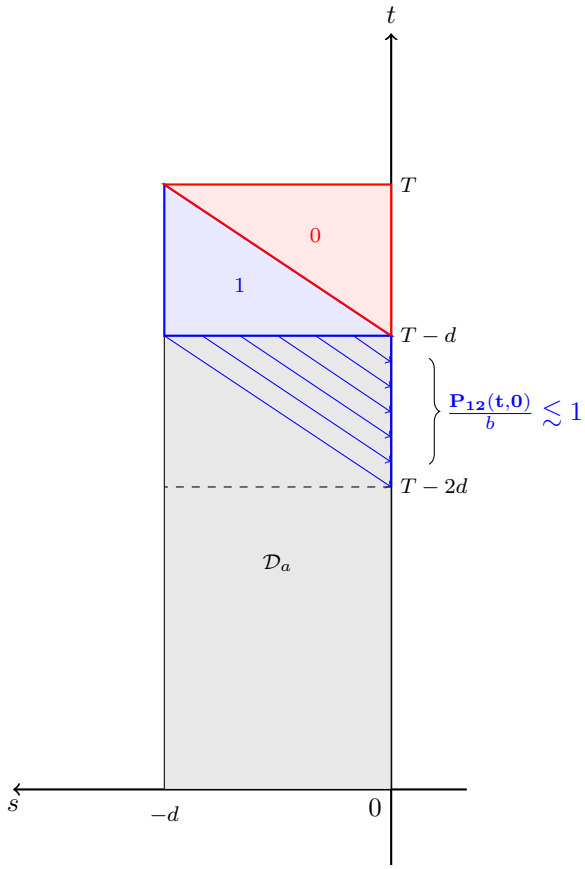
First, note that since $P_{\hat{22}}$ is a transport of P_{11} which takes the form $P_{\hat{22}}(t,s) = \sigma^2 P_{11}(t+s+d)1_{t+s+d\leq T} = \sigma^2 1_{t+s+d\leq T}$. On this slice, the kernel P_{11} can be expressed in the following integral form

$$P_{11}(t) = 1 - \left(\frac{b}{\sigma}\right)^2 \int_t^{T-d} \left(\frac{\mathbf{P_{12}(s,0)}}{\mathbf{b}}\right)^2 ds, \tag{4.3.8}$$

for $t \in [T-2d, T-d]$. Looking at $P_{12}(\cdot,0)$, we have

$$P_{12}(t,0) = b - \sigma^{-2} \int_t^{T-d} P_{12}(x,0) \times \underbrace{P_{22}(x,t-x,0)}_{\lesssim b^2} dx,$$

see also (4.A.4). On the right, we represent the value of the normalized kernel P_{12}/b in the different areas of the domain $[T-2d, T] \times [-d, 0]$. If we visualize the evolution of the normalized kernel P_{12}/b in a backward way on the slice $[T-2d, T-d]$, we see that this term is equal to a transport of its value on the boundary $\frac{P_{12}(T-d,s)}{b} = 1$, represented by the blue arrows, minus the integral of a positive source term which is independent of $t \in [T-2d, T-d] \mapsto P_{11}(t)$.



Consequently, the delay $d > 0$ makes the integral term in (4.3.8) independent of $t \in [T-2d, T-d] \mapsto P_{11}(t)$ and of the order of $d\left(\frac{b}{\sigma}\right)^2$. If this quantity is too large, the kernel P_{11} can then reach negative values, thus making $P_{\hat{22}}$ negative on the next slice $[T-3d, T-2d]$ and therefore preventing the system (4.2.9)-(4.2.10)-(4.2.11) from having a solution. Repeating this argument from slice to slice of size d in a backward manner induces the aforementioned sufficient condition. Note that these arguments break down when $d = 0$. These arguments are precisely developed in Appendix 4.A.

4.4 Deep learning scheme

4.4.1 A quick reminder of PINNs and Deep Galerkin method for PDEs

In order to solve (4.2.9)-(4.2.10)-(4.2.11), we will make use of neural networks in the spirit of the emerging Physics Informed Neural Networks (PINNs) and Deep Galerkin literatures, see [SS18] and [RPK19] to name just a few. We first recall some of the main ideas. Assume we have a nonlinear partial differential equation of the form

$$\begin{aligned} \partial_t u + \mathcal{N}(u) &= 0, \qquad \text{on } \Omega, \\ u &= g, \qquad \text{on } \partial\Omega, \end{aligned} \tag{4.4.1}$$

where $\mathcal{N}$ is a nonlinear operator, Ω a bounded open subset and g a function on the boundary of the domain. The main idea is to approximate the solution u to (4.4.1) by

a deep neural network. Let us call $t \mapsto u(t, \Theta)$ this network, where Θ and t denote respectively its parameters and a generic element of $\Omega \cup \partial\Omega$. The goal is to find a Θ so that $t \mapsto u(t, \Theta)$ satisfies (4.4.1). To do so, the idea is to proceed by minimizing the mean square error loss

$$\begin{aligned}
\mathcal{L}(\Theta, \mathcal{T}) &= \mathcal{L}_u(\Theta, \mathcal{T}) + \mathcal{L}_f(\Theta, \mathcal{T}), \\
\mathcal{L}_u(\Theta, \mathcal{T}) &= \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} |(\partial_t + \mathcal{N}) u(t, \Theta)|^2 1_{t \in \Omega}, \\
\mathcal{L}_f(\Theta, \mathcal{T}) &= \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} |u(t, \Theta) - g(t)|^2 1_{t \in \partial\Omega},
\end{aligned}$$

where $\mathcal{L}_f$ is the loss associated with the initial and boundary constraints on $\partial\Omega$, $\mathcal{L}_u$ the loss associated to the PDE constraint $\partial_t u + \mathcal{N}(u) = 0$ on Ω and $\mathcal{T}$ a random subset of $\partial\Omega \cup \Omega$.

4.4.2 A tailor-made algorithm for Riccati partial differential equations

Although (4.2.9)-(4.2.10)-(4.2.11) naturally fits the framework of PINNs, we make use of the structure exhibited on the operator P to build a tailor-made algorithm to approximate the system of Riccati transport PDEs (4.2.9)-(4.2.10)-(4.2.11).

Step 1: We define one neural network for each kernel $P_{11}, P_{12}, P_{\hat{2}\hat{2}}, P_{22}$, as described in Figure 4.2. Note that usually a unique neural network is used as a surrogate to the function that is to be approximated.

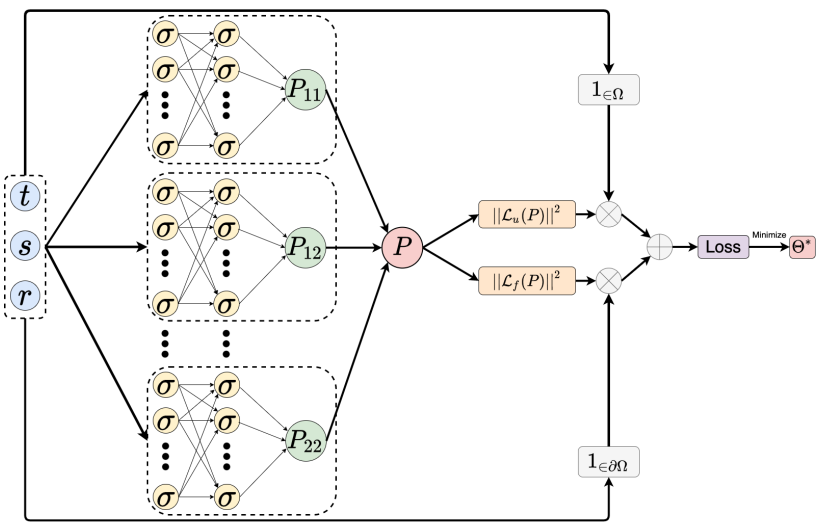


Figure 4.2: Structure of the model used to solve (4.2.9)-(4.2.10)-(4.2.11).

Step 2: We build specific loss functionals for each of our neural networks. To illustrate this, consider for instance the constraint imposed on the derivative of $t \mapsto P_{11}(t)$:

$$\dot{P}_{11}(t) = \frac{P_{12}(t, 0)^2}{P_{\hat{2}\hat{2}}(t, 0)}, \qquad t \in [0, T]. \tag{4.4.2}$$

Note here that, as in the previous sections, we use the convention $0^2/0 = 0$. As a result

(4.4.2) can be rewritten as

$$\dot{P}_{11}(t) = \frac{P_{12}(t,0)^2}{P_{\hat{2}2}(t,0)}, \qquad t \leq T-d,$$
$$\dot{P}_{11}(t) = 0, \qquad t > T-d.$$

Thus, a natural contribution to the total loss function to enforce (4.4.2) would be

$$\mathcal{L}_{11}(\Theta_{11},\Theta_{12},\Theta_{\hat{2}2},\mathcal{T}) = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\left(\partial_t P_{11}(\Theta_{11},t) - \frac{P_{12}(\Theta_{12},t,0)^2}{P_{\hat{2}2}(\Theta_{\hat{2}2},t,0)}\right)^2, \qquad \mathcal{T}\subset[0,T],$$

and the natural gradient descent step associated to the constraint (4.4.2) would be

$$\Theta_i \leftarrow \Theta_i - \eta\nabla_{\Theta_i}\mathcal{L}_{11}(\Theta_{11},\Theta_{12},\Theta_{\hat{2}2},\mathcal{T}), \qquad i\in\{11,12,\hat{2}2\},$$

Consequently, the constraint (4.4.2) a priori entails the updating of P_{11}, P_{12} and $P_{\hat{2}2}$. In particular, it requires to compute the gradient of $(\Theta_{12},\Theta_{\hat{2}2}) \mapsto \sum_{t\in\mathcal{T}}\frac{P_{12}(\Theta_{12},t,0)^2}{P_{\hat{2}2}(\Theta_{\hat{2}2},t,0)}$ which is expected to be highly unstable as $t,s \mapsto P_{\hat{2}2}(t,s)$ vanishes for $t+s \geq T-d$. To mitigate this issue, the term $t \mapsto \frac{P_{12}(\Theta_{12},t,0)^2}{P_{\hat{2}2}(\Theta_{\hat{2}2},t,0)}$ is considered as an exogenous source term for P_{11} which is fixed when we train P_{11}.

$$\dot{P}_{11}(t) = \underbrace{\frac{P_{12}(t,0)^2}{P_{\hat{2}2}(t,0)}}_{\text{Seen as a fixed exogenous source term when } P_{11} \text{ is trained}}, t\in[0,T].$$

To implement this idea, a second set of neural networks $(\tilde{P}_k(\tilde{\Theta}_k))_{k\in\{11,12,\hat{2}2,22\}}$ is initialized with $\tilde{\Theta}_k = \Theta_k$ for $k\in\{11,12,\hat{2}2,22\}$ at initialization. These additional networks are then used as surrogates to the right-hand side source terms and will not be used for the computation of the gradients of the losses. They will only be updated at the end of each batch training. Consequently, the gradient descent scheme implemented for each batch $\mathcal{T}$ is the following:

$$\text{Step 1}\begin{cases} \Theta_{11} & \leftarrow \Theta_{11} - \eta\nabla_{\Theta_{11}}\left(\mathcal{L}^r_{11}(\Theta_{11},\tilde{\Theta}_{12},\tilde{\Theta}_{\hat{2}2},\mathcal{T}) + \mathcal{L}^f_1(\Theta_{11},\mathcal{T})\right) \\ \Theta_{12} & \leftarrow \Theta_{12} - \eta\nabla_{\Theta_{12}}\left(\mathcal{L}^r_{12}(\Theta_{12},\tilde{\Theta}_{\hat{2}2},\tilde{\Theta}_{22},\mathcal{T}) + \mathcal{L}^b_{12}(\tilde{\Theta}_{11},\Theta_{12},\mathcal{T}) + \mathcal{L}^f_{12}(\Theta_{12},\mathcal{T})\right) \\ \Theta_{\hat{2}2} & \leftarrow \Theta_{\hat{2}2} - \eta\nabla_{\Theta_{\hat{2}2}}\left(\mathcal{L}^r_{\hat{2}2}(\Theta_{\hat{2}2},\mathcal{T}) + \mathcal{L}^b_{\hat{2}2}(\tilde{\Theta}_{11},\Theta_{\hat{2}2},\mathcal{T}) + \mathcal{L}^f_{\hat{2}2}(\Theta_{\hat{2}2},\mathcal{T})\right) \\ \Theta_{22} & \leftarrow \Theta_{22} - \eta\nabla_{\Theta_{22}}\left(\mathcal{L}^r_4(\Theta_{\hat{2}2},\Theta_{22},\tilde{\Theta}_{22},\mathcal{T}) + \mathcal{L}^b_{22}(\Theta_{12},\Theta_{22},\tilde{\Theta}_{22},\mathcal{T}) + \mathcal{L}^f_{22}(\Theta_{22},\mathcal{T})\right), \end{cases}$$
$$\text{Step 2}\left\{\tilde{\Theta}_k \leftarrow \Theta_k, \qquad k\in\{11,12,\hat{2}2,22\},\right. \tag{4.4.3}$$

where the $\mathcal{L}^r_i$'s stand for the *residual loss*, the $\mathcal{L}^b_i$'s stand for the *boundary loss* and the $\mathcal{L}^f_i$'s stand for the *final loss*. The precise definitions of the losses are given below.

For each neural network $P_k(\Theta_k)$, $k\in\{11,12,\hat{2}2,22\}$, a *follower network* is initialized $\tilde{P}_k(\tilde{\Theta}_k)$, $k\in\{11,12,\hat{2}2,22\}$. These follower networks serve as surrogate for the source terms in (4.2.9)-(4.2.10)-(4.2.11). They condition the loss functionals that are used to train the P_k's and are updated at the end of each batch as described in Algorithm 3.

Losses of P_{11}:

$$\mathcal{L}^r_{11}(\Theta_{11},\tilde{\Theta}_{12},\tilde{\Theta}_{\hat{2}2}) = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\left(\partial_t P_{11}(t,\Theta_{11}) - \frac{\tilde{P}_{12}(t,0,\tilde{\Theta}_{12})^2}{\tilde{P}_{\hat{2}2}(t,0,\tilde{\Theta}_{\hat{2}2})}\right)^2,$$
$$\mathcal{L}^f_{11}(\Theta_{11},\mathcal{T}) = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\left((P_{11}(t,\Theta_{11})-1)^2\,1_{t=T}\right).$$

Losses of P_{12}:

$$\mathcal{L}^r_{12}(\Theta_{12}, \tilde{\Theta}_{\hat{2}\hat{2}}, \tilde{\Theta}_{22}, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{t,s\in\mathcal{T}} \left((\partial_t - \partial_s) P_{12}(t,s,\Theta_{12}) - \frac{\tilde{P}_{12}(t,0,\tilde{\Theta}_{12})\tilde{P}_{22}(t,s,0,\tilde{\Theta}_{22})}{\tilde{P}_{\hat{2}\hat{2}}(t,0,\tilde{\Theta}_{\hat{2}\hat{2}})} \right)^2,$$

$$\mathcal{L}^b_{12}(\tilde{\Theta}_{11}, \Theta_{12}, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{t,s\in\mathcal{T}} \left(P_{12}(t,s,\Theta_{12}) - b\tilde{P}_{11}(t,\tilde{\Theta}_{11}) \right)^2 1_{s=0,t\neq 0},$$

$$\mathcal{L}^f_{12}(\Theta_{12}, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{t,s\in\mathcal{T}} \left(P_{12}(t,s,\Theta_{12})^2 1_{t=T} \right).$$

Losses of $P_{\hat{2}\hat{2}}$:

$$\mathcal{L}^r_{\hat{2}\hat{2}}(\Theta_{\hat{2}\hat{2}}, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{t,s\in\mathcal{T}} \left((\partial_t - \partial_s) P_{\hat{2}\hat{2}}(t,s,\Theta_{\hat{2}\hat{2}}) \right)^2,$$

$$\mathcal{L}^b_{\hat{2}\hat{2}}(\tilde{\Theta}_{11}, \Theta_{\hat{2}\hat{2}}, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{t,s\in\mathcal{T}} \left(P_{\hat{2}\hat{2}}(t,s,\Theta_{\hat{2}\hat{2}}) - \sigma^2\tilde{P}_{11}(t,\tilde{\Theta}_{11}) \right)^2 1_{s=0,t\neq T},$$

$$\mathcal{L}^f_{\hat{2}\hat{2}}(\Theta_{\hat{2}\hat{2}}, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{t,s\in\mathcal{T}} \left(P_{\hat{2}\hat{2}}(t,s,\Theta_{\hat{2}\hat{2}})^2 1_{t=T} \right).$$

Losses of P_{22}:

$$\mathcal{L}^r_{22}(\Theta_{22}, \tilde{\Theta}_{22}, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{t,s,r\in\mathcal{T}} \left((\partial_t - \partial_s - \partial_r) P_4(t,s,r,\Theta_{22}) - \frac{\tilde{P}_{22}(t,0,r,\tilde{\Theta}_{22})\tilde{P}_{22}(t,s,0,\tilde{\Theta}_{22})}{\tilde{P}_{\hat{2}\hat{2}}(t,0,\tilde{\Theta}_{\hat{2}\hat{2}})} \right)^2,$$

$$\begin{aligned}\mathcal{L}^b_{22}(\tilde{\Theta}_{12}, \Theta_{22}, \mathcal{T}) &= \frac{1}{|\mathcal{T}|} \sum_{t,s,r\in\mathcal{T}} \left(P_{22}(t,s,r,\Theta_{22}) - b\tilde{P}_{22}(t,s,\tilde{\Theta}_{12}) \right)^2 1_{r=0,t\neq T} \\ &\quad + \frac{1}{|\mathcal{T}|} \sum_{t,s,r\in\mathcal{T}} \left(P_{22}(t,s,r,\Theta_{22}) - b\tilde{P}_{22}(t,r,\tilde{\Theta}_{12}) \right)^2 1_{s=0,t\neq T},\end{aligned}$$

$$\mathcal{L}^f_{22}(\Theta_{22}, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{t,s,r\in\mathcal{T}} \left(P_{22}(t,s,r,\Theta_{22})^2 1_{t=T} \right).$$

4.5 Applications to mean-variance portfolio selection with execution delay

4.5.1 One asset with delay

We now aim at solving the celebrated example of mean-variance portfolio selection, see [Mar52], with execution delay in the spirit of the problem of hedging of European options with execution delay presented in [FF14]. We present here the settings. Let us consider a standard Black-Scholes financial market, composed of a risk-less asset with zero interest rate

$$S^0_t = 1, \quad t \in [0,T],$$

and a risky asset with dynamics

$$dS_t = S_t \left\{ (\sigma\lambda)\, dt + \sigma dW_t \right\}, \quad t \in [0,T],$$

Algorithm 3: Deep learning scheme to solve (4.2.9)-(4.2.10)-(4.2.11)

Initialize:
the learning rate η, the neural networks $P(\Theta) = (P_k(\Theta_k))_k$ and $\tilde{P}(\tilde{\Theta}) = (\tilde{P}_k(\tilde{\Theta}_k))_k$. ;
;

Copy the weights $\tilde{\Theta}_k \leftarrow \Theta_k$, $k \in \{11, 12, \hat{22}, 22\}$;
;

For each batch:
Randomly sample $\mathcal{T} \subset \partial\Omega \cup \Omega$;
Compute the gradient $\nabla_\Theta \mathcal{L}(\Theta, \tilde{\Theta}, \mathcal{T})$ as in (4.4.3);
Update $\Theta \leftarrow \Theta - \eta \nabla_\Theta \mathcal{L}(\Theta, \tilde{\Theta}, \mathcal{T})$;
Update $\tilde{\Theta} \leftarrow \Theta$;
Return: The set of optimized parameters Θ^*.

where λ and σ are constants representing respectively the risk premium and the volatility of the risky asset. At every time $t \in [0, T]$ the investor chooses, based on the information $\mathcal{F}_t$, to allocate the amount of money $\alpha_t \in \mathbb{R}$ into the risky asset. However, due to execution delays this order will be executed at time $t + d$. Set N_t^α (respectively N_t^0) the number of risky (respectively risk-less) shares held at time t. Then, given an investment strategy $\alpha \in \mathcal{A}$, the value $(X_t^\alpha)_{t\in[0,T]}$ of the portfolio, that we suppose self-financing, follows the dynamics

$$\begin{aligned} dX_t^\alpha &= N_t^\alpha dS_t + \underbrace{(dN_t^\alpha)S_t + (dN_t^0)S_t^0}_{=0\ ,\ \text{self-financing}} \\ &= \underbrace{N_t S_t}_{\alpha_{t-d}} \{(\sigma\lambda)dt + \sigma dW_t\}. \end{aligned}$$

Consequently, the controlled state equation of the portfolio's value is of the form

$$\begin{cases} dX_t^\alpha = \alpha_{t-d}\left((\sigma\lambda)\,dt + \sigma dW_t\right), & t \in [0, T], \\ X_0 = x_0, \quad \alpha_s = \gamma_s, & \forall s \in [-d, 0], \end{cases}$$

with $x_0 > 0$ and $\gamma \in L^2([-d, 0], \mathbb{R})$. The Mean-Variance portfolio selection problem in continuous-time consists in solving the following constrained problem

$$\begin{cases} \min_{\alpha\in\mathcal{A}} \mathrm{Var}(X_T^\alpha) \\ \text{s.t. } \mathbb{E}[X_T^\alpha] = c. \end{cases} \tag{4.5.1}$$

It is well-known that problem (4.5.1) is equivalent to the following **max-min problem**, see [Pha09, Section 6.6.2]

$$\max_{\eta\in\mathbb{R}} \min_{\alpha\in\mathcal{A}} \mathbb{E}\left[(X_T^\alpha - (c - \eta))^2\right] - \eta^2. \tag{4.5.2}$$

Thus, solving problem (4.5.1) involves two steps. First, the internal minimization problem in terms of the Lagrange multiplier η has to be solved. Second, the optimal value of η for the external maximization problem has to be determined. Thus, with $\xi = c - \eta$, we first define the **Inner optimization problem:**

$$\min_{\alpha\in\mathcal{A}} \mathbb{E}\left[\left(X_T^\alpha - \xi\right)^2\right]. \tag{4.5.3}$$

Note that, by setting $\tilde{X}^\alpha = X^\alpha - \xi$, the inner problem (4.5.3) fits into the delayed LQ control problem analysed in Section 4.3. We first solve the inner optimization problem (4.5.3) in the following lemma.

Lemma 4.5.1. *Fix $\eta \in \mathbb{R}$ and $\xi = c - \eta$. Assume $T < d\mathcal{N}(d, (\sigma\lambda), \sigma)$ and define $\alpha^*(\xi)$ as the investment strategy*

$$\alpha_t^*(\xi) = \frac{-1_{t\leq T-d}}{P_{\hat{22}}(t,0)}\left\{(X_t^{\alpha^*} - \xi)P_{12}(t,0) + \int_{t-d}^t \alpha_s^*(\xi)P_{22}(t,0,s-t)ds\right\}, \tag{4.5.4}$$

where P denotes the solution to (4.2.9)-(4.2.10)-(4.2.11) in the sense of Definition 4.2.1. Then, the inner minimization problem (4.5.3) admits $\alpha^*(\xi)$ as an admissible optimal feedback strategy and the optimal value is

$$V_0(\xi) = P_{11}(0)(x_0 - \xi)^2 + R(x_0 - \xi, \gamma), \tag{4.5.5}$$

where $R(\gamma)$ denotes the cost associated to the initial investment strategy γ on $[-d, 0]$

$$R(x,\gamma) = 2x\int_{-d}^0 \gamma_s P_{12}(0,s)ds + \int_{-d}^0 \gamma_s^2 P_{\hat{22}}(0,s)ds + \int_{[-d,0]^2} \gamma_s\gamma_u P_{22}(0,s,r)dsdr.$$

Proof. First, note that Proposition 4.3.2 yields the existence and uniqueness of a solution P to (4.2.9)-(4.2.10)-(4.2.11). Furthermore the admissibility of $\alpha^*(\xi)$ results from Proposition 4.3.1. For any $\alpha \in \mathcal{A}$, define $\tilde{X}_t^\alpha = X_t^\alpha - \xi$. Then, by Itô's formula we have

$$\begin{cases} d\tilde{X}_t^\alpha = \alpha_{t-d}\left((\sigma\lambda)dt + \sigma dW_t\right), & t \in [0,T], \\ \tilde{X}_0 = x_0 - \xi, \quad \alpha_s = \gamma_s, & \forall s \in [-d, 0]. \end{cases}$$

As a result, $\tilde{X}^\alpha$ and X^α have the same dynamics and $\tilde{X}_T^\alpha = X_T^\alpha - \xi$ so that problem (4.5.3) can be alternatively written as

$$\min_{\alpha\in\mathcal{A}} \mathbb{E}\left[\left(\tilde{X}_T^\alpha\right)^2\right].$$

Thus, the optimality of $\alpha^*(\xi)$ and the value (4.5.5) are immediately given by the verification theorem 4.3.1. □

Theorem 4.5.1. *Assume $T < d\mathcal{N}(d, (\sigma\lambda), \sigma)$. Then, the optimal investment strategy for the maximization problem (4.5.1) is given by $a^*(\xi^*)$ defined in (4.5.4) with $\xi^* = c - \eta^*$ and*

$$\begin{aligned} \eta^* &= \frac{K(\gamma) + P_{11}(0)(x_0 - c)}{1 - P_{11}(0)}, \\ K(\gamma) &= \int_{-d}^0 \gamma_s P_{12}(0,s)ds. \end{aligned} \tag{4.5.6}$$

Furthermore, the value of (4.5.1) is

$$\begin{aligned} \mathrm{Var}(X_T^{\alpha^*}) = &\frac{P_{11}(0)}{1 - P_{11}(0)}\left(x_0 - c + K(\gamma)\right)^2 \\ &+ \int_{-d}^0 \gamma_s^2 P_{\hat{22}}(0,s)ds + \int_{[-d,0]^2} \gamma_s\gamma_u P_{22}(0,s,r)dsdr. \end{aligned} \tag{4.5.7}$$

Proof. As $T < d\mathcal{N}(d, (\sigma\lambda), \sigma)$, Proposition 4.3.2 ensures the existence and uniqueness of a solution P to (4.2.9)-(4.2.10)-(4.2.11). From Lemma 4.5.1 and (4.5.2), we have that the max-min problem (4.5.2), which is equivalent to (4.5.1), reduces to

$$\begin{aligned}&\max_{\eta\in\mathbb{R}}\left\{V_0(c-\eta)-\eta^2\right\}\\&=\max_{\eta\in\mathbb{R}}\left\{P_{11}(0)\left(x_0-(c-\eta)\right)^2+R(x_0-(c-\eta),\gamma)-\eta^2\right\}.\end{aligned}$$

Furthermore, since $T < d\mathcal{N}(d, (\sigma\lambda), \sigma)$, Proposition 4.3.2 ensures $0 < P_{11}(0) < 1$ so that the maximization problem is strictly concave. Consequently, η^* given by (4.5.6) is the optimal parameter. Setting $\xi^* = c - \eta^*$ in (4.5.4) and (4.5.5) results in the optimality of $\alpha^*(\xi^*)$, and the optimal value (4.5.7) for the mean-variance problem (4.5.1). □

Remark 4.5.1. *In the absence of pre-investment strategy, $\gamma = 0$, we recover the usual form of the efficient frontier formula*

$$\mathrm{Var}(X_T^{\alpha^*}) = \frac{P_{11}(0)}{1-P_{11}(0)}\left(x_0-c\right)^2.$$

Our observations from the simulations are the following.

Efficient frontier: In Figure 4.3, we plot the efficient frontier for different delays d. Note that the greater the delay, the greater is the variance. This could have been foreseen by observing that, when the initial control is set to 0, i.e. $\gamma = 0_{L^2}$, the value function takes the form $V(x, 0_{L^2}) = P_{11}(0)x^2$, see (4.3.2). As the value function is clearly an increasing function of the delay, the terminal variance of the portfolio $\mathrm{Var}(X_T^{\alpha^*}) = \frac{P_{11}(0)}{1-P_{11}(0)}\left(x_0-c\right)^2$ is also an increasing function of the delay.

Destabilization effect : In Figure 4.4, we plot different scenarios of portfolio allocation. We observe a destabilization effect and a supplement of volatility induced by the delay feature. We also note the tendency to invest more aggressively for greater values of the delay, as the investor has less time to ensure that the promised yield is achieved. We propose the following interpretation: In the classical setting, where $d = 0$, if Y^* denotes the optimal portfolio value process, the optimal investment strategy is of the form $\alpha_t^* = -\frac{b}{\sigma^2}(Y^{\alpha^*} - \mu^*)$ for a certain constant $\mu^* > c$. It can then easily be shown that $Y^* \leq \mu^*$. Thus, the optimal strategy consists in aiming from below at a fixed target μ^*. When $d > 0$, the optimal control is composed of an additional *inertial term*

$$\alpha_t^*(\xi^*) = \frac{-1_{t\leq T-d}}{P_{\hat{2}\hat{2}}(t,0)}\Bigg\{\underbrace{(X_t^{\alpha^*}-\xi^*)P_{12}(t,0)}_{\text{Usual mean-reverting term}}+\underbrace{\int_{t-d}^{t}\alpha_s^*(\xi)P_{22}(t,0,s-t)ds}_{\text{New inertial term}}\Bigg\},$$

so that, contrary to the case where $d = 0$, the optimal control does not cancel when the target ξ^* is attained. Also, note that at every time t, the agent doesn't have any control on the near future from t to $t + d$.

Kernel P : In Figure 4.5, we plot the kernels $P_{11}, P_{12}, P_{\hat{2}\hat{2}}$, and P_{22}. Note the discontinuity between $\mathcal{D}_b$ and $\mathcal{D}_c$ also described in Figure 4.1.

4.5.2 One asset with delay and one without

To further explore the effect of the delay on the control, we now study a toy example where the investor has two investment opportunities, one with a delayed execution and

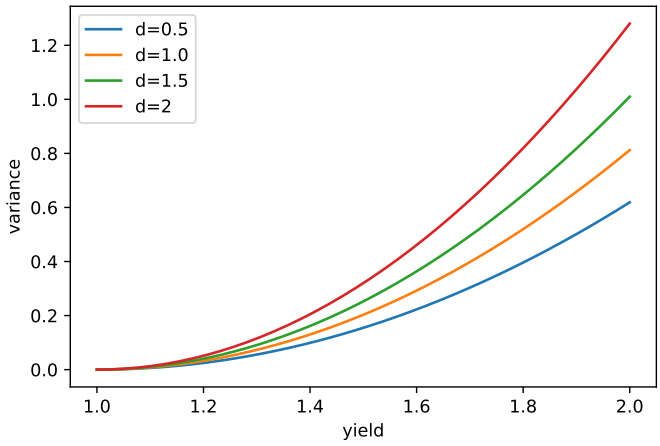


Figure 4.3: Efficient frontier with $\sigma = 1$, $\lambda = 0.5$, $T = 5$ and $\gamma \equiv 0$.

one without. More precisely, consider the following portfolio dynamic

$$\begin{cases} dX_t^{(\alpha,\beta)} = \alpha_t \left\{(\sigma_1\lambda_1)dt + \sigma_1 dW_t^1\right\} + \beta_{t-d}\left\{(\sigma_2\lambda_2)dt + \sigma_2 dW_t^2\right\}, \quad t \in [0,T], \\ X_0 = x_0, \quad \beta_s = \gamma_s, \quad s \in [-d,0], \\ \langle W^1, W^2\rangle_t = \rho t, \end{cases}$$

where $x_0 > 0$ and $\gamma \in L^2([-d,0],\mathbb{R})$, together with the same optimization objective (4.5.1) as before. Here, α_t and β_t correspond respectively to the amounts of money the investor decides to invest at time t in the undelayed and the delayed risky assets. The constants λ_i and σ_i represent respectively the risk premium and the volatility of the risky asset i. Following the heuristic approach of Section 4.2, we define the following set of Riccati-PDEs on $[0,T] \times [-d,0]^2$

$$\begin{aligned} &\dot{P}_{11}(t) = \lambda_1^2 P_{11}(t) + \frac{P_{12}(t,0)^2}{P_{\hat{2}\hat{2}}(t,0)}, \\ &(\partial_t - \partial_s)(P_{12})(t,s) = \lambda_1^2 P_{12}(t,s) + \frac{P_{12}(t,0)P_{22}(t,s,0)}{P_{\hat{2}\hat{2}}(t,0)}, \\ &(\partial_t - \partial_s)(P_{\hat{2}\hat{2}})(t,s) = 0, \\ &(\partial_t - \partial_s - \partial_r)(P_{22})(t,s,r) = \lambda_1^2 \frac{P_{12}(t,s)P_{12}(t,r)}{P_{11}(t)} + \frac{P_{22}(t,s,0)P_{22}(t,0,r)}{P_{\hat{2}\hat{2}}(t,0)}, \end{aligned} \tag{4.5.8}$$

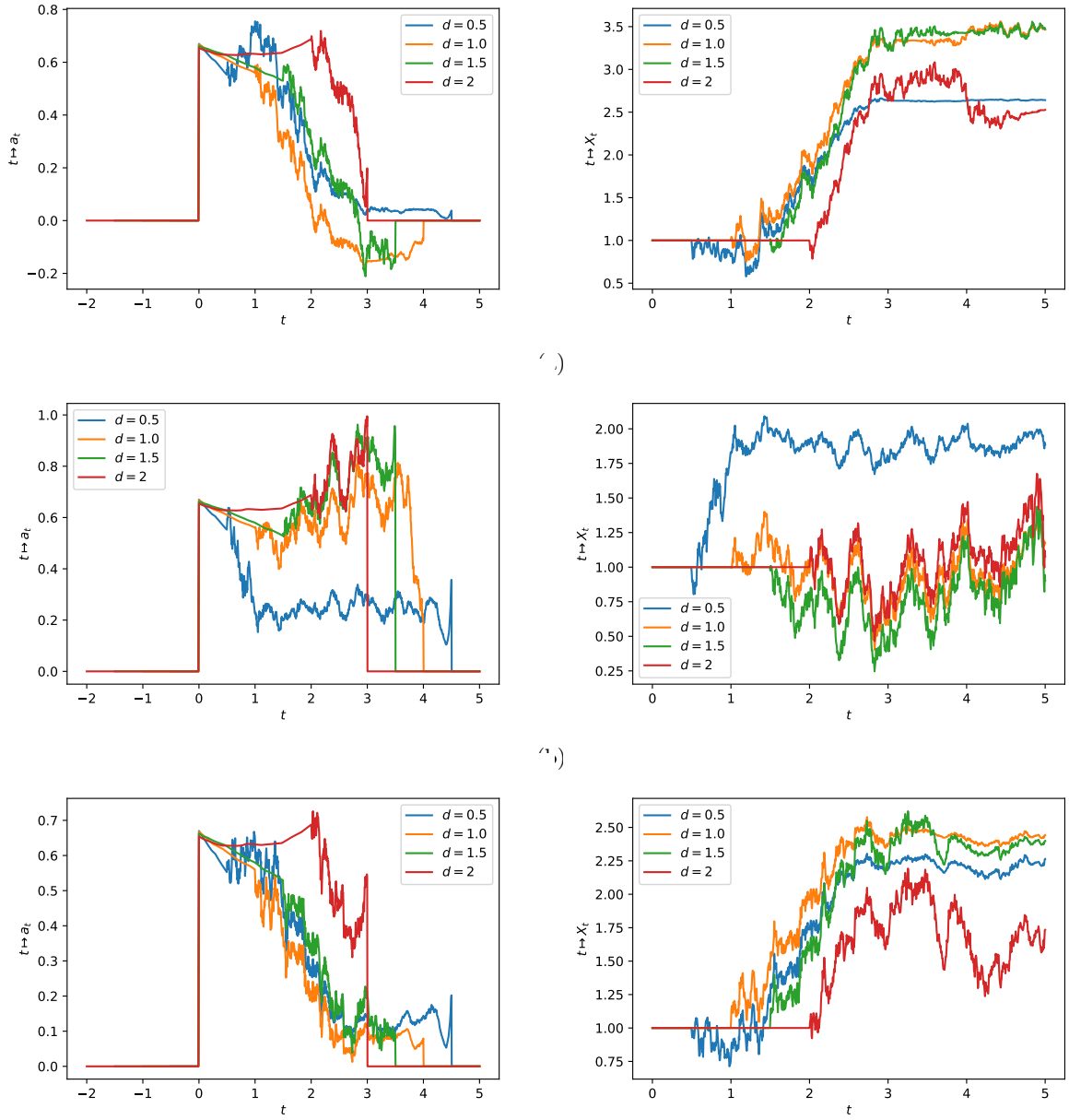
accompanied by the boundary conditions, for almost any $t, s \in [0,T] \times [-d,0]$

$$\begin{aligned} &P_{12}(t,-d) = \lambda_2\sigma_2\left(1 - \rho\frac{\lambda_1}{\lambda_2}\right)P_{11}(t), \qquad P_{\hat{2}\hat{2}}(t,-d) = \sigma_2^2\left(1-\rho^2\right)P_{11}(t), \\ &P_{22}(t,s,-d) = \lambda_2\sigma_2\left(1 - \rho\frac{\lambda_1}{\lambda_2}\right)P_{12}(t,s), \quad P_{22}(t,-d,s) = \lambda_2\sigma_2\left(1 - \rho\frac{\lambda_1}{\lambda_2}\right)P_{12}(t,s), \end{aligned} \tag{4.5.9}$$

and the terminal constraints

$$P_{11}(T) = 1, \qquad P_{12}(T,s) = P_{\hat{2}\hat{2}}(T,s) = P_{22}(T,s,r) = 0, \tag{4.5.10}$$

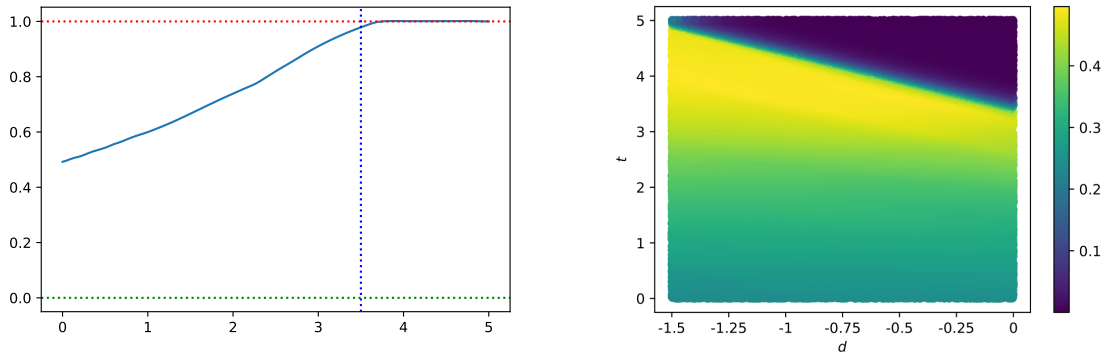
for almost every $s, r \in [-d,0]$.
As in the previous section, we first solve the inner optimization problem 4.5.3.



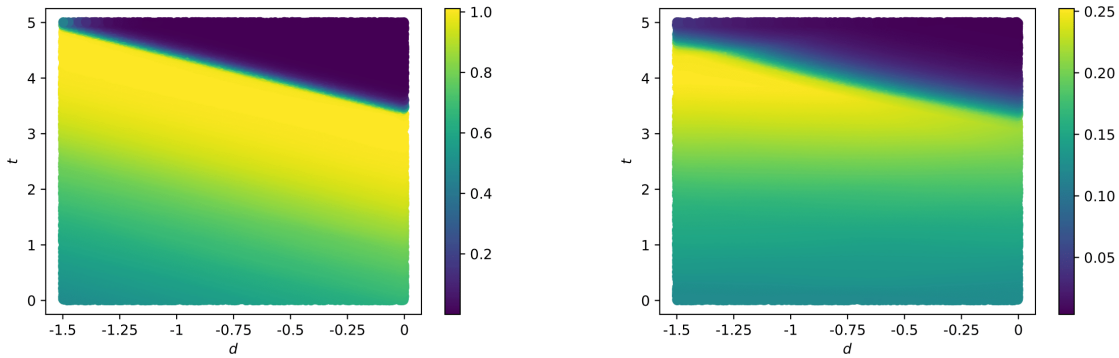
(c)

Figure 4.4: Different scenarios of the optimal portfolio with $c = 1.6$, $\sigma = 1$, $\lambda = 0.5$, and $T = 5$. Left: $t \mapsto \alpha^*$, right: $t \mapsto X_t^*$. Note the destabilization effect and the supplement of volatility induced by the delay feature. Note also the tendency to invest more aggressively the delayed investor has, as she has less time to ensure the promised yield. $\xi^*(d = 0.5) = 2.57$, $\xi^*(d = 1) = 2.68$, $\xi^*(d = 1.5) = 2.80$, $\xi^*(d = 2) = 2.97$.

Lemma 4.5.2. *Fix $\eta \in \mathbb{R}$ and $\xi = c - \eta$. Assume (4.5.8)-(4.5.9)-(4.5.10) admits a piecewise absolutely continuous solution with $P_{\hat{22}}(t) > 0$ for any $t \leq T - d$, and define*



(a) Right: $t \mapsto P_{11}(t)$. Left: $t, s \mapsto P_{12}(t, s)$.



(b) Right: $t \mapsto P_{\hat{22}}(t, s)$. Left: $t, s \mapsto P_{22}(t, s, 0)$.

Figure 4.5: Numerical results of Algorithm 3 with $\sigma = 1$, $\lambda = 0.5$, $d = 1.5$, and $T = 5$.

the couple $(\alpha^*, \beta^*)(\xi)$ *as the investment strategies*

$$\alpha_t^*(\xi) = -\left\{\frac{\lambda_1}{\sigma_1}(X_t^* - \xi) + \rho\frac{\sigma_2}{\sigma_1}\beta_{t-d}^* + \frac{\lambda_1}{\sigma_1 P_{11}(t)}\int_{t-d}^{t}\beta_s^*(\xi)P_{12}(t, s-t)ds\right\}.$$

$$\beta_t^*(\xi) = \frac{-1_{t\leq T-d}}{P_{\hat{22}}(t,0)}\left\{P_{12}(t,0)(X_t^* - \xi) + \int_{t-d}^{t}\beta_s^*(\xi)P_{22}(t,0,r-t)dr\right\},$$

where X^* *denotes the state process* $X^{(\alpha^*,\beta^*)}$. *Then, the inner minimization problem* (4.5.3) *admits* $(\alpha^*(\xi), \beta^*(\xi))$ *as an optimal feedback strategy and the optimal value is*

$$V_0(\xi) = P_{11}(0)(x_0 - \xi)^2 + R(x_0 - \xi, \gamma),$$

where $R(\gamma)$ *denotes the cost associated to the initial investment strategy* γ *on* $[-d, 0]$

$$R(x, \gamma) = 2x\int_{-d}^{0}\gamma_s P_{12}(0, s)ds + \int_{-d}^{0}\gamma_s^2 P_{\hat{22}}(0, s)ds + \int_{[-d,0]^2}\gamma_s\gamma_u P_{22}(0, s, r)dsdr.$$

Proof. The proof is similar to the one of Lemma 4.5.1. □

Finally, the parameter η^* and efficient frontier $\mathrm{Var}(X_T^*) = f(c)$ are given by the same formulas (4.5.6) and (4.5.7) as in in the mono-asset case, γ being the pre-investment strategy of the delayed asset.

Remark 4.5.2. *One surprise that emerges is that the "buy the good stock sell the bad one" criterion is unchanged for the delayed asset. Indeed, the sign of the control for this asset is still given by the sign of* $1-\rho\frac{\lambda_1}{\lambda_2}$*, that fixes the sign of the* P_{12} *and* P_{22}*, as it would be in the case without delay*[†]*, see the boundary conditions* (4.5.9)*. But this threshold disappears in the undelayed asset's control as now only the term* $\frac{\lambda_1}{\sigma_1}$ *remains in the mean-reverting term.*

Numerical simulations: To exhibit the effect of the correlation ρ, we generate two independent Brownian motions $\left(W_t^1\right)_{t\in[0,T]}$ and $(B_t)_{t\in[0,T]}$ and define the Brownian motion $\left(W_t^2\right)_{t\in[0,T]}$ as

$$W_t^2 = \rho W_t^1 + \sqrt{1-\rho^2}B_t, \quad t\in[0,T].$$

We then compare different scenarios with different values of correlation ρ and delay d while fixing W^1 and B. The numerical simulations can be found in Figures 4.6, 4.7 and 4.8. As it could have been expected, we see from (4.5.9) and Figure 4.6, that the greater ρ is, the more favored the undelayed asset is.

[†]When $d=0$, recall that $\alpha_t^* = \frac{\lambda_1 P_t}{\sigma_1(1-\rho^2)}(1-\rho\frac{\lambda_2}{\lambda_1})(\xi^*-X_t^*)$ and $\beta^* = \frac{\lambda_2 P_t}{\sigma_2(1-\rho^2)}(1-\rho\frac{\lambda_1}{\lambda_2})(\xi^*-X_t^*)$ with P being a positive function and $\xi^*\geq X^*$. Thus, in the classical setting, the buy or sell thresholds are $(1-\rho\frac{\lambda_2}{\lambda_1})$ and $(1-\rho\frac{\lambda_1}{\lambda_2})$.

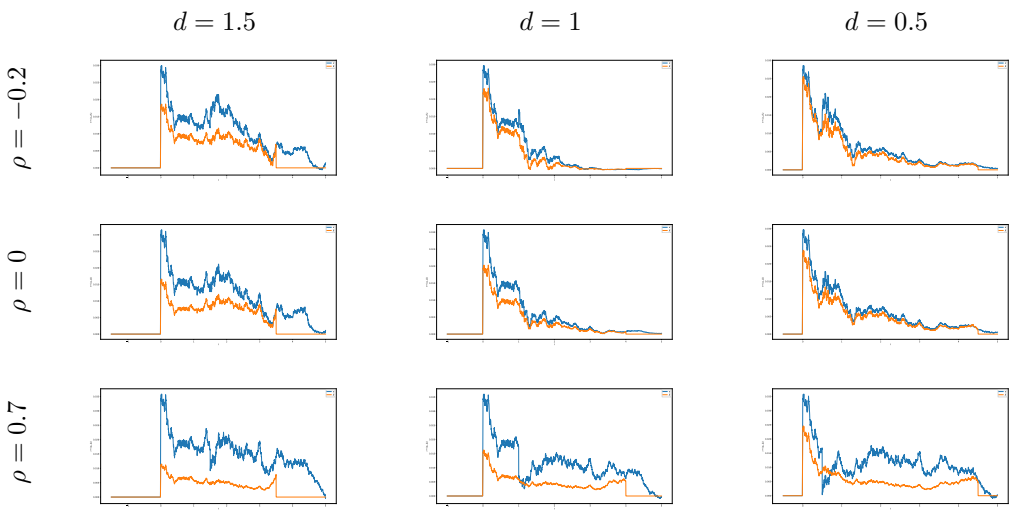


Figure 4.6: $t \mapsto (\alpha_t^*, \beta_t^*)$, with $\sigma_1 = \sigma_2 = 1$, $\lambda_1 = \lambda_2 = 0.5$ and $T = 5$. Blue : α^*, orange : β^*. The same realizations of W and B were used for all experiments. Note that the more positively correlated the assets are, the more favored the undelayed asset is.

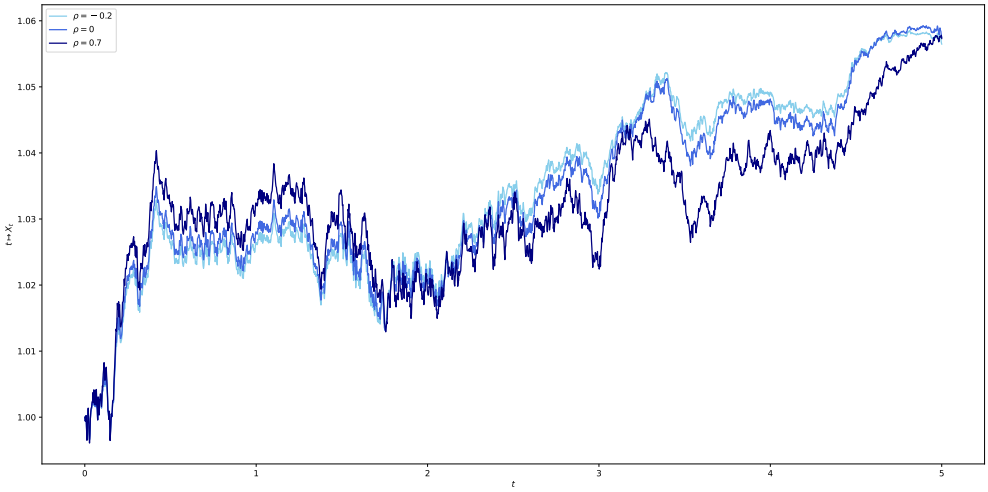


Figure 4.7: $t \mapsto X_t^*$, with $\sigma_1 = \sigma_2 = 1$, $\lambda_1 = \lambda_2 = 0.5$, $T = 5$ and $d = 1.5$ for $\rho = -0.7$, 0 and 0.7. The same realizations of the Brownian motions W^1 and B was used for all experiments.

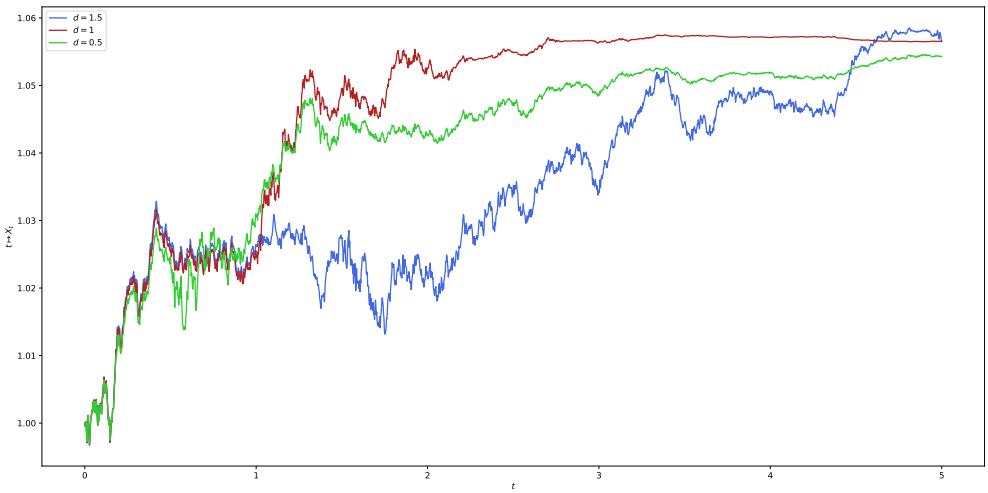


Figure 4.8: $t \mapsto X_t^*$, with $\sigma_1 = \sigma_2 = 1$, $\lambda_1 = \lambda_2 = 0.5$, $T = 5$ and $\rho = -0.7$ for $d = 1.5$, 1 and 0.5. The same realizations of the Brownian motions W^1 and B was used for all experiments.

4.A Proof of Proposition 4.3.2

Our proof extends [Ale+71, Theorem 5] to the case where the volatility is controlled. It consists in slicing the domain $\mathcal{D}$ in slices of size d and proceeding by a backward recursion. More precisely, we show existence, uniqueness, and positiveness of the first coordinates, over a sequence of slices $\left([T-(n+1)d, T-nd] \times [-d,0]^2\right)_n$. We then concatenate the sequence of absolutely continuous solutions obtained, which yields a piece-wise absolutely continuous solution. In each slice, the proof consists of the following steps

1. Show that there exists a unique solution on a small interval;
2. Prove that the local solution is Lipshitz;
3. As a result extend the solution to the whole slice.

We finally concatenate the sequence of solutions obtained above.

4.A.1 Slice $t \in [T-d, T]$, initialization

On $\mathcal{D}_b \cup \mathcal{D}_c$, the constraints (4.2.9)-(4.2.10)-(4.2.11) on $P_{12}, P_{\hat{22}}$ and P_{22} reduce to linear homogeneous transport equations admitting closed form solutions given, for every $(t,s,r) \in \mathcal{D}_b \cup \mathcal{D}_c$, by

$$P_{12}(t,s) = bP_{11}(t+s+d)1_{t+s+d\leq T}, \qquad P_{\hat{22}}(t,s) = \sigma^2 P_{11}(t+s+d)1_{t+s+d\leq T},$$
$$P_{22}(t,s,r) = b^2 P_{11}(t+s\vee r+d)1_{t+s\vee r+d\leq T}.$$

Or, as $P_{11}(t) = 1$ for any $t \geq T-d$, we then have for every $(t,s,r) \in \mathcal{D}_b \cup \mathcal{D}_c$

$$P_{11}(t) = 1, \qquad P_{12}(t,s) = b1_{t+s+d\leq T},$$
$$P_{\hat{22}}(t,s) = \sigma^2 1_{t+s+d\leq T}, \qquad P_{22}(t,s,r) = b^2 1_{t+s\vee r+d\leq T}.$$

The existence and uniqueness in the sense of Definition 4.2.1 are thus trivially proved on $[T-d, T]$.

4.A.2 Slice $[T-2d, T-d]$

On $[T-2d, T-d] \times [-d,0]^2$, we have $P_{\hat{22}}(t,s) = \sigma^2 P_{11}(t+s+d)$ so that $P_{\hat{22}}(t,0) = \sigma^2 P_{11}(t+d) = \sigma^2$. Consequently, the system (4.2.9)-(4.2.10)-(4.2.11) reduces to

$$\begin{aligned}
\dot{P}_{11}(t) &= \frac{P_{12}(t,0)^2}{\sigma^2}, \\
(\partial_t - \partial_s)(P_{12})(t,s) &= \frac{P_{12}(t,0)P_{22}(t,s,0)}{\sigma^2}, \\
(\partial_t - \partial_s - \partial_r)(P_{22})(t,s,r) &= \frac{P_{22}(t,s,0)P_{22}(t,0,r)}{\sigma^2},
\end{aligned} \tag{4.A.1}$$

with terminal conditions

$$P_{11}(T-d) = 1, \qquad P_{12}(T-d,s) = b, \qquad P_{22}(T-d,s,r) = b^2, \tag{4.A.2}$$

and boundary constraints

$$P_{12}(t,-d) = bP_{11}(t), \qquad P_{22}(t,s,-d) = bP_{12}(t,s). \tag{4.A.3}$$

Thus, for every $(t,s,r) \in [T-2d, T-d] \times [-d,0]^2$, the set of equations (4.A.1) and constraints (4.A.2)-(4.A.3) can be rewritten in the following integral form

$$\begin{aligned}
P_{11}(t) &= 1 - \sigma^{-2} \int_t^{T-d} P_{12}(x,0)^2 dx, \\
P_{12}(t,s) &= bP_{11}((T-d) \wedge (t+s+d)) \\
&\quad - \sigma^{-2} \int_t^{(T-d)\wedge(t+s+d)} P_{12}(x,0) P_{22}(x, t+s-x, 0) dx, \\
P_{22}(t,s,r) &= bP_{12}((T-d) \wedge (t+s\wedge r+d), (s-r) \vee (r-s) - d) \\
&\quad - \sigma^{-2} \int_t^{(T-d)\wedge(t+s\wedge r+d)} P_{22}(x, t+s-x, 0) P_{22}(x, 0, t+r-x) dx.
\end{aligned} \tag{4.A.4}$$

We then make use of the following lemma to prove local existence of a solution.

Lemma 4.A.1. *There exists $\tau \in (0,d]$ such that system (4.A.4) has a unique absolutely continuous solution on $[T-\tau-d, T-d] \times [-d,0]^2$.*

Proof. Let $\tau \in (0,d]$ and $\mathcal{S}_\tau$ denote the Banach space of absolutely continuous functions $\xi = (\xi_1(\cdot), \xi_2(\cdot,\cdot), \xi_3(\cdot,\cdot,\cdot))$ defined on

$$\mathcal{D}_\tau = \{(t,s,r) |\ T-d-\tau \le t \le T-d, -d \le s,r \le 0\},$$

endowed with the sup-norm

$$\|\xi\|_\infty = \|\xi_1\|_\infty + \|\xi_2\|_\infty + \|\xi_3\|_\infty,$$

where $\|\xi_1\|_\infty, \|\xi_2\|_\infty$ and $\|\xi_3\|_\infty$ denote, with a slight abuse of notation, the respective sup-norm on $[T-d-\tau, T-d]$, $[T-d-\tau, T-d] \times [-d,0]$ and $[T-d-\tau, T-d] \times [-d,0]^2$. Let $\mathcal{B}_\tau$ denote the ball in $\mathcal{S}_\tau$

$$\mathcal{B}_\tau = \{(\xi_1,\xi_2,\xi_3) \in \mathcal{S}_\tau : \quad \|\xi_1 - 1\| \le 1/2, \quad \|\xi_2 - b\| \le |b|/2, \quad \|\xi_3 - b^2\| \le b^2/2\},$$

On $\mathcal{B}_\tau$, we denote by $\phi = (\phi_1, \phi_2, \phi_3)$ the operator defined as follows

$$\begin{aligned}
(\phi_1\xi)(t) &= 1 - \sigma^{-2} \int_t^{T-d} \xi_2(x,0)^2 dx \\
(\phi_2\xi)(t,s) &= b\phi_1(\xi)((T-d) \wedge (t+s+d)) \\
&\quad - \sigma^{-2} \int_t^{(T-d)\wedge(t+s+d)} \xi_2(r,0)\xi_3(r, t+s-x, 0) dx \\
(\phi_3\xi)(t,s,r) &= b\phi_2(\xi)((T-d) \wedge (t+s\wedge r+d), (s-r) \vee (r-s) - d) \\
&\quad - \sigma^{-2} \int_t^{(T-d)\wedge(t+s\wedge r+d)} \xi_3(x, t+s-x, 0)\xi_3(x, 0, t+r-x) dx.
\end{aligned}$$

Clearly, there exists $\tilde{\tau} > 0$ such that for any $\tau \le \tilde{\tau}$, $\phi(\mathcal{B}_\tau) \to \mathcal{B}_\tau$. We show a contraction property on ϕ. For any $\xi, \xi' \in \mathcal{B}_\tau$, we have the following inequalities

$$\begin{aligned}
\|\phi_1(\xi) - \phi_1(\xi')\|_\infty &\le 4\tau\sigma^{-2}|b|\|\xi_2 - \xi_2'\|_\infty, \\
\|\phi_2(\xi) - \phi_2(\xi')\|_\infty &\le 4\tau\sigma^{-2}\left(|b|\|\xi_2 - \xi_2'\|_\infty + |b|^2\|\xi_3 - \xi_3'\|_\infty\right) \\
&\quad + |b|\|\phi_1(\xi) - \phi_1(\xi')\|_\infty, \\
\|\phi_3(\xi) - \phi_3(\xi')\|_\infty &\le |b|\|\phi_2(\xi) - \phi_2(\xi')\|_\infty + 4\tau\sigma^{-2}|b|^2\|\xi_3 - \xi_3'\|_\infty.
\end{aligned}$$

Consequently, the operator ϕ satisfies

$$\|\phi(\xi) - \phi(\xi')\|_\infty \leq \tau m \|\xi - \xi'\|_\infty,$$

where $m > 0$ depends on b and σ. Therefore, for $\tau < \tilde{\tau} \wedge m^{-1}$, the operator ϕ is a contraction of $\mathcal{B}_\tau$ into itself. Thus, ϕ admits a unique fixed point in $\mathcal{B}_\tau$, which is solution to (4.A.4) on $\mathcal{D}_\tau$.

□

Lemma 4.A.2. *Let $\xi = (\xi_1, \xi_2, \xi_3)$ denote the absolutely continuous solution of (4.A.4) on $\mathcal{D}_\tau$ from Lemma 4.A.1. Then ξ is Lipschitz in each variable on $\mathcal{D}_\tau$.*

Proof. As ξ_1, ξ_2 and ξ_3 are continuous on $\mathcal{D}_\tau$, there exists a constant $m > 0$ such that $|\xi_1| \wedge |\xi_2| \wedge |\xi_3| \leq m$ on $\mathcal{D}_\tau$. Thus, ξ_1 is Lipschitz with constant $\kappa = m^2\sigma^{-2}$. Let us now show that ξ_2 and ξ_3 are Lipschitz in the s-variable. Fix $t \in [T-d-\tau, T-d]$ and $\eta > 0$. Then, for any $s \in [-d, 0]$ such that $s + \eta \in [-d, 0]$, we have

$$\begin{aligned}
|\xi_2(t,s) - \xi_2(t,s+\eta)| \leq & \kappa\eta + \sigma^{-2}\left| \int_t^{(T-d)\wedge(t+s+\eta+d)} \xi_2(x,0)\xi_3(x,t+s+\eta-x,0)dx \right. \\
& \left. - \int_t^{(T-d)\wedge(t+s+d)} \xi_2(x,0)\xi_3(x,t+s-x,0)dx \right| \\
& \leq \kappa\eta + \mathbf{I}(t,s) + \mathbf{II}(t,s),
\end{aligned}$$

Since $|\xi_2| \leq m$, it yields

$$\begin{aligned}
\mathbf{I}(t,s) \leq & \int_t^{(T-d)\wedge(t+s+d)} \big|\xi_2(x,0)\big|\big|\xi_3(x,t+s+\eta-x,0) - \xi_3(x,t+s-x,0)\big|dx \\
\leq & m \int_t^{(T-d)\wedge(t+s+d)} \epsilon(x)dx,
\end{aligned}$$

where ϵ is defined as

$$\epsilon(x) = \sup_{\substack{s,r \\ \in[-d,0]^2}} |\xi_3(x,s,r) - \xi_3(x,s+\eta,r)| + \sup_{s\in[-d,0]} |\xi_2(x,s) - \xi_2(x,s+\eta)|.$$

Futhermore, as $|\xi_2| \wedge |\xi_3| \leq m$ on $\mathcal{D}_\tau$, we have

$$\begin{aligned}
\mathbf{II}(t,s) \leq & \int_{(T-d)\wedge(t+s+d)}^{(T-d)\wedge(t+s+\eta+d)} |\xi_2(x,0)\xi_3(x,t+s+\eta-x,0)|dx \\
\leq & m^2\eta.
\end{aligned}$$

Consequently, for any $t \in [T-d-\tau, T-d]$, we obtain

$$\sup_s |\xi_2(t,s) - \xi_2(t,s+\eta)| \leq m^2\eta + m\int_t^{T-d} \epsilon(r)dr. \tag{4.A.5}$$

Looking at the equation of ξ_3 in system (4.A.4), we obtain in a similar manner

$$|\xi_3(t,s,r) - \xi_3(t,s+\eta,r)| \leq |b|\mathbf{I}(t,s,r) + \sigma^{-2}\mathbf{II}(t,s,r). \tag{4.A.6}$$

An application to the triangle inequality combined with (4.A.5) and the lipshitzianity of ξ_1 leads to

$$\begin{aligned}\mathbf{I}(t,s,r) \leq & |\xi_2((T-d)\wedge(t+(s+\eta)\wedge r+d),(s+\eta-r)\vee(r-(s+\eta))-d) \\ & -\xi_2((T-d)\wedge(t+s\wedge r+d),(s-r)\vee(r-s)-d)| \\ \leq & (\kappa+m^2(1+\sigma^{-2}))\eta+m\int_{(T-d)\wedge(t+s\wedge r+d)}^{T-d}\epsilon(x)dx \\ \leq & (1+2\kappa)\eta+m\int_t^{T-d}\epsilon(x)dx.\end{aligned} \tag{4.A.7}$$

Furthermore

$$\begin{aligned}\mathbf{II}(t,s,r) \leq & \Big|\int_t^{(T-d)\wedge(t+(s+\eta)\wedge r+d)}\xi_3(x,t+s+\eta-x,0)\xi_3(x,0,t+r-x)dx \\ & -\int_t^{(T-d)\wedge(t+s\wedge r+d)}\xi_3(x,t+s-x,0)\xi_3(x,0,t+u-x)dx\Big| \\ \leq & \int_t^{(T-d)\wedge(t+s\wedge r+d)}|\xi_3(x,0,t+r-x)||\xi_3(x,t+s-x,0) \\ & -\xi_3(x,t+(s+\eta)-x,0)|dx \\ & +\int_{(T-d)\wedge(t+s\wedge r+d)}^{(T-d)\wedge(t+(s+\eta)\wedge r+d)}|\xi_3(x,t+(s+\eta)-x,0)\xi_3(r,0,t+r-x)|dx \\ \leq & m^2\eta+\int_t^{T-d}\epsilon(r)dr\end{aligned} \tag{4.A.8}$$

Thus, inequality (4.A.7) together with (4.A.8) and (4.A.6) yield the existence of a positive constant $c>0$, independent of η, such that

$$\sup_{\substack{s,r\\ \in[-d,0]^2}}|\xi_3(t,s,r)-\xi_3(t,s+\eta,r)| \leq c\left(\eta+\int_t^{T-d}\epsilon(r)dr\right),$$

which, combined with (4.A.5) leads, for any $t\in[T-d-\tau,T-d]$, to

$$\epsilon(t)\leq c\left(\eta+\int_t^{T-d}\epsilon(r)dr\right).$$

Consequently, an application to Gronwall's lemma yields $\epsilon(t)\leq m'\eta$ on $[T-d-\tau,T-d]$, with $m'>0$. Thus, ξ_2 and ξ_3 are Lipschitz in the s-variable. The arguments for showing that ξ_2 and ξ_3 are Lipschitz in the t-variable and ξ_3 Lipschitz in the r-variable follow the same line. □

Lemma 4.A.3. *There exists a unique absolutely continuous solution $\xi=(\xi_1,\xi_2,\xi_3)$ of (4.A.4) on $[T-2d,T-d]\times[-d,0]^2$ such that $\xi_1\geq 1-d\left(\frac{b}{\sigma}\right)^2>0$.*

Proof. Let $\theta\in[T-2d,T-d)$ denote the lower limit of all τ's such that there exists an absolutely continuous solution (ξ_1,ξ_2,ξ_3) to (4.A.4) on $[\theta,T-d]$. Assume $\theta>T-2d$. From Lemma 4.A.2, ξ_1, ξ_2 and ξ_3 are Lipschitz in each variable and thus admit a limit, when $t\to\theta$, which is Lipschitz. Therefore, the argument of Lemma (4.A.1) can be repeated to extend the existence and uniqueness of the solution of system (4.A.4) on

$[\xi, T-d]$ for $T-2d \leq \xi < \theta$. As a result, we necessarily have $\theta = T-2d$. It remains to prove that $0 < \xi_1$. For this, note that since ξ_1 is solution to (4.A.4), we have

$$\|\xi_1 - 1\|_\infty \leq \frac{d}{\sigma^2} \sup_{\substack{t\in \\ [T-2d,T-d]}} |\xi_2(t,0)|^2. \tag{4.A.9}$$

By injecting the boundary condition (4.A.2) into the system (4.A.4), one notes that $t \in [T-2d, T-d] \mapsto \xi_2(t,0)$ is solution to

$$\xi_2(t,0) = b - \sigma^{-2} \int_t^{T-d} \xi_2(x,0)\xi_3(x,t-x,0)dx, \qquad T-2d \leq t \leq T-d.$$

Or, for every $t \in [T-2d, T-d]$, $f_t : x \in [t, T-d] \mapsto f_t(x) := \xi_3(x, t-x, 0)$ takes only positive values as f_t is solution to the system

$$\begin{aligned} f_t(x) &= b^2 - \sigma^{-2} \int_x^{T-d} f_t(u)\xi_3(u,0,x-u)du, \qquad x \in [t, T-d], \\ f_t(T-d) &= b^2, \end{aligned}$$

which can be proven to admit, through a contraction proof in the Banach space $C([t, T-d], \mathbb{R})$, a unique positive solution since ξ and its derivatives are bounded. Similarly, we also have $\xi_2(t,0) \geq 0$ for any $t \in [T-2d, T-d]$. As a result, we have $sign(\xi_2) = sign(b)$ and

$$\sup_{\substack{t\in \\ [T-2d,T-d]}} |\xi_2(t,0)| \leq |b|. \tag{4.A.10}$$

Consequently, (4.A.9) and (4.A.10) yield that for any $T-2d \leq t \leq T-d$, we have $\xi_1 \geq 1 - d\left(\frac{b}{\sigma}\right)^2 = a_2 > 0$ as $\mathcal{N}(d,b,\sigma)$ is assumed to be greater than 2. □

Finally, by setting $P_{11}(t) = \xi_1(t)$, $P_{12}(t,s) = \xi_2(t,s)$, $P_{22}(t,s,r) = \xi_3(t,s,r)$ and $P_{\hat{22}}(t,s) = \xi_1(t+s+d)$ for any $(t,s,r) \in [T-2d, T-d] \times [-d,0]^2$, Lemma 4.A.3 yields the existence and uniqueness of a solution P to (4.2.9)-(4.2.10)-(4.2.11) in the sense of definition 4.2.1 on $[T-2d, T-d]$. The concatenation of the unique solution of (4.2.9)-(4.2.10)-(4.2.11) on $[T-d, T]$ and $[T-2d, T-d]$ leads to a unique solution on $[T-2d, T]$.

4.A.3 From slice $[T-nd, T]$ to $[T-(n+1)d, T]$

Let n be an integer such that $2 \leq n < \mathcal{N}(d,b,\sigma)$. Assume that there exists a solution P to (4.2.9)-(4.2.10)-(4.2.11) in the sense of Definition 4.2.1 on $[T-nd, T]$ such that $0 < a_n \leq P_{11}(t) \leq 1$, for any $t \geq T-nd$. Recall the Definition (4.3.7) of $(a_n)_{n\geq 0}$. Consider the following system on $[T-(n+1)d, T-nd] \times [-d,0]^2$

$$\begin{aligned} P_{11}(t) =& P_{11}(T-nd) - \int_t^{T-nd} \frac{P_{12}(x,0)^2}{\sigma^2 P_{11}(x+d)} dx, \\ P_{12}(t,s) =& bP_{11}((T-nd)\wedge(t+s+d)) - \int_t^{(T-nd)\wedge(t+s+d)} \frac{P_{12}(x,0)P_{22}(x,t+s-x,0)}{\sigma^2 P_{11}(x+d)} dx, \\ P_{22}(t,s,r) =& bP_{12}((T-nd)\wedge(t+s\wedge r+d), (s-r)\vee(r-s)-d) \\ & - \int_t^{(T-nd)\wedge(t+s\wedge r+d)} \frac{P_{22}(x,t+s-x,0)P_{22}(x,0,t+r-x)}{\sigma^2 P_{11}(x+d)} dx. \end{aligned} \tag{4.A.11}$$

Note that this system is the same as (4.A.4), the only difference being the term $x \in [T-(n+1)d, T-nd] \mapsto P_{11}(x+d)$ which comes from the previous slice $[T-nd, T-(n-1)d]$. Therefore, it can be considered as a positive continuous coefficient by induction hypothesis. As result, existence and uniqueness on $[T-(n+1)d, T-nd]$ can be proven in the same fashion as in Lemmas 4.A.1-4.A.2-4.A.3. It remains to prove that $P_{11}(t) \geq a_{n+1}$ for any $t \in [T-(n+1)d, T-nd]$. As in Lemma 4.A.3, and by using the induction hypothesis, we have

$$|P_{12}(t,-d)| \leq |bP_{11}(T-nd)| \leq |b|, \qquad t \in [T-(n+1)d, T-nd]. \tag{4.A.12}$$

Furthermore, P_{11} satisfies (4.A.11) on $[T-(n+1)d, T-nd]$, which, combined with $P_{11} \geq a_n$ on $[T-nd, T-(n-1)d]$ and (4.A.12) yields

$$\begin{aligned} P_{11}(t) &\geq P_{11}(T-nd) - \frac{d}{a_n}\left(\frac{b}{\sigma}\right)^2 \\ &\geq a_n - \frac{d}{a_n}\left(\frac{b}{\sigma}\right)^2 = a_{n+1} > 0, \end{aligned}$$

for any $t \in [T-(n+1)d, T-nd]$, which ends the proof.

Part III

Differential learning for non-linear PDEs

Chapter 5

Differential learning methods for solving fully nonlinear PDEs

Abstract We propose machine learning methods for solving fully nonlinear partial differential equations (PDEs) with convex Hamiltonian. Our algorithms are conducted in two steps. First the PDE is rewritten in its dual stochastic control representation form, and the corresponding optimal feedback control is estimated using a neural network. Next, three different methods are presented to approximate the associated value function, i.e., the solution of the initial PDE, on the entire space-time domain of interest. The proposed deep learning algorithms rely on various loss functions obtained either from regression or pathwise versions of the martingale representation and its differential relation, and compute simultaneously the solution and its derivatives. Compared to existing methods, the addition of a differential loss function associated to the gradient, and augmented training sets with Malliavin derivatives of the forward process, yields a better estimation of the PDE's solution derivatives, in particular of the second derivative, which is usually difficult to approximate. Furthermore, we leverage our methods to design algorithms for solving families of PDEs when varying terminal condition (e.g. option payoff in the context of mathematical finance) by means of the class of DeepOnet neural networks aiming to approximate functional operators. Numerical tests illustrate the accuracy of our methods on the resolution of a fully nonlinear PDE associated to the pricing of options with linear market impact, and on the Merton portfolio selection problem.

Keywords: Fully nonlinear PDEs, deep learning, differential learning, option pricing with market impact.

5.1 Introduction

This section is devoted to the resolution of fully nonlinear partial differential equations (PDEs) of the form

$$\begin{cases} \partial_t u + H(x, D_x u, D_x^2 u) &= 0, \qquad (t,x) \in [0,T) \times \mathbb{R}^d, \\ u(T,x) &= g(x), \quad x \in \mathbb{R}^d, \end{cases} \tag{5.1.1}$$

where the Hamiltonian $H : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^{d\times d} \to \mathbb{R} \cup \{\infty\}$ is a lower semi-continuous convex function with respect to the two last arguments (z, γ), and g a measurable function on $\mathbb{R}^d$. The numerical resolution of this class of PDE is a notorious challenging problem, and it is especially difficult to obtain a good approximation of the second spatial derivative $D_x^2 u$ of the solution in this fully nonlinear context. In the last years, significant progress has been achieved towards these challenges with several numerical methods using techniques from deep learning, see the recent surveys by [Bec+20] and [GPW21]: A first class of approximation algorithms, called *Physics Informed Neural Network* (PINNs) [RPK19], also known as *Deep Galerkin method* (DGM) [SS18], directly approximates the solution to the PDE by a neural network, and its partial derivatives by automatic differentiation, by minimizing the loss function arising from the residual of the PDE evaluated on a random grid in the space-time domain. A second class of algorithms relies on the backward stochastic differential representation of the PDE in the semi-linear case by minimizing either a global loss function (see [EHJ17], and extensions in [BEJ19], [Ji+20], [NR21]), or sequence of loss functions from backward recursion (see [HPW20], and variations-extensions in [PWG21], [Bec+21], [GPW21]).

In this section, we consider numerical methods for fully nonlinear PDEs based on machine learning techniques that are conducted in two steps. The starting point of our approach is to rewrite the PDE (5.1.1) with convex Hamiltonian in its stochastic control representation form following the duality arguments of [STZ+13]. An approximation of the associated optimal feedback control is then obtained using a neural network by a global optimization, as described in [HE16] and [GM05]. Based on this control approximation, two main approaches using neural networks are then developed in order to approximate the value function, hence the solution of the initial PDE on the space-time domain, which is formulated as a conditional expectation with respect to the approximate optimal state process.

The first one, called *Differential regression learning*, is inspired by [HS20]. In this paper, the authors compute conditional expectations of an option payoff in the spirit of the Longstaff-Schwartz method [LS01], by parametrizing it with a neural network and performing the regression simultaneously on the value and on the derivative of this neural network. The addition of a regression loss on the derivative, where the derivative of the network, computed by automatic differentiation, is regressed against the pathwise derivative of the conditional expectation integrand, improves the estimation of the first derivative of the conditional expectation and empirically speeds up the training by allowing to train the network on smaller batches. We adapt this method to our context. Indeed, having approximated the optimal control of the stochastic control problem associated to the PDE, the associated value function can be expressed as the conditional expectation of a running payoff of optimally controlled state trajectories, while its gradient is represented also as a conditional expectation formula by differentiation of the payoff. This representation formulae provide two loss functions that will be minimized alternately in order to learn by neural network approximation both the solution of the PDE and its gradient.

The second approach, called Pathwise learning, is inspired by [VŠS21], where the authors compute the conditional expectation of a payoff by using the Feynman-Kac formula to derive a pathwise control variate corresponding to the hedging strategy. In their work, the derivative of the conditional expectation value, present in the hedging integral, is approximated by a neural network and optimized so as to minimize the variance of the conditional expectation estimator. This approach is analogous to the one derived in [PBS01], with the addition of machine learning techniques. In our case, given the approximation of the stochastic control associated to the PDE, a martingale representation of the payoff is derived on optimally controlled trajectories. Our first pathwise method, called *Pathwise martingale learning*, uses this martingale represen-

tation to train the value and the first derivative of a neural network. This method is also in the spirit of the deep BSDE method of [Bec+20], but the minimization of our loss function provides directly an approximation of the solution and its gradient on the space-time domain. Our second method, called *Pathwise differential learning*, considers furthermore the derivative of this martingale representation, computed by automatic differentiation, which gives another loss function to be minimized in order to train neural networks for approximating the value function and its first and second derivatives. Such differential representation has been also considered in the recent paper [NAO21] for designing a deep learning scheme with one-step loss functions as in the deep backward approach in [HPW20] for solving forward backward SDEs with new estimation and error control of the Z process. Actually, the addition of this derivative loss function permits a better approximation of the terminal condition of the PDE and improves the overall approximation of the PDE solution's value and derivatives on the entire domain.

Finally, we leverage our deep learning algorithms for solving families of PDEs when varying the terminal condition. In other words, the input is a function g_K with parameter K, and the output is the solution to the PDE with terminal condition g_K. This is performed by means of the class of DeepOnet neural networks aiming to approximate functional operators. These networks, introduced in [LJK19], rely on a universal approximation theorem for operators [CC95] stating that a neural network with a single hidden layer can approximate accurately any nonlinear continuous operator. The DeepOnet realizes this theorem in practice and can be used to learn the mapping between the terminal function of a PDE and its solution. More examples and algorithms based on DeepOnet for solving family of PDEs are postponed for further development.

The outline of this section is organized as follows. In Section 5.2, we present the problem, recall the dual stochastic control representation of fully nonlinear PDEs and outlines the different methods. In Section 5.3, we present the theory of the Differential regression learning method, give the expression of the losses used to train the neural network and present the advantages of adding a loss to train the first derivative of the neural network. In Section 5.4, the Pathwise and Pathwise differential methods are developed and the expressions of the losses used to train the neural network are given. In Section 5.5, the implementation details and pseudo-codes of the different algorithms are presented along with validation tests and numerical results of the three methods on the Merton porfolio selection problem and on the Black-Scholes with linear market impact PDE. Finally, Section 5.6 presents a method to solve nonlinear parabolic PDEs with parametric terminal condition g_K for parameter values K in a compact set. The codes of our algorithms are available on https://colab.research.google.com/drive/1xyE1U3SqN4Hjia2d3pOsCXCCDGWRUqsH?usp=sharing.

Notations. We end this introduction with some notations that will be used in the sequel of the section. The scalar product between two vectors b and z is denoted by $b \cdot z$, and $|\cdot|$ is the Euclidian norm. Given two matrices $A = (A_{ij})$ and $B = (B_{ij})$, we denote by $A : B = \mathrm{Tr}(A^\top B) = \sum_{i,j} A_{ij} B_{ij}$ its inner product, and by $|A|$ the Frobenius norm of A. Here $\top$ is the transpose matrice operator. $\mathbb{S}^d$ is the set of $d \times d$ symmetric matrices with real coefficients equipped with the partial order: $\gamma_1 \leq \gamma_2$ iff $\gamma_2 - \gamma_1 \in \mathbb{S}^d_+$, the set of positive semidefinite matrices in $\mathbb{S}^d$.

Let $\boldsymbol{M} = (\boldsymbol{M}_{i_1 i_2 i_3}) \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ be a tensor of order 3. For $p = 1, 2, 3$, the p-mode product of $\boldsymbol{M}$ with a vector $b = (b_i) \in \mathbb{R}^{d_p}$, is denoted by $\boldsymbol{M} \bullet_p b$, and it is a tensor of order 2, i.e. a matrix defined elementwise as

$$(\boldsymbol{M} \bullet_1 b)_{i_2 i_3} = \sum_{i_1=1}^{d_1} M_{i_1 i_2 i_3} b_{i_1}, \ (\boldsymbol{M} \bullet_2 b)_{i_1 i_3} = \sum_{i_2=1}^{d_2} M_{i_1 i_2 i_3} b_{i_2}, \ (\boldsymbol{M} \bullet_3 b)_{i_1 i_2} = \sum_{i_3=1}^{d_3} M_{i_1 i_2 i_3} b_{i_3}$$

The p-mode product of a 3-th order tensor $\boldsymbol{M} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ with a matrix $B = (B_{ij}) \in \mathbb{R}^{d_p \times d}$, also denoted by $\boldsymbol{M} \bullet_p B$, is a 3-th order tensor defined elementwise as

$$\begin{aligned}
\big(\boldsymbol{M} \bullet_1 B\big)_{\ell i_2 i_3} &= \sum_{i_1=1}^{d_1} M_{i_1 i_2 i_3} B_{i_1 \ell}, \quad \big(\boldsymbol{M} \bullet_2 B\big)_{i_1 \ell i_3} = \sum_{i_2=1}^{d_2} M_{i_1 i_2 i_3} B_{i_2 \ell} \\
\big(\boldsymbol{M} \bullet_3 B\big)_{i_1 i_2 \ell} &= \sum_{i_3=1}^{d_3} M_{i_1 i_2 i_3} B_{i_3 \ell}.
\end{aligned}$$

Finally, the tensor contraction (or partial trace) of a 3-th order tensor $\boldsymbol{M} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ whose dimensions d_p and d_q are equal is denoted as $\mathrm{Tr}_{p,q}\boldsymbol{M}$. This tensor contraction is a tensor of order 1, i.e. a vector, defined elementwise as

$$\big(\mathrm{Tr}_{1,2}\boldsymbol{M}\big)_{i_3} = \sum_{\ell=1}^{d_1} M_{\ell \ell i_3}, \ \big(\mathrm{Tr}_{1,3}\boldsymbol{M}\big)_{i_2} = \sum_{\ell=1}^{d_1} M_{\ell i_2 \ell}, \ \big(\mathrm{Tr}_{2,3}\boldsymbol{M}\big)_{i_1} = \sum_{\ell=1}^{d_2} M_{i_1 \ell \ell}.$$

5.2 Dual stochastic control representation of fully nonlinear PDE

We consider a fully nonlinear partial differential equation (PDE) of parabolic type:

$$\begin{cases} \partial_t u + H(x, D_x u, D_x^2 u) &= 0, \qquad (t,x) \in [0,T) \times \mathbb{R}^d, \\ u(T,x) &= g(x), \quad x \in \mathbb{R}^d, \end{cases} \tag{5.2.1}$$

where the Hamiltonian $H : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^{d \times d} \to \mathbb{R} \cup \{\infty\}$ is a lower semi-continuous convex function w.r.t the two last arguments (z, γ), and g a measurable function on $\mathbb{R}^d$. As it is usual, we assume that $H(x,z,\gamma) = H(x,z,\gamma^\top)$, and that $\gamma \in \mathbb{S}^d \mapsto H(x,z,\gamma)$ is nondecreasing,

Without loss of generality, we may then assume that H is in a Bellman form:

$$H(x,z,\gamma) = \sup_{a \in A} \big[\mathrm{b}(x,a).z + \frac{1}{2}\sigma\sigma^\top(x,a) : \gamma + f(x,a)\big], \quad (x,z,\gamma) \in \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{S}^d, \tag{5.2.2}$$

for some measurable functions $b : \mathbb{R}^d \times A \to \mathbb{R}^d$, $\sigma : \mathbb{R}^d \times A \to \mathbb{R}^{d \times m}$, $f : \mathbb{R}^d \times A \to \mathbb{R}$, and with A some subset of $\mathbb{R}^q$. Indeed, such form may arise directly from the dynamic programming equation of a stochastic control problem. Otherwise, it can be written in this formulation by following the duality argument as in [STZ+13]. We introduce the concave conjugate of the function $H(x,z,\gamma)$ w.r.t. the last two variables, i.e.

$$\mathrm{f}(x,b,c) := \inf_{z \in \mathbb{R}^d, \gamma \in \mathbb{R}^{d \times d}} \big[H(x,z,\gamma) - b \cdot z - \frac{1}{2}c : \gamma\big], \quad x \in \mathbb{R}^d, b \in \mathbb{R}^d, c \in \mathbb{R}^{d \times d},$$

and notice that $\mathrm{f}(x,b,c) = \mathrm{f}(x,b,c^\top)$ as $H(x,z,\gamma) = H(x,z,\gamma^\top)$. By the Fenchel-Moreau duality relation, we then get

$$\begin{aligned}
H(x,z,\gamma) &= \sup_{b \in \mathbb{R}^d, c \in \mathbb{S}^d} \big[b \cdot z + \frac{1}{2}c : \gamma + \mathrm{f}(x,b,c)\big] \\
&= \sup_{(b,c) \in D_f} \big[b \cdot z + \frac{1}{2}c : \gamma + \mathrm{f}(x,b,c)\big], \qquad \text{for } x \in \mathbb{R}^d, z \in \mathbb{R}^d, \gamma \in \mathbb{S}^d,
\end{aligned}$$

where $D_f := \{(b,c) \in \mathbb{R}^d \times \mathbb{S}^d : \mathrm{f}(x,b,c) > -\infty\} \subset \mathbb{R}^d \times \mathbb{S}^d_+$ by the nondecreasing monotonicity of $\gamma \mapsto H(x,z,\gamma)$. By assuming that H is uniformly continuous in x, we

notice that the domain D_f does not depend on x. Since for any $c \in \mathbb{S}^d_+$, there exists a unique $\mathrm{s} \in \mathbb{S}^d_+$ s.t. $c = \mathrm{s}^2$, the above duality relation is in the Bellman form (5.2.2) with $a = (b, \mathrm{s}) \in A = \{(b, \mathrm{s}) \in \mathbb{R}^d \times \mathbb{S}^d_+ : \mathrm{f}(x, b, \mathrm{s}^2) > -\infty\}$, $\mathrm{b}(x, a) = b$, $\sigma(x, a) = \mathrm{s}$, $f(x, a) = \mathrm{f}(x, b, \mathrm{s}^2)$.

It is well-known that the solution to the PDE (5.2.1) with an Hamiltonian H as in (5.2.2) admits the stochastic representation:

$$u(t,x) = \sup_{\alpha \in \mathcal{A}} \mathbb{E}\Big[g(X_T^{t,x,\alpha}) + \int_t^T f(X_s^{t,x,\alpha}, \alpha_s)\mathrm{d}s\Big], \quad (t,x) \in [0,T] \times \mathbb{R}^d, \tag{5.2.3}$$

where $X = X^{t,x,\alpha}$ is solution to the stochastic differential equation

$$\mathrm{d}X_s = \mathrm{b}(X_s, \alpha_s)\mathrm{d}s + \sigma(X_s, \alpha_s)\mathrm{d}W_s, \quad t \le s \le T, \quad X_t = x, \tag{5.2.4}$$

on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_{0 \le t \le T}, \mathbb{P})$ along with a m-dimensional Brownian motion W, and the control $\alpha \in \mathcal{A}$ is a pair of $\mathbb{F}$-progressively measurable processes valued in A, satisfying suitable integrability conditions for ensuring under some Lipschitz assumptions on the coefficients b, σ that the SDE (5.2.4) admits a unique strong solution.

Problem (5.2.3) is a standard stochastic control problem with controlled Markov state process X governed by (5.2.4), and it is well-known that when it exists the optimal control $\hat{\alpha} \in \mathcal{A}$ is in closed-loop (or feedback) form, i.e.

$$\hat{\alpha}_s = \hat{\mathrm{a}}(s, \hat{X}_s^{t,x}), \quad t \le s \le T, \ (t,x) \in [0,T] \times \mathbb{R}^d,$$

for some measurable function $\hat{\mathrm{a}} : [0,T] \times \mathbb{R}^d \to A \subset \mathbb{R}^q$, where $\hat{X}$ is the state process controlled by $\hat{\alpha}$. From the time-consistency of the stochastic control problem (5.2.3), we notice that this feedback form $\hat{\mathrm{a}}$ does not depend on the starting point $(t,x) \in [0,T] \times \mathbb{R}^d$ of the value function. Furthermore, the value function u is given by the conditional expectation

$$u(t,x) = \mathbb{E}\Big[g(\hat{X}_T) + \int_t^T f(\hat{X}_s, \hat{\mathrm{a}}(s, \hat{X}_s))\mathrm{d}s \big| \hat{X}_t = x\Big]. \tag{5.2.5}$$

Our resolution method for approximating a solution to (5.2.1) on the whole domain $[0,T] \times \mathbb{R}^d$ is performed in two steps:

1. First, following the deep learning approach in [HE16], we shall use neural networks functions $\mathrm{a}_\theta : [0,T] \times \mathbb{R}^d \to A \subset \mathbb{R}^q$, to approximate the optimal feedback control $\hat{b}$, by maximizing over parameters θ the objective function

$$J(\theta) = \mathbb{E}\Big[g(X_T^\theta) + \int_0^T f(X_t^\theta, \mathrm{a}_\theta(t, X_t^\theta))\mathrm{d}t\Big], \tag{5.2.6}$$

   where X^θ solves

$$dX_t^\theta = \mathrm{b}(X_t^\theta, \mathrm{a}_\theta(t, X_t^\theta))\mathrm{d}t + \sigma(X_t^\theta, \mathrm{a}_\theta(t, X_t^\theta))\mathrm{d}W_t, \quad 0 \le t \le T,$$

   with initial condition X_0^θ distributed acccording to some law μ_0 on $\mathbb{R}^d$. We denote by θ^* the "optimal parameter" that maximizes $J(\theta)$, and set $\mathrm{a}^* = \mathrm{a}_{\theta^*}$. We denote by $X^* = X^{\theta^*}$ an approximation of the optimal state process $\hat{X}$. For the numerical implementation, we discretize in time the process X^θ and the integral over f in (5.2.6), and apply a stochastic gradient ascent algorithm based on samples of X^θ. The pseudo-code is presented in Algorithm 4.

2. Once we get an approximation of the optimal feedback control, we could in principle compute $u(t,x)$ from the Feynman-Kac representation (5.2.5) by Monte-Carlo simulations of X^*. However, with the purpose of solving the PDE (5.2.1) on the whole domain, this has to be performed for every point $(t,x) \in [0,T] \times \mathbb{R}^d$, which is not feasible in practice. Instead, we apply three types of differential learning methods for approximating simultaneously the value function u, as well as its derivative: (i) the first one, called *differential regression learning*, is directly inspired from the original approach in [HS20], and gives an approximation of u and its first derivative $D_x u$ from the minimization of two loss functions based on least-square regressions, (ii) the second one in the spirit of [PBS01], [VŠS21], which uses a contingent claim hedging strategy as a Monte Carlo control variate, approximates the value function and its first derivative from the minimization of a single loss function based on the martingale representation in (5.2.5), and is refered to as *pathwise martingale learning* method. (iii) the third one, called *pathwise differential learning*, provides in addition an accurate approximation of the second derivative $D_x^2 u$ of u. We develop these three methods and present their pseudo-codes in the next sections.

Notice that since the neural network a^* is by nature a suboptimal feedback policy, the approximation computed in the second step provides a lower bound for the value function u solution to the PDE.

5.3 Differential regression learning

From the conditional expectation representation (5.2.5), and its fundamental characterization property as an L^2-regression, we have

$$u(t,\hat{X}_t) = \arg\min_{v_t} \mathbb{E}\big|\hat{Y}_T^t - v_t(\hat{X}_t)\big|^2(\hat{X}_t), \qquad \text{for all } t \in [0,T],$$

where the target payoff is

$$\hat{Y}_T^t = g(\hat{X}_T) + \int_t^T f(\hat{X}_s, \hat{\mathrm{a}}(s,\hat{X}_s))\mathrm{d}s, \quad t \in [0,T], \tag{5.3.1}$$

and the argmin is taken over measurable real-valued functions v_t on $\mathbb{R}^d$ s.t. $v_t(\hat{X}_t)$ is square-integrable.

This suggests to use a class of neural networks (NN) functions ϑ^η on $[0,T] \times \mathbb{R}^d$, with parameters η, for approximating the value function u, and a loss function

$$\begin{aligned}\hat{L}_{val}(\eta) &= \mathbb{E}\Big[\int_0^T \big|\hat{Y}_T^t - \vartheta^\eta(t,\hat{X}_t)\big|^2 \mathrm{d}t\Big] \\ &\simeq \mathbb{E}\Big[\int_0^T \big|Y_T^{*,t} - \vartheta^\eta(t,X_t^*)\big|^2 \mathrm{d}t\Big] =: L_{val}^*(\eta),\end{aligned} \tag{5.3.2}$$

where

$$Y_T^{*,t} = g(X_T^*) + \int_t^T f(X_s^*, \mathrm{a}^*(s,X_s^*))\mathrm{d}s, \quad t \in [0,T]. \tag{5.3.3}$$

As pointed out in [HS20], the training of the loss function L_{val}^* in (5.3.2) would require a vast number of samples (often of order millions) to learn accurate approximation of the value function, and is furthermore prone to overfitting. Indeed, by training a neural network to minimize L_{val}^*, we would obtain a function which interpolates the random points generated during training. This comes with two shortcomings. First, a large

number of training samples is needed to get satisfactory values of the solution and a good enough generalisation to untrained domains. Second, the functions obtained by this method are usually noisy. If we are interested in the derivatives of the PDE solution, as it is the case in finance for example, where *greeks* are computed in order to hedge contingent claims, the solution computed might not be accurate enough. Some standard methods, such as Ridge and Lasso penalisations allow to reduce overfitting but come at the cost of adding bias and an arbitrary penalty and do not ensure that the derivative of the network will be a good approximation of the derivative of the PDE solution. To circumvent these issues, and following the idea in [HS20], we propose to consider furthermore the learning of the derivative of the value function. This method relies on pathwise differentiation of the target payoff $\hat{Y}_T^t$ for deriving the gradient of the value function (see Chapter 7 in [Gla13]):

$$D_x u(t, \hat{X}_t) = \mathbb{E}\Big[\hat{Z}_T^t \big| \mathcal{F}_t\Big], \quad t \in [0, T], \tag{5.3.4}$$

where $\hat{Z}_T^t = D_{\hat{X}_t} \hat{Y}_T^t$ (recall that $\hat{Y}_T^t$ is a function of $\hat{X}_t$ as the control $\hat{\mathrm{a}}$ is a feedback function of t and $\hat{X}_t$). This suggests to complete the learning of the value function together with its derivative by considering furthermore the loss function

$$\begin{aligned}\hat{L}_{der}(\eta) &= \mathbb{E}\Big[\int_0^T \big|\hat{Z}_T^t - D_x \vartheta^\eta(t, \hat{X}_t)\big|^2 \mathrm{d}t\Big] \\ &\simeq \mathbb{E}\Big[\int_0^T \big|Z_T^{*,t} - D_x \vartheta^\eta(t, X_t^*)\big|^2 \mathrm{d}t\Big] =: L_{der}^*(\eta),\end{aligned}$$

where $Z_T^{*,t} = D_{X_t^*} Y_T^{*,t}$ valued in $\mathbb{R}^d$, is obtained by automatic differentiation as

$$Z_T^{*,t} = \big(D_{X_t} X_T\big)^\top D_x g(X_T) + \int_t^T \big(D_{X_t} X_s\big)^\top D_x f^{\mathrm{a}^*}(s, X_s) \mathrm{d}s, \quad t \in [0, T], \tag{5.3.5}$$

where we denote by $f^{\mathrm{a}^*}(t, x) = f(x, \mathrm{a}^*(t, x))$, and assuming that g and f are continuously differentiable. Notice that $D_x f^{\mathrm{a}^*} = D_x f + (D_x \mathrm{a}^*)^\top D_a f$, where the derivatives $D_x \mathrm{a}^*$ of the approximate optimal feedback control in the class of neural networks can be efficiently computed by automatic differentiation. Here, to alleviate notations, we have dropped the superscript $*$ for the state $X = X^*$. Actually, when g and f are only piecewise-differentiable, the above relation still holds when the marginal law of X_s is absolutely continuous with respect to Lebesgue measure on $\mathbb{R}^d$, which is satisfied under nondegeneracy conditions on the diffusion coefficients (see Theorem 2.3.2 in [Nua95]). We recall that the flow derivative of the optimal state process, valued in $\mathbb{R}^{d \times d}$, is solution to the SDE (see e.g. [Pro05])

$$D_{X_t} X_s = I_d + \int_t^s D_x \mathrm{b}^{\mathrm{a}^*}(r, X_r) D_{X_t} X_r \mathrm{d}r + D_x \sigma_j^{\mathrm{a}^*}(r, X_r) D_{X_t} X_r \mathrm{d}W_r^j, \; t \le s \le T, \tag{5.3.6}$$

where we denote by $\mathrm{b}^{\mathrm{a}^*}(t, x) = \mathrm{b}(x, \mathrm{a}^*(t, x))$, $\sigma^{\mathrm{a}^*}(t, x) = \sigma(x, \mathrm{a}^*(t, x))$, and use the Einstein summation convention over the repeated index $j = 1, \ldots, d$, with $\sigma_j^{\mathrm{a}^*}$ (resp. σ_j) the j-th column of the matrix σ^{a^*} (resp. σ). Notice that $D_x \mathrm{b}^{\mathrm{a}^*} = D_x \mathrm{b} + D_a \mathrm{b} D_x \mathrm{a}^*$, and $D_x \sigma_j^{\mathrm{a}^*} = D_x \sigma_j + D_a \sigma_j D_x \mathrm{a}^*$.

Remark 5.3.1. *We have an alternative representation (5.3.4) for the gradient of v, which avoids smoothness assumptions on the coefficients. It is expressed with $\hat{Z}_T^t$ given by (see [MZ02]):*

$$\hat{Z}_T^t = g(\hat{X}_T) \hat{H}_T^t + \int_t^T f(\hat{X}_s, \hat{\mathrm{a}}(s, \hat{X}_s)) \hat{H}_s^t \mathrm{d}s, \quad t \in [0, T], \tag{5.3.7}$$

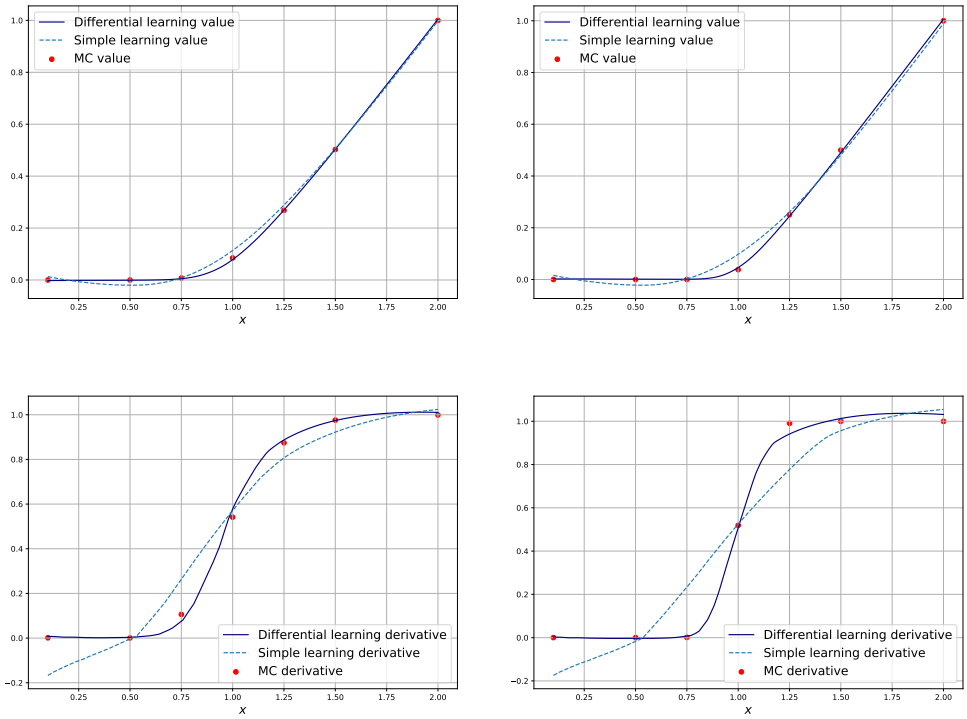
with the so-called Malliavin weights $\hat{H}_s^t$, $t \leq s$, *given by*

$$H_s^t = \frac{1}{s-t} \int_t^s \sigma^{-1}(t, \hat{X}_t)^\top \sigma^{-1}(r, \hat{X}_r) D_{\hat{X}_t} \hat{X}_r \sigma(t, \hat{X}_t) \mathrm{d}W_r,$$

where $\sigma^{-1} = \sigma^\top(\sigma\sigma^\top)^{-1}$ *is the right-inverse of the matrix* σ *assumed to be of full rank. Therefore, in the loss function* L^*_{der}, *instead of* $Z_T^{*,t}$ *as in* (5.3.5), *we can use alternately* $Z_T^{*,t}$ *as in* (5.3.7), *with* $\hat{X}$ *approximated by* X^*, *and* $\hat{\mathrm{a}}$ *approximated by* a^*.

To illustrate the interest of learning the derivative of the value function, we plot in Figure 5.1 the results obtained by learning the value and the derivative (*Differential regression learning*) or by learning just the value (*Simple learning*) of the call option price with market impact (see the application presented in Section 5.5.6).

As a reference, we compute the option price on chosen points $(t, x) \in \mathbb{R}_+ \times \mathbb{R}$ by Monte-Carlo, as explained in section 5.5.4.



(a) $t = 0.5$ (b) $t = 0.9$

Figure 5.1: Value function values (first line) and derivatives (second line) obtained by *Differential Learning* (navy curve) (Algorithm 5), *Simple learning* (blue dashed curve) and Monte Carlo (red dots) plotted as functions of x, for fixed values of t.

In the differential regression learning method, we use a combination of the loss L^*_{val} and L^*_{der} in the training of the neural networks for approximating the value function and its derivative. We train alternately the value and the derivative of the network by taking a gradient step to minimize L^*_{val} every even number of epochs and a gradient step to minimize L^*_{der} every odd number of epochs. An alternative method would be to minimize a convex combination of L^*_{val} and L^*_{der} with weights w_{val} and $w_{der} = 1 - w_{val}$. These weights could be chosen by performing a grid search or a random

search, such as advocated in [BB12] for the choice of neural network hyperparameters. Some theoretical arguments could also be derived in order to choose these weights, as it has been done in [MOB21] to optimally choose the weights of the losses associated to different constraints of a PDE when using Physics Informed Neural Networks [RPK19]. Our approach proves to be effective, as shown by the numerical results in Sections 5.5.5 and 5.5.6, and avoids the need to chose a value for this additional hyperparameter. The algorithmic implementation and the pseudo-codes are described in Section 5.5.2.

5.4 Pathwise learning

5.4.1 Pathwise martingale learning

This approach is based on the martingale representation related to relation (5.2.5), which leads by Itô's formula to the equation:

$$\hat{Y}_T^t = u(t,\hat{X}_t) + \int_t^T \big(D_x u(s,\hat{X}_s)\big)^\top \sigma^{\hat{\mathrm{a}}}(s,\hat{X}_s)\mathrm{d}W_s, \quad t\in[0,T], \tag{5.4.1}$$

where we recall that $\hat{Y}_T^t$ is given in (5.3.1), and denote $\sigma^{\hat{\mathrm{a}}}(t,x) = \sigma(x,\hat{\mathrm{a}}(t,x))$. This suggests to use a class of neural networks (NN) functions ϑ^η on $[0,T]\times\mathbb{R}^d$, with parameters η, for approximating the value function u, and a loss function

$$\begin{aligned}\hat{L}_{mar}(\eta) =\ & \mathbb{E}\Big[\int_0^T \big|\hat{Y}_T^t - \vartheta^\eta(t,\hat{X}_t) - \int_t^T \big(D_x\vartheta^\eta(s,\hat{X}_s)\big)^\top \sigma^{\hat{\mathrm{a}}}(s,\hat{X}_s)\mathrm{d}W_s\big|^2\mathrm{d}t\Big]\\ \simeq\ & \mathbb{E}\Big[\int_0^T \big|Y_T^{*,t} - \vartheta^\eta(t,X_t^*) - \int_t^T \big(D_x\vartheta^\eta(s,X_s^*)\big)^\top \sigma^{\mathrm{a}^*}(s,X_s^*)\mathrm{d}W_s\big|^2\mathrm{d}t\Big]\\ =:\ & L^*_{mar}(\eta).\end{aligned}$$

Recall that Y^* is the process defined in (5.3.3) when using the approximate optimal neural network $\mathrm{a}^\star$. Alternately, we can use two classes of neural networks: one ϑ^η from $[0,T]\times\mathbb{R}^d$ into $\mathbb{R}$, with parameters η, for the approximation of u, and a second one $\mathcal{Z}^\delta$ from $[0,T]\times\mathbb{R}^d$ into $\mathbb{R}^d$, with parameters δ, for the approximation of $D_x u$. We then consider a loss function

$$\tilde{L}^*_{mar}(\eta,\delta) := \mathbb{E}\Big[\int_0^T \big|Y_T^{*,t} - \vartheta^\eta(t,X_t^*) - \int_t^T \mathcal{Z}^\delta(s,X_s^*)^\top \sigma^{\mathrm{a}^*}(s,X_s^*)\mathrm{d}W_s\big|^2\mathrm{d}t\Big].$$

Notice that compared to the deep BSDE approach in [HJE17], which considers a loss function from the misfit between the l.h.s (the target) and r.h.s. of (5.4.1) at time 0, namely

$$\tilde{L}^*_{DBSDE}(y_0,\delta) := \mathbb{E}\Big[\big|Y_T^{*,0} - y_0 - \int_0^T \mathcal{Z}^\delta(s,X_s^*)^\top \sigma^{\mathrm{a}^*}(s,X_s^*)\mathrm{d}W_s\big|^2\Big],$$

our loss functions L^*_{mar} or $\tilde{L}^*_{mar}$ take into account the misfit between the l.h.s and r.h.s. of (5.4.1) at any time $t\in[0,T]$, since our goal is to approximate the solution u (and its derivative) on the whole domain $[0,T]\times\mathbb{R}^d$ (and not only at time $t=0$).

5.4.2 Pathwise differential learning

We can further compute the pathwise derivative in the martingale representation relation (5.4.1) in order to obtain a second estimator linking the first and second derivatives of

$u(t,x)$. Indeed, by [EKQP97], we have

$$D_{\hat{X}_t}\hat{Y}_T^t = D_x u(t,\hat{X}_t) + \int_t^T \Big(\big[D_x\sigma^{\hat{\mathrm{a}}}(s,\hat{X}_s) \bullet_3 D_{\hat{X}_t}\hat{X}_s\big] \bullet_1 D_x u(s,\hat{X}_s) + \sigma^{\hat{\mathrm{a}}}(s,\hat{X}_s)^\top D_x^2 u(s,\hat{X}_s) D_{\hat{X}_t}\hat{X}_s \Big)^\top \mathrm{d}W_s, \quad t\in[0,T].$$

This suggests to use a class of neural networks (NN) functions ϑ^η on $[0,T]\times\mathbb{R}^d$, with parameters η, for approximating the value function u, and a loss function

$$L^*_{dermar}(\eta) = \mathbb{E}\Big[\int_0^T \Big| Z_T^{*,t} - D_x\vartheta^\eta(t,X_t) - \int_t^T \Big(\big[D_x\sigma^{\mathrm{a}^*}(s,X_s) \bullet_3 D_{X_t}X_s\big] \bullet_1 D_x\vartheta^\eta(s,X_s) + \sigma^{\mathrm{a}^*}(s,X_s)^\top D_x^2\vartheta^\eta(s,X_s) D_{X_t}X_s \Big)^\top \mathrm{d}W_s \Big|^2 \mathrm{d}t\Big],$$

where we recall that $Z_T^{*,t}$ is given in (5.3.5), and we omit the superscript $*$ in the approximation of the optimal state process $X = X^*$. Alternatively, we can use three classes of neural networks: one ϑ^η from $[0,T]\times\mathbb{R}^d$ into $\mathbb{R}$, with parameters η, for the approximation of u, a second one $\mathcal{Z}^\delta$ from $[0,T]\times\mathbb{R}^d$ into $\mathbb{R}^d$, with parameters δ, for the approximation of $D_x u$, and a third one Γ^ϵ from $[0,T]\times\mathbb{R}^d$ into $\mathbb{S}^d$, with parameters ϵ, for the approximation of $D_{xx}u$, and consider a loss function

$$\tilde{L}^*_{dermar}(\delta,\epsilon) = \mathbb{E}\Big[\int_0^T \Big| Z_T^{*,t} - \mathcal{Z}^\delta(t,X_t) - \int_t^T \Big(\big[D_x\sigma^{\mathrm{a}^*}(s,X_s) \bullet_3 D_{X_t}X_s\big] \bullet_1 \mathcal{Z}^\delta(s,X_s) + \sigma^{\mathrm{a}^*}(s,X_s)^\top \Gamma^\epsilon(s,X_s) D_{X_t}X_s \Big)^\top \mathrm{d}W_s \Big|^2 \mathrm{d}t\Big].$$

As with the Differential regression learning method presented in Section 5.3, the neural network can be trained either by minimising a convex combination of the losses L^*_{mar} and L^*_{dermar} or by minimising these losses individually. During our numerical experiment, we minimised these two losses individually, but contrary to the algorithm used in the Differential regression learning, for each epoch, a gradient step was made to minimise L^*_{mar} and then another one was made to minimize L^*_{dermar}.

Remark 5.4.1. *This optimisation scheme was found to be more effective when optimising the neural network parameters for the Pathwise differential learning. For the Differential regression learning method, making a gradient step on only one of these two losses at each epoch gave better results. The difference between these two optimization schemes lies in the fact that if both the losses L^*_{mar} and L^*_{dermar} are optimised during an epoch, these two losses are computed using the "old" network weights η, then these weights are modified two times, first by making a gradient step to minimize L^*_{mar}, and then by taking another gradient step to minimize L^*_{dermar}, as written below*

One epoch:

$$\eta' \leftarrow \eta - \nabla_\eta L^*_{mar}(\eta),$$
$$\eta'' \leftarrow \eta' - \nabla_\eta L^*_{dermar}(\eta).$$

When the network is optimised by minimizing alternatively one of these two losses at each epoch, as in the Differential regression method, one of the losses, say L^*_{mar}, is computed using the "old" weights η. A gradient step is then made to minimize this loss

and obtain new network weights $\eta^{'}$, which are then used in the next epoch to compute the loss L^*_{mar} and make the next gradient step, as written below

$$\begin{aligned} &\textit{One epoch:} \\ &\qquad \eta^{'} \leftarrow \eta - \nabla_\eta L^*_{mar}(\eta), \\ &\textit{Next epoch:} \\ &\qquad \eta^{''} \leftarrow \eta^{'} - \nabla_\eta L^*_{dermar}(\eta^{'}). \end{aligned}$$

5.5 Numerical results

In this section, we detail the implementation of the methods presented above. Our algorithms are discretised in time for the training of the processes and for the integrals that appear in the loss functions, and which are approximated by Riemann sums. In the sequel, we are then given a mesh grid $\mathcal{T}_N = \{0 = t_0 < t_1 < \ldots < t_N = T\}$ of $[0, T]$ with $deltat_n = t_{n+1} - t_n$, for $n = 0, \ldots, N-1$.

Our codes are written in Python and we use the Tensorflow library to implement the neural networks and compute the derivatives present in our calculations by auto-differentiation (AAD).

Finally, we illustrate our results with some examples of applications in finance.

5.5.1 Approximation of the optimal control

As a first step, we consider a neural network a_θ from $[0, T] \times \mathbb{R}^d$ into $A \subset \mathbb{R}^q$ for the approximation of the feedback control, and the associated discretised state process

$$X^\theta_{t_{n+1}} = X^\theta_{t_n} + \mathrm{b}(X^\theta_{t_n}, \mathrm{a}_\theta(t_n, X^\theta_{t_n}))\Delta t + \sigma(X^\theta_{t_n}, \mathrm{a}_\theta(t_n, X^\theta_{t_n}))\Delta W_{t_n}, \quad n = 0, \ldots, N-1,$$

starting from $X_0 \sim \mu_0$ (probability distribution on $\mathbb{R}^d$), and where $\Delta W_{t_n} = W_{t_{n+1}} - W_{t_n}$. As in [Hur+21], to constrain the output of the neural network a_θ to be in the control space A, we define a custom activation function σ_A for the output layer of the network. This activation function σ_A is chosen depending on the form of the control space A. When $A = \mathbb{R}^q$, σ_A is equal to the identity function. When the control space if of the form $A = \prod_{i=1}^q [a_i, \infty)$, one can take the component-wise ReLU activation function (possibly shifted and scaled); when $A = \prod_{i=1}^q [a_i, b_i]$, for $a_i \leq b_i$, $i = 1, ..., q$, one can take the component-wise sigmoid activation function (possibly shifted and scaled). For the numerical experiments presented here, we used the ELU (Exponential Linear Unit) activation function, defined as

$$ELU(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0, \end{cases}$$

with parameter α for the hidden layers.

The structure of the neural networks used to approximate the feedback control is represented in Figure 5.2. It is composed of two dense feed-forward sub-networks composed of two layers of n neurons, taking respectively the time t and the state x as input. The outputs of these two sub-network, which are in $\mathbb{R}^n$, are then concatenated and inputed in a third dense network composed of two layers of n neurons and a last layer of q neurons which outputs the approximation of the control in $\mathbb{R}^q$. This structure adds more flexibility compared to the network structure usually implemented, where the time and state variables are directly concatenated and passed through a dense feed-forward

network. It allows to use different activation functions in each sub-network and adapts well to situations where the network is used to approximate a function which has very different behaviors in its time and state variables. The same structure is used for the neural networks used to approximate the value function by Differential Regression learning or Pathwise learning. In the applications presented in this section, we used neural networks with $n = 50$ neurons per layer and $q = 1$.

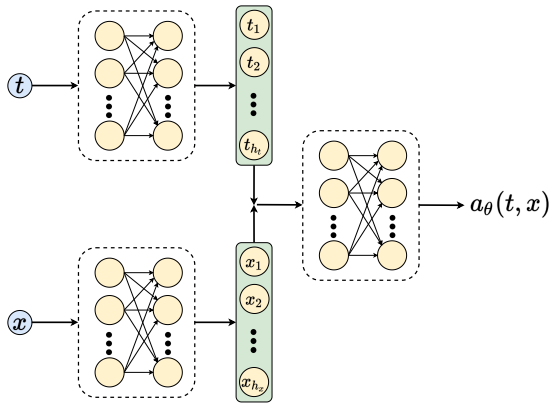


Figure 5.2: Structure of the neural network used to approximate the optimal control and value function.

Remark 5.5.1. *For the approximation of the optimal control for the example of the Merton portfolio selection, presented in Section 5.5.5, the neural network used had the same architecture as the one described above but used 20 neurons per layer and hyperbolic tangent activations functions. This configuration proved to work better empirically, as the exploding gradient problem was encountered while training a network using 50 neurons and ELU activation function.*

For the training of the neural network control a_θ, we use a batch of M independent trajectories $\{x_{t_n}^{m,\theta}, t_n \in \mathcal{T}_N\}$, $m = 1, \ldots, M$, of $\{X_{t_n}^{\theta}, t_n \in \mathcal{T}_N\}$, and apply a stochastic gradient ascent method to the empirical gain function:

$$J_M(\theta) = \frac{1}{M} \sum_{m=1}^{M} \Big[g(x_T^{m,\theta}) + \sum_{n=0}^{N-1} f\big(x_{t_n}^{m,\theta}, \mathrm{a}_\theta(t_n, x_{t_n}^{m,\theta})\big) \Delta t_n \Big].$$

The pseudo-code is described in Algorithm 4. The output of this algorithm yields a parameter θ^*, and so an approximation of the optimal feedback control with $\mathrm{a}^* = \mathrm{a}_{\theta^*}$, and of the associated optimal state process with $X^* = X^{\theta^*}$. In the sequel, to alleviate notations, we shall omit the superscript $*$, and simply denote a and X.

5.5.2 Differential regression learning algorithm

We consider a neural network ϑ^η from $[0,T] \times \mathbb{R}^d$ into $\mathbb{R}$ for the approximation of the value function. The derivatives $D_x g$, $D_x f$, $D_x \mathrm{b}$, $D_x \sigma$, $D_x \mathrm{a}$, and $D_x \vartheta^\eta$ that appear in the differential regression learning methods are computed straightforwardly by auto-differentiation. Concerning the flow derivative of the approximate optimal state process, it is computed by time discretization of (5.3.6), which can be efficiently obtained by storing the one-step derivatives:

$$D_{X_{t_n}} X_{t_{n+1}} \quad = \quad I_d + D_x b^{\mathrm{a}^*}(t_n, X_{t_n}) \Delta t_n + \sum_{j=1}^{d} D_x \sigma_j^{\mathrm{a}^*}(t_n, X_{t_n}) \Delta W_{t_n}^j, \quad n = 0, \ldots, N-1,$$

Algorithm 4: Deep learning scheme to solve the stochastic control problem (5.2.3)

Result: A set of optimized parameters θ^*;
Initialize the learning rate l and the neural network a_θ;
Generate an $\mathbb{R}^{N+1}$-valued time grid $0 = t_0 < t_1 < \ldots < t_N = T$ with time steps $(\Delta t_n)_{n=0,\ldots,N-1}$;
Generate a batch of M starting points $X_0 \sim \mu_0$ and Brownian increments $(\Delta W_{t_n})_{n=0,\ldots,N-1}$ in R^d;
for *each batch element* m **do**
  Compute the trajectory $(x_{t_n}^{m,\theta})_{n=0,\ldots,N}$ through the scheme

$$x_{t_{n+1}}^{m,\theta} = x_{t_n}^{m,\theta} + \mathrm{b}(x_{t_n}^{m,\theta}, \mathrm{a}_\theta(t_n, x_{t_n}^{m,\theta}))\Delta t_n + \sigma(x_{t_n}^{m,\theta}, \mathrm{a}_\theta(t_n, x_{t_n}^{m,\theta}))\Delta w_{t_n}^m,$$

  from the generated starting point $x_{t_0}^m$ and Brownian increments $(\Delta w_{t_n}^m)_{n=0,\ldots,N-1}$;
end
for *each epoch* **do**
  Compute the batch loss

$$J_M(\theta) = \frac{1}{M}\sum_{m=1}^{M}\Big[g(x_T^{m,\theta}) + \sum_{n=0}^{N-1} f\big(x_{t_n}^{m,\theta}, \mathrm{a}_\theta(t_n, x_{t_n}^{m,\theta})\big)\Delta t_n\Big]$$

  Compute the gradients $\nabla_\theta J_M(\theta)$;
  Update $\theta \leftarrow \theta - l\nabla_\theta J_M(\theta)$;
end
Return: The set of optimized parameters θ^*;

and then use the chain rule

$$D_{X_{t_n}} X_{t_p} = D_{X_{t_n}} X_{t_{n+1}} \cdots D_{X_{t_{p-1}}} X_{t_p}, \quad \text{for } n < p \in [\![0, N]\!]$$

The target payoff and its derivative are then computed as

$$Y_T^{t_n} = g(X_T) + \sum_{p=n}^{N-1} f^{\mathrm{a}^*}(t_p, X_{t_p})\Delta t_p, \quad n = 0, \ldots, N,$$

$$Z_T^{t_n} = \big(D_{X_{t_n}} X_T\big)^\top D_x g(X_T) + \sum_{p=n}^{N-1} \big(D_{X_{t_n}} X_{t_p}\big)^\top D_x f^{\mathrm{a}^*}(t_p, X_{t_p})\Delta t_p,$$

with the convention that the above sum over p is zero when $n = N$.

For the training of the neural network ϑ^η, we use a batch of M independent samples $(x_{t_n}^m, y_T^{m,t_n}, z_T^{m,t_n})$, $m = 1, \ldots, M$, of $(X_{t_n}, Y_T^{t_n}, Z_T^{t_n})$, $n = 0, \ldots, N$, and apply stochastic gradient descent for the minimization of the mean squared error functions

$$MSE_{val}(\eta) = \frac{1}{M}\sum_{m=1}^{M}\sum_{n=0}^{N-1}\big|y_T^{m,t_n} - \vartheta^\eta(t_n, x_{t_n}^m)\big|^2\Delta t_n$$

$$MSE_{der}(\eta) = \frac{1}{M}\sum_{m=1}^{M}\sum_{n=0}^{N-1}\frac{1}{\|z_T^{t_n}\|^2}\big|z_T^{m,t_n} - D_x\vartheta^\eta(t_n, x_{t_n}^m)\big|^2\Delta t_n.$$

Here, as in [HS20] (Appendix 2), we normalize the derivative loss by the L_2 norm of the target derivative computed along the batch dimension $\|z_T^{t_n}\|^2 := \big(\sum_{m=1}^M \sqrt{z_T^{t_n}}\big)^2$.

The pseudo-code is described in Algorithm 5.

Remark 5.5.2. *Notice that in the above expressions of the mean squared errors, the neural network is trained from the time $t_0 = 0$ up to time $t_{N-1} < T$ and is thus not trained on the terminal condition of the PDE (5.2.1). This choice is justified by the fact that the neural network is already implicitly trained to fit the terminal condition at time*

T since the terminal function g appears in the losses. Furthermore, we observed that the regularity of the terminal function affects the performance of the neural network. If the terminal function is at least of class C^1, no problem arises as the regularity of the solution to the PDE (5.2.1) is the same on the domain $[0,T) \times \mathbb{R}^d$ and on the terminal domain $\{T\} \times \mathbb{R}^d$. If we use a neural network ϑ^η of regularity C^1, we will then be able to approximate the PDE solution on the entire domain $[0,T] \times \mathbb{R}^d$. However, if the terminal condition's regularity is less than C^1, it will be difficult for the neural network to approximate the PDE solution on the entire domain $[0,T] \times \mathbb{R}^d$. Indeed, the solution of parabolic PDEs is often smoother than its terminal (or initial) condition, thus the neural network will have to approximate a function that has a continuous first derivative on the domain $[0,T) \times \mathbb{R}^d$ and a discontinuous first derivative on $\{T\} \times \mathbb{R}^d$. If the neural network used is not C^1, which is the case for a network with ReLU activation functions for example, the network will give a good approximation of the terminal condition but will give a worst fit of the solution on the domain $[0,T) \times \mathbb{R}^d$, and particularly of its derivatives. On the contrary, if the neural network used is C^1, which is the case when the ELU activation function is used, the network will give a good approximation of the solution on $[0,T) \times \mathbb{R}^d$ but will give a worse approximation of the terminal condition. The difficulty thus comes from the fact that we try to obtain a solution on the entire domain of the PDE. As the terminal condition is known and the quantity of interest is the solution of the PDE (5.2.1) on the domain $[0,T) \times \mathbb{R}^d$, we choose to use a C^1 neural network trained on this domain.

Algorithm 5: Deep learning scheme for Differential Regression learning

Result: A set of optimized parameters η^*;
Initialize the learning rate l, the neural networks ϑ^η;
Generate an $\mathbb{R}^{N+1}$-valued time grid $0 = t_0 < t_1 < ... < t_N = T$ with time steps $(\Delta t_n)_{n=0,\dots,N-1}$;
Generate a batch of M starting points $X_0 \sim \mu_0$ and Brownian increments $(\Delta W_{t_n})_{n=0,\dots,N}$ in R^d;
for *each batch element* m **do**
  Compute the trajectory $(x^m_{t_n})_{n=0,\dots,N}$ through the scheme

$$x^m_{t_{n+1}} = x^m_{t_n} + \mathrm{b}^{\mathrm{a}^*}(t_n, x^m_{t_n})\Delta t_n + \sigma^{\mathrm{a}^*}(t_n, x^m_{t_n})\Delta w^m_{t_n},$$

  from the generated starting point $x^m_{t_0}$, Brownian increments $(\Delta w^m_{t_n})_{n=0,\dots,N-1}$ and previously trained control $a = a_{\theta^*}$;
  Compute the value and derivative targets $(y^{m,t_n}_T)_{n=0,\dots,N}$ and $(z^{m,t_n}_T)_{n=0,\dots,N}$;
end
for *each epoch* **do**
  if *Epoch number is even* **then**
    Compute, for every batch element m, the integral $\sum_{n=0}^{N-1} |y^{m,t_n}_T - \vartheta^\eta(t_n, x^m_{t_n})|^2 \Delta t_n$;
    Compute the batch loss $MSE_{val}(\eta)$;
    Compute the gradient $\nabla_\eta MSE_{val}(\eta)$;
    Update $\eta \leftarrow \eta - l\nabla_\eta MSE_{val}(\eta)$;
  end
  else
    Compute, for every batch element m, the integral $\sum_{n=0}^{N-1} |z^{m,t_n}_T - D_x\vartheta^\eta(t_n, x^m_{t_n})|^2 \Delta t_n$.;
    Compute the batch loss $MSE_{der}(\eta)$;
    Compute the gradient $\nabla_\eta MSE_{der}(\eta)$;
    Update $\eta \leftarrow \eta - l\nabla_\eta MSE_{der}(\eta)$;
  end
end
Return: The set of optimized parameters η^*;

5.5.3 Pathwise learning algorithms

We consider a neural network ϑ^η from $[0,T] \times \mathbb{R}^d$ into $\mathbb{R}$ for the approximation of the value function. For the training of this neural network, we use a batch of M independent samples $(x^m_{t_n}, y^{m,t_n}_T, \Delta w^m_{t_n})$, $m = 1, \dots, M$, of $(X_{t_n}, Y^{t_n}_T, \Delta W_{t_n})$, $n = 0, \dots, N$, and apply

stochastic gradient descent for the minimization of the mean squared error function

$$MSE_{mar}(\eta) = \frac{1}{M}\sum_{m=1}^{M}\sum_{n=0}^{N-1}\Big| y_T^{m,t_n} - \vartheta^\eta(t_n, x_{t_n}^m) - \sum_{p=n}^{N-1}\big(D_x\vartheta^\eta(t_p, x_{t_p}^m)\big)^\top \sigma(x_{t_p}^m, \mathrm{a}^*(t_p, x_{t_p}^m))\Delta w_{t_p}^m\Big|^2 \Delta t_n, \tag{5.5.1}$$

The pseudo-code for the pathwise martingale learning is described in Algorithm 6.

Algorithm 6: Deep learning scheme for Pathwise martingale learning with 1 NN

Result: A set of optimized parameters η^*;
Initialize the learning rate l, the neural networks ϑ^η;
Generate an $\mathbb{R}^{N+1}$-valued time grid $0 = t_0 < t_1 < ... < t_N = T$ with time steps $(\Delta t_n)_{n=0,\dots,N-1}$;
Generate a batch of M starting points $X_0 \sim \mu_0$ and Brownian increments $(\Delta W_{t_n})_{n=0,\dots,N}$ in R^d;
for *each batch element* m **do**
  Compute the trajectory $(x_{t_n}^m)_{n=0,\dots,N}$ through the scheme

$$x_{t_{n+1}}^m = x_{t_n}^m + \mathrm{b}^{\mathrm{a}^*}(t_n, x_{t_n}^m)\Delta t_n + \sigma^{\mathrm{a}^*}(t_n, x_{t_n}^m)\Delta w_{t_n}^m,$$

  from the generated starting point $x_{t_0}^m$, Brownian increments $(\Delta w_{t_n}^m)_{n=0,\dots,N-1}$ and previously trained control $a = a_{\vartheta^*}$;
  Compute the value target $(y_T^{m,t_n})_{n=0,\dots,N}$;
end
for *each epoch* **do**
  Compute, for every batch element m, the integral $\sum_{n=0}^{N-1}\Big|y_T^{m,t_n} - \vartheta^\eta(t_n, x_{t_n}^m) - \sum_{p=n}^{N-1}\big(D_x\vartheta^\eta(t_p, x_{t_p}^m)\big)^\top\sigma^{\mathrm{a}^*}(t_p, x_{t_p}^m)\Delta w_{t_p}^m\Big|^2\Delta t_n$;
  Compute the batch loss $MSE_{mar}(\eta)$;
  Compute the gradient $\nabla_\eta MSE_{mar}(\eta)$;
  Update $\eta \leftarrow \eta - l\nabla_\eta MSE_{mar}(\eta)$;
end
Return: The set of optimized parameters η^*;

Alternately, we can use another neural network $\mathcal{Z}^\delta$ from $[0,T]\times\mathbb{R}^d$ into $\mathbb{R}^d$ for the approximation of the gradient of the solution, and then use the mean squared error function

$$\tilde{MSE}_{mar}(\eta,\delta) = \frac{1}{M}\sum_{m=1}^{M}\sum_{n=0}^{N-1}\Big| y_T^{m,t_n} - \vartheta^\eta(t_n, x_{t_n}^m) - \sum_{p=n}^{N-1}\big(\mathcal{Z}^\delta(t_p, x_{t_p}^m)\big)^\top\sigma^{\mathrm{a}^*}(x_{t_p}^m)\Delta w_{t_p}^m\Big|^2\Delta t_n.$$

The pseudo-code for the pathwise martingale learning with two neural networks is described in Algorithm 10 in Appendix 5.A.

For the differential version of this algorithm, we also use a neural network ϑ^η from $[0,T]\times\mathbb{R}^d$ into $\mathbb{R}$ for the approximation of the value function that we train by using the same batch of M independent samples and applying stochastic gradient descent for the minimization of both the mean squared error functions defined in (5.5.1) and the following one:

$$
\begin{aligned}
MSE_{dermar}(\eta) = \frac{1}{M}\sum_{m=1}^{M}\sum_{n=0}^{N-1}\Big| & z_T^{m,t_n} - D_x\vartheta^\eta(t_n, x_{t_n}^m) \\
& - \sum_{p=n}^{N-1}\Big(\big[D_x\sigma^{\mathrm{a}^*}(t_p, x_{t_p}^m) \bullet_3 D_{x_{t_n}} x_{t_p}^m\big] \bullet_1 D_x\vartheta^\eta(t_p, x_{t_p}^m) \\
& + \sigma^{\mathrm{a}^*}(t_p, x_{t_p}^m)^\top D_{xx}\vartheta^\eta(t_p, x_{t_p}^m) D_{x_{t_n}} x_{t_p}^m\Big)^\top \Delta w_{t_p}^m\Big|^2 \Delta t_n.
\end{aligned}
$$

The pseudo-code is described in Algorithm 7.

Algorithm 7: Deep learning scheme for Pathwise differential learning with 1 NN

Result: A set of optimized parameters η^*;
Initialize the learning rate l, the neural networks ϑ^η;
Generate an $\mathbb{R}^{N+1}$-valued time grid $0 = t_0 < t_1 < ... < t_N = T$ with time steps $(\Delta t_n)_{n=0,\dots,N-1}$;
Generate a batch of M starting points $X_0 \sim \mu_0$ and Brownian increments $(\Delta W_{t_n})_{n=0,\dots,N}$ in R^d;
for *each batch element* m **do**
  Compute the trajectory $(x_{t_n}^m)_{n=0,\dots,N}$ through the scheme

$$x_{t_{n+1}}^m = x_{t_n}^m + \mathrm{b}^{\mathrm{a}^*}(t_n, x_{t_n}^m)\Delta t_n + \sigma^{\mathrm{a}^*}(t_n, x_{t_n}^m)\Delta w_{t_n}^m,$$

  from the generated starting point $x_{t_0}^m$, Brownian increments $(\Delta w_{t_n}^m)_{n=0,\dots,N-1}$ and previously trained control $a = a_{\theta^*}$;
  Compute the value and derivative targets $(y_T^{m,t_n})_{n=0,\dots,N}$ and $(z_T^{m,t_n})_{n=0,\dots,N}$;
end
for *each epoch* **do**
  Compute, for every batch element m, the integral
  $\sum_{n=0}^{N-1}\Big|y_T^{m,t_n} - \vartheta^\eta(t_n, x_{t_n}^m) - \sum_{p=n}^{N-1}\big(D_x\vartheta^\eta(t_p, x_{t_p}^m)\big)^\top\sigma^{\mathrm{a}^*}(t_p, x_{t_p}^m)\Delta w_{t_p}^m\Big|^2\Delta t_n$;
  Compute the batch loss $MSE_{mar}(\eta)$;
  Compute the gradient $\nabla_\eta MSE_{mar}(\eta)$;
  Update $\eta \leftarrow \eta - l\nabla_\eta MSE_{mar}(\eta)$;
  Compute, for every batch element m, the integral
  $\sum_{n=0}^{N-1}\Big|z_T^{m,t_n} - D_x\vartheta^\eta(t_n, x_{t_n}^m) - \sum_{p=n}^{N-1}\Big(\big[D_x\sigma^{\mathrm{a}^*}(t_p, x_{t_p}^m) \bullet_3 D_{x_{t_n}} x_{t_p}^m\big] \bullet_1 D_x\vartheta^\eta(t_p, x_{t_p}^m)$
  $+\sigma^{\mathrm{a}^*}(t_p, x_{t_p}^m)^\top D_{xx}\vartheta^\eta(t_p, x_{t_p}^m) D_{x_{t_n}} x_{t_p}^m\Big)^\top \Delta w_{t_p}^m\Big|^2\Delta t_n$;
  Compute the batch loss $MSE_{dermar}(\eta)$;
  Compute the gradient $\nabla_\eta MSE_{dermar}(\eta)$;
  Update $\eta \leftarrow \eta - l\nabla_\eta MSE_{dermar}(\eta)$;
end
Return: The set of optimized parameters η^*;

Alternately, in addition to the NN ϑ^η for u, we can use neural networks $\mathcal{Z}^\delta$ and Γ^ϵ for the gradient and the Hessian that we train with the loss function

$$
\begin{aligned}
\tilde{MSE}_{dermar}(\delta, \epsilon) = \frac{1}{M}\sum_{m=1}^{M}\sum_{n=0}^{N-1}\Big| & z_T^{m,t_n} - \mathcal{Z}^\delta(t_n, x_{t_n}^m) \\
& - \sum_{p=n}^{N-1}\Big(\big[D_x\sigma^{\mathrm{a}^*}(t_p, x_{t_p}^m) \bullet_3 D_{x_{t_n}} x_{t_p}^m\big] \bullet_1 \mathcal{Z}^\delta(t_p, x_{t_p}^m) \\
& + \sigma^{\mathrm{a}^*}(t_p, x_{t_p}^m)^\top \Gamma^\epsilon(t_p, x_{t_p}^m) D_{x_{t_n}} x_{t_p}^m\Big)^\top \Delta w_{t_p}^m\Big|^2 \Delta t_n,
\end{aligned}
$$

where $D_{x_{t_n}} x_{t_p}^m$ denotes the flow derivative of the approximate optimal state process at time t_p w.r.t the state at time t_n along the path m of the batch.

The pseudo-code for this version using three neural networks is described in Algorithm 11 in Appendix 5.A.

5.5.4 Validation tests

The deep learning methods described above provide an approximation ϑ of the solution u to the PDE, which relies up-front on the approximation a^θ of the optimal control $\hat{\mathrm{a}}$ arising from the dual stochastic control representation.

We can test and validate the convergence and accuracies of these approximations as follows. On the one hand, as in the DGM and PINN methods, see [SS18] and [RPK19], we can compute the losss $\mathcal{L}_{res}$ and $\mathcal{L}_{term}$, associated respectively to the residual and to the terminal condition of the partial differential equation (5.2.1)

$$\mathcal{L}_{res} := \frac{1}{|\mathcal{T}||\chi|} \sum_{t\in\mathcal{T}, x\in\chi} \Big|\partial_t \vartheta^\eta + H(x, D_x\vartheta^\eta, D_x^2\vartheta^\eta)\Big|^2, \tag{5.5.2}$$

$$\mathcal{L}_{term} := \frac{1}{|\chi|} \sum_{x\in\chi} \Big|\vartheta^\eta(T,x) - g(x)\Big|^2,$$

where the time grid $\mathcal{T}$ is composed of times that were not used during the network training, so as to verify the generalisation of the solution obtained, and χ is a bounded space grid in $\mathbb{R}^d$. As the state diffusion $(X_t^\theta)_{0\le t\le T}$ is not bounded, the bounds of the space grid are fixed arbitrarily depending on the domain of interest. Alternatively, these bounds could be chosen based on the distribution of the values attained during the computation of the diffusion scheme of $(X_t^\theta)_{0\le t\le T}$.

On the other hand, by noting that the optimal control should satisfy the optimality condition

$$D_a\mathrm{b}(x,\hat{\mathrm{a}}).D_x u(t,x) + \frac{1}{2}\mathrm{Tr}_{1,2}\big(D_a\sigma\sigma^\top(x,\hat{\mathrm{a}}) \bullet_2 D_{xx}u(t,x)\big) + D_a f(x,\hat{\mathrm{a}}) = 0,$$

we can check the accuracy the approximation a_θ of the optimal control by computing the following loss

$$\mathcal{L}_{optim} = \frac{1}{|\mathcal{T}||\chi|} \sum_{t\in\mathcal{T}, x\in\chi} \Big|D_a\mathrm{b}(x,\mathrm{a}_\theta)D_x\vartheta^\eta(t,x) + \frac{1}{2}\mathrm{Tr}_{1,2}\big(D_a\sigma\sigma^\top(x,\mathrm{a}_\theta)\big) \bullet_2 D_{xx}\vartheta^\eta(t,x)\big)$$

$$+\ D_a f(x,\mathrm{a}_\theta)\Big|^2,$$

on the same grid $\mathcal{T}\times\chi$ as before.

Another validation method, which is more graphical, consists in approximating numerically the optimal control as described in Section 5.5.1 and then computing the value function and its first derivative for some chosen points (t,x) by Monte Carlo simulations:

$$\vartheta_{MC}(t,x) = \frac{1}{M}\sum_{m=1}^{M}\Big[g(x_{t_N}^{m,t,x}) + \sum_{p=n}^{N-1} f(x_{t_p}^{m,t,x}, \mathrm{a}_\theta(t_p, x_{t_p}^{m,t,x}))\Delta t_p\Big],$$

$$\begin{aligned} D_x\vartheta_{MC}(t,x) = \frac{1}{M}\sum_{m=1}^{M}\Big[&\big(D_x x_{t_N}^{m,t,x}\big)^\top D_x g(x_{t_N}^{m,t,x}) \\ &+ \sum_{p=n}^{N-1}\Big(\big(D_x x_{t_N}^{m,t,x}\big)^\top D_x f(x_{t_p}^{m,x}, \mathrm{a}_\theta(t_p, x_{t_p}^{m,t,x})) \\ &+ \big(D_x\mathrm{a}_\theta(t_p, x_{t_p}^{m,t,x}) D_x x_{t_p}^{m,t,x}\big)^\top D_a f(x_{t_p}^{m,t,x}, \mathrm{a}_\theta(t_p, x_{t_p}^{m,t,x}))\Big)\Delta t_p\Big], \end{aligned}$$

with $t = t_n$. We then plot these Monte Carlo points alongside the value functions obtained by using neural networks to check that the machine learning methods described in the previous sections are able to approximate the value function corresponding to the approximated optimal control a_θ.

5.5.5 Example of Merton portfolio selection

We consider the Bellman equation:

$$\begin{cases} \partial_t u + \sup_{a\in\mathbb{R}} \left[axbD_x u + \frac{1}{2}a^2x^2\sigma^2 D_x^2 u \right] &= 0, \qquad (t,x) \in [0,T) \times (0,\infty), \\ u(T,x) &= g(x), \quad x \in (0,\infty), \end{cases}$$

which arises from the Merton portfolio selection problem where an agent invests a proportion $\alpha = (\alpha_t)_t$ of her wealth $X = X^\alpha$ in a stock following a Black-Scholes model with rate of return $b \in \mathbb{R}$, and constant volatility $\sigma > 0$. The controlled wealth dynamics is then governed by

$$dX_t = X_t\alpha_t b\mathrm{d}t + X_t\alpha_t\sigma\mathrm{d}W_t,$$

and the goal of the investor is to maximize over α her expected terminal wealth $\mathbb{E}[g(X_T)]$, with g some utility function, i.e. concave and nondecreasing, on $(0,\infty)$.

When the utility function g is of power type, i.e. $g(x) = x^\gamma/\gamma$, for some $\gamma < 1$, $\gamma \neq 0$, it is well-known that the optimal control is constant equal to

$$\hat{\mathrm{a}} = \frac{b}{\sigma^2(1-\gamma)},$$

while the value function is explicitly given by

$$u(t,x) = e^{\rho(T-t)}g(x), \qquad \text{with} \qquad \rho = \frac{b^2}{2\sigma^2}\frac{\gamma}{1-\gamma}.$$

These closed-form expressions serve as benchmarks for comparing our results computed by the differential learning algorithms.

In order to check that the value function approximation obtained is a lower bound of the true one, we compute in Table 5.1 the difference between the closed form value function and the estimation of the value function on points (t,x) obtained by computing the expectation (5.2.5) by Monte Carlo on $1e^6$ trajectories controlled by the Deep Learning approximation of the optimal control.We compute the value functions on a grid $t \in \{0, 0.5, 0.9\}$, $x \in \{1e^{-2}, 0.5, 0.75, 1, 1.25, 1.5, 2\}$ with parameters $b = 0.2$, $\sigma = 0.2$ and power utility with exponent $\gamma = 0.5$ and present in the table the difference between the closed form value and the Monte Carlo approximation. For clarity of presentation we present the results averaged over t in this table.

	$x = 1e^{-2}$	$x = 0.5$	$x = 0.75$	$x = 1$	$x = 1.25$	$x = 1.5$	$x = 2$
Difference closed - MC	$2.699e^{-4}$	$1.811e^{-3}$	$2.208e^{-3}$	$2.543e^{-3}$	$2.839e^{-3}$	$3.106e^{-3}$	$3.582e^{-3}$

Table 5.1: Difference between the closed form value function and the value computed by Monte Carlo on $1e^6$ trajectories on points $x \in \{1e^{-2}, 0.5, 0.75, 1, 1.25, 1.5, 2\}$ and averaged over times $t \in \{0, 0.5, 0.9\}$ for the Merton problem with parameters $b = 0.2$, $\sigma = 0.2$ and power utility with exponent $\gamma = 0.5$.

We compute in Table 5.2, the residual losses defined in (5.5.2) for the NN ϑ^η obtained by the various deep learning methods: the differential learning scheme (Algorithm 5),

the pathwise martingale learning with 1 NN (Algorithm 6) and the pathwise differential learning with 1 NN (Algorithm 7). Since two gradient steps are performed during each training epoch of the Pathwise differential learning method, we indicate the training time for 500 epochs for this method whereas the training time of the two other methods is indicated for 1000 epochs. For each of these algorithms, 8192 starting points and Brownian trajectories are used in order to train the neural networks. We also provide the training time for each of these algorithms. On this table we see that the Pathwise differential learning method is the slowest to train but yields the best results in terms of residual and terminal losses. We see that for all three methods the difference between the residual loss only and the sum of residual and terminal loss is small, meaning that with all three methods the neural network managed to fit the terminal condition of the PDE during the training.

	Diff. regr. learning	Path. 1NN	Path. diff. 1NN
Residual loss	$1.538e^{-1}$	$1.752e^{-1}$	$7.872e^{-2}$
Residual loss + terminal loss	$1.548e^{-1}$	$1.758e^{-1}$	$7.894e^{-2}$
Training time	274s 1000 epochs	297s 1000 epochs	525s 500 epochs

Table 5.2: Residual and boundary losses computed on a 1002x1002 time and space grid with $t \in [0, 0.9]$ and $x \in [1e^{-2}, 2]$ for the Merton problem with parameters $b = 0.2$, $\sigma = 0.2$ and power utility with exponent $\gamma = 0.5$.

We plot the value function $\vartheta^\eta(t,x)$ and its derivatives $\partial_x \vartheta^\eta(t,x)$ and $\partial_{xx} v_\eta(t,x)$ for fixed values $t = 0$, $t = 0.5$, $t = 0.9$, and for parameter values $b = 0.2$, $\sigma = 0.2$, $\gamma = 0.5$, and compare it with the closed-form solution of the problem. Figure 5.3 corresponds to the Differential regression learning method, Figure 5.4 corresponds to the pathwise martingale learning while Figure 5.5 corresponds to the pathwise differential learning method. These graphs are coherent with the results presented in Table 5.2. We see that the Pathwise differential provides the best fit, in particular for the first and second derivatives of the solution, followed by the Differential regression method. The Pathwise method provides a good fit for the value of the PDE solution but does not manage to fit very well its first and second derivatives for small values of x.

5.5.6 Example of the Black-Scholes model with linear market impact

We consider the option pricing problem with linear market impact as studied in [Loe18], which leads to a nonlinear Black Scholes (BS) equation in the form (5.2.1) with g the option payoff and an Hamiltonian H given on $(0,\infty) \times \mathbb{R}$ by

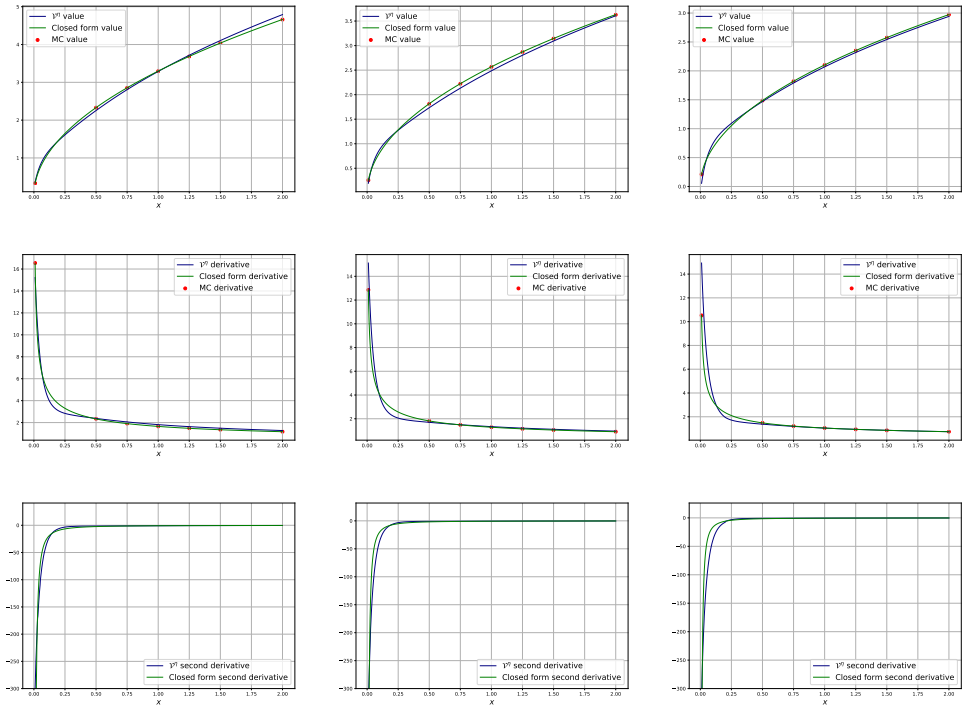
$$H(x,\gamma) = \begin{cases} \frac{1}{2}\sigma^2 \frac{x^2\gamma}{1-\lambda x^2 \gamma}, & \text{if } \lambda x^2 \gamma < 1 \\ \infty, & \text{otherwise.} \end{cases} \tag{5.5.3}$$

where $\sigma > 0$ is the volatility in the BS model, and λ is a nonnegative constant related to the linear market impact. Notice that H can be written in Bellman form as

$$H(x,\gamma) = \sup_{a \geq 0} \big[\frac{1}{2} a x^2 \gamma - \frac{1}{2\lambda}(\sqrt{a} - \sigma)^2\big],$$

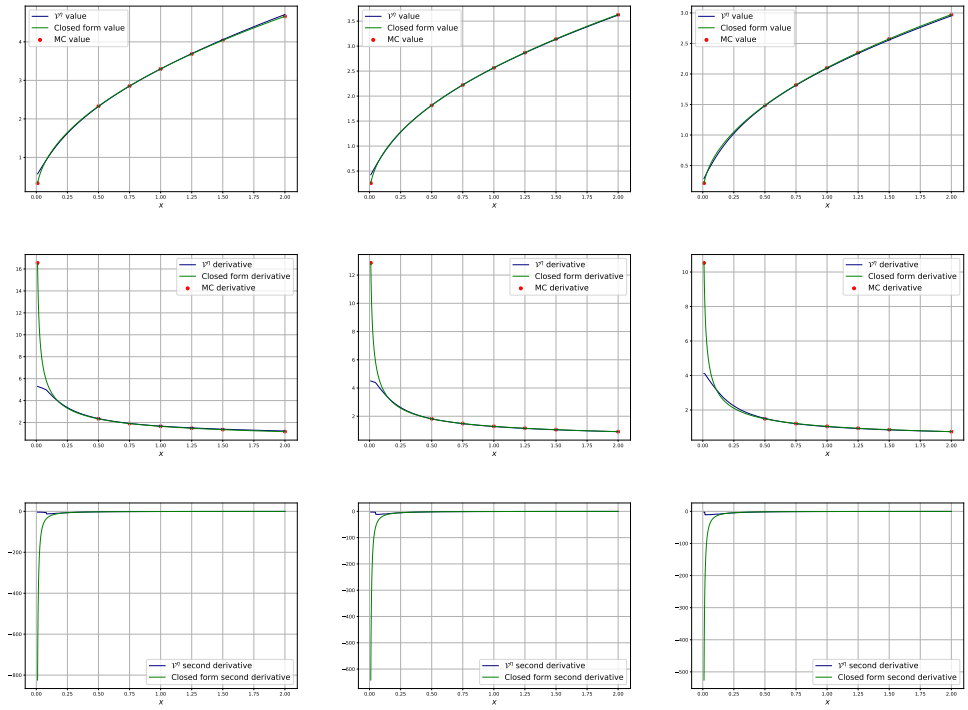
which corresponds to the dual stochastic control representation of the option price u as

$$u(t,x) = \sup_{\alpha} \mathbb{E}\Big[g(X_T^{t,x,\alpha}) - \frac{1}{2\lambda}\int_t^T \big(\sqrt{\alpha_s} - \sigma\big)^2 \mathrm{d}s\Big], \tag{5.5.4}$$



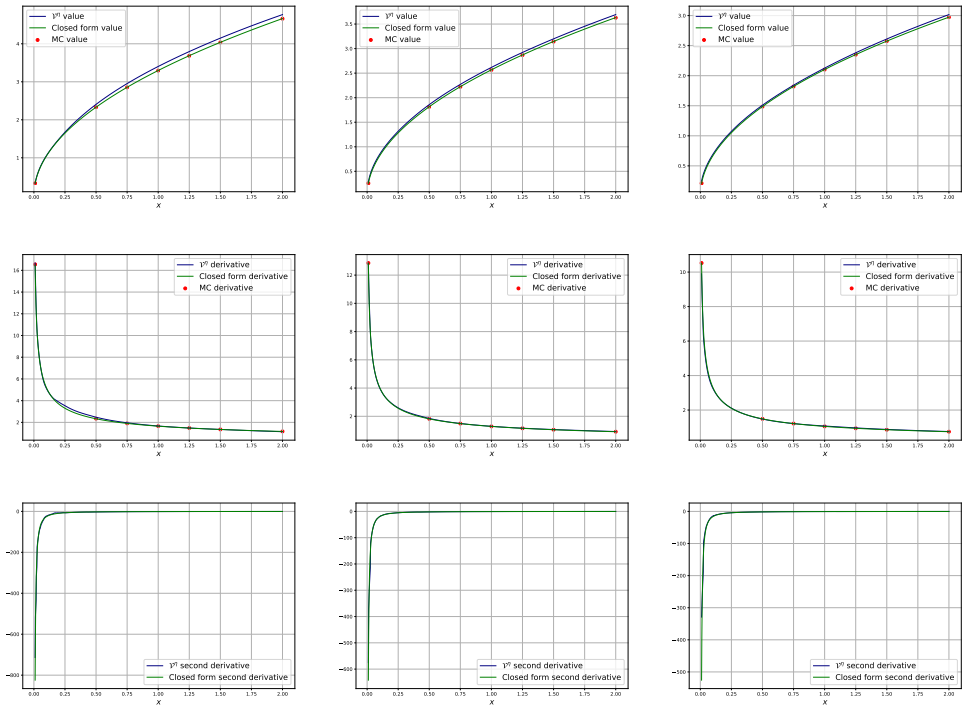
(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 5.3: Value function ϑ^η and its first and second derivative obtained by Differential regression Learning (Algorithm 5) for the Merton problem with parameters $b = 0.2$, $\sigma = 0.2$ and power utility with exponent $\gamma = 0.5$, plotted as functions of x, for fixed values of t.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 5.4: Value function ϑ^η and its first and second derivative obtained by Pathwise learning (Algorithm 6) for the Merton problem with parameters $b = 0.2$, $\sigma = 0.2$ and power utility with exponent $\gamma = 0.5$, plotted as functions of x, for fixed values of t.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 5.5: Value function ϑ^η and its first and second derivative obtained by Pathwise differential learning (Algorithm 7) for the Merton problem with parameters $b = 0.2$, $\sigma = 0.2$ and power utility with exponent $\gamma = 0.5$, plotted as functions of x, for fixed values of t.

where $X = X^{t,x,\alpha}$ is governed by the controlled dynamics

$$\mathrm{d}X_s = X_s\sqrt{\alpha_s}\mathrm{d}W_s, \quad t \leq s \leq T,\ X_t = x > 0,$$

with a control process α valued in $\mathbb{R}_+$.

We shall apply the various differential learning methods to this problem for two examples of option payoff.

5.5.6.1 Closed-form solution for a logarithmic terminal cost

We first consider the toy example where the option payoff g is logarithmic: $g(x) = \ln x$. Indeed, in this case, we can check that the solution to the pricing PDE (5.2.1) with H as in (5.5.3) is given in closed-form by

$$u(t,x) = \ln(x) - \frac{\sigma^2}{2(1+\lambda)}(T-t),$$

while the optimal control to the dual stochastic control representation (5.5.4) is constant equal to

$$\hat{\mathrm{a}}(t,x) = \left(\frac{\sigma}{1+\lambda}\right)^2.$$

In order to check that the value function approximation obtained is a lower bound of the true one, we compute in Table 5.3 the difference between the closed form value function and the estimation of the value function on points (t,x) obtained by computing the expectation (5.2.5) by Monte Carlo on $1e^6$ trajectories controlled by the Deep Learning approximation of the optimal control. We compute the value functions on a grid $t \in \{0, 0.5, 0.9\}$, $x \in \{1e^{-2}, 0.5, 0.75, 1, 1.25, 1.5, 2\}$ with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$ and present in the table the difference between the closed form value and the Monte Carlo approximation. For clarity of presentation we present the results averaged over t in this table.

	$x = 1e^{-2}$	$x = 0.5$	$x = 0.75$	$x = 1$	$x = 1.25$	$x = 1.5$	$x = 2$
Difference closed - MC	$2.673e^{-4}$	$2.042e^{-4}$	$2.058e^{-4}$	$2.066e^{-4}$	$2.099e^{-4}$	$2.147e^{-4}$	$2.300e^{-4}$

Table 5.3: Difference between the closed form value function and the value computed by Monte Carlo on $1e^6$ trajectories on points $x \in \{1e^{-2}, 0.5, 0.75, 1, 1.25, 1.5, 2\}$ and averaged over times $t \in \{0, 0.5, 0.9\}$ for Black-Scholes problem with linear market impact factor $\lambda = 5e^{-3}$ and parameter $\sigma = 0.2$.

In Table 5.4, we compute the residual losses defined in (5.5.2) for the NN ϑ^η obtained by the various deep learning methods: the differential learning scheme (Algorithm 5), the pathwise martingale learning with 1 NN (Algorithm 6), the pathwise differential learning with 1 NN (Algorithm 7). We also provide the training time for 500 epochs for each of these algorithms. On this table, we see that the Pathwise learning methods yield better results than the Differential regression learning methods. The difference between the residual loss only and the sum of the residual and terminal loss is small in all three methods, meaning that both the PDE solution's derivatives and terminal condition have been learned by the neural network.

We plot the value function $\vartheta^\eta(t,x)$ and its derivatives $\partial_x\vartheta^\eta(t,x)$ and $\partial_{xx}v_\eta(t,x)$ for fixed values $t = 0$, $t = 0.5$, $t = 0.9$, parameter $\sigma = 0.2$ and linear market impact factor $\lambda = 5e^{-3}$, and compare it with the closed-form solution of the problem. Figure 5.6

	Diff. regr. learning	Path. 1NN	Path. diff. 1NN
Residual loss	$2.046e^{-3}$	$3.484e^{-4}$	$6.644e^{-4}$
Residual loss + terminal loss	$2.179e^{-3}$	$3.864e^{-4}$	$6.758e^{-4}$
Training time (500 epochs)	163s	130s	525s

Table 5.4: Residual and boundary losses computed on a 1002x1002 time and space grid with $t \in [0, 0.9]$ and $x \in [0.1, 2]$ for a terminal logarithmic payoff, with parameter $\sigma = 0.2$ and linear market impact factor $\lambda = 5e^{-3}$.

corresponds to the Differential regression learning method, Figure 5.7 corresponds to the pathwise martingale learning while Figure 5.8 corresponds to the pathwise differential learning method. On these graphs, we can see that the three methods, as well as the Monte-Carlo values ϑ_{MC} and $D_x\vartheta_{Mc}$, yield good approximations of the PDE solution and its derivatives. Notice however that the pathwise learning (see Figure 5.7) does not provide a good approximation of the second derivative on the boundary points of the grid, namely the points that were not explored by the simulations, but when combining with the differential learning (see Figure 5.8), it greatly improves the approximation of the second derivative.

5.5.6.2 Call-option type terminal condition

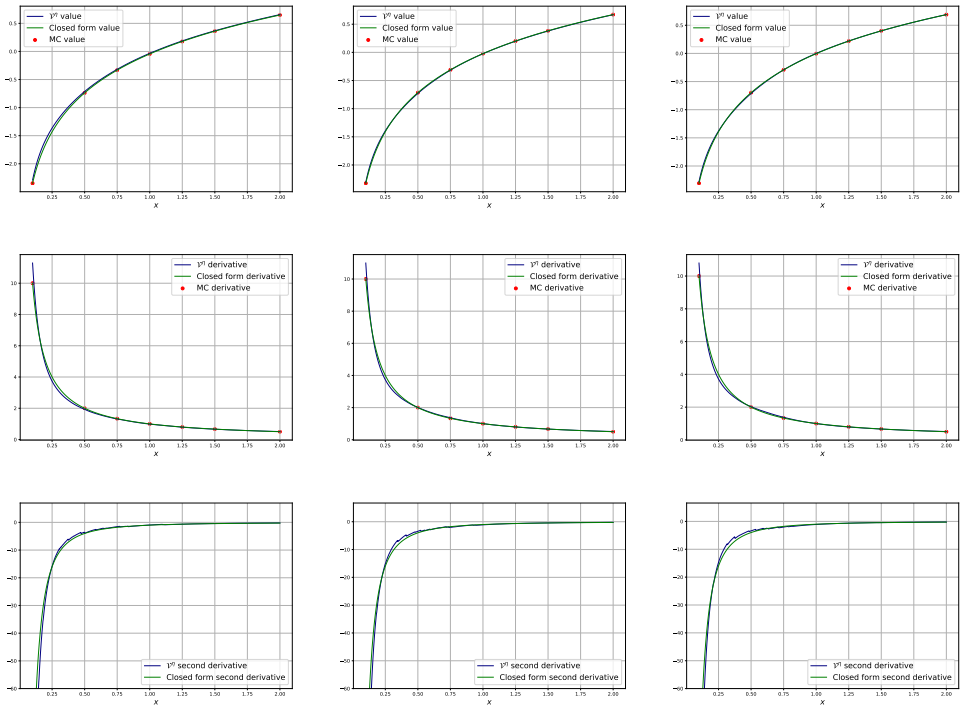
We now consider a usual call option payoff with strike $K = 1$, hence a function g equal to: $g(x) = \max(x - 1, 0)$.

Again, we compute in Table 5.5, the residual losses defined in (5.5.2) for the NN ϑ^η obtained by the various deep learning methods: the differential learning scheme (Algorithm 5), the pathwise martingale learning with 1 NN (Algorithm 6) and the pathwise differential learning with 1 NN (Algorithm 7). We also provide the training time for each of these algorithms. On this table, we see that the Differential regression learning and the Pathwise differential methods yield better results than the "simple" Pathwise method. Despite having the lowest residual loss, the Pathwise method gives the biggest terminal loss. This shows that, while the time and second space derivatives of the neural network give a low residual loss corresponding to the Hamiltonian (5.5.3), the network does not manage to fit the terminal function. This phenomenon is also present, to a lesser extent, in the approximation given by the Differential regression learning method. Out of the three methods, the Pathwise differential yields the smallest residual and terminal losses.

	Diff. regr. learning	Path. 1NN	Path. diff. 1NN
Residual loss	$2.998e^{-4}$	$2.756e^{-4}$	$2.283e^{-4}$
Residual loss + terminal loss	$3.972e^{-4}$	$1.004e^{-3}$	$2.538e^{-4}$
Training time	262s 1000 epochs	299s 1000 epochs	584s 500 epochs

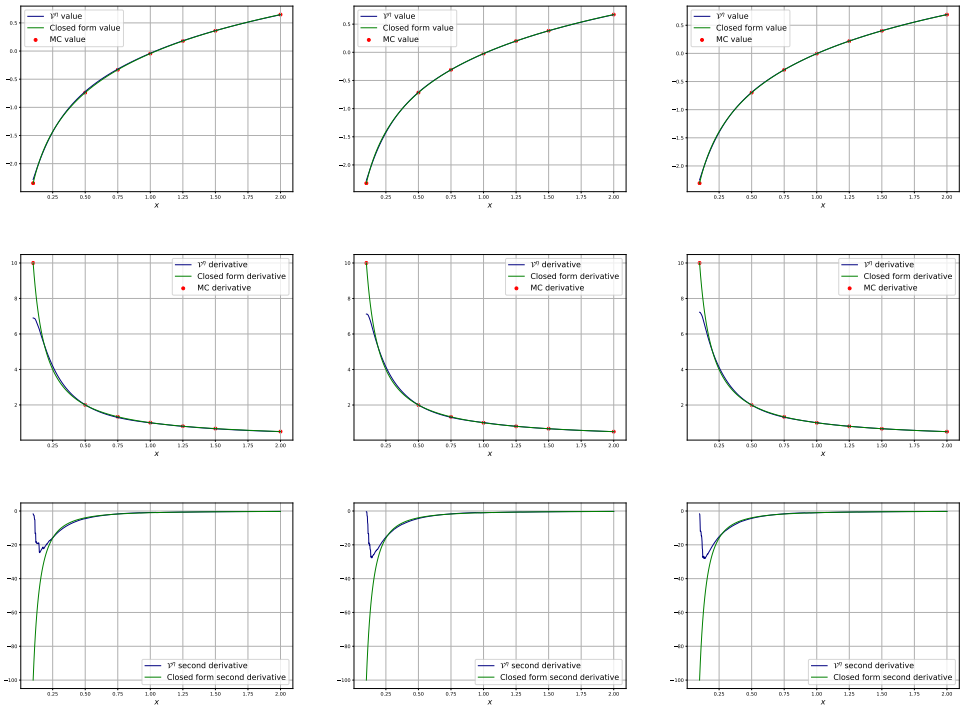
Table 5.5: Residual and boundary losses computed on a 1002x1002 time and space grid with $t \in [0, 0.9]$ and $x \in [0.1, 2]$ for a terminal call-option payoff $g(x) = \max(x - 1, 0)$, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$.

We plot again the value function $\vartheta^\eta(t, x)$ and its derivative $\partial_x\vartheta^\eta(t, x)$ for fixed values $t = 0$, $t = 0.5$, $t = 0.9$, parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$,



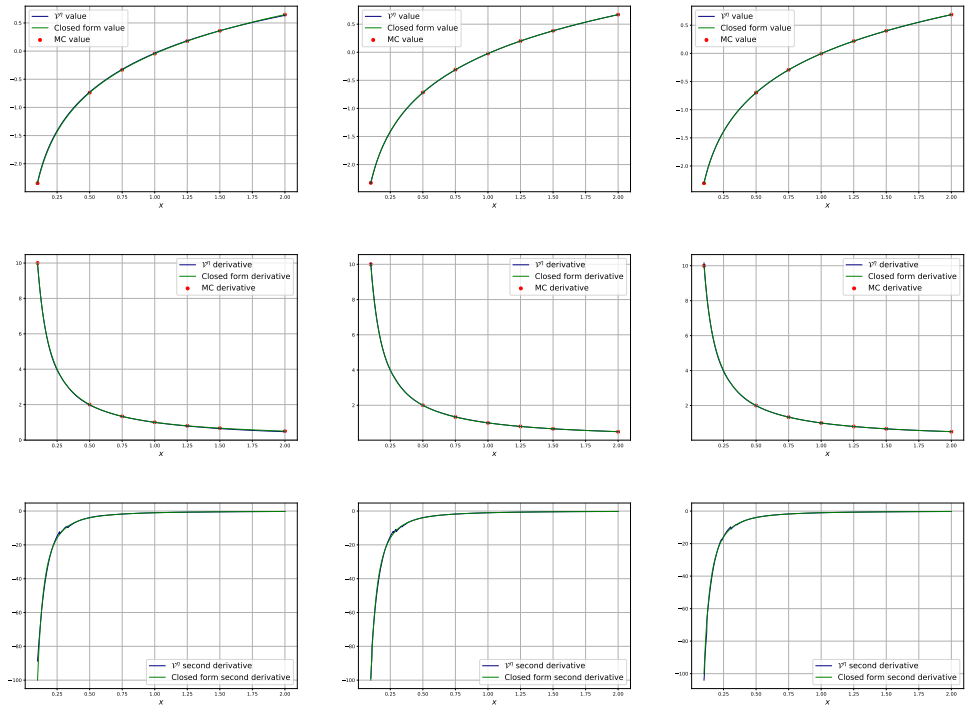
(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 5.6: Value function ϑ^η and its first and second derivative obtained by Differential Regression Learning (Algorithm 5) for a logarithmic option payoff, with parameter $\sigma = 0.2$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

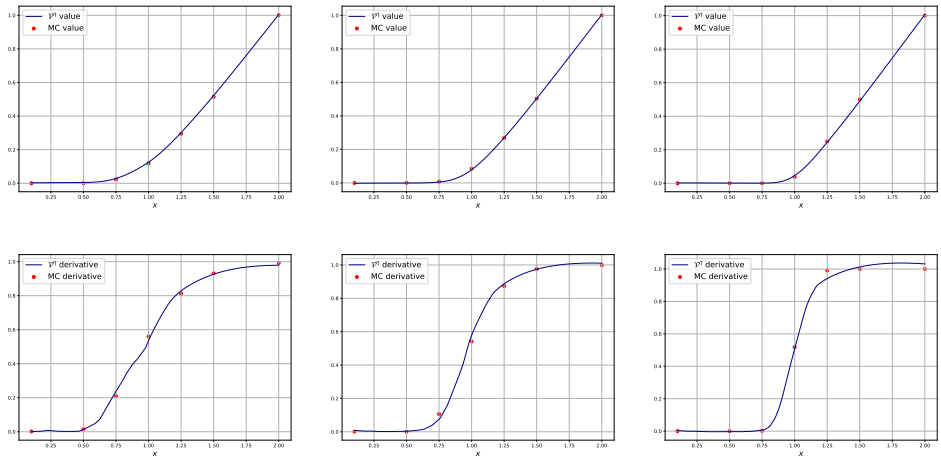
Figure 5.7: Value function ϑ^η and its first and second derivative obtained by Pathwise Learning (Algorithm 6) for a logarithmic option payoff, with parameter $\sigma = 0.2$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 5.8: Value function ϑ^η and its first and second derivative obtained by Pathwise Differential Learning (Algorithm 7) for a logarithmic option payoff, with parameter $\sigma = 0.2$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.

and compare it with the Monte-Carlo estimation obtained. The Figure 5.9 corresponds to the Differential regression learning method, Figure 5.10 corresponds to the pathwise martingale learning while Figure 5.11 corresponds to the Pathwise differential learning method. Graphically, the results of the Differential regression learning and the Pathwise differential learning methods are very close to the points obtained by Monte Carlo estimation for the value and the first derivative. The Pathwise methods does not give a good approximation of the value and the derivatives for values of x smaller than 1. The difference of performance between the Pathwise and the Differential pathwise methods is analogous to the one observed between Differential regression learning and "simple" regression learning in Figure 5.1, demonstrating the interest of adding a regression term for the derivative of the neural network.



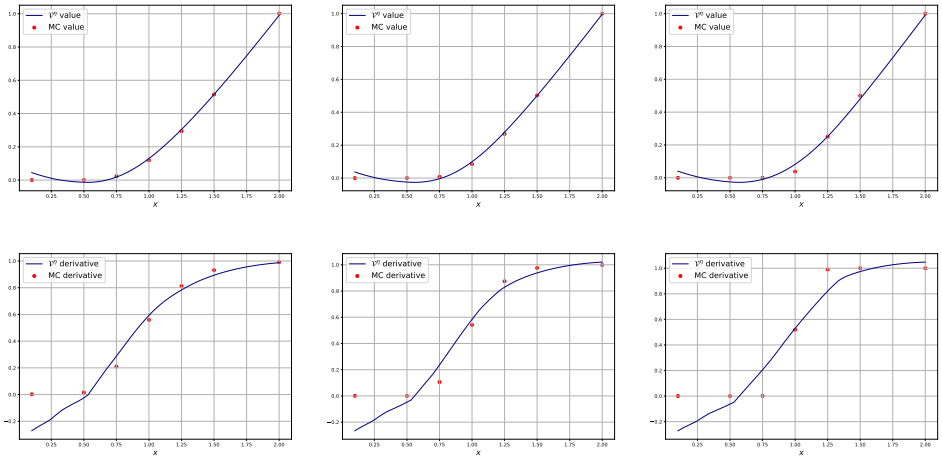
(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 5.9: Value function ϑ^{η} and its derivative obtained by Differential Regression Learning (Algorithm 5) for a call option with strike 1, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.

5.6 Further step: resolution for parametric terminal functions

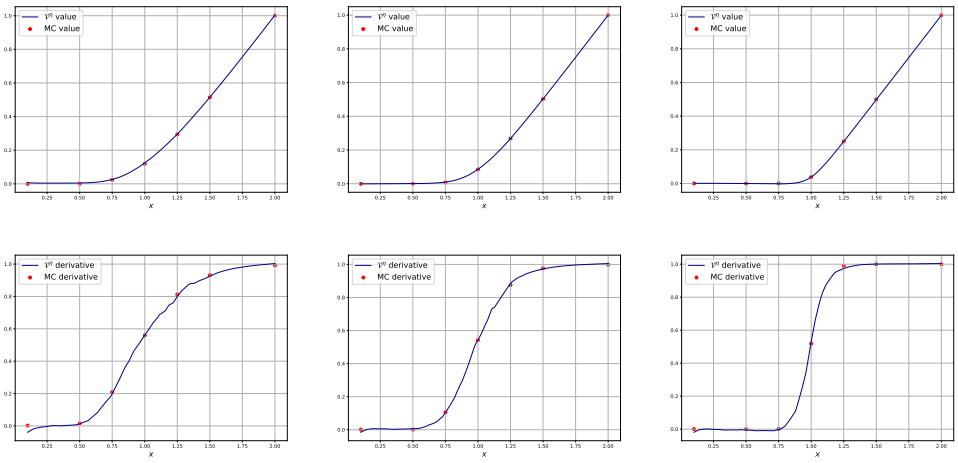
5.6.1 Theory and network structure

In line with the works in [VŠS21], [GW22], [RME22], our next goal is to design a Machine Learning method allowing us to directly obtain a solution of problem (5.2.3) for a parametric terminal condition g_K, for every value of the parameter $K \in \mathbb{R}^p$ in a compact set. In other words, we aim to learn the operator that maps the payoff function parameter K to the solution of the PDE with terminal condition g_K. We want to be able to train the neural networks once and for all on a selection of parameter values and obtain a network which takes a couple (t, x) and the parameter value K and outputs the solution of the problem. In [GW22] the authors solve parametic PDEs by using a variant of highway networks [SGS15], which are feedforward networks where each dense layer has an additional parameter called *gate* which allows the layer to output a combination of the unmodified input and of the output of the affine and activation operations, alleviating the vanishing gradient problem and allowing to train deeper networks. Their network



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 5.10: Value function ϑ^η and its derivative obtained by Pathwise Learning (Algorithm 6) for a call option with strike 1, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 5.11: Value function ϑ^η and its derivative obtained by Pathwise Differential Learning (Algorithm 7) for a call option with strike 1, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.

takes the time, space and PDE parameter as input and is trained in the spirit of the Deep Galerkin method. The PDE residuals is computed from the neural network on a random time, space and parameter grid and is minimized in order to approximate the PDE solution. In [VŠS21] the authors give four methods to solve parametric PDEs using a fully connected network taking the time, space and PDE parameter as input and minimizing a loss averaged over random parameter values. In this section we follow the methodology developped in [RME22] by relying on a class of neural networks, called DeepONet, presented in [LJK19], and aiming to approximate functional operators. This method is based on the following universal approximation theorem for operator, due to Chen and Chen [CC95].

Theorem 5.6.1. *Suppose that σ is a continuous non-polynomial function, X is a Banach space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$, G is a nonlinear continuous operator, which maps V into $C(K_2)$. Then for any $\epsilon > 0$, there are positive integers n, p, m, constants c_i^k, ξ_{ij}^k, θ_i^k, $\zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, ..., n$, $k = 1, ..., p$, $j = 1, ..., m$, such that*

$$\left\| G(u)(y) - \underbrace{\sum_{k=1}^{p} \sum_{i=1}^{n} c_i^k \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{branch\ net} \underbrace{\sigma\left(w_k y + \zeta_k\right)}_{trunk\ net} \right\| < \epsilon,$$

holds for all $u \in V$ and $y \in K_2$.

The network used in [LJK19] is composed of two sub networks, the *branch net*, which takes the terminal function estimated on a fixed number of points called *sensors* as input, and the trunk net, which takes the time and space coordinates as input. In our case, as in [RME22], the *branch net* takes the parametric terminal function estimated on a grid of *sensors*, and will be trained for random values of the function's parameter. We represent the structure of this neural network in Figure 5.12.

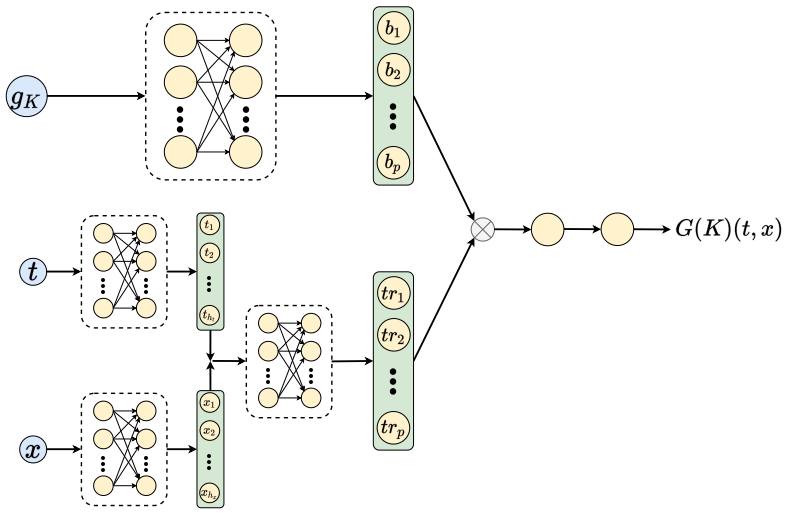


Figure 5.12: Structure of the DeepONet network.

As our numerical methods proceed in two stages, by first approximating the optimal control of problem (5.2.3) with a neural network and then approximating the associated value function with another network, we again use a DeepONet to approximate the optimal control and then another one to approximate the value function u by differential learning.

While the authors of [LJK19] train a DeepONet in a supervised manner, we train our network in an **unsupervised way**.

As before, we start by training the control network a_θ in order to approximate the optimal control of problem. We use a batch of M independent trajectories $\{x_{t_n}^{m,K_m,\theta}, t_n \in \mathcal{T}_N\}$, $m = 1, \dots, M$, of $\{X_{t_n}^{K_m,\theta}, t_n \in \mathcal{T}_N\}$, where the K superscript denotes that the trajectory is driven by a control with input parameter K, and M random parameter values K_m randomly sampled from a distribution μ_K with compact support in $\mathbb{R}^p$, and apply a stochastic gradient ascent method to the empirical gain function:

$$J_M(\theta) = \frac{1}{M}\sum_{m=1}^{M}\Big[g(x_T^{m,K_m,\theta}, K_m) + \sum_{n=0}^{N-1} f\big(x_{t_n}^{m,K_m,\theta}, \mathrm{a}_\theta(t_n, x_{t_n}^{m,K_m,\theta}, K_m)\big)\Delta t_n\Big].$$

The pseudo-code is described in Algorithm 8. The output of this algorithm yields a parameter θ^*, and so an approximation of the optimal feedback control with $\mathrm{a}^* = \mathrm{a}_{\theta^*}$, and of the associated optimal state process with $X^* = X^{\theta^*}$. In the sequel, to alleviate notations, we shall omit the superscript $*$, and simply denote a and X.

Algorithm 8: Deep learning scheme to solve the stochastic control problem (5.2.3)

Result: A set of optimized parameters θ^*;
Initialize the learning rate l and the neural network a_θ;
Generate an $\mathbb{R}^{N+1}$-valued time grid $0 = t_0 < t_1 < \dots < t_N = T$ with time steps $(\Delta t_n)_{n=0,\dots,N-1}$;
Generate a batch of M starting points $X_0 \sim \mu_0$, Brownian increments $(\Delta W_{t_n})_{n=0,\dots,N-1}$ in R^d and parameter values $K \sim \mu_K$;
for *each batch element* m **do**
  Compute the trajectory $(x_{t_n}^{m,K_m,\theta})_{n=0,\dots,N}$ through the scheme

$$x_{t_{n+1}}^{m,K_m,\theta} = x_{t_n}^{m,K_m,\theta} + \mathrm{b}(x_{t_n}^{m,K_m,\theta}, \mathrm{a}_\theta(t_n, x_{t_n}^{m,K_m,\theta}, K_m))\Delta t_n + \sigma(x_{t_n}^{m,K_m,\theta}, \mathrm{a}_\theta(t_n, x_{t_n}^{m,K_m,\theta}, K_m))\Delta w_{t_n}^m,$$

  from the generated starting point $x_{t_0}^m$, Brownian increments $(\Delta w_{t_n}^m)_{n=0,\dots,N-1}$ and parameter K_m;
end
for *each epoch* **do**
  Compute the batch loss

$$J_M(\theta) = \frac{1}{M}\sum_{m=1}^{M}\Big[g(x_T^{m,K_m,\theta}, K_m) + \sum_{n=0}^{N-1} f\big(x_{t_n}^{m,K_m,\theta}, \mathrm{a}_\theta(t_n, x_{t_n}^{m,K_m,\theta}, K_m)\big)\Delta t_n\Big]$$

  Compute the gradients $\nabla_\theta J_M(\theta)$;
  Update $\theta \leftarrow \theta - l\nabla_\theta J_M(\theta)$;
end
Return: The set of optimized parameters θ^*;

From this optimal control approximation, the value function is then approximated through the Differential regression learning algorithm presented in Section 5.5.2 modified in order to train the network for different values of the terminal function parameter K. The target payoff and its derivative are then computed as

$$Y_T^{K_m,t_n} = g(X_T^{K_m}, K_m) + \sum_{q=n}^{N-1} f^{\mathrm{a}^*}(t_q, X_{t_q}^{K_m})\Delta t_q, \quad n = 0, \dots, N,$$

$$Z_T^{K_m,t_n} = \big(D_{X_{t_n}} X_T^{K_m}\big)^\top D_x g(X_T^{K_m}, K_m) + \sum_{q=n}^{N-1}\big(D_{X_{t_n}} X_{t_q}^{K_m}\big)^\top D_x f^{\mathrm{a}^*}(t_q, X_{t_q}^{K_m})\Delta t_p,$$

with the convention that the above sum over q is zero when $n = N$. For the training of the neural network ϑ^η, we use a batch of M independent samples $(x_{t_n}^{m,K_m}, y_T^{m,K_m,t_n}, z_T^{m,K_m,t_n})$,

$m = 1, \dots, M$, of $(X_{t_n}^{K_m}, Y_T^{K_m,t_n}, Z_T^{K_m,t_n})$, $n = 0, \dots, N$ and M random parameter values K^m randomly sampled from a distribution μ_K with compact support in $\mathbb{R}^p$, and apply stochastic gradient descent for the minimization of the mean squared error functions

$$MSE_{val}(\eta) = \frac{1}{M} \sum_{m=1}^{M} \sum_{n=0}^{N-1} \left| y_T^{m,K_m,t_n} - \vartheta^\eta(t_n, x_{t_n}^{m,K_m}, K_m) \right|^2 \Delta t_n$$

$$MSE_{der}(\eta) = \frac{1}{M} \sum_{m=1}^{M} \sum_{n=0}^{N-1} \frac{1}{\|z_T^{K_m,t_n}\|^2} \left| z_T^{m,K_m,t_n} - D_x\vartheta^\eta(t_n, x_{t_n}^{m,K_m}, K_m) \right|^2 \Delta t_n.$$

The pseudo-code is described in Algorithm 9.

Algorithm 9: Deep learning scheme for Differential Regression learning

Result: A set of optimized parameters η^*;
Initialize the learning rate l, the neural networks ϑ^η;
Generate an $\mathbb{R}^{N+1}$-valued time grid $0 = t_0 < t_1 < \dots < t_N = T$ with time steps $(\Delta t_n)_{n=0,\dots,N-1}$;
Generate a batch of M starting points $X_0 \sim \mu_0$, Brownian increments $(\Delta W_{t_n})_{n=0,\dots,N}$ in R^d and parameter values $K \sim \mu_K$;
for *each batch element* m **do**
  Compute the trajectory $(x_{t_n}^{m,K_m})_{n=0,\dots,N}$ through the scheme

$$x_{t_{n+1}}^{m,K_m} = x_{t_n}^{m,K_m} + \mathrm{b}^{\mathrm{a}^*}(t_n, x_{t_n}^{m,K_m})\Delta t_n + \sigma^{\mathrm{a}^*}(t_n, x_{t_n}^{m,K_m})\Delta w_{t_n}^m,$$

  from the generated starting point $x_{t_0}^m$, Brownian increments $(\Delta w_{t_n}^m)_{n=0,\dots,N-1}$, parameter K_m and previously trained control $a = a_{\theta^*}$;
  Compute the value and derivative targets $(y_T^{m,K_m,t_n})_{n=0,\dots,N}$ and $(z_T^{m,K_m,t_n})_{n=0,\dots,N}$;
end
for *each epoch* **do**
  if *Epoch number is even* **then**
    Compute, for every batch element m, the integral $\sum_{n=0}^{N-1} \left| y_T^{m,K_m,t_n} - \vartheta^\eta(t_n, x_{t_n}^{m,K_m}, K_m) \right|^2 \Delta t_n$;
    Compute the batch loss $MSE_{val}(\eta)$;
    Compute the gradient $\nabla_\eta MSE_{val}(\eta)$;
    Update $\eta \leftarrow \eta - l\nabla_\eta MSE_{val}(\eta)$;
  end
  else
    Compute, for every batch element m, the integral $\sum_{n=0}^{N-1} \left| z_T^{m,K_m,t_n} - D_x\vartheta^\eta(t_n, x_{t_n}^{m,K_m}, K_m) \right|^2 \Delta t_n.$;
    Compute the batch loss $MSE_{der}(\eta)$;
    Compute the gradient $\nabla_\eta MSE_{der}(\eta)$;
    Update $\eta \leftarrow \eta - l\nabla_\eta MSE_{der}(\eta)$;
  end
end
Return: The set of optimized parameters η^*;

5.6.2 Application to the Black-Scholes model with linear market impact

Using this DeepONet based algorithm, we revisit the resolution of the nonlinear Black Scholes equation presented in Section 5.5.6. In this Section, we use the Algorithms 8 and 9 in order to solve the PDE for a terminal function corresponding to a call option payoff $g(x, K) = \max(x - K, 0)$ with parameter (or *strike*) $K \in \mathbb{R}_+^*$.

The *branch net* of the **DeepONet used to approximate the control** as a function of the terminal function g_K is a standard feed-forward network composed of two layers with 50 neurons and use the *tanh* activation function. The *trunk net* has the same structure as the network used in the previous sections and represented in Figure 5.2. It is composed of two sub-networks taking respectively the time t and state x as an input and each composed of two layers of 50 neurons using the *tanh* activation function. The outputs of these two sub-networks are contatenated into a vector of $\mathbb{R}^{100}$ which passes through two additionnal layers of 50 neurons using *tanh* activation. The output

of the *branch net* and the *trunk net*, which have the same dimension, are then combined through a dot product whose output passes through a layer of one neuron using *tanh* activation and a layer of one neuron using the *Parametric ReLU* activation function ensuring, as explained in Section 5.5.1, that the control obtained belongs to the control space A.

The **DeepONet used to approximate the value function** u shares the same structure. The branch and trunk net have the same structures as the ones used in the control DeepONet with the same number of layers and neurons per layer and with *Swish* activation function, defined as

$$Swish(x) = \frac{x}{1 + e^{-x}}.$$

After the dot product, the output also passes through a layer composed of one neuron using *Swish* activation function and a last layer of one neuron using no activation function.

Remark 5.6.1. *For the control DeepONet, we used the tanh activation function instead of the ELU activation used in the previous sections as we encountered loss divergences during the training of the DeepONet with ELU activation. Since the tanh activation is bounded, the problem was resolved using this function.*

For the value DeepONet, we used the Swish activation function instead of the ELU activation used in the previous sections as it empirically gave better results. As the ELU function's second derivative is discontinuous, a kink was observed on the value function's derivative we obtained with the DeepONet. This effect was not present when performing the "simple" regression with a standard network in the previous sections, probably because the regression problem is simpler and the true value function fitted with a better accuracy. Since the Swish activation is of class C^∞, we obtained better results, without kinks, using this function.

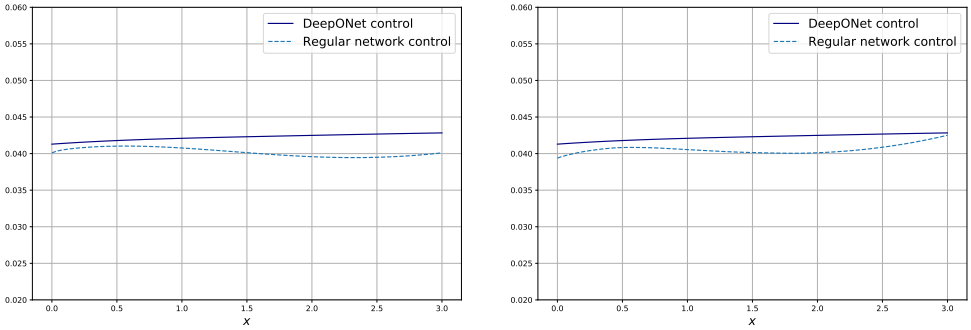
For both trainings, we use the Adam optimizer with a learning rate equal to $1e^{-3}$ and train the network on 8192 random trajectories and strike values. In order to test the generalization power of our method, we train the control and value neural networks on strikes randomly sampled from $\mathcal{U}([0.25, 0.75] \cup [1.5, 2])$ and test the network on strikes chosen in $[0.2, 2.1]$. We plot in figure 5.13 below the optimal control approximation obtained along for $K = 1$ along with the control approximation obtained by Algorithm 4 (denoted *regular network control*) which serves as a reference.

We compute in Table 5.6, the residual losses defined in (5.5.2) for the DeepONet ϑ^η by the Differential Regression Learning scheme (Algorithm 9). We compute the residual losses on a 102x102 linearly spaced time and space grid with $t \in [0, 0.9]$ and $x \in [0, 3]$ for a terminal call option payoff for strikes $K \in \{0.2,\ 0.5,\ 1,\ 1.75,\ 2,\ 2.1\}$, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$.

	$K = 0.2$	$K = 0.5$	$K = 1$	$K = 1.75$	$K = 2$	$K = 2.1$
Residual loss + terminal loss	$1.918e^{-3}$	$1.461e^{-4}$	$1.700e^{-3}$	$1.162e^{-3}$	$2.002e^{-3}$	$2.175e^{-3}$

Table 5.6: Residual and boundary losses computed on a 102x102 time and space grid with $t \in [0, 0.9]$ and $x \in [0, 3]$ for a terminal call option payoff $g(x) = \max(x - K, 0)$ for $K \in \{0.2,\ 0.5,\ 1,\ 1.75,\ 2,\ 2.1\}$, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$.

We plot below the value function $\vartheta^\eta(t, x)$ and its derivative $\partial_x \vartheta^\eta(t, x)$ obtained by Differential Regression Learning with DeepONet networks, for fixed values $t = 0$, $t = 0.5$,



(a) $t = 0$ (b) $t = 0.9$

Figure 5.13: Control a_θ obtained by global method with DeepONet (Algorithm 8) for a call option with strike $K = 1$, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.

$t = 0.9$, parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$, and compare it with the Monte-Carlo estimation obtained. We plot these value functions for strike values $K \in \{0.5,\ 2\}$ inside the training domain and $K \in \{0.2,\ 1,\ 2.1\}$ outside the training domain.

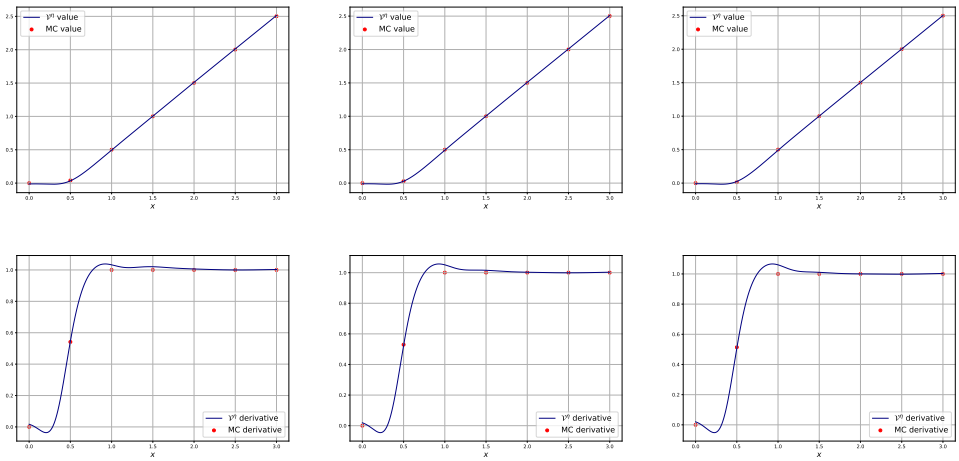


Figure 5.14: Value function ϑ^η (first line) and its derivative (second line) obtained by Differential Regression Learning (Algorithm 9) for a terminal call option payoff with strike $K = 0.5$, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.

On these graphs we see that the estimation of the value of the value function is good and consistent with the Monte Carlo estimator for every strike, inside or outside of the training domain. The estimation of the derivative of the value function is not as good and we can see on the graphs that we get the worst results for the value of the strike closest to zero, $K = 0.2$ (Figure 5.16), and for values of the strike out of the training domain, $K = 1$ (Figure 5.17) and $K = 2.1$ (Figure 5.18).

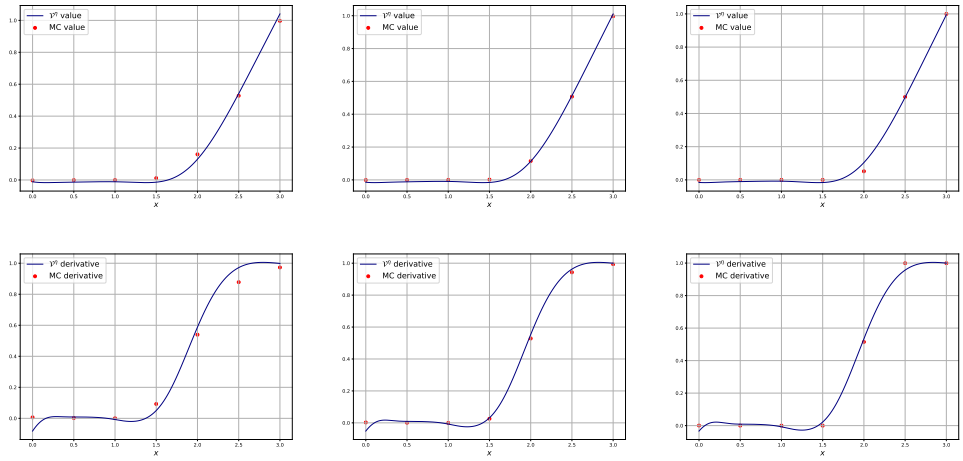
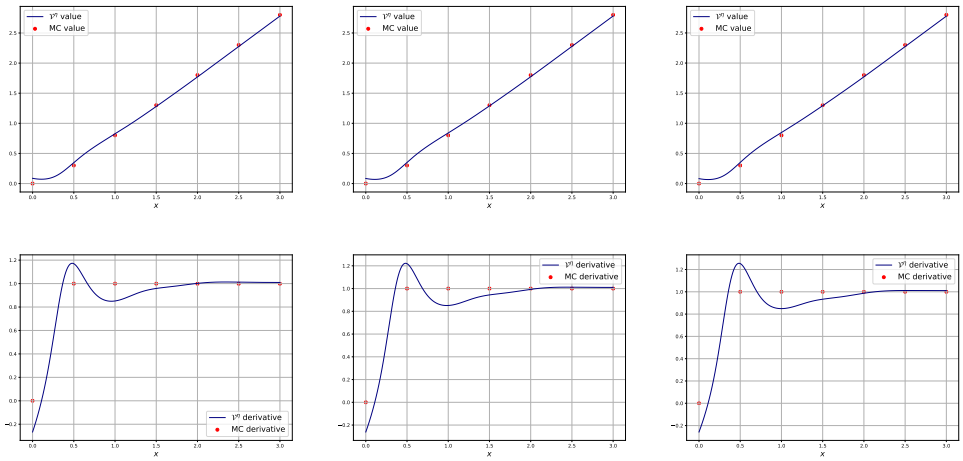
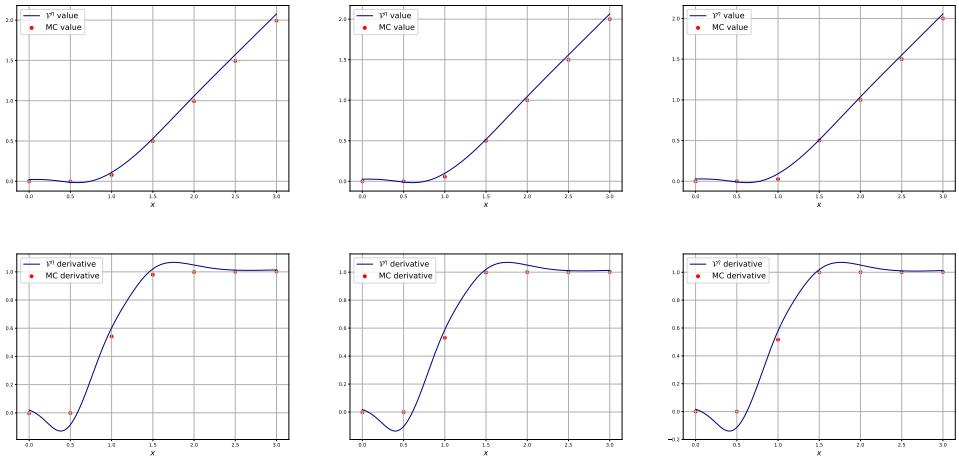


Figure 5.15: Value function ϑ^η (first line) and its derivative (second line) obtained by Differential Regression Learning (Algorithm 9) for a terminal call option payoff with strike $K = 2$, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 5.16: Value function ϑ^η (first line) and its derivative (second line) obtained by Differential Regression Learning (Algorithm 9) for a terminal call option payoff with strike $K = 0.2$, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 0.9$

Figure 5.17: Value function ϑ^η (first line) and its derivative (second line) obtained by Differential Regression Learning (Algorithm 9) for a terminal call option payoff with strike $K = 1$, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.

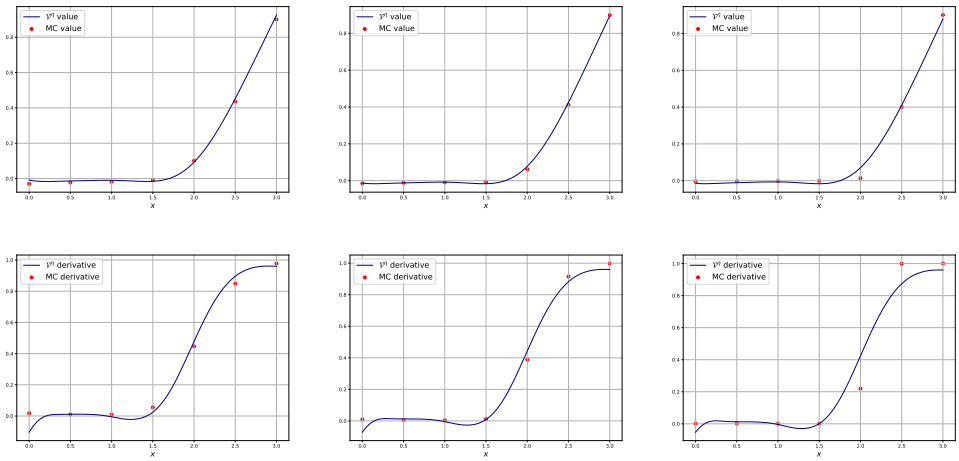


Figure 5.18: Value function ϑ^η (first line) and its derivative (second line) obtained by Differential Regression Learning (Algorithm 9) for a terminal call option payoff with strike $K = 2.1$, with parameter $\sigma = 0.3$ and linear market impact factor $\lambda = 5e^{-3}$, plotted as functions of x, for fixed values of t.

5.A Alternative algorithms using multiple neural networks

We present below the Algorithm 10, which is the version of Algorithm 6 using two neural networks, described in Section 5.5.3.

Algorithm 10: Deep learning scheme for Pathwise martingale learning with 2 NN

Result: A set of optimized parameters η^*, δ^*;
Initialize the learning rate l, the neural networks ϑ^η, $\mathcal{Z}^\delta$;
Generate an $\mathbb{R}^{N+1}$-valued time grid $0 = t_0 < t_1 < ... < t_N = T$ with time steps $(\Delta t_n)_{n=0,\dots,N-1}$;
Generate a batch of M starting points $X_0 \sim \mu_0$ and Brownian increments $(\Delta W_{t_n})_{n=0,\dots,N}$ in R^d;
for *each batch element* m **do**
  Compute the trajectory $(x^m_{t_n})_{n=0,\dots,N}$ through the scheme

$$x^m_{t_{n+1}} = x^m_{t_n} + \mathrm{b}^{\mathrm{a}^*}(t_n, x^m_{t_n})\Delta t_n + \sigma^{\mathrm{a}^*}(t_n, x^m_{t_n})\Delta w^m_{t_n},$$

  from the generated starting point $x^m_{t_0}$, Brownian increments $(\Delta w^m_{t_n})_{n=0,\dots,N-1}$ and previously trained control $a = a_{\theta^*}$;
  Compute the value target $(y^{m,t_n}_T)_{n=0,\dots,N}$;
end
for *each epoch* **do**
  Compute, for every batch element m, the integral
  $\sum_{n=0}^{N-1} \left| y^{m,t_n}_T - \vartheta^\eta(t_n, x^m_{t_n}) - \sum_{p=n}^{N-1} \left(\mathcal{Z}^\delta(t_p, x^m_{t_p})\right)^\top \sigma^{\mathrm{a}^*}(t_p, x^m_{t_p})\Delta w^m_{t_p} \right|^2 \Delta t_n$;
  Compute the batch loss $\tilde{MSE}_{mar}(\eta, \delta)$;
  Compute the gradient $\nabla_\eta \tilde{MSE}_{mar}(\eta, \delta)$ and $\nabla_\delta \tilde{MSE}_{mar}(\eta, \delta)$;
  Update $\eta \leftarrow \eta - l\nabla_\eta \tilde{MSE}_{mar}(\eta, \delta)$, $\delta \leftarrow \delta - l\nabla_\delta \tilde{MSE}_{mar}(\eta, \delta)$;
end
Return: The set of optimized parameters η^*, δ^*;

In the same way, the Algorithm 11 below is the version of Algorithm 7 using three neural networks described in Section 5.5.3.

Algorithm 11: Deep learning scheme for Pathwise differential learning with 3 NN

Result: A set of optimized parameters η^*;
Initialize the learning rate l, the neural networks ϑ^η;
Generate an $\mathbb{R}^{N+1}$-valued time grid $0 = t_0 < t_1 < ... < t_N = T$ with time steps $(\Delta t_n)_{n=0,\dots,N-1}$;
Generate a batch of M starting points $X_0 \sim \mu_0$ and Brownian increments $(\Delta W_{t_n})_{n=0,\dots,N}$ in R^d;
for *each batch element* m **do**
  Compute the trajectory $(x^m_{t_n})_{n=0,\dots,N}$ through the scheme

$$x^m_{t_{n+1}} = x^m_{t_n} + \mathrm{b}^{\mathrm{a}^*}(t_n, x^m_{t_n})\Delta t_n + \sigma^{\mathrm{a}^*}(t_n, x^m_{t_n})\Delta w^m_{t_n},$$

  from the generated starting point $x^m_{t_0}$, Brownian increments $(\Delta w^m_{t_n})_{n=0,\dots,N-1}$ and previously trained control $a = a_{\theta^*}$;
  Compute the value and derivative targets $(y^{m,t_n}_T)_{n=0,\dots,N}$ and $(z^{m,t_n}_T)_{n=0,\dots,N}$;
end
for *each epoch* **do**
  Compute, for every batch element m, the integral
  $\sum_{n=0}^{N-1} \left| y^{m,t_n}_T - \vartheta^\eta(t_n, x^m_{t_n}) - \sum_{p=n}^{N-1} \left(D_x\vartheta^\eta(t_p, x^m_{t_p})\right)^\top \sigma^{\mathrm{a}^*}(t_p, x^m_{t_p})\Delta w^m_{t_p} \right|^2 \Delta t_n$;
  Compute the batch loss $MSE_{mar}(\eta)$;
  Compute the gradient $\nabla_\eta MSE_{mar}(\eta)$;
  Update $\eta \leftarrow \eta - l\nabla_\eta MSE_{mar}(\eta)$;
  Compute, for every batch element m, the integral
  $\sum_{n=0}^{N-1} \Big| z^{m,t_n}_T - D_x\vartheta^\eta(t_n, x^m_{t_n}) - \sum_{p=n}^{N-1} \Big(\left[D_x\sigma^{\mathrm{a}^*}(t_p, x^m_{t_p}) \bullet_3 D_{x_{t_n}} x^m_{t_p}\right] \bullet_1 D_x\vartheta^\eta(t_p, x^m_{t_p})$
  $+\sigma^{\mathrm{a}^*}(t_p, x^m_{t_p})^\top D_{xx}\vartheta^\eta(t_p, x^m_{t_p}) D_{x_{t_n}} x^m_{t_p} \Big)^\top \Delta w^m_{t_p} \Big|^2 \Delta t_n$;
  Compute the batch loss $MSE_{dermar}(\eta)$;
  Compute the gradient $\nabla_\eta MSE_{dermar}(\eta)$;
  Update $\eta \leftarrow \eta - l\nabla_\eta MSE_{dermar}(\eta)$;
end
Return: The set of optimized parameters η^*;

Bibliography

[AD11] Daniel Andersson and Boualem Djehiche. "A maximum principle for SDEs of mean-field type". In: *Applied Mathematics & Optimization* 63.3 (2011), pp. 341–356.

[Ale+71] Yogish Alekal, Pavol Brunovsky, DH Chyung, and E Lee. "The quadratic problem for systems with time delays". In: *IEEE Transactions on Automatic Control* 16.6 (1971), pp. 673–687.

[AZ99] Patrick K Asea and Paul J Zak. "Time-to-build and cycles". In: *Journal of economic dynamics and control* 23.8 (1999), pp. 1155–1175.

[Bam08] Mauro Bambi. "Endogenous growth and time-to-build: The AK case". In: *Journal of Economic Dynamics and Control* 32.4 (2008), pp. 1015–1040.

[BB12] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization." In: *Journal of machine learning research* 13.2 (2012).

[Bec+20] Christain Beck, Martin Hutzenthaler, Arnulf Jentzen, and Benno Kuckuck. "An overview on deep learning-based approximation methods for partial differential equations". In: *arXiv preprint: 2012.12348* (2020).

[Bec+21] Chistian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. "Deep splitting method for parabolic PDEs". In: *SIAM Journal on Scientific Computing* 43.5 (2021).

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

[Ben+07] Alain Bensoussan, Giuseppe Da Prato, Michel C Delfour, and Sanjoy K Mitter. *Representation and control of infinite dimensional systems*. Springer Science & Business Media, 2007.

[BFG12] Mauro Bambi, Giorgio Fabbri, and Fausto Gozzi. "Optimal policy and consumption smoothing effects in the time-to-build AK model". In: *Economic Theory* 50.3 (2012), pp. 635–669.

[BP19] Matteo Basei and Huyên Pham. "A weak martingale approach to linear-quadratic McKean–Vlasov stochastic control problems". In: *Journal of Optimization Theory and Applications* 181.2 (2019), pp. 347–382.

[BT22] Bruno Bouchard and Xiaolu Tan. "Understanding the dual formulation for the hedging of path-dependent options with price impact". In: *The Annals of Applied Probability* 32.3 (2022), pp. 1705–1733.

[Car+18] René Carmona, Jean-Pierre Fouque, Seyyed Mostafa Mousavi, and Li-Hsien Sun. "Systemic risk and stochastic games with delay". In: *Journal of Optimization Theory and Applications* 179.2 (2018), pp. 366–399.

[CC95] Tianping Chen and Hong Chen. "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems". In: *IEEE Transactions on Neural Networks* 6.4 (1995), pp. 911–917.

[Cha12] Sohail Chand. "On tuning parameter selection of lasso-type methods-a monte carlo study". In: *Proceedings of 2012 9th international Bhurban conference on applied sciences & technology (IBCAST)*. IEEE. 2012, pp. 120–129.

[Che+13] Caihua Chen, Xindan Li, Caleb Tolman, Suyang Wang, and Yinyu Ye. "Sparse portfolio selection via quasi-norm regularization". In: *arXiv preprint arXiv:1312.6350* (2013).

[dAVV12] Hippolyte d'Albis, Emmanuelle Augeraud-Véron, and Alain Venditti. "Business cycle fluctuations and learning-by-doing externalities in a one-sector model". In: *Journal of Mathematical Economics* 48.5 (2012), pp. 295–308.

[DeM+09] Victor DeMiguel, Lorenzo Garlappi, Francisco J Nogales, and Raman Uppal. "A generalized approach to portfolio optimization: Improving performance by constraining portfolio norms". In: *Management science* 55.5 (2009), pp. 798–812.

[DFNP19] Carmine De Franco, Johann Nicolle, and Huyên Pham. "Bayesian learning for the Markowitz portfolio selection problem". In: *International Journal of Theoretical and Applied Finance* 22.07 (2019), p. 1950037.

[DL09] Ran Duchin and Haim Levy. "Markowitz versus the Talmudic portfolio diversification strategies". In: *The Journal of Portfolio Management* 35.2 (2009), pp. 71–74.

[DM72] Michel C Delfour and Sanjoy K Mitter. "Hereditary differential systems with constant delays. I. General case". In: *Journal of Differential Equations* 12.2 (1972), pp. 213–235.

[DPT22] Mao Fabrice Djete, Dylan Possamaï, and Xiaolu Tan. "McKean–Vlasov optimal control: the dynamic programming principle". In: *The Annals of Probability* 50.2 (2022), pp. 791–833.

[EHJ17] Weinan E., Jiequn Han, and Arnulf Jentzen. "Deep Learning-Based numerical methods for high dimensional parabolic partial differential equations and backward stochastic differential equations". In: *Commun. Math. Stat.* 5.4 (2017), pp. 349–380.

[EK81] Nicole El Karoui. "Les aspects probabilistes du contrôle stochastique". In: *École d'été de Probabilités de Saint-Flour IX-1979*. Springer, 1981, pp. 73–238.

[EKQP97] Nicole El Karoui, Marie-Claire Quenez, and Shige Peng. "Backward stochastic differential applications in finance". In: *Mathematical Finance* 7.1 (1997), pp. 1–71.

[FF14] Giorgio Fabbri and Salvatore Federico. "On the infinite-dimensional representation of stochastic controlled systems with delayed control in the diffusion term". In: *Mathematical Economics Letters* 2.3-4 (2014), pp. 33–43.

[FGG10] Salvatore Federico, Ben Goldys, and Fausto Gozzi. "HJB equations for the optimal control of differential equations with delays and state constraints, I: regularity of viscosity solutions". In: *SIAM Journal on Control and Optimization* 48.8 (2010), pp. 4910–4937.

[FHZ10] Frank J Fabozzi, Dashan Huang, and Guofu Zhou. "Robust portfolios: contributions from operations research and finance". In: *Annals of operations research* 176.1 (2010), pp. 191–220.

[FL16] Markus Fischer and Giulia Livieri. "Continuous time mean-variance portfolio optimization through the mean field approach". In: *ESAIM: Probability and Statistics* 20 (2016), pp. 30–44.

[FZ20] Jean-Pierre Fouque and Zhaoyu Zhang. "Deep learning methods for mean field control problems with delay". In: *Frontiers in Applied Mathematics and Statistics* 6 (2020), p. 11.

[Gla13] Paul Glasserman. *Monte Carlo methods in financial engineering.* Vol. 53. Springer Science & Business Media, 2013.

[GLT20] Jérôme Gava, William Lefebvre, and Julien Turc. "Beyond carry and momentum in government bonds". In: *The Journal of Fixed Income* 29.4 (2020), pp. 48–74.

[GM05] E. Gobet and R. Munos. "Sensitivity analysis using Itô-Malliavin calculus and martingales, and application to stochastic optimal control". In: *SIAM J. Control Optim.* 43.5 (2005), pp. 1676–1713.

[GM19] Olivier Guéant and Iuliia Manziuk. "Deep reinforcement learning for market making in corporate bonds: beating the curse of dimensionality". In: *Applied Mathematical Finance* 26.5 (2019), pp. 387–452.

[GMP20] Olivier Guéant, Iuliia Manziuk, and Jiang Pu. "Accelerated share repurchase and other buyback programs: what neural networks can bring". In: *Quantitative Finance* 20.8 (2020), pp. 1389–1404.

[GMS09] Fausto Gozzi, Carlo Marinelli, and Sergei Savin. "On controlled linear diffusions with delay in a model of optimal advertising under uncertainty with memory effects". In: *Journal of optimization theory and applications* 142.2 (2009), pp. 291–321.

[GPW21] Maximilien Germain, Huyên Pham, and Xavier Warin. "Neural networks-based algorithms for stochastic control and PDEs in finance". In: *to appear in Machine learning for financial markets: a guide to contemporary practices* (2021).

[GRM05] Fausto Gozzi, Sociali di Roma, and Carlo Marinelli. "Stochastic Optimal Control of Delay Equations Arising in Advertising Models". In: *Stochastic Partial Differential Equations and Applications-VII* (2005), p. 133.

[Gro19] Thomas Hakon Gronwall. "Note on the derivatives with respect to a parameter of the solutions of a system of differential equations". In: *Annals of Mathematics* (1919), pp. 292–296.

[Guo+22] Ivan Guo, Nicolas Langrené, Grégoire Loeper, and Wei Ning. "Robust utility maximization under model uncertainty via a penalization approach". In: *Mathematics and Financial Economics* 16.1 (2022), pp. 51–88.

[GW22] Kathrin Glau and Linus Wunderlich. "The deep parametric PDE method and applications to option pricing". In: *Applied Mathematics and Computation* 432 (2022), p. 127355.

[Hal+77] Robert E Hall, Christopher A Sims, Franco Modigliani, and William Brainard. "Investment, interest rates, and the effects of stabilization policies". In: *Brookings papers on economic activity* 1977.1 (1977), pp. 61–121.

[HE16] Jiequn Han and Weinan E. "Deep learning approximation for stochastic control problems". In: *Deep Reinforcement Learning Workshop, NIPS, arXiv preprint: 1611.07422* (2016).

[HH21] Jiequn Han and Ruimeng Hu. "Recurrent neural networks for stochastic control problems with delay". In: *Mathematics of Control, Signals, and Systems* 33.4 (2021), pp. 775–795.

[HJE17] Jiequn Han, Arnulf Jentzen, and Weinan E. "Solving high-dimensional partial differential equations using deep learning". In: *Proc. Natl. Acad. Sci. USA* 115 (2017).

[HJW18] Jiequn Han, Arnulf Jentzen, and E Weinan. "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences* 115.34 (2018), pp. 8505–8510.

[HL22] Ruimeng Hu and Mathieu Lauriere. "Recent Developments in Machine Learning Methods for Stochastic Control and Games". In: *Recent Developments in Machine Learning Methods for Stochastic Control and Games (May 13, 2022)* (2022).

[HLLR16] Pierre Henry-Labordere, Christian Litterer, and Zhenjie Ren. "A dual algorithm for stochastic control problems: Applications to uncertain volatility models and CVA". In: *SIAM Journal on Financial Mathematics* 7.1 (2016), pp. 159–182.

[HNW93] Ernst Hairer, Syvert P Nørsett, and Gerhard Wanner. *Solving ordinary differential equations I. Nonstiff problems, volume 8 of.* 1993.

[HPW20] Côme Huré, Huyên Pham, and Xavier Warin. "Deep backward schemes for high-dimensional nonlinear PDEs". In: *Mathematics of Computation* 89.324 (2020), pp. 1547–1579.

[HS20] Brian Norsk Huge and Antoine Savine. "Differential Machine Learning". In: *Available at SSRN 3591734* (2020).

[HSX15] Michael Ho, Zheng Sun, and Jack Xin. "Weighted elastic net penalized mean-variance portfolio design and computation". In: *SIAM Journal on Financial Mathematics* 6.1 (2015), pp. 1220–1244.

[Hur+21] Côme Huré, Huyên Pham, Achref Bachouch, and Nicolas Langrené. "Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis". In: *SIAM Journal on Numerical Analysis* 59.1 (2021), pp. 525–557.

[Huz+02] Mihai Huzmezan, William A Gough, Guy A Dumont, and Sava Kovac. "Time delay integrating systems: a challenge for process control industries. A practical solution". In: *Control Engineering Practice* 10.10 (2002), pp. 1153–1161.

[Ich82] Akira Ichikawa. "Quadratic control of evolution equations with delays in control". In: *SIAM Journal on control and optimization* 20.5 (1982), pp. 645–668.

[IP19] Amine Ismail and Huyên Pham. "Robust Markowitz mean-variance portfolio selection under ambiguous covariance matrix". In: *Mathematical Finance* 29.1 (2019), pp. 174–207.

[JD07] Elias Jarlebring and Tobias Damm. "The Lambert W function and the spectrum of some multidimensional time-delay systems". In: *Automatica* 43.12 (2007), pp. 2124–2128.

[Ji+20] Shaolin Ji, Shige Peng, Ying Peng, and Xichuan Zhang. "Three algorithms for solving high-dimensional fully coupled FBSDE through deep learning". In: *IEEE Intelligent Systems* 35.3 (2020), pp. 71–84.

[KP82] Finn E Kydland and Edward C Prescott. "Time to build and aggregate fluctuations". In: *Econometrica: Journal of the Econometric Society* (1982), pp. 1345–1370.

[LJK19] Lu Lu, Pengzhan Jin, and George Em Karniadakis. "Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators". In: *arXiv preprint arXiv:1910.03193* (2019).

[LLP20] William Lefebvre, Gregoire Loeper, and Huyên Pham. "Mean-variance portfolio selection with tracking error penalization". In: *Mathematics* 8.11 (2020), p. 1915.

[LLP22] William Lefebvre, Grégoire Loeper, and Huyên Pham. "Differential learning methods for solving fully nonlinear PDEs". In: *arXiv preprint arXiv:2205.09815* (2022).

[LM21] William Lefebvre and Enzo Miller. "Linear-quadratic stochastic delayed control and deep learning resolution". In: *Journal of Optimization Theory and Applications* 191.1 (2021), pp. 134–168.

[Loe18] Grégoire Loeper. "Option pricing with linear market impact and nonlinear black–scholes equations". In: *Annals of Applied Probability* 28.5 (2018), pp. 2664–2726.

[LR14] Qian Lin and Frank Riedel. "Optimal consumption and portfolio choice with ambiguity". In: *arXiv preprint arXiv:1401.1639* (2014).

[LS01] Francis A Longstaff and Eduardo S Schwartz. "Valuing American options by simulation: a simple least-squares approach". In: *The review of financial studies* 14.1 (2001), pp. 113–147.

[LZ16] Jun Liu and Xudong Zeng. "Correlation ambiguity". In: *Available at SSRN 2692692* (2016).

[Mar52] Harry Markowitz. "Portfolio Selection". In: *The Journal of Finance* 7.1 (1952), pp. 77–91. DOI: 10.1111/j.1540-6261.1952.tb01525.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1952.tb01525.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1952.tb01525.x.

[MOB21] Remco van der Meer, Cornelis W Oosterlee, and Anastasia Borovykh. "Optimally weighted loss functions for solving pdes with neural networks". In: *Journal of Computational and Applied Mathematics* (2021), p. 113887.

[MPZ15] Anis Matoussi, Dylan Possamaï, and Chao Zhou. "Robust utility maximization in nondominated models with 2BSDE: the uncertain volatility model". In: *Mathematical Finance* 25.2 (2015), pp. 258–287.

[MRT10] Sébastien Maillard, Thierry Roncalli, and Jérôme Teiletche. "The properties of equally weighted risk contribution portfolios". In: *The Journal of Portfolio Management* 36.4 (2010), pp. 60–70.

[MZ02] Jin Ma and Jianfeng Zhang. "Representation theorems for backward stochastic differential equations". In: *Annals of Applied Probability* 12.4 (2002), pp. 1390–1418.

[NAO21] Balint Negyesi, Kristoffer Andersson, and Cornelis Oosterlee. "The One step Malliavin scheme: new discretization of BSDEs implemented with deep learning regressions". In: *arXiv preprint arXiv:2110.05421* (2021).

[NN18] Ariel Neufeld and Marcel Nutz. "Robust utility maximization with Lévy processes". In: *Mathematical Finance* 28.1 (2018), pp. 82–105.

[NR21] Nikolas Nüskens and Lorenz Richter. "Interpolating between BSDEs and PINNs: deep learning for elliptic and parabolic boundary value problems". In: *arXiv:2112.03749* (2021).

[Nua95] David Nualart. *The Malliavin calculus and related topics.* Springer-Verlag, Berlin, 1995.

[Pau77] Wilfried Pauwels. "Optimal dynamic advertising policies in the presence of continuously distributed time lags". In: *Journal of Optimization Theory and Applications* 22.1 (1977), pp. 79–89.

[PBS01] Marc Potters, Jean-Philippe Bouchaud, and Dragan Sestovic. "Hedged Monte-Carlo: low variance derivative pricing with objective probabilities". In: *Physica A: Statistical Mechanics and its Applications* 289.3-4 (2001), pp. 517–525.

[Pha09] Huyên Pham. *Continuous-time stochastic control and optimization with financial applications.* Vol. 61. Springer Science & Business Media, 2009.

[Pin16] Mustafa Pinar. "On robust mean-variance portfolios". In: *Optimization* 65.5 (2016), pp. 1039–1048.

[Pro05] Philip E Protter. "Stochastic differential equations". In: *Stochastic integration and differential equations.* Springer, 2005, pp. 249–361.

[PW17] Huyên Pham and Xiaoli Wei. "Dynamic programming for optimal control of stochastic McKean–Vlasov dynamics". In: *SIAM Journal on Control and Optimization* 55.2 (2017), pp. 1069–1101.

[PWG21] Huyên Pham, Xavier Warin, and Maximilien Germain. "Neural networks-based backward scheme for fully nonlinear PDEs". In: *SN Partial Differential Equations and Applications* 2.1 (2021), pp. 1–24.

[PWZ22] Huyen Pham, Xiaoli Wei, and Chao Zhou. "Portfolio diversification and model uncertainty: A robust dynamic mean-variance approach". In: *Mathematical Finance* 32.1 (2022), pp. 349–404.

[RME22] Carl Remlinger, Joseph Mikael, and Romuald Elie. "Robust Operator Learning to Solve PDE". In: (2022).

[Ron13] Thierry Roncalli. *Introduction to risk parity and budgeting.* CRC Press, 2013.

[RPK19] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[Run+19] Francesco Rundo, Francesca Trenta, Agatino Luigi di Stallo, and Sebastiano Battiato. "Grid trading system robot (gtsbot): A novel mathematical algorithm for trading fx market". In: *Applied Sciences* 9.9 (2019), p. 1796.

[Set74] Suresh P Sethi. "Sufficient conditions for the optimal control of a class of systems with continuous lags". In: *Journal of Optimization Theory and Applications* 13.5 (1974), pp. 545–552.

[SGS15] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Training very deep networks". In: *Advances in neural information processing systems* 28 (2015).

[Sip+11] Rifat Sipahi, Silviu-Iulian Niculescu, Chaouki T Abdallah, Wim Michiels, and Keqin Gu. "Stability and stabilization of systems with time delay". In: *IEEE Control Systems Magazine* 31.1 (2011), pp. 38–65.

[SS18] Justin Sirignano and Konstantinos Spiliopoulos. "DGM: A deep learning algorithm for solving partial differential equations". In: *Journal of Computational Physics* 375 (2018), pp. 1339–1364.

[STZ+13] H Mete Soner, Nizar Touzi, Jianfeng Zhang, et al. "Dual formulation of second order target problems". In: *The Annals of Applied Probability* 23.1 (2013), pp. 308–347.

[TG99] Yu-Chu Tian and Furong Gao. "Control of integrator processes with dominant time delay". In: *Industrial & engineering chemistry research* 38.8 (1999), pp. 2979–2983.

[TLT20] Van-Dai Ta, Chuan-Ming Liu, and Direselign Addis Tadesse. "Portfolio optimization-based stock prediction using long-short term memory network in quantitative trading". In: *Applied Sciences* 10.2 (2020), p. 437.

[Tso11] John D Tsoukalas. "Time to build capital: Revisiting investment-cash-flow sensitivities". In: *Journal of Economic Dynamics and Control* 35.7 (2011), pp. 1000–1016.

[VŠS21] Marc Sabate Vidales, David Šiška, and Lukasz Szpruch. "Unbiased Deep Solvers for Linear Parametric PDEs". In: *Applied Mathematical Finance* 28.4 (2021), pp. 299–329.

[YZ99] Jiongmin Yong and Xun Yu Zhou. *Stochastic controls: Hamiltonian systems and HJB equations*. Vol. 43. Springer Science & Business Media, 1999.

[ZL00] Xun Yu Zhou and Duan Li. "Continuous-time mean-variance portfolio selection: A stochastic LQ framework". In: *Applied Mathematics and Optimization* 42.1 (2000), pp. 19–33.

[Zou06] Hui Zou. "The adaptive lasso and its oracle properties". In: *Journal of the American statistical association* 101.476 (2006), pp. 1418–1429.

Stochastic control methods applied to portfolio construction, control with delay and PDE solving

The present thesis deals with stochastic control problems and methods applied to the resolution of problems arising in the field of quantitative finance, such as portfolio selection and resolution of non linear PDEs associated to the construction of investment strategies and the pricing of derivative products. It is divided into three parts.

In the first part, we solve a mean variance portfolio selection problem where the portfolio is penalized by a distance between the wealth invested in each of its assets and the composition of a reference portfolio with fixed weights. The optimal control and value function are obtained in closed form and an analogue of the efficient frontier formula is obtained in the limit where the penalisation tends to zero. The robustness of this allocation is tested on simulated prices with parameter misspecification.

The second part deals with the delayed control of stochastic differential equations. We solve a simple linear quadratic stochastic control problem where the control appears both in the drift and diffusion part of the state and is affected by a delay. The expressions of the optimal control and value function are obtained in terms of the solution of a system of coupled Riccati PDEs for which the existence and uniqueness of a solution is proven, provided that a condition, combining the time horizon, the delay, the drift and the volatility of the state SDE is satisfied. A deep learning method is used to solve the system Riccati PDEs in the context of Markovitz portfolio selection with execution delay.

In the third part, three methods based on deep learning are defined in order to solve fully non linear PDEs with convex Hamiltonian. These methods use the stochastic representation form of the PDE, whose optimal control is approximated numerically, in order to obtain three different estimators of the PDE solution based on regression or pathwise versions of the martingale representation and its differential relation, and compute simultaneously the solution and its derivatives. We further leverage our methods to design algorithms for solving families of PDEs with parametric terminal condition by means of DeepOnet neural networks.

Keywords: Continuous-time mean-variance problem, tracking error, robustified allocation, parameter misspecification, Linear-quadratic stochastic control, delay, Riccati PDEs, Markowitz portfolio allocation, Fully nonlinear PDEs, deep learning, differential learning, option pricing with market impact.

List of papers being part of this thesis:

- William Lefebvre, Gregoire Loeper, and Huyên Pham. "Mean-variance portfolio selection with tracking error penalization". In: *Mathematics* 8.11 (2020), p. 1915.
- William Lefebvre and Enzo Miller. "Linear-quadratic stochastic delayed control and deep learning resolution". In: *Journal of Optimization Theory and Applications* 191.1 (2021), pp. 134–168.
- William Lefebvre, Grégoire Loeper, and Huyên Pham. "Differential learning methods for solving fully nonlinear PDEs". In: *arXiv preprint arXiv:2205.09815* (2022).