Deep learning methods for high dimensional PDE's

An application to N-agent games

by

Carlos Daniel Contreras Quiroz

Advisor: Mauricio Junca

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Master in
Mathematics

at the
Universidad de los Andes
2023

Acknowledgements

I want to acknowledge all the people whose work has brought me where I am in life. Without them, nothing I have accomplished would be possible.

I am grateful for all my family, as they make my life happier. Especially, I would like to thank Mom and Dad for their constant support in all ways.

Also, I would like to thank my professors, from whom I have learned a lot and have inspired me to work harder every day. Particularly, I want to thank Mauricio for his help and support during this project, but also Michael, Sasha and Ahmed, all have contributed enormously to my academic and personal path.

Contents

List of Figures

Chapter 1

Introduction

Partial differential equations are common tools for modelling phenomena in various disciplines. We can find many of them in physics, biology, engineering and finance. However, their mathematical behavior is as diverse as the areas in which they can be applied. Even if we can establish a partial classification using some properties of the equations, there cannot exit a complete theory that contains all our understanding about them

By the other hand, in general we have no access to analytical solutions to such models, and thus me may require numerical methods to obtain approximate solutions. To do this, we have well studied tools as finite element, finite differences, finite volumes and spectral methods. There is strong theory that supports rigorously its convergence and when can they be used and when they can fail.

However, applying these methods to particular problems is restricted in many cases by the nature of the equations they are applied to. For example, the aforementioned methods suffer from what is called *the curse of dimensionality*, i.e, their computational complexity grows exponentially in the dimension of the problem. Hence, they are restricted to low dimensional cases($n = 1, 2, 3, 4$). This is a problem in areas like financial mathematics or image denoising, where the dimension of the PDE to be solved is determined by the number of underlying assets in the former case, and the size of the image in the latter. Moreover, there may exist other problems besides the dimension of the equation, for example, many non-linearities or complex domains are usually intractable.

Another area where those problems have appeared is in data analysis and machine learning problems. For instance, the complexity of some non-linear regression models grows exponentially with the size of the subjacent data. For those problems, we have developed many powerful tools that allow us to deal with high-dimensional and non-linear problems. Particularly, neural networks have had considerable successful to model problems with such complexities[1].

Therefore, trying to replicate this achievement in machine learning in the context of partial differential equations, recently have emerged new perspectives to approximate solutions using the same set of tools. Between them, we can find Physics Informed Neural Networks(PINNS) [2, 3] and Fourier Neural Operators (FNO)[4]. Empirical evidence suggest that those methods can approximate solutions where classical discretization methods cannot[5, 6], despite usually not being competitive in terms of accuracy in the cases where both work. Moreover, there is not yet a well established theoretical framework that justify when and how they should be used, but it is rapidly evolving subject of study.

In this work we focus on a family of these new methods, named the Deep BSDE methods, that was originally proposed in [7]. They exploit the close relation that exists between forward backward stochastic differential equations and a certain class of non-linear partial differential equations. Hence, our problem can be transformed in a learning one for a neural network approximation that we will train using data associated to a Monte Carlo simulation of diffusion paths.

We review some theoretical background that is needed to establish the method, and perform some numerical tests for solving a control problem using the Hamilton-Jacobi-Bellman in high-dimensions. Finally, we use such tools for solving a game theory problem with interacting agents trying to optimize individual cost functions.

Chapter 2

Backward stochastic differential equations and PDEs

When addressing deterministic optimal control problems of dynamical systems, there are two approaches, one involving Bellman's dynamic programming principle, and the other relying on the Pontryagin's maximum principle. The former approach leads to a partial differential equation, the Hamilton-Jacobi-Bellman equation, to be solved for the value function and the optimal control of the process. The latter leads to a system of ordinary differential equations, one equation forward in time for the state and one backward in time for its adjoint.

The stochastic version of these problems is solved by methods analogous to those of the deterministic case. However, there are issues with desirable mathematical properties of solutions when we state them extending directly the ones proposed by deterministic methods. That is the case of the stochastic version of the Pontryagin's maximum principle, in which the backward differential equation cannot be stated directly as an SDE with terminal condition, as the solution is not guaranteed to be adapted to the filtration generated by the brownian motion.

The theory of backward stochastic differential equations (BSDEs) emerged in Bismut's [8] early work, and later generalized by Pardoux and Peng [9], as an attempt to formalize the application of the stochastic maximum principle. Here we give an introduction and compilation of results about them based on [10, 11, 12, 13], including its relation with a certain class of nonlinear parabolic partial differential equations, which will be the main tool for the method explained in the following chapters.

2.1 Backward stochastic differential equations

2.1.1 Motivation

Let's introduce the necessity for a different formulation of stochastic differential equations through an example [12]. In the usual setting for a stochastic differential equation (SDE), we specify the evolution of a $\mathbb{R}^n$-valued stochastic process X_t through its dynamics and an initial value $x_0 \in \mathbb{R}^n$(possibly random), in the form

$$X_t = x_0 + \int_0^t \mu(t, X_t)dt + \int_0^t \sigma(t, X_t)dW_t, \tag{2.1}$$

or equivalently,

$$\begin{aligned} dX_t &= \mu(t, X_t)dt + \sigma(t, X_t)dW_t \\ X_0 &= x_0, \end{aligned} \tag{2.2}$$

where W_t is a d-dimensional Brownian motion process and the stochastic integral is defined in the Ito sense.

We know that, under some Lipschitz and boundedness conditions for the drift μ and the volatility σ, the equation with initial condition (2.2) has a unique solution which is adapted with respect to the filtration $\mathbb{F} = (\mathcal{F}_t)_t$ generated by W_t.

Now, what happens if we consider the problem (2.2) with a terminal condition at time $T > 0$? Consider, for instance, the particular case with $\mu(t, X_t) = \sigma(t, X_t) = 0$, and a square-integrable $\mathcal{F}_T$-measurable random variable $\xi \in L^2(0, T)$ for which we try to solve the problem of finding a process Y_t such that

$$\begin{aligned} dY_t &= 0 \\ Y_t(T) &= \xi. \end{aligned} \tag{2.3}$$

This equation has a unique solution given by $Y(t) = \xi$, which is not necessarily $\mathcal{F}_t-$measurable for every $0 \leq t \leq T$, and therefore (2.3) may not have solution in the usual SDE sense.

Despite this, we can try to solve this problem reinterpreting the solution to (2.3) based on the following representation theorem.

Theorem 2.1.1 (Martingale representation theorem [14])**.** *Let $(M_t)_{0 \leq t \leq T}$ be a continuous $\mathbb{R}^n$-valued square-integrable martingale with respect to $\mathcal{F}_t$, the augmented filtration generated by an d-dimensional Brownian motion $(W_t)_t$. Then, there is a unique $\mathbb{R}^{n \times d}$-valued $\mathcal{F}_t$-adapted stochastic process $f(s)$, with $\mathbb{E}[\int_0^T |f|^2 dt] < \infty$, such that*

$$M_t = M_0 + \int_0^t f(s)dW_s \quad \textit{for} \quad t \in [0, T], \tag{2.4}$$

where the uniqueness is interpreted in the mean squared norm.

We can intend to enforce the solution Y_t to be $\mathcal{F}_t-$measurable for every $0 \leq t \leq T$ by taking its conditional expectation with respect to the evolving σ-algebra

$$Y(t) := \mathbb{E}[\xi | \mathcal{F}_t], \tag{2.5}$$

which satisfies the terminal condition $Y(T) = \xi$, since ξ is $\mathbb{F}_T$-measurable. Thus, as a consequence of the Martingale representation theorem 2.1.1, we conclude that there exist a square-integrable $\mathcal{F}_t$-measurable process Z_t such that

$$Y(t) = Y(0) + \int_0^t Z_s dW_s \quad \text{for} \quad t \in [0, T], \tag{2.6}$$

which can be written as

$$\begin{aligned} dY_t &= Z_t dW_t \\ Y(T) &= \xi \end{aligned} \tag{2.7}$$

Therefore, problem (2.3) can be reinterpreted as in problem (2.7), that we will denote as a bacward stochastic differential equation (BSDE) , in which we seek a pair of processes (Y_t, Z_t) that will provide an adapted solution to our original problem. Indeed, the process Z_t will "steer" the system so that the process Y_t remains adapted, and is thus called a control process. It is not possible to revert time as $t \to T - t$ as the filtration goes only in one direction [15].

Finally, we can write this equation in another form. Note that (2.7) is a forward SDE problem, hence we can solve for $Y(0)$ in the integral form , and so we have

$$Y(0) = \xi - \int_0^T Z_s dW_s, \tag{2.8}$$

that is inserted in (2.6) to obtain

$$Y(t) = \xi - \int_0^T Z_s dW_s + \int_0^t Z_s dW_s = \xi - \int_t^T Z_s dW_s \quad \forall t \in [0, T], \tag{2.9}$$

which is the standard way to write the BSDE in integral form.

2.1.2 Some useful theorems

Now that we have motivated the use of BSDEs, we follow [16] to provide a formal definition and prove that under certain regularity conditions, we can ensure the existence of a solution for that kind of equations.

Let be $(\Omega, \mathcal{F}, \mathbb{P})$ a probability space and $T > 0$ a fixed horizon time. We consider a d-dimensional Brownian motion $W = (W_t)_{t\in[0,T]}$ and let $\mathbb{F} = (\mathcal{F}_t)_{t\in[0,T]}$ be the corresponding natural augmented filtration (i.e with the completeness and right continuity conditions).

Denote by $\mathbb{S}^2(0,T)$ the set of $\mathbb{R}$-valued progressively measurable processes Y_t such that

$$\mathbb{E}\left[\sup_{0\le t\le T} |Y_t|^2\right] < \infty, \tag{2.10}$$

and by $\mathbb{H}^2(0,T)^d$ the set of $\mathbb{R}^d$-valued progressively measurable processes Z_t such that

$$\mathbb{E}\left[\int_0^T |Z_t|^2 dt\right] < \infty. \tag{2.11}$$

Here we consider the backward stochastic differential equation

$$\begin{aligned} dY_t &= -f(t, Y_t, Z_t)dt + Z_t \cdot dW_t \\ Y(T) &= \xi \end{aligned} \tag{2.12}$$

Definition 2.1.1. *A solution to the BSDE (2.12) is a pair $(Y,Z) \in \mathbb{S}^2(0,T) \times \mathbb{H}^2(0,T)^d$ such that*

$$Y_t = \xi + \int_t^T f\left(s, Y_s, Z_s\right) ds - \int_t^T Z_s \cdot dW_s, \quad 0 \le t \le T \tag{2.13}$$

Now we establish an existence and uniqueness theorem for $\mathbb{R}$-valued process, which can be extended to $\mathbb{R}^m$-valued processes.

Assumptions 2.1.2. *Let (ξ, f) satisfy*

I. $\xi \in L^2(\Omega, \mathcal{F}_T, \mathbb{P}; \mathbb{R})$

II. $f : \Omega \times [0,T] \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ *such that*

a) $f(\cdot, t, y, z)$, written $f(t,y,z)$ for simplicity, is progressively measurable for all y, z

b) $f(t,0,0) \in \mathbb{H}^2[0,T]$

c) f is uniformly Lipschitz in (y,z), i.e ,there exist a constant C_f such that for all $y_1, y_2 \in \mathbb{R} \times \mathbb{R}$ and $z_1, z_2 \in \mathbb{R}^d \times \mathbb{R}^d$ we have

$$\left|f\left(t, y_1, z_1\right) - f\left(t, y_2, z_2\right)\right| \le C_f\left(\left|y_1 - y_2\right| + \left|z_1 - z_2\right|\right) \quad a.s \tag{2.14}$$

Theorem 2.1.2 (Existence and uniqueness of solutions to BSDEs [16])**.** *Given a pair (ξ, f), called the terminal condition and the driver of the BSDE, that satisfy the assumptions 2.1.2 , there exist a unique solution (Y,Z) to the backward stochastic differential equation (2.12).*

To give a demonstration we will need the following inequalities about SDEs, whose proofs will be omitted.

Theorem 2.1.3 (Doob's martingale inequality [14])**.** *Let $\{M_t\}_t \ge 0$ be a $\mathbb{R}^m$-valued martingale in $L^p(\Omega; \mathbb{R}^m)$. Let $[0,T]$ be a bounded interval with $T > 0$ and let $p > 1$. Then*

$$\mathbb{E}\left[\sup_{0 \le t \le T} |M_t|^p\right] \le \left(\frac{p}{p-1}\right)^p \mathbb{E}[|M_T|^p], \tag{2.15}$$

in particular, if $p = 2$,

$$\mathbb{E}\left[\sup_{0 \le t \le T} |M_t|^2\right] \le 4\mathbb{E}[|M_T|^2]. \tag{2.16}$$

Theorem 2.1.4 (Burkholder-Davis-Gundy inequality [14])**.** *Let $g \in L^2(\mathbb{R}^+; \mathbb{R}^{m \times d})$. Define for $t \ge 0$*

$$x(t) = \int_0^t g(s) dW_s \quad \textit{and} \quad A(t) = \int_0^t |g(s)|^2 ds$$

then, for every $p > 0$ there exist universal positive constants c_p, C_p, depending only on p, such that the following inequalities hold,

$$c_p \mathbb{E}[|A(t)|^{\frac{p}{2}}] \le \mathbb{E}\left[\sup_{0 \le s \le t} |x(s)|^p\right] \le C_p \mathbb{E}[|A(t)|^{\frac{p}{2}}], \tag{2.17}$$

in particular, if $p = 1$, we can take $c_p = \frac{1}{2}$ and $C_p = 4\sqrt{2}$.

Proof of theorem 2.1.2 . Here we give a fixed point argument. To do it, lets consider a pair of process $(U, V) \in \mathbb{S}^2(0,T) \times \mathbb{H}^2(0,T)^d$ and, as in the motivation example, consider the martingale

$$M_t = \mathbb{E}\left[\xi + \int_0^T f(s, U_s, V_s)ds \middle| \mathcal{F}_t\right], \tag{2.18}$$

which is square-integrable under the hypothesis on (ξ, f). Using to the martingale representation theorem 2.1.1, we deduce the existence and uniqueness of a process $Z_s \in \mathbb{H}^2(0,T)^d$ such that

$$M_t = M_0 + \int_0^t Z_s \cdot dW_s. \tag{2.19}$$

Now, define the process Y_t for $0 \le t \le T$ as

$$\begin{aligned} Y_t &= \mathbb{E}\left[\xi + \int_t^T f(s, U_s, V_s)ds \middle| \mathcal{F}_t\right] = \mathbb{E}\left[\xi + \int_0^T f(s, U_s, V_s)ds - \int_0^t f(s, U_s, V_s)ds \middle| \mathcal{F}_t\right] \\ &= M_t - \int_0^t f(s, U_s, V_s)ds \end{aligned} \tag{2.20}$$

and note that from this and using (2.19), Y_t satisfies

$$\begin{aligned} Y_t &= M_0 + \int_0^t Z_s \cdot dW_s - \int_0^t f(s, U_s, V_s)ds \\ &= \xi + \int_t^T f(s, U_s, V_s)ds - \int_t^T Z_s \cdot dW_s. \end{aligned} \tag{2.21}$$

Thus, consider the function $\Phi : \mathbb{S}^2(0,T) \times \mathbb{H}^2(0,T)^d \to \mathbb{S}^2(0,T) \times \mathbb{H}^2(0,T)^d$ that maps the pair (U, V) to the pair (Y, Z) constructed as above, $\Phi(U, V) = (Y, Z)$. Note that it is well-defined as the Z process is unique, and by Doob's martingale inequality 2.1.3 we have

$$\mathbb{E}\left[\sup_{0 \le t \le T} \left|\int_t^T Z_s \cdot dW_s\right|^2\right] \le 4\mathbb{E}\left[\int_0^T |Z_s|^2 ds\right] < \infty, \tag{2.22}$$

and therefore, by assumptions I, IIa) and IIb), Y_t lies in $\mathbb{S}^2(0,T)$. Also note that a solution to the BSDE (2.12) is a fixed point of Φ. We will show that such fixed point exist by showing it is a contraction if we endow the $\mathbb{S}^2(0,T) \times \mathbb{H}^2(0,T)^d$ space with the metric

$$\|(Y, Z)\|_\beta = \left(\mathbb{E}\left[\int_0^T e^{\beta s}(|Y_s|^2 + |Z_s|^2)ds\right]\right)^{\frac{1}{2}}, \tag{2.23}$$

where $\beta > 0$ is a parameter to be chosen later.

To show that Φ is a contraction, let $(U,V),(U',V') \in \mathbb{S}^2(0,T) \times \mathbb{H}^2(0,T)^d$ and $(Y,Z) = \Phi(U,V)$, $(Y',Z') = \Phi(U',V')$. We denote $(\bar{U},\bar{V}) = (U-U',V-V')$, $(\bar{Y},\bar{Z}) = (Y-Y',Z-Z')$ and $\bar{f}_t = f(t,U_t,V_t) - f(t,U'_t,V'_t)$.

Using equation (2.21), we know that $\bar{Y}_s$ satisfies

$$\bar{Y}_s = -\int_0^t \bar{f}_s ds + \int_0^t \bar{Z}_s \cdot dW_s \tag{2.24}$$

So let's apply Ito's formula to the process $e^{\beta s}|\bar{Y}_s|^2$ between 0 and T to obtain

$$\begin{aligned} e^{\beta T}|\bar{Y_T}|^2 = |\bar{Y_0}|^2 &+ \int_0^T (\beta e^{\beta s}|\bar{Y_s}|^2 - 2e^{\beta s}\bar{Y_s} \cdot \bar{f}_s + e^{\beta s}|\bar{Z}_s|^2)ds \\ &+ \int_0^T 2e^{\beta s}\bar{Y}_s\bar{Z}_s \cdot dW_s. \end{aligned} \tag{2.25}$$

Observe that we can apply the Burkholder-Davis-Gundy inequality 2.1.4 with $p=1$ to the following expectation of the supremum associated with the last term

$$\begin{aligned} \mathbb{E}\left[\sup_{0\le t\le T}\left|\int_0^t 2e^{\beta s}\bar{Y}_s\bar{Z}_s \cdot dW_s\right|\right] &\le 4\sqrt{2}\,\mathbb{E}\left[\left(\int_0^T 4e^{2\beta s}|\bar{Y}_s|^2|\bar{Z}_s|^2 ds\right)^{\frac{1}{2}}\right] \\ &\le 4\sqrt{2}e^{\beta T}\,\mathbb{E}\left[\sup_{0\le t\le T}|Y_t|^2 + \int_0^T |\bar{Z}_s|^2 ds\right] \\ &< \infty, \end{aligned} \tag{2.26}$$

which shows that the local martingale $\int_0^t 2e^{\beta s}\bar{Y}_s\bar{Z}_s \cdot dW_s$ is actually a uniformly integrable martingale and therefore its expected value remains constant zero. Also, note that $\bar{Y}_T = Y_T - Y'_T = \xi - \xi = 0$.

Using these facts, take the expected value to (2.25) and reorder terms to obtain

$$\begin{aligned} \mathbb{E}\left|\bar{Y_0}\right|^2 + \mathbb{E}\left[\int_0^T e^{\beta s}\left(\beta\left|\bar{Y_s}\right|^2 + \left|\bar{Z_s}\right|^2\right)ds\right] &= 2\mathbb{E}\left[\int_0^T e^{\beta s}\bar{Y_s}\cdot\bar{f_s}ds\right] \\ \le 2C_f\mathbb{E}\left[\int_0^T e^{\beta s}\left|\bar{Y_s}\right|\left(\left|\bar{U_s}\right| + \left|\bar{V_s}\right|\right)ds\right] &\quad \text{(by condition } IIc)) \\ \le 4C_f^2\mathbb{E}\left[\int_0^T e^{\beta s}\left|\bar{Y_s}\right|^2 ds\right] + \frac{1}{2}\mathbb{E}\left[\int_0^T e^{\beta s}\left(\left|\bar{U_s}\right|^2 + \left|\bar{V_s}\right|^2\right)ds\right], \end{aligned} \tag{2.27}$$

so if we choose $\beta = 1 + 4C_f^2$ and ignore the $\mathbb{E}\left|\bar{Y_0}\right|^2$ term, we obtain

$$\mathbb{E}\left[\int_0^T e^{\beta s}\left(\left|\bar{Y_s}\right|^2 + \left|\bar{Z_s}\right|^2\right)ds\right] \le \frac{1}{2}\mathbb{E}\left[\int_0^T e^{\beta s}\left(\left|\bar{U_s}\right|^2 + \left|\bar{V_s}\right|^2\right)ds\right], \tag{2.28}$$

which is $\|(\Phi(U,V))\|_\beta \le \frac{1}{2}\|(U,V)\|_\beta$, that means Φ is a contraction in a Banach space, as $\mathbb{S}^2(0,T) \times \mathbb{H}^2(0,T)^d$ is the product of Banach spaces, and therefore has a unique fixed point. ■

As in the every differential equation, there are cases where we can provide an explicit solution. The next theorem provides one for the BSDE with linear generator

Theorem 2.1.5 (Linear BSDEs [16])**.** *Let A_t,B_t be bounded progressively measurable processes with values in $\mathbb{R}$ and $\mathbb{R}^d$, C_t a process in $\mathbb{H}^2(0,T)$ and $\xi \in L^2(\Omega, \mathcal{F}_T, \mathbb{P}, \mathbb{R})$. Then, the linear backward stochastic differential equation*

$$\begin{aligned} &dY_t = -(A_tY_t + Z_t \cdot B_t + C_t)dt + Z_t \cdot dW_t \\ &Y_T = \xi \end{aligned} \tag{2.29}$$

has a unique solution, and is given by the formula

$$\Gamma_t Y_t = E\left[\Gamma_T \xi + \int_t^T \Gamma_s C_s ds \mid \mathcal{F}_t\right], \tag{2.30}$$

where Γ_t is the solution to the adjoint process

$$\begin{aligned} &d\Gamma_t = \Gamma_t(A_tdt + B_t \cdot dW_t) \\ &\Gamma_0 = 1 \end{aligned} \tag{2.31}$$

Proof. First apply Ito's formula to $\Gamma_t Y_t$ to obtain

$$\begin{aligned} d(\Gamma_tY_t) &= Y_t d\Gamma_t + \Gamma_t dY_t + d\Gamma_t dY_t \\ &= Y_t(\Gamma_t A_t dt + \Gamma_t B_t \cdot dW_t) + \Gamma_t(-(A_tY_t + Z_t \cdot B_t + C_t)dt + Z_t \cdot dW_t) \\ &\quad + \Gamma_t Z_t \cdot B_t dt \\ &= -\Gamma_t C_t dt + \Gamma_t(Z_t + Y_tB_t) \cdot dW_t, \end{aligned} \tag{2.32}$$

that can be written in integral form as

$$\Gamma_t Y_t + \int_0^t \Gamma_s C_s ds = Y_0 + \int_0^t \Gamma_s \left(Z_s + Y_s B_s\right) \cdot dW_s. \tag{2.33}$$

We will show, as in the proof of theorem 2.1.2, that the stochastic integral in the last expression, which is a local martingale, is in fact a uniformly integrable martingale. We have $\mathbb{E}\big[\sup_{0\le t\le T} |\Gamma_t|^2\big] < \infty$,since A_t and B_t are bounded. Also, let's denote b_∞ the upper bound on B_t, then the following inequalities hold

$$\begin{aligned} \mathbb{E}\left[\sup_{0\le t\le T}\left|\int_0^t \Gamma_s \left(Z_s + Y_s B_s\right) \cdot dW_s\right|\right] &\le 4\sqrt{2}\,\mathbb{E}\left[\left(\int_0^T |\Gamma_s|^2 |Z_s + Y_s B_s|^2 ds\right)^{\frac{1}{2}}\right] \\ &\qquad \text{(By BDG inequality 2.1.4)} \\ &\le \frac{4\sqrt{2}}{2} E\left[\sup_{0\le t\le T} |\Gamma_t|^2 + 2\int_0^T |Z_t|^2\,dt + 2b_\infty^2 \int_0^T |Y_t|^2\,dt\right] \\ &< \infty. \end{aligned} \tag{2.34}$$

Consequently, the right-hand side of is a uniformly integrable martingale, and so, if we take expected values to the equality (2.33), we have

$$\begin{aligned}\Gamma_t Y_t + \int_0^t \Gamma_s C_s ds &= \mathbb{E}\left[\Gamma_T Y_T + \int_0^T \Gamma_s C_s ds \middle| \mathcal{F}_t\right] \\ &= \mathbb{E}\left[\Gamma_T \xi + \int_0^T \Gamma_s C_s ds \middle| \mathcal{F}_t\right]\end{aligned} \tag{2.35}$$

and, as $\int_0^t \Gamma_s C_s ds$ is $\mathcal{F}_t$-measurable, we obtain

$$\Gamma_t Y_t = \mathbb{E}\left[\Gamma_T \xi + \int_t^T \Gamma_s C_s ds \middle| \mathcal{F}_t\right], \tag{2.36}$$

that is what we wanted to prove. The control solution Z_t can be obtained by the martingale representation theorem 2.1.1 applied to this process. ■

Finally, we state the next comparison principle for solution of BSDEs

Theorem 2.1.6 (Comparison principle for BSDEs [16]). *Let (ξ_1, f_1) and (ξ_2, f_2) two pairs of terminal conditions and generators satisfying assumptions 2.1.2,and let $(Y_{1,t}, Z_{1,t})$ and $(Y_{2,t}, Z_{2,t})$ the solutions to their corresponding BSDE. Suppose that*

1. $\xi_1 \leq \xi_2$ *a.s*
2. $f_1(t, Y_{1,t}, Z_{1,t}) \leq f_2(t, Y_{1,t}, Z_{1,t})$ $dt \times d\mathbb{P}$*-a.e*
3. $f_2(t, Y_{1,t}, Z_{1,t}) \in \mathbb{H}^2(0,T)$

Then $Y_{1,t} \leq Y_{2,t}$ for all $0 \leq t \leq T$, a.s. Furthermore, if $Y_{2,0} \leq Y_{1,0}$, then $Y_{1,t} = Y_{2,t}$ for $t \in [0,T]$. In particular, if $\mathbb{P}(\xi_1 < \xi_2) > 0$ or $f_1(t,\cdot,\cdot) < f_2(t,\cdot,\cdot)$ on a set with strictly positive measure $dt \times d\mathbb{P}$ then $Y_{1,0} < Y_{2,0}$.

Proof. To simplify notation, we give a proof with $d = 1$. We denote $\bar{Y_t} = Y_{2,t} - Y_{1,t}$ and $\bar{Z_t} = Z_{2,t} - Z_{1,t}$. Then $(\bar{Y_t}, \bar{Z_t})$ satisfy the BSDE

$$\begin{aligned}&d\bar{Y_t} = -\left(\Delta_t^y \bar{Y_t} + \Delta_t^z \bar{Z_t} + \bar{f_t}\right) dt + \bar{Z_t} dW_t \\ &\bar{Y_T} = \xi_2 - \xi_1,\end{aligned} \tag{2.37}$$

where

$$\begin{aligned}\Delta_t^y &= \frac{f_2\left(t, Y_{2,t}, Z_{2,t}\right) - f_2\left(t, Y_{1,t}, Z_{2,t}\right)}{Y_{2,t} - Y_{1,t}} 1_{Y_{2,t}-Y_{1,t}\neq 0} \\ \Delta_t^z &= \frac{f_2\left(t, Y_{1,t}, Z_{2,t}\right) - f_2\left(t, Y_{1,t}, Z_{1,t}\right)}{Z_{2,t} - Z_{1,t}} 1_{Z_{2,t}-Z_{1,t}\neq 0} \\ \bar{f_t} &= f_2\left(t, Y_{1,t}, Z_{1,t}\right) - f_1\left(t, Y_{1,t}, Z_{1,t}\right).\end{aligned} \tag{2.38}$$

By assumption, f_2 is Lipschitz in y, z, hence Δ_t^y and Δ_t^z are bounded. Moreover, $\bar{f_t} \in \mathbb{H}^2(0,T)$. Therefore, the solution to (2.37) is given by theorem 2.1.5 as

$$\Gamma_t \bar{Y_t} = \mathbb{E}\left[\Gamma_T\left(\xi_2 - \xi_1\right) + \int_t^T \Gamma_s \bar{f_s} ds \middle| \mathcal{F}_t\right], \tag{2.39}$$

where Γ_t satisfies

$$\begin{aligned} d\Gamma_t &= \Gamma_t(\Delta_t^y dt + \Delta_t^z dW_t) \\ \Gamma_0 &= 1. \end{aligned} \tag{2.40}$$

Note that Γ_t is strictly positive. We conclude the stated result using that $\xi_2 - \xi_2 \geq 0$ by assumption 1), and $\bar{f}_t \geq 0$ by assumption 2).

■

2.1.3 Forward-Backward stochastic differential equations

Now we consider a special case of backward stochastic differential equations in which the randomness of the drift enters through a process satisfying a forward stochastic differential equation. In its more general form, the problem is stated as find three processes $(X_t, Y_t, Z_t) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^{m\times d}$ such that

$$\begin{aligned} dX_s &= \mu(t, X_s, Y_s, Z_s)ds + \sigma(s, X_s, Y_s, Z_s)dW_s \\ X_t &= x \\ dY_s &= -f(s, X_s, Y_s, Z_s)ds + Z_s dW_s \\ Y_T &= g(X_T), \end{aligned} \tag{2.41}$$

for all $t \leq s \leq T$, where μ, σ and g are known functions, and x is the initial condition at starting time s. This coupled system is called a forward-backward stochastic differential equation (FBSDE).

This problem is rather difficult, as the coupling between the processes may forbid a solution to exist. There are conditions on μ, σ, g where we can establish the existence and uniqueness of solutions to the former system, but their detailed proof is very technical and thus is not presented here, see [10].

However, we can say something simpler about the decoupled case

$$\begin{aligned} dX_s &= \mu(s, X_s)ds + \sigma(s, X_s)dW_s \\ X_t &= x, \\ dY_s &= -f(s, X_s, Y_s, Z_s)ds + Z_s dW_s \\ Y_T &= g(X_T). \end{aligned} \tag{2.42}$$

for all $t \leq s \leq T$.

In this case, if μ and σ satisfy enough regularity conditions to ensure that a solution to the forward SDE in (2.42) exists, for example, if they are Lipschitz and bounded, then we can solve it for the process X_t and insert the solution into the backward equation in 2.42 and solve for the backward process. However, the main property of FBSDEs is that the solution process (Y, Z) of the BSDE can be written as a deterministic function of time and the state process, in this case the solution is said to be *markovian.*

Let's establish this assertions in the following theorem.

Assumptions 2.1.3. *Let (μ, σ, f, g). There exist a constant $C > 0$ such that for all x, y, t*

I. $|\mu(t,x)-\mu(t,y)|+|\sigma(t,x)-\sigma(t,y)|\leq C(1+|x-y|)$

II. $|f(t,x,y_1,z_1)-f(t,x,y_2,z_2)|\leq C(|y_1-y_2|+|z_1-z_2|)$

III. $|\sigma(t,x)|+|\mu(t,x)|\leq C(1+|x|)$

IV. $|f(t,x,y,z)|+|g(x)|\leq C(1+|x|^p)$ *para* $p\geq\frac{1}{2}$

Theorem 2.1.7 (Existence and markovianity of solutions of FBSDEs [17])**.** *Under assumptions 2.1.3, the uncoupled forward-backward stochastic differential equation (2.42) has a unique solution $(X_s^{t,x},Y_s^{t,x},Z_s^{t,x})$ starting from x at time t. Moreover, $(Y_s^{t,x},Z_s^{t,x})$ is adapted to the future σ-algebra of W after t, i.e, it is $\mathcal{F}_s^t$-adapted where for each $s\in[t,T]$ we define $\mathcal{F}_s^t=\sigma(W_u-W_t,t\leq u\leq s)$. In particular, $Y_t^{t,x}$ is deterministic and for $0\leq s\leq t$ we have $Y_s^{t,x}=Y_t^{t,x}$ and $Z_s^{t,x}=0$.*

Proof. The first part about existence and uniqueness of solution to the FBSDE follows from the fact that in assumptions 2.1.3, I and III are the standard Lipschitz and linear growth conditions that guarantee the existence of a solution for the forward process [14], and that II and IV are sufficient conditions to ensure the existence of the solution to the backward process from theorem 2.1.2.

For the second part, consider the translated Brownian motion W' and its associated filtration given by $W'_s=W_{t+s}-W_t$ for $0\leq s\leq T-t$ and $\mathcal{F}'_s:=\mathcal{F}^t_{t+s}$ or $0\leq s\leq T-t$. Let $X_s'^{0,x}$ be the adapted solution to the SDE

$$\begin{aligned}&dX'_s=\mu(s,X'_s)dt+\sigma(s,X'_s)dW_s\\&X'_0=x.\end{aligned}\tag{2.43}$$

By the the uniqueness provided by the former theorems, we have $X_s^{t,x}=X_{s-t}'^{0,x}$ a.s for $s\in[0,T-t]$, hence $X_s^{t,x}$ is $\mathcal{F}_s^t$-adapted.

Now consider the associated $\mathcal{F}'$-adapted solution (Y'_s,Z'_s) with $s\in[0,T-t]$ to the BSDE

$$\begin{aligned}&dY'_s=-f(s+t,X'_s,Y'_s,Z'_s)ds+Z'_s\cdot dW_s\\&Y'_{T-t}=g(X'_{T-t}).\end{aligned}\tag{2.44}$$

We have that (Y'_{s-t},Z'_{s-t}) with $s\in[t,T]$ is also a solution of the backward equation in (2.42) in $[t,T]$. Hence, by the uniqueness provided before we have that $(Y'_{s-t},Z'_{s-t})=(Y_s^{t,x},Z_s^{t,x})$ for $s\in[t,T]$, therefore (Y'_{s-t},Z'_{s-t}) is $\mathcal{F}_s^t$-adapted.

■

From now on, we will denote by

$$v(t,x):=Y_t^{t,x},\tag{2.45}$$

the deterministic function of t and x provided by the last theorem. We also notice that $Y_t=v(t,X_t)$ for $t\in[0,T]$.

2.2 The Feynman-Kac formulas

Now we shall establish the connection between stochastic differential equations with parabolic linear partial differential equations and its non-linear generalization based on backward stochastic differential equations.

2.2.1 The linear Feynman-Kac formula

We will start by the linear case to introduce the necessity for a non-linear generalization. Consider the $\mathbb{R}^n$-valued process $X_s^{t,x}$ defined to be the solution in $s \in [t, \infty)$ of the SDE

$$\begin{aligned} dX_s &= \mu(s, X_s)ds + \sigma(s, X_s)dW_s \\ X_t &= x, \end{aligned} \tag{2.46}$$

where, again, W_t is a d-dimensional Brownian motion, μ is $\mathbb{R}^n$-valued function, σ is a $(n \times d)$-valued matrix of functions and $x \in \mathbb{R}^n$ is the initial condition. We have the following estimate

Theorem 2.2.1 ([18]). *Let μ and σ satisfy conditions I and III of assumptions 2.1.3, then, there exist a constant $C > 0$ such that the solution to 2.46 satisfies*

$$\mathbb{E}\left[\sup_{t\le s\le T} |X_s|^2\right] \le C(1 + \mathbb{E}\left[|x|^2\right])e^{C(T-t)}. \tag{2.47}$$

Now, for a fixed $T > 0$, consider the following parabolic PDE with terminal condition for the function $v(t,x) : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}$,

$$\begin{aligned} &\frac{\partial v}{\partial t} + \mathcal{L}_t v - k(t,x)v + f(t,x) = 0 \\ &v(T,x) = g(x), \end{aligned} \tag{2.48}$$

where $f(x,t)$ and $g(x)$ are some $\mathbb{R}$-valued continuous functions, $k(x,t)$ is a non-negative $\mathbb{R}$-valued function, and $\mathcal{L}_t$ is the *generator* of the process X_s, defined as

$$\begin{aligned} \mathcal{L}_t v &= \mu(t,x) \cdot D_x v(t,x) + \frac{1}{2}\operatorname{Tr}(\sigma(t,x)\sigma^T(t,x)D_{xx}^2 v(t,x)) \\ &= \sum_{i=1}^n \mu_i(t,x)\frac{\partial v}{\partial x_i}(t,x) + \sum_{i,k=1}^n a_{i,k}(t,x)\frac{\partial^2 v}{\partial x_i \partial x_k}(t,x), \end{aligned} \tag{2.49}$$

where we denote by $a_{i,k}$ the coefficients of the *diffusion matrix*, calculated as

$$a_{i,k} = \sum_{j=1}^d \sigma_{i,j}(t,x)\sigma_{k,j}(t,x). \tag{2.50}$$

The linear Feynman-Kac formula establishes a connection between the process satisfying (2.46) and the classical solution to equation (2.48) as follows

Assumptions 2.2.1. *Let μ and σ satisfy conditions I and III of assumptions 2.1.3, and assume that*

I. The functions $g(x) : \mathbb{R}^n \to \mathbb{R}$, $f(t,x) : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ and $k(t,x) : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^+$ are continuous.

II. The function $v(t,x) : [0,T] \times \mathbb{R}^n \to \mathbb{R}$ is continuous in $[0,T] \times \mathbb{R}^n$, one time differentiable in t, and two times differentiable in x, and satisfies the PDE (2.48). Moreover, its first derivative in x is bounded, i.e , $|D_x v(t,x)| < M$ for some constant M and all $t \in [0,T]$ and $x \in \mathbb{R}^n$.

Theorem 2.2.2 (Linear Feynman-Kac formula [18])**.** *Under the assumptions 2.2.1, the solution $v(t,x)$ to the equation (2.48) admits the stochastic representation*

$$v(t,x) = \mathbb{E}\Big[g(X_T^{t,x})e^{-\int_t^T k(s,X_s^{t,x})ds} + \int_t^T f(s,X_s^{t,x})e^{-\int_t^s k(u,X_u^{t,x})du} ds\Big], \tag{2.51}$$

for all $t \in [0,T]$ and $x \in \mathbb{R}^n$. In particular, this solution is unique.

Proof. In order to simplify notation, we set $X_s = X_s^{x,t}$. Let's apply Ito's formula to the process $v(s,X_s)e^{-\int_t^s k(u,X_u)du}$ in $s \in [t,T]$ to obtain

$$\begin{aligned} e^{-\int_t^T k(u,X_u)du} v(T,X_T) &= e^{-\int_t^t k(u,X_u)du} v(t,X_t) \\ &+ \int_t^T e^{-\int_t^s k(u,X_u)du} \left(\frac{\partial v}{\partial t}(s,X_s) - k(s,X_s)v(s,K_s) + \mathcal{L}_s v(s,X_s)\right) ds \\ &+ \int_t^T e^{-\int_t^s k(u,X_u)du} D_x v(s,X_s) \cdot \sigma(s,X_s) dW_s \\ &= v(t,X_t) - \int_t^T e^{-\int_t^s k(u,X_u)du} f(s,X_s) ds \\ &+ \int_t^T e^{-\int_t^s k(u,X_u)du} D_x v(s,X_s) \cdot \sigma(s,X_s) dW_s, \end{aligned} \tag{2.52}$$

and therefore, using that $X_t = x$, $v(T,X_t) = g(X_T)$ and solving for $v(x,t)$, we have

$$\begin{aligned} v(t,x) = g(X_T)e^{-\int_t^T k(u,X_u)du} + \int_t^T f(s,X_s)e^{-\int_t^s k(u,X_u)du} ds \\ - \int_t^T e^{-\int_t^s k(u,X_u)du} D_x v(s,X_s) \cdot \sigma(s,X_s) dW_s. \end{aligned} \tag{2.53}$$

To obtain the desired formula, we take expectation to this expression, and observe that the stochastic integral is a square integrable martingale by assumption 2.2.1 *III* on $D_x v$, the non-negativity of k, the linear growth condition on σ and the estimation 2.2.1, therefore it's expected value is constant 0. ■

Note that we required that equation (2.48) has a classical smooth solution, for which we need some regularity conditions on μ, σ and growth conditions on f and g to ensure the uniform ellipticity of $\mathcal{L}_t$. Also note, that assumptions 2.2.1 are rather restrictive, especially the boundedness of D_x, but can be relaxed imposing some quadratic growth condition on v and additional growth condition on f and g. However, even if there is no smooth solution to this problem, the Feynman-Kac formula may provide a solution with other meaning, which will be named a viscosity solution and will be defined in what follows.

In addition, this formula is useful for approximating solutions $v(t,x)$ to PDEs of the form (2.48), even in high dimensions, where classical methods fails because of the *curse of dimensionality.* This expected value can be approximated by Monte-Carlo simulation using sample paths, whose rate of convergence is independent on the dimension n of the underlying process. However, its use is limited to linear equations that may not be useful for certain problems.

By the other hand, if we consider a bounded domain and add boundary conditions to (2.48), the we have a similar formula for the solution that can be obtained as before. We state it informally in the following theorem omitting long technical definitions and relying on the intuitive definition of reflected process.

Theorem 2.2.3 (Linear Feynman-Kac formula with boundary conditions [19],[11])**.** *Consider equation (2.48) in a bounded smooth domain Ω, with boundary conditions*

$$v(t,x) = h_d(t,x) \quad \forall x \in \Gamma_d \tag{2.54}$$

$$\frac{\partial v}{\partial \vec{n}}(t,x) = h_n(t,x) \quad \forall x \in \Gamma_n \tag{2.55}$$

where $\Gamma_d \dot{\cup} \Gamma_n = \partial\Omega$ is a disjoint partition of the boundary with outer normal vector $\vec{n}$. Let

$$\tau = \inf\{s \in [t,T] | X_s^{t,x} \in \Gamma_d\} \tag{2.56}$$

the stopping time associated with the first exit of the domain, and define

$$\Psi(t,x) = \begin{cases} g(x) & (t,x) \in \{T\} \times \Omega \\ h_d(t,x) & (t,x) \in (t,T) \times \Gamma_d \end{cases} \tag{2.57}$$

Then, under the assumptions 2.2.1, the solution $v(t,x)$ to the equation (2.48) admits the stochastic representation

$$v(t,x) = \mathbb{E}\left[\Psi(\tau, X_\tau^{t,x}) e^{-\int_t^T k(s,X_s^{t,x})ds} + \int_t^T f(s,X_s^{t,x}) e^{-\int_t^s k(u,X_u^{t,x})du} ds\right], \tag{2.58}$$

for all $t \in [0,T]$ and $x \in \mathbb{R}^n$, where $X_s^{t,x}$ is the solution of a reflected stochastic differential equation starting at t,x with reflection boundary Γ_n. In particular, this solution is unique.

Note that we have not defined precisely what means to be a solution to a reflected stochastic differential equation, see [11].

2.2.2 The non-linear Feynman-Kac formula

Now we deal with a more general PDE than (2.48). We consider, for some fixed $T > 0$, the problem

$$\begin{aligned}&\frac{\partial v}{\partial t}(t,x) + \mathcal{L}_t v(t,x) + f(t,x,v(t,x),\sigma(t,x)'D_x v(t,x)) = 0\\ &v(T,x) = g(x),\end{aligned} \tag{2.59}$$

for all $t \in [0,T]$ and $x \in \mathbb{R}^n$, and where $f : \mathbb{R}\times\mathbb{R}^n\times\mathbb{R}\times\mathbb{R}^d \to \mathbb{R}$ is a non-linear function.

We will associate the solution to this problem with a forward-backward stochastic differential equation of the form (2.42). We have an easy association given by the following theorem

Theorem 2.2.4 (Verification theorem [16])**.** *Let $v(t,x)$ be a classical solution to 2.59, that is continuous on $[0,T]\times\mathbb{R}^n$, one time differentiable in t and two times differentiable in x and satisfy the linear growth condition $|v(t,x)| \le L(1+|x|)$ for some $L > 0$ and all $x \in \mathbb{R}^n$ $t \in [0,T]$. Also, let its first space derivative satisfy the growth condition $|D_x v(t,x)| \le C(1+|x|^q)$ for some $C > 0$, $q > 0$ and all $x \in \mathbb{R}^n$ $t \in [0,T]$.Then, the pair (Y,Z) defined by*

$$Y_t = v(t,X_t) \quad Z_t = \sigma(t,X_t)'D_x v(t,X_t) \tag{2.60}$$

is the solution to the backward stochastic differential equation in 2.42.

Proof. Apply Ito's formula to $Y_t = v(t,X_t)$ to obtain

$$\begin{aligned} dY_t &= \left(\frac{\partial v}{\partial t}(t,X_t) + \mathcal{L}_t v(t,X_t)\right)dt + \sigma(t,X_t)'D_x v(t,X_t)\cdot dW_t\\ &= -f(t,X_t,v(t,X_t,D_x v(t,X_t)))dt + \sigma(t,X_t)'D_x v(t,X_t)\cdot dW_t\\ &= -f(t,X_t,Y_t,Z_t)dt + Z_t dW_t\end{aligned} \tag{2.61}$$

and observing that $Y_T = v(T,X_T) = g(X_T)$, we have the first part, as this process is in $\mathbb{S}^2(0,T)\times\mathbb{H}^2(0,T)^d$ due to the growth condition of v and $D_x v$. ■

Nevertheless, the reciprocal affirmation y somewhat more complicated. If we have a solution (Y_t,Z_t) to the FBSDE (2.42), not necessarily $v(t,x) = Y_t^{t,x}$ will be a classical smooth solution to (2.59) as it may not exist due to the non-linearity. Nevertheless, we can define a new weaker notion of solution as follows

Definition 2.2.2. *Let $v(t,x) : \mathbb{R}\times\mathbb{R}^n \to \mathbb{R}$ be a locally bounded continuous function. Then*

- *$v(t,x)$ is called a viscosity sub-solution of (2.59) if $v(T,x) \le g(x)$ for $x \in \mathbb{R}^n$, and for all $\phi \in C^{1,2}([0,T]\times\mathbb{R}^n)$ such that the map $v(t,x) - \phi(t,x)$ attains a local maximum at $(t,x) \in [0,T]\times\mathbb{R}^n$ it holds*

$$\frac{\partial \phi}{\partial t} + \mathcal{L}_t\phi + f(t,x,v(x,t),\sigma(t,x)'D_x v(t,x)) \ge 0 \tag{2.62}$$

- *$v(t,x)$ is called a viscosity super-solution of 2.59 if $v(T,x) \geq g(x)$ for $x \in \mathbb{R}^n$, and for all $\phi \in C^{1,2}([0,T] \times \mathbb{R}^n)$ such that the map $v(t,x) - \phi(t,x)$ attains a local minimum at $(t,x) \in [0,T] \times \mathbb{R}^n$ it holds*

$$\frac{\partial \phi}{\partial t} + \mathcal{L}_t \phi + f(t,x,v(x,t),\sigma(t,x)' D_x v(t,x)) \leq 0 \tag{2.63}$$

- *If $v(t,x)$ is a sub-solution and a super-solution it is called a viscosity solution of* (2.59)*.*

In particular, note that this definition does not require the smoothness of $v(t,x)$.

With this new concept of solution, we can establish the reverse relation as follows

Theorem 2.2.5 (Representation theorem [16])**.** *Let (X,Y,Z) be the solution to the uncoupled FBSDE* (2.42) *and set $v(t,x) = Y_t^{t,x}$. Then, v is a continuous function and is a viscosity solution to 2.59.*

Proof. Step 1: Continuity of $v(t,x)$:
Let $(t_1,x_1),(t_2,x_2) \in [0,T] \times \mathbb{R}^n$, with $t_1 \leq t_2$. For lighten the notation we write $X_s^i = X_s^{t_i,x_i}$, $i = 1,2$.

$$\begin{aligned}\left|Y_t^1 - Y_t^2\right|^2 = &\left|g\left(X_T^1\right) - g\left(X_T^2\right)\right|^2 - \int_t^T \left|Z_s^1 - Z_s^2\right|^2 ds \\ &+ 2\int_t^T \left(Y_s^1 - Y_s^2\right) \cdot \left(f\left(s, X_s^1, Y_s^1, Z_s^1\right) - f\left(s, X_s^2, Y_s^2, Z_s^2\right)\right) ds \\ &- 2\int_t^T \left(Y_s^1 - Y_s^2\right)' \left(Z_s^1 - Z_s^2\right) dW_s\end{aligned} \tag{2.64}$$

Then,

$$\begin{aligned}
&E\left[\left|Y_t^1-Y_t^2\right|^2\right]+E\left[\int_t^T\left|Z_s^1-Z_s^2\right|^2ds\right]\\
=&E\left[\left|g\left(X_T^1\right)-g\left(X_T^2\right)\right|^2\right]\\
&+2E\left[\int_t^T\left(Y_s^1-Y_s^2\right)\cdot\left(f\left(s,X_s^1,Y_s^1,Z_s^1\right)-f\left(s,X_s^2,Y_s^2,Z_s^2\right)\right)ds\right]\\
\leq&E\left[\left|g\left(X_T^1\right)-g\left(X_T^2\right)\right|^2\right]\\
&+2E\left[\int_t^T\left|Y_s^1-Y_s^2\right|\left|f\left(s,X_s^1,Y_s^1,Z_s^1\right)-f\left(s,X_s^2,Y_s^1,Z_s^1\right)\right|ds\right]\\
&+2C_fE\left[\int_t^T\left|Y_s^1-Y_s^2\right|\left(\left|Y_s^1-Y_s^2\right|+\left|Z_s^1-Z_s^2\right|\right)ds\right]\\
\leq&E\left[\left|g\left(X_T^1\right)-g\left(X_T^2\right)\right|^2\right]\\
&+E\left[\int_t^T\left|f\left(s,X_s^1,Y_s^1,Z_s^1\right)-f\left(s,X_s^2,Y_s^1,Z_s^1\right)\right|^2ds\right]\\
&+\left(1+4C_f^2\right)E\left[\int_t^T\left|Y_s^1-Y_s^2\right|^2ds+\frac{1}{2}E\int_t^T\left|Z_s^1-Z_s^2\right|^2ds\right],
\end{aligned}\tag{2.65}$$

So, using the Gronwall's lemma we obtain

$$\begin{aligned}
E\left[\left|Y_t^1-Y_t^2\right|^2\right]\leq&E\left[\left|g\left(X_T^1\right)-g\left(X_T^2\right)\right|^2\right]+E\left[\int_t^T\left|f\left(s,X_s^1,Y_s^1,Z_s^1\right)-f\left(s,X_s^2,Y_s^1,Z_s^1\right)\right|^2ds\right]\\
&+\left(1+4C_f^2\right)E\left[\int_t^T\left|Y_s^1-Y_s^2\right|^2ds\right]
\end{aligned}\tag{2.66}$$

and, by Gronwall's lemma,

$$\begin{aligned}
E\left[\left|Y_t^1-Y_t^2\right|^2\right]\leq C\Big\{&E\left[\left|g\left(X_T^1\right)-g\left(X_T^2\right)\right|^2\right]\\
&+E\left[\int_t^T\left|f\left(s,X_s^1,Y_s^1,Z_s^1\right)-f\left(s,X_s^2,Y_s^1,Z_s^1\right)\right|^2ds\right]\Big\}.
\end{aligned}\tag{2.67}$$

This, combined with the continuity of f and g in x, and the continuity of $X^{t,x}$ in (t,x), shows the mean-squared continuity of $Y_s^{t,x}$, and so the continuity of the map $(t,x)\to v(t,x)$. *Step 2: $v(t,x)$ is a viscosity solution.*
Let's prove it is a super-solution. Assume by contradiction that

$$-\frac{\partial\varphi}{\partial t}(t,x)-\mathcal{L}\varphi(t,x)-f\left(t,x,v(t,x),\left(D_x\varphi\right)'(t,x)\sigma(x)\right)>0.\tag{2.68}$$

By continuity of f,φ and its derivatives, there exists $h,\varepsilon>0$ such that for all $t\leq s\leq t+h,|x-y|\leq\varepsilon$,

$$\begin{aligned}
&v(s,y)\leq\varphi(s,y)\\
&-\frac{\partial\varphi}{\partial t}(s,y)-\mathcal{L}\varphi(s,y)-f\left(s,y,v(s,y),\left(D_x\varphi\right)'(s,y)\sigma(y)\right)>0.
\end{aligned}$$

Let $\tau = \inf\left\{s \geq t : \left|X_s^{t,x} - x\right| \geq \varepsilon\right\} \wedge (t+h)$, and consider the pair

$$\left(Y_s^1, Z_s^1\right) = \left(Y_{s\wedge\tau}^{t,x}, 1_{[0,\tau]}(s) Z_s^{t,x}\right), \quad t \leq s \leq t+h$$

By construction, $\left(Y_s^1, Z_s^1\right)$ solves the BSDE

$$\begin{aligned} -dY_s^1 &= 1_{[0,\tau]}(s) f\left(s, X_s^{t,x}, u\left(s, X_s^{t,x}\right), Z_s^1\right) ds - Z_s^1 dW_s, \quad t \leq s \leq t+h, \\ Y_{t+h}^1 &= u\left(\tau, X_\tau^{t,x}\right) \end{aligned}$$

On the other hand, the pair

$$\left(Y_s^2, Z_s^2\right) = \left(\varphi\left(s, X_{s\wedge\tau}^{t,x}\right), 1_{[0,\tau]}(s) D_x\varphi\left(s, X_s^{t,x}\right)' \sigma\left(X_s^{t,x}\right)\right), \quad t \leq s \leq t+h$$

satisfies, by Itô's formula, the BSDE

$$\begin{aligned} -dY_s^2 &= -1_{[0,\tau]}(s)\left(\frac{\partial\varphi}{\partial t} + \mathcal{L}\varphi\right)\left(s, X_s^{t,x}\right) - Z_s^2 dW_s, \quad t \leq s \leq t+h, \\ Y_{t+h}^1 &= \varphi\left(\tau, X_\tau^{t,x}\right). \end{aligned}$$

From the inequalities (2.2.2), and the strict comparison principle, we deduce $Y_0^1 < Y_0^2$, i.e. $u(t,x) < \varphi(t,x)$, then we have a contradiction. ■

Note that this representation can be restated in a more similar form to the linear Feynman-Kac formula as

Theorem 2.2.6. *Under assumptions the same assumptions as 2.2.5, the function defined by*

$$v(t,x) := Y_t^{t,x} = \mathbb{E}\left[g\left(X_T^{t,x}\right) + \int_t^T f\left(s, X_s^{t,x}, Y_s^{t,x}, Z_s^{t,x}\right) ds\right], \tag{2.69}$$

where (X_t, Y_T, Z_T) is the solution to the FBSE 2.42 restricted to $[t,T]$, is a viscosity solution to the parabolic PDE (2.59).

Finally, we can state a similar informal result for nonlinear equations in a bounded domain with boundary conditions in analogy with Theorem 2.2.3.

Theorem 2.2.7 ([11]). *Under the same hypothesis of Theorem 2.2.4 and assuming v is a classical solution that also satisfies the Dirichlet and Neumann conditions as in Theorem 2.2.3 with $h_n(t,x) = 0$, then the pair $(Y_t = v(t, X_t))$, $Z_t = \sigma(t, X_t)' D_x v(t, X_t)$ is the solution to a backward stochastic differential equation similar to (2.42) with the forward process reflected at the Neumann boundary and stopped at the first time it hits the Dirichlet boundary. And conversely, if (X, Y, Z) is a solution to the BSDE with the forward process reflecting at the Neumann boundary and stopped at the Dirichlet boundary, then $v(t,x) := Y_t^{t,x}$ is continuous and is a viscosity solution to the nonlinear PDE that also satisfies boundary conditions.*

Chapter 3

Deep Learning Methods for PDEs

Partial differential equations (PDE's) are ubiquitous among the tools for modeling complex phenomena in all sciences. However, we almost never have explicit solutions for them, making it difficult to describe those phenomenons and make accurate predictions about them. Hence, we need numerical methods to provide approximate solutions to those equations, for example, classical methods are finite differences, finite elements and spectral methods. Those rely on different discretizations of the particular problem that we can use for calculating approximations in different forms and with varying levels of accuracy. Since the advent of fast computers and efficient tools for programming them, this process is effective for many kinds of problems.

Now, when we attempt to solve numerically some particular problem, we need to play with the trade-off between accuracy of the approximate solution and the computational cost needed to obtain it. Indeed, with those classical methods, a small approximation error requires a finer grid, which implies more computational resources to store and process the information required by the method. In consequence, for some problems, we may not be able to calculate an accurate enough solution in a feasible computational time.

This is the case for high dimensional PDE's, for which the size of discretization usually scales exponentially with the number of points used for each dimension. For example, if we try to use a finite difference scheme in a 100-dimensional unit square $[0,1]^{100}$ with N points in each dimension, we would need N^{100} points in total, making it impossible to even store them in a computer. In practice, high dimension can be considered as low as $d > 4$, for which traditional methods cannot be used as regularly. This problem is known as the *curse of dimensionality*, a term established by Bellman when considering problems in dynamic programming.

High dimensional PDE's appear in many contexts, such as asset pricing, image denoising, statistical physics, many-body quantum mechanics, optimal control and game theory. Therefore, there is a necessity for numerical methods that are able to overcome this difficulty. Early attempts to solve this kind of problems used the connection between stochastic diffusions and parabolic PDE's, as we seemed in the preceding chapter. In fact, if the PDE is linear, the linear Feynman-Kac 2.2.2 formula can be used to provide an approximate solution by computing the expectation using simulated paths of the process through the Monte-Carlo approach. The convergence of this formulation is independent of the dimension of the underlying process, and therefore does not suffer from the curse dimensionality.

Nevertheless, if we try a similar approach using the non-linear Feynman-Kac formula 2.2.6 for more general non-linear equations, we would have to deal with solving numerically the associated BSDE. There are numerical methods to approximate the set of solution processes (X, Y, Z), but they are not as simple as an Euler-Maruyama discretization for a forward process. Generally, they require the computation of conditional expectations that almost never are computationally cheap and hence is not a straightforward generalization of the former linear approach. Despite this, some progress has been made under this formulation, see for example [20]. Other solutions methods are based in fixed point iterations and branching methods [21].

Representing functions in a high dimensional space is a problem encountered in many other areas of applied mathematics. Particularly, in recent times, the analysis and inference on big amounts of data has emerged as the fascinating research area of *machine learning*. Many methods have been proposed for this goal, for example, regression methods, support vector machines and tree methods. Nonetheless, the approach that has encountered more success when trying to approximate high dimensional functions using big amounts of data is deep learning. In this setting, we parametrize functions using structures that use composition of simpler function for approximate complex ones, these structures are called neural networks. We refer the reader to [1] for a first exposition of the topic.

The idea of using this neural network parametrization of functions to solve PDE's can be tracked to the 80's, when a perceptron layer approximation was proposed to approximate solutions to PDEs. However, due to the high computational cost of training a neural network, a successful attempt was not achieved until recently[22].

This is a very new area of research, for which many open questions remain. Particularly, it is not well understood yet if the curse of dimensionality is solved, even if there is work for certain equations that ensures it. Also, there is not yet a good understanding of why different classes of neural networks are useful to approximate certain functions and how to tune adequately its parameters to do it efficiently. In consequence, even if it is possible to give a convergence proof for certain cases, most algorithms rely on empirical experimentation and heuristic arguments to provide reasonable approximate solutions.

In this chapter we review some of these methods, implement them for toy examples and perform a comparison of speed, accuracy and practical usefulness for solving PDE's.

3.1 Unbounded problems

Let's start with problems in free space. In the same setup as Theorem 2.2.4, we deal with the following equation with terminal condition

$$\begin{aligned}&\frac{\partial v}{\partial t}(t,x)+\mu(t,x)\cdot D_x v(t,x)+\frac{1}{2}\operatorname{Tr}(\sigma(t,x)\sigma^T(t,x)D^2_{xx}v(t,x))\\&+f(t,x,v(t,x),\sigma(t,x)'D_x v(t,x))=0\\&v(T,x)=g(x),\end{aligned}\tag{3.1}$$

that can also be written as

$$\begin{aligned}&\frac{\partial v}{\partial t}(t,x)+\mathcal{L}_t v(t,x)+f(t,x,v(t,x),\sigma(t,x)'D_x v(t,x))=0\\&v(T,x)=g(x),\end{aligned}\tag{3.2}$$

using the infinitesimal generator $\mathcal{L}_t$ of the forward process X as in (2.49).

We have proven that we can construct a viscosity solution to this equation by setting $v(t,x)=Y_t^{t,x}$, where Y is the solution process to the FBSDE

$$\begin{aligned}&dX_s=\mu(s,X_s)ds+\sigma(s,X_s)dW_s\\&X_t=x,\\&dY_s=-f(s,X_s,Y_s,Z_s)ds+Z_s dW_s\\&Y_T=g(X_T).\end{aligned}\tag{3.3}$$

Moreover, we have that $Y_t=v(t,X_t)$ and $Z_t=\sigma(t,X_t)'D_x v(t,X_t)$.

3.1.1 Deep BSDE method

The first deep learning algorithm that was successfully applied to solve equation (3.1) was proposed by Han, E and Jentzen [7, 23]. This algorithm aims to approximate $Y_0=v(0,x)$ for some point $x\in\mathbb{R}^n$, and is similar in spirit to the stochastic shooting method for ODE's.

Here we discretize the time domain $0=t_0<t_1<\cdots<t_{N-1}<t_N=T$ and the FBSDE system a forward equation using the Euler-Maruyama scheme for $n=0,\ldots,N-1$,

$$X_{t_{n+1}}\approx X_{t_n}+\mu\left(t_n,X_{t_n}\right)\Delta t_n+\sigma\left(t_n,X_{t_n}\right)\Delta W_n\tag{3.4}$$

and

$$\begin{aligned}v\left(t_{n+1},X_{t_{n+1}}\right)\approx{}&v\left(t_n,X_{t_n}\right)-f\left(t_n,X_{t_n},v\left(t_n,X_{t_n}\right),\sigma'\left(t_n,X_{t_n}\right)D_x v\left(t_n,X_{t_n}\right)\right)\Delta t_n\\&+\sigma\left(t_n,X_{t_n}\right)'D_x u\left(t_n,X_{t_n}\right)\Delta W_n,\end{aligned}\tag{3.5}$$

where $\Delta t_n=t_{n+1}-t_n$ and $\Delta W_n\sim\mathcal{N}(0,\Delta t_n)$.

The main idea of this algorithm is to transform the problem in a learning one approximating the unknown product $\sigma\left(t_n,X_{t_n}\right)'D_x u\left(t_n,X_{t_n}\right)$ with a fully coupled neural network for each time step, i.e

$$\sigma\left(t_n,X_{t_n}\right)'D_x u\left(t_n,X_{t_n}\right)\approx\mathcal{Z}_n(X_{t_n}|\theta_n),\tag{3.6}$$

where θ_n denotes the parameters of the neural network at time t_n. Each one of these networks receives as inputs the simulated paths (3.4) and therefore its input layers have d neurons. Furthermore, the desired solution $v(0,x)$ and its derivative $D_x v(0,x)$ also will be parameters to be learned in the model, it means $v(0,x) \approx \theta_{v_0}$ and $D_x v(0,x) \approx \theta_{D_x v_0}$. Thus, the total set of parameters to be optimized is

$$\theta = \{\theta_{v_0}, \theta_{D_x v_0}, \theta_1, \theta_2, \cdots, \theta_n\}. \tag{3.7}$$

If we need the solution $v(0,x)$ for all x in some region Ω, we can choose to parametrize $v(0,x) \approx \theta_{v_0}$ with a neural network and simulate the process X_t with random initial conditions in Ω.

The set of parameters will be optimized such that the stacked solution $\hat{u}(\{X_{t_n}\}_0^N, \{W_{t_n}\}_0^N)$, constructed with (3.5), resembles the terminal condition $g(X_{t_N})$. This is achieved defining the loss function

$$\ell(\theta) = \mathbb{E}\big[|g(X_{t_N}) - \hat{u}(\{X_{t_n}\}_0^N, \{W_{t_n}\}_0^N)|^2\big], \tag{3.8}$$

which will be minimized using deep learning standard methods for training, for example, the ADAM optimizer.

The overall method can be thought as a constrained minimization problem of the form

$$\begin{aligned} &\inf_{\theta} \hat{u}(\{X_{t_n}\}_0^N, \{W_{t_n}\}_0^N) \\ s.t \quad & X_0 = \xi, \quad Y_0 = \theta_{v_0} \\ & X_{t_{n+1}} = X_{t_n} + \mu(t_n, X_{t_n}) \Delta t_n + \sigma(t_n, X_{t_n}) \Delta W_n \\ & Z_{t_n} = \mathcal{Z}_n(X_{t_n}) \\ & Y_{t_{n+1}} = Y_{t_n} - f(t_n, X_{t_n}, Y_{t_n}, Z_{t_n})\Delta t + Z'_{t_n} \Delta W_n \end{aligned} \tag{3.9}$$

where ξ is random variable uniformly distributed on Ω. And it can be summarized in the diagram shown in figure 3.1.

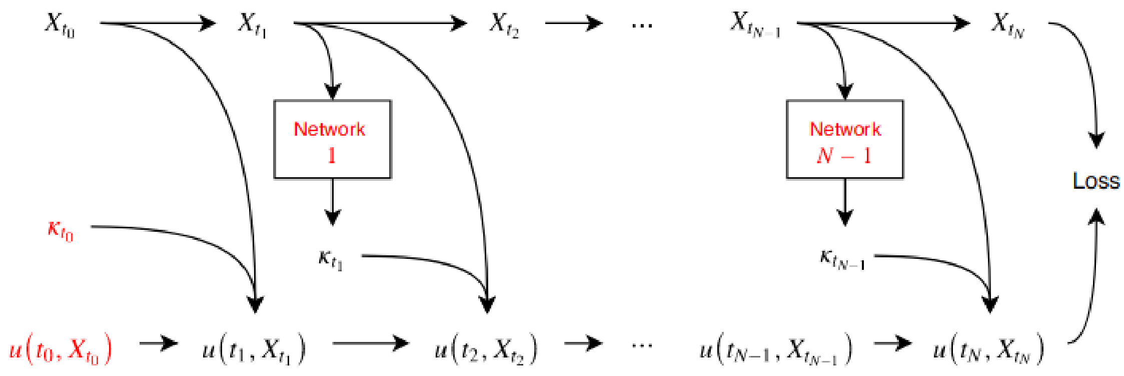


Figure 3.1: Deep BSDE diagram [24]. In red are the parameters to be optimized in the algorithm

The convergence of this algorithm has been proved in [25] for fully coupled FBSDEs. The assumptions needed are very general, and the proof is rather technical, so for the sake of brevity, we only state some imprecise results.

Denote by X_t, Y_t, Z_t the exact solution to the FBSDE (3.3), and by $X_{t_i}^\pi, Y_{t_i}^\pi, Y_{t_i}^\pi$ the discrete solution to the constrained optimization problem (3.9). Also, let's denote $h = \max_i \Delta t_i$. The first result states that the simulation error can be bounded through the value of the loss function (3.8).

Theorem 3.1.1 (Error of discretization is bounded by loss function [25]). *Under some assumptions, there exist a constant C, independent of h, d and n, such that for sufficiently small h*

$$\sup_{t\in[0,T]}\left(E\left|X_t-\hat{X}_t^\pi\right|^2+E\left|Y_t-\hat{Y}_t^\pi\right|^2\right)+\int_0^T E\left|Z_t-\hat{Z}_t^\pi\right|^2 \ \mathrm{d}t \leq C\left[h+E\left|g\left(X_T^\pi\right)-Y_T^\pi\right|^2\right]$$

where $\hat{X}_t^\pi = X_{t_i}^\pi, \hat{Y}_t^\pi = Y_{t_i}^\pi, \hat{Z}_t^\pi = Z_{t_i}^\pi$ for $t \in [t_i, t_{i+1})$

The second result establishes that the optimal value of the loss function can be small if the approximation capability of the family of parametric functions (neural networks) is good enough. Denote by $\mathcal{N}_0'$ and $\{\mathcal{N}_i\}_0^{N-1}$ the parametric function spaces generated by neural networks, then we have

Theorem 3.1.2 (Optimal loss function is bounded by approximation error [25]). *Under some assumptions, there exists a constant C, independent of $\Delta t, d$ and n, such that for sufficiently small Δt,*

$$\begin{aligned}
&\inf_{\mu_0^\pi\in\mathcal{N}_0',\phi_i^\pi\in\mathcal{N}_i} E\left|g\left(X_T^\pi\right)-Y_T^\pi\right|^2\\
&\leq C\left\{h+\inf_{\mu_0^\pi\in\mathcal{N}_0',\phi_i^\pi\in\mathcal{N}_i}\left[E\left|Y_0-\mu_0^\pi(\xi)\right|^2\right.\right.\\
&\left.\left.+\sum_{i=0}^{N-1} E\left|E\left[\tilde{Z}_{t_i} \mid X_{t_i}^\pi, Y_{t_i}^\pi\right]-\phi_i^\pi\left(X_{t_i}^\pi, Y_{t_i}^\pi\right)\right|^2 h\right]\right\},
\end{aligned}$$

where $\tilde{Z}_{t_i} = h^{-1}E\left[\int_{t_i}^{t_{i+1}} Z_t \ \mathrm{d}t \mid \mathcal{F}_{t_i}\right]$. If μ and σ are independent of Y, the term $E\left[\tilde{Z}_{t_i} \mid X_{t_i}^\pi, Y_{t_i}^\pi\right]$ can be replaced with $E\left[\tilde{Z}_{t_i} \mid X_{t_i}^\pi\right]$.

Neural networks are a promising candidate for such approximation space of functions, as there are results in regard to the universal approximation and complexity of neural networks.

Note that, in practice, we cannot minimize exactly the loss function in the space of parametric functions, as the methods we use are generally iterative. Also, it is not known how better is the approximation depending on the width, deep and connections of the neural network, so the capability of approximation is not well understood yet. Therefore, this results only establish the convergence of the method in a general setting and are not useful for estimate the real velocity of convergence, the achievable loss in the training stage, nor the real accuracy of the approximate solution.

Now, we highlight the major drawbacks of this method

1. The number of neural networks to train grows linearly with the number of time steps in the discretization, making it very computationally costly to use small time steps.

2. We only have a full solution at time $t = 0$. At intermediate times we only have approximate solutions evaluated on sample paths $v(t, X_t)$. Therefore, we would need many of them to represent accurately the solution in the desired region.
3. Moreover, nothing guarantees that this intermediate steps resembles accurately the real solutions in between steps. This requirement is not well encoded in the loss function.
4. The time structure of the problem is not reflected in the separate approximations for each step, at least not directly.
5. It may be unstable or converge to saddle points, see [26].

Moreover, the structure of the neural networks is crucial for the practical convergence of the algorithm. The original work by [7] used fully coupled neural networks with 3 layers, the *relu* activation function and batch normalization, this structure is represented in figure 3.2. However, many changes can be made to accelerate the process of training or to achieve lower optimal losses. All of them are inspired by practical evidence and currently there is not enough understanding of how to choose theoretically the hyperparameters to reach the best convergence we can. Some of these modifications are explained below.

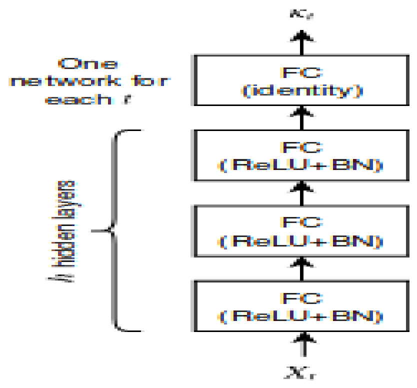


Figure 3.2: Neural network structure [24].

Merged Deep BSDE

The first such modification was proposed in [24]. In this work, the authors propose to use a single neural network to approximate the Z process for all time steps. Thus, this reduced model has less parameters to optimize, making this process faster, and also adds regularity to the computed gradients, which means that close in time parametrizations should be close for a given x.

Moreover, it was noted that using all information available at time t, like Y_t and $g(X_t)$, increases the performance and optimal loss. Thus, in order to merge all neural networks in a single one, we have to add new dimensions to the neural network to include the additional variables, hence we use a neural network $\mathcal{N}^\theta : (t, x) \in \mathbb{R}^{n+3} \to \mathbb{R}^n$ that uses $(t_n, Y_{t_n}, g(X_{t_n}), X_{t_n})$ as inputs to approximate Z_{t_n} for each t_n in the discretization. Note that Z_0 is also obtained through such network and therefore is not longer a parameter to be optimized. This process is summarized in the figure 3.3.

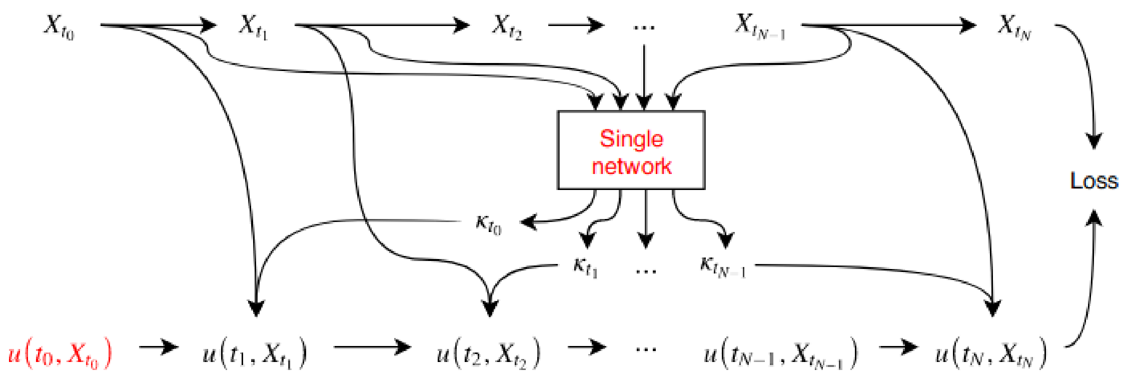


Figure 3.3: Merged Deep BSDE diagram [24]. In red are the parameters to be optimized in the algorithm

The structure of the neural network needs to be adapted to this new setup. In this case, the activation functions may be changed for the ELU function and batch normalization was not performed after each layer as the process in no longer stationary. This structure is represented in figure 3.4.

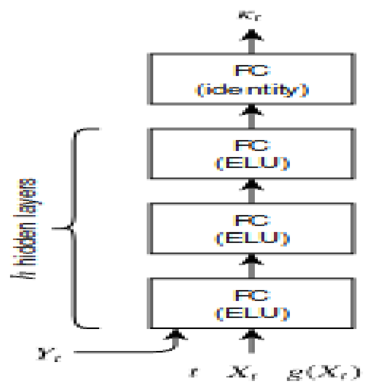


Figure 3.4: Neural network structure for merged BSDE [24].

Residual Merged Deep BSDE

Finally, another useful modification to the merged deep BSDE scheme was to modify the network structure adding shortcut connections between layers. This new configuration is represented in figure 3.5.

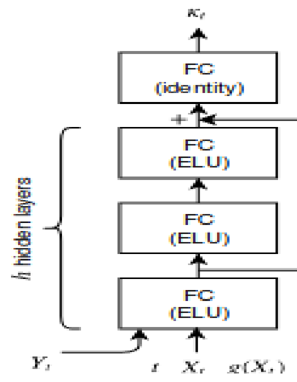


Figure 3.5: Neural network structure for merged residual BSDE [24].

3.1.2 Raissi's method

To circumvent some of the problems encountered with the deep BSDE method and its variants, Raissi [27] proposed a new scheme based on the same stochastic formulation. In this approach, the solution $v(t, x)$ is directly approximated with a neural network $\mathcal{N}(x, t)$ that takes as inputs x and t instead of its gradient as before.

Now, the constraints on problem (3.9) are relaxed and they are enforced weakly through the neural network loss function given by

$$\ell(\theta) := \mathbb{E}\left[\sum_{i=0}^{N-1} \Phi\left(t_i, X_{t_i}, Y_{t_i}, Y_{t_{i+1}}, \Delta W_{t_i}\right) + \left(g\left(X_{t_N}\right) - Y_{t_N}\right)^2\right] \tag{3.10}$$

where

$$\begin{aligned}\Phi\left(t_i, X_{t_i}, Y_{t_i}, Y_{t_{i+1}}, \Delta W_{t_i}\right) = \Big(Y_{t_{i+1}} - Y_{t_i} + f\left(t_i, X_{t_i}, Y_{t_i}, \sigma'\left(t_i, X_{t_i}\right)\widehat{Z}_{t_i}\right)\left(\Delta t_i\right) \\ -\widehat{Z}'_{t_i}\sigma\left(t_i, X_{t_i}\right)\left(\Delta W_{t_i}\right)\Big)^2,\end{aligned} \tag{3.11}$$

and $\widehat{Z}_{t_i}$ is calculated with automatic differentiation in the neural network $\widehat{Z}_{t_i} = \hat{D}\mathcal{N}(t_i, X_{t_i})$. With this approach we can compute the solution at all times t and point in space x.

3.1.3 An example

Let's test those algorithms with a toy problem from control theory using the Hamilton-Jacobi-Bellman equation. For completeness, we briefly review the standard formulation of such problems in Appendix A.

We will perform a non-exhaustive comparison between the preceding algorithms by solving, just for fun, the Hamilton-Jacobi-Bellman equation associated to a linear-quadratic regulator in dimension 6.

Suppose that we have three stochastic particles in the plane, whose common state is described by the stochastic process $X_t = (\vec{x}_1, \vec{x}_2, \vec{x}_3) = (x_1, y_1, x_2, y_2, x_3, y_3) \in \mathbb{R}^6$, that satisfies the linear dynamics given by

$$\begin{aligned} dX_t &= 2\sqrt{\lambda}\alpha_t dt + \sqrt{2\nu} dW_t \\ X_0 &= x_0, \end{aligned} \tag{3.12}$$

where α_t is a joint control you can select for all particles individually and λ, ν are constants representing the strength of the control we exert and the magnitude of noise the particles feel from its environment.

We wish to bring the three particles to the center of the unit square in the plane at time $T = 1.0$, i.e the desired terminal state is $X_t = (0.5, 0.5, 0.5, 0.5, 0.5, 0.5)$, while trying to avoid that the particles are close to each other during their trajectories. Thus, we will try to choose α_t to minimize the cost

$$J(\alpha_t) = \mathbb{E}\left[\int_0^T (|\alpha_t|^2 + F(t, X_t))dt + g(X_T)\right], \tag{3.13}$$

where we model the avoiding condition with the $F(t, X_t)$ term given by

$$F(t, X_t) = F(t, (\vec{x}_1, \vec{x}_2, \vec{x}_3)) = C\left(e^{-\frac{|\vec{x}_1-\vec{x}_2|^2}{\sigma}} + e^{-\frac{|\vec{x}_1-\vec{x}_3|^2}{\sigma}} + e^{-\frac{|\vec{x}_2-\vec{x}_3|^2}{\sigma}}\right), \tag{3.14}$$

with C, σ fixed constants, and we also model the desired terminal condition through the terminal cost $g(X_T)$ given by

$$g(x) = |x - (0.5, 0.5, 0.5, 0.5, 0.5, 0.5)|^2. \tag{3.15}$$

Therefore, as described in Appendix A, to solve this problem we may try to solve the Hamilton-Jacobi-Bellman equation for the value function $V(t, x)$

$$\frac{\partial V}{\partial t} + \nu\Delta V - \lambda|\nabla V|^2 + F(t, x) = 0 \tag{3.16}$$

subject to the terminal condition

$$V(T, x) = g(x). \tag{3.17}$$

Note that we have access to a probabilistic representation of such value function given by (A.27). We may simulate sample paths to calculate the expected value at a given space-time point using the Monte-Carlo approach. However, this is not very useful for our purposes, as we require the optimal controls $\hat{\alpha}(t, x) = -\sqrt{\lambda}D_xV(t, x)$, and with this representation we do not have access to gradients of the solution.[1] Despite this, it will be useful to test the convergence of our numerical methods comparing the obtained solutions in a single point.

Note also that it is not very practical to apply traditional discretization methods as finite differences or finite elements. The first problem we face is that the domain is not bounded,

[1]Please ignore the fact that we can obtain analytically the expected value with the known density of Brownian motion, i.e. the heat kernel, and then take its derivative to obtain the desired controls, as nobody would want to perform such long and tedious calculation.

and even if we focus on some small region of interest with artificial boundary condition, the discretization of a 6-dimensional space is not computationally feasible.

Thus, we may interpret this problem in the framework previously developed. To do that, note that (3.16) is the same as (3.1) if we choose

$$\mu(t,x) = 0 \qquad \sigma(t,x) = \sqrt{2\nu}\mathbb{I}_{6\times 6} \tag{3.18}$$

and

$$f(t,x,V(t,x),\sigma(t,x)'D_xV(t,x)) = -\lambda|D_xV|^2 + F(t,x). \tag{3.19}$$

Therefore, we can apply the deep BSDE solver and all its variants to solve this PDE. We implemented these in the Python language using the Pytorch framework with its automatic differentiation and neural networks capabilities.

In this example we use $\lambda = 1.0$ and $\nu = 0.01$. For all methods, we discretized time using a step of $\Delta t = \frac{1}{120}$. The sample paths were simulated using the Euler-Maruyama scheme.

We trained all neural networks using the ADAM optimizer with a learning rate of $\eta = 0.01$ and using batches of 64 sample paths to calculate the expectations in loss functions.

In the deep BSDE method, for every subnet approximating Z_{t_n} we used a fully coupled neural network with 2 layers with 16 neurons each, using a batch normalization layer in between to prevent gradients exploding and the ReLu as activation function. We also parametrize the initial solution and its gradient with fully coupled neural networks with the same structure as before.

In the merged deep BSDE method, we use a 3 layer fully coupled neural network with 20 neurons in each layer, with the ELU activation function to parametrize the mapping $(t,x) \to Z(t,x)$.

In the residual merged deep BSDE method, we use a 3 layer fully coupled neural network with 20 neurons in each layer, with the ELU activation function, and with a shortcut connection as depicted in Figure 3.5 to parametrize the mapping $(t,x) \to Z(t,x)$.

For the Raissi's method we used a 4 layer fully coupled neural network with 15 neurons in each layers to parametrize the approximated solution $\varphi(t,x)$. We note that this method is not well suited for computing gradients and therefore controls, as the automatic differentiation of the parametrized solution is slow and not very precise.

In Figure 3.6, we depict the training error with respect to the exact solution at the point $x = (0.5, 0.5, 0.5, 0.5, 0.5, 0.5)$ at $t = 0$, obtained using the formula (A.27) with 20000 sample paths, for those four methods. Note the instability that may occur when changing the network structure in the case of the merged residual method, but also the speed of convergence that can be achieved when it is well chosen, as in the case of merged fully coupled method. Note also that the number of training stages do not reflect efficiently the time needed to compute the solution, as in our case the Raissi's method took much longer to achieve that accuracy, probably due to the need of use automatic differentiation for computing the gradients in the FBSDE discretization.

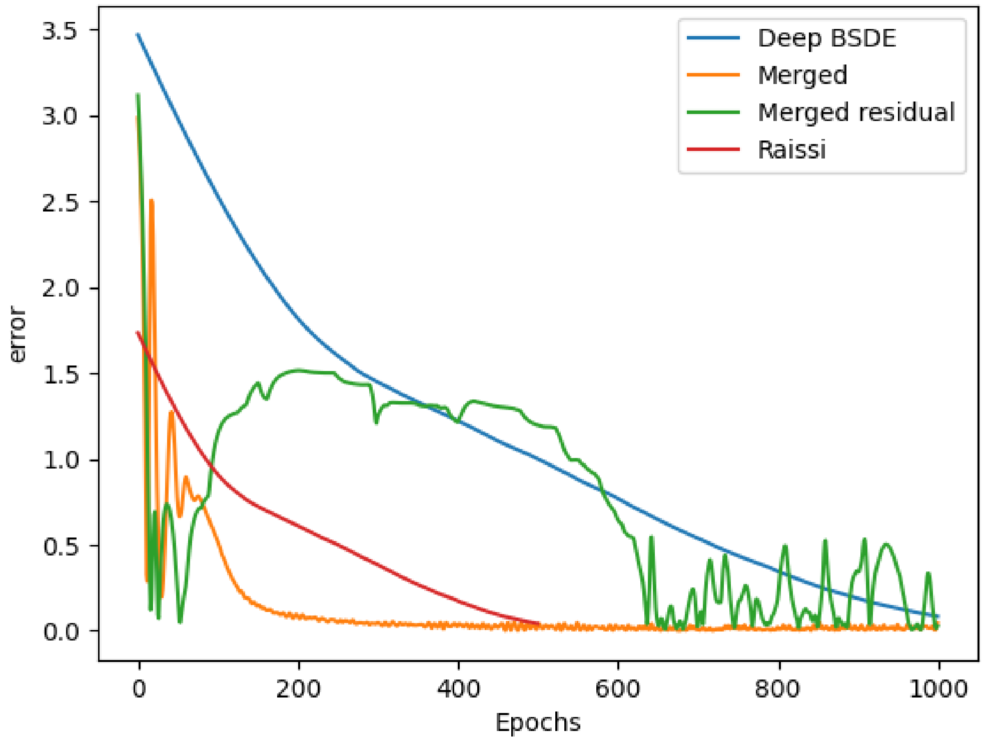


Figure 3.6: Accuracy in the training process

In Figure 3.7, an optimal controlled path is shown. Note that the particles try to reach the center of the unit square, but as the get closer, the cost associated running for proximity increments, thus it is not effective to continue advancing to the center. Hence, each particle will try to stay the closest possible to the center at final time, while avoiding being near each other. Finally, it is worth to note that we did not try to perform an exhaustive analysis of advantages for each one because we found that practical performance were very dependent on implementation details, thus do not provide a homogeneous framework to perform fair conclusions.

We note that when we optimize the parameters of the neural networks, there is maximum achievable error depending on the network structure and time discretization size. In this toy example we reached an estimated error with order of magnitude 10^{-2}. Fewer error would require more epochs in the training stage and smaller time step.

3.2 Bounded problems

In this section, we consider PDE's with boundary conditions of the form

$$\frac{\partial v}{\partial t}(t,x) + \mathcal{L}v(t,x) + f(t,x,v(t,x),\sigma(t,x)' D_x v(t,x)) = 0 \tag{3.20a}$$

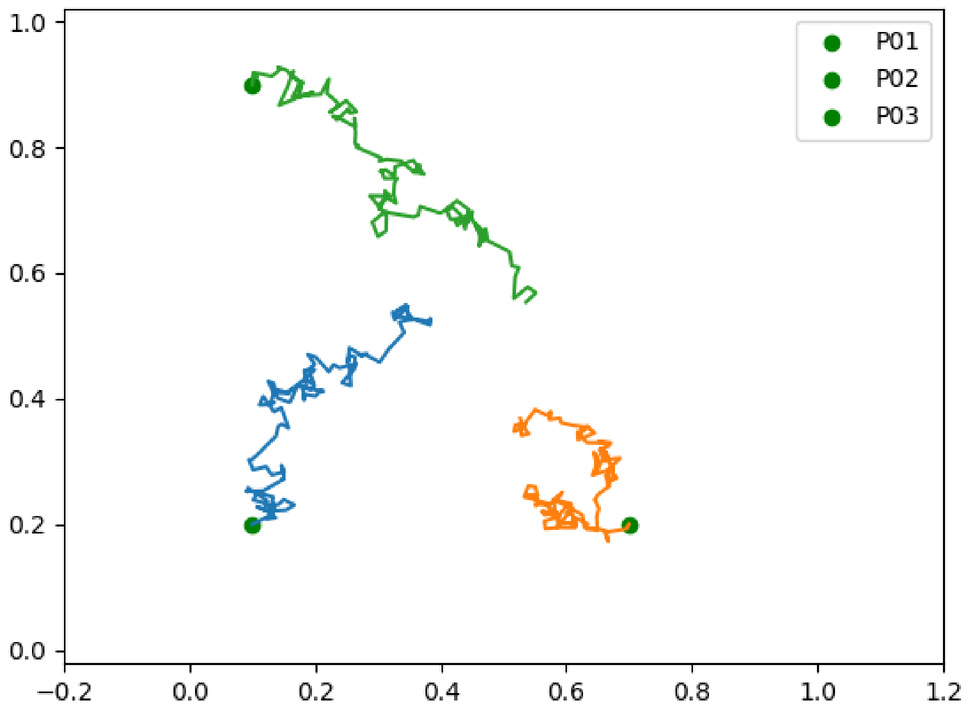


Figure 3.7: Optimal controlled sampled path

with terminal condition

$$v(T, x) = g(x) \quad \forall x \in \Omega, \tag{3.20b}$$

and Dirichlet and Neumman conditions

$$v(t, x) = h_d(x) \quad \forall x \in \Gamma_D \text{ and } \forall t \in [0, T] \tag{3.20c}$$

$$\frac{\partial v}{\partial n}(t, x) = h_n(x) \quad \forall x \in \Gamma_N \text{ and } \forall t \in [0, T], \tag{3.20d}$$

where Ω is a bounded domain, regular enough for a solution to exist, and $\Gamma_D \dot{\cup} \Gamma_N = \partial\Omega$. Here again $\mathcal{L}$ represents the infinitesimal generator of the process X that is solution to (3.3).

We will assume the following

Assumptions 3.2.1. *The following holds*

I. *The domain Ω is bounded with piecewise smooth boundary*

II. *The boundary problem (3.20) admits a unique classical solution $v \in C^{1,2}([0, t] \times \Omega) \cap C(\bar{\Omega} \times [0, T])$. Moreover, the gradient of v satisfies a polynomial growth condition in x, i.e , $|D_x v(t, x)| \leq C(1 + |x|^p)$ for $(t, x) \in [0, T] \times \Omega$ and some $C, q \geq 0$*

III. The forward process in (3.3) *admits a unique strong solution globally in time.*

If, in addition to the PDE, we impose Neumman/Dirichlet boundary conditions, the previous approaches must be modified because the Feynman-Kac formula does not account for them explicitly. We will try two different approaches.

3.2.1 Deep Galerkin method

The first approach is named the deep Galerkin method (DGM) and was proposed in [3]. This method is also called Physics Informed Neural Networks (PINN's). It does not rely on a probabilistic representation of the solution, but on the explicit form (3.20).

The DGM algorithm approximates the solution $v(t,x)$ with a deep neural network $\varphi(t,x|\theta)$, where θ are the network's parameters. We will choose this function aiming to minimize the loss function given by

$$\begin{aligned}\mathcal{L}(\varphi) :=& \alpha_{int}\mathcal{L}_{\text{DGM,int}}(\varphi) + \alpha_T\mathcal{L}_{\text{DGM,T}}(\varphi) + \alpha_d\mathcal{L}_{\text{DGM,d}}(\varphi) + \alpha_n\mathcal{L}_{\text{DGM,n}}(\varphi) \\ :=& \alpha_{int}\left\|\frac{\partial\varphi}{\partial t} + \mathcal{L}\varphi + f(t,x,\varphi(t,x),\sigma(t,x)'D_x\varphi(t,x))\right\|^2_{[0,T]\times\Omega,\nu_1} \\ &+\alpha_T\left\|\varphi(T,x|\theta) - g(x)\right\|^2_{\Omega,\nu_2} \\ &+\alpha_d\left\|\varphi(t,x) - h_d(x)\right\|^2_{[0,T]\times\partial\Gamma_D,\nu_3} \\ &+\alpha_n\left\|\hat{n}\cdot D_x\varphi(t,x) - h_n(x)\right\|^2_{[0,T]\times\partial\Gamma_N,\nu_4}\end{aligned}, \tag{3.21}$$

where we use the notation for the norms $\|f(y)\|^2_{\mathcal{Y},\nu} = \int_{\mathcal{Y}} |f(y)|^2 d\nu$, given a positive probability density ν on $\mathcal{Y}$. Here, $J(\varphi)$ measure how well ϕ satisfies the PDE with its initial and boundary conditions weighted by the α coefficients. If $J(\varphi) = 0$, then φ is a solution to (3.20).

To obtain the minimizing parameters θ, we will perform a training procedure of the neural network with a stochastic gradient descent algorithm. We will sample points according to $\nu_1, \nu_2, \nu_3, \nu_4$ and calculate a Monte-Carlo approximation of the functional loss (3.21) using automatic differentiation to calculate the derivatives appearing in the PDE operator and boundary conditions. With this approximation, we will take a descent step in the gradient direction calculated again using automatic differentiation.

Using the universal approximation theorem for single layer neural networks and some standard assumptions on the behavior of the problem, we can establish the strong convergence in $L^2(\Omega \times [0,T])$ of this algorithm as the number of neurons in the layer tends to infinity, assuming we are able to find exactly the optimal parameters of the network in each size. The reader may consult [3] for such proof.

However, the practical convergence presents difficulties as it depends strongly on a set of well-chosen hyper-parameters, such as the learning rate of the descent method, the batch sizes, the weights in the loss function, the network architecture, as well as the sampling densities. There is not a straightforward way to choose such parameters, thus the convergence is

Note also that there is a big operational cost involved in the computation of derivatives appearing in the loss function through automatic differentiation. For operators using fully coupled second order derivatives, this cost becomes prohibitive in high dimensions. Therefore, we need alternatives for problems with these restrictions.

3.2.2 Interpolating BSDEs with PINNs

To circumvent these issues, we may combine the powerful ideas of stochastic representation of solutions of PDEs with the forcing technique used in the PINNs approach. This method was proposed by [28], and its main idea is to define a parameter to interpolate between the losses used by each scheme to combine the strengths of both. In the same spirit of the last methods, we parameterize the PDE's solution by a neural network ϕ and try to adjust its parameters to give a good enough approximation of the desired solution.

In this method, we use the Ito's formula applied to $v(X_t, t)$ to obtain

$$v(T, X_T) - v(0, X_0) = \int_0^T \left(\frac{\partial}{\partial s} + \mathcal{L}\right) v(s, X_s) ds + \int_0^T \sigma' D_x v(s, X_s) \cdot dW_s, \tag{3.22}$$

and, inspired in this relation for the interior process and the forcing of boundary conditions used in the DGM method, we aim to minimize the *diffusion loss* with interpolation parameter $\mathrm{t} \in (0, \infty)$ defined as

$$\mathcal{L}_{\text{diff}}^{\mathrm{t}}(\varphi) = \alpha_{\text{int}}\, \mathcal{L}_{\text{diff,int}}^{\mathrm{t}}(\varphi) + \alpha_{\mathrm{T}} \mathcal{L}_{\text{diff, T}}^{\mathrm{t}}(\varphi) + \alpha_{\mathrm{d}} \mathcal{L}_{\text{diff, d}}^{\mathrm{t}}(\varphi) + \alpha_{\mathrm{n}} \mathcal{L}_{\text{diff, n}}^{\mathrm{t}}(\varphi) \tag{3.23}$$

where

$$\begin{aligned}
\mathcal{L}_{\text{diff,int}}^{\mathrm{t}}(\varphi) &= \mathbb{E}\Bigg[\Bigg(\varphi(\mathcal{T}, X_{\mathcal{T}}) - \varphi(t_0, X_{t_0}) - \int_{t_0}^{\mathcal{T}} \sigma^\top \nabla \varphi(s, X_s) \cdot \mathrm{d}W_s \\
&\quad + \int_{t_0}^{\mathcal{T}} f\left(s, X_s, \varphi(s, X_s), \sigma^\top \nabla \varphi(s, X_s)\right) \mathrm{d}s\Bigg)^2\Bigg], \\
\mathcal{L}_{\text{diff,T}}^{\mathrm{t}}(\varphi) &= \mathbb{E}\left[\left(\varphi\left(T, X^{(T)}\right) - g\left(X^{(T)}\right)\right)^2\right] \\
\mathcal{L}_{\text{diff, d}}^{\mathrm{t}}(\varphi) &= \mathbb{E}\left[\left(\varphi\left(t^{\mathrm{d}}, X^{\mathrm{d}}\right) - h_d\left(t^{\mathrm{d}}, X^{\mathrm{d}}\right)\right)^2\right], \\
\mathcal{L}_{\text{diff, n}}^{\mathrm{t}}(\varphi) &= \mathbb{E}\left[\left(D_x \varphi(t^{\mathrm{n}}, X^{\mathrm{n}}) - h_n(t^{\mathrm{n}}, X^{\mathrm{n}})\right)^2\right].
\end{aligned} \tag{3.24}$$

Here, the process $(X_t)_{0 \le t \le \mathcal{T}}$ is the solution to (3.3) with initial condition $(X_0, t_0) \sim \nu_{\Omega \times [0,T]}$, where $\nu_{\Omega \times [0,T]}$ is a measure with full support on $\Omega \times [0, T]$, and maximal trajectory length $\mathrm{t} > 0$. In addition, the stopping time $\mathcal{T} := (t_0 + \mathrm{t}) \wedge \tau \wedge T$ refers to the random final time associated with the realization of the path X_t when it either hit the boundary at time $\tau = \inf\{t > 0 : X_t \notin \Omega\}$ or runs for $t_0 + \mathrm{t}$ or reaches time T. Furthermore, using the same notation as in the case of DGM, $X^{(T)} \sim \nu_2$, $(t^{\mathrm{n}}, X^{\mathrm{d}}) \sim \nu_3$ and $(t^{\mathrm{n}}, X^{\mathrm{n}}) \sim \nu_4$.

In this case, the data inside the domain is not sampled as points with distribution ν_1 as in DGM, but along trajectories of the diffusion. Hence, we do not need to calculate second derivatives of φ explicitly trough automatic differentiation, but are approximated using the underlying process X_t. Note also that using a maximal trajectory length we avoid computing large first exit times that may arise depending on Ω or long simulation times if T is large.

Now, let's prove that this loss function is indeed suitable for the problem (3.20).

Theorem 3.2.1 ([28]). *Let $\mathcal{F} \subset C^{1,2}([0,t] \times \Omega) \cap C(\bar{\Omega} \times [0,T])$ be the function space of neural networks. Then, under assumptions 2.1.3 and 3.2.1, for $\varphi \in \mathcal{F}$, the following are equivalent*

I. φ fulfills the boundary value problem

II. The difussion loss vanishes on φ

$$\mathcal{L}^{\mathrm{t}}_{diff}(\varphi) = 0 \tag{3.25}$$

(3.20).

Proof. $I \implies II$) Assume that φ satisfies (3.20) and X_s is the strong solution to the forward process (3.3), then, by Ito's formula, we have

$$\varphi(\mathcal{T}, X_{\mathcal{T}}) = v(0, X_0) + \int_{t_0}^{\mathcal{T}} \left(\frac{\partial}{\partial s} + \mathcal{L}\right) \varphi(s, X_s) ds + \int_{t_0}^{\mathcal{T}} \sigma' D_x \varphi(s, X_s) \cdot dW_s, \tag{3.26}$$

almost surely. It follows that $\mathcal{L}^{\mathrm{t}}_{\mathrm{diff,int}}(\varphi) = 0$, and similarly, as φ satisfies terminal and boundary conditions, we have $\mathcal{L}^{\mathrm{t}}_{\mathrm{diff,T}}(\varphi) = \mathcal{L}^{\mathrm{t}}_{\mathrm{diff,d}}(\varphi) = \mathcal{L}^{\mathrm{t}}_{\mathrm{diff,n}}(\phi) = 0$, so $\mathcal{L}^{\mathrm{t}}_{\mathrm{diff}}(\varphi) = 0$

$II \implies I$) Observe that $\mathcal{L}^{\mathrm{t}}_{\mathrm{diff,int}}(\varphi) = 0$ implies that

$$\varphi(\mathcal{T}, X_{\mathcal{T}}) = \varphi(0, X_0) + \int_{t_0}^{\mathcal{T}} \sigma' D_x \varphi(s, X_s) \cdot dW_s - \int_{t_0}^{\mathcal{T}} f(s, X_s \varphi(s, X_s), \sigma' D_x \varphi(s, X_s)) ds, \tag{3.27}$$

almost surely, and note that the same holds for ϕ replaced by v (3.22). Now, define the processes $\tilde{Y}_s = \phi(s, X_s)$,$\tilde{Z}_s = \sigma' D_x \phi(s, X_s)$ and $Y_s = v(s, X_s)$,$Z_s = \sigma' D_x v(s, X_s)$. Observe that those are W_t-progressively measurable and square integrable. Moreover, (3.27) and (3.22) means that they satisfy a BSDE with terminal condition $\xi = \varphi(\mathcal{T}, X_{\mathcal{T}})$ on $[t_0, \mathcal{T}]$. Thus, by the uniqueness of solution in Theorem 2.1.5, we have that $Y = \tilde{Y}$ and $Z = \tilde{Z}$. Observe also that $v(t_0, X_{t_0}) = Y^{t_0, X_{t_0}} = \tilde{Y}^{t_0, X_{t_0}} = \varphi(t_0, X_{t_0})$. Hence, we conclude that $v = \varphi$ $\nu_{\Omega} \times [0,T]-$ almost surely, and thus φ is the solution to (3.20) using that $\nu_{\Omega \times [0,T]}$ has full support. ∎

By the other hand, observe that this diffusion loss, in the case of an unbounded problem, where $\alpha_d = \alpha_n = 0$, is an interpolation between the weak BSDE loss defined as

$$\begin{aligned}\mathcal{L}_{BSDE} = \mathbb{E}\Big[\Big(g(X_T) - \varphi(t_0, X_{t_0}) - \int_{t_0}^{T} \sigma' D_x \varphi(s, X_s) \cdot dW_s \\ + \int_{t_0}^{T} f(s, X_s, \varphi(s, X_s), \sigma' D_x \varphi(s, X_s) ds)\Big)^2\Big]\end{aligned} \tag{3.28}$$

and the DGM loss as the following theorem shows.

Theorem 3.2.2 ([28])**.** *Let $\phi \in \mathcal{F}$. Assuming that the sampling distributions ν_1 for interior points in the DGM loss and $\nu_{\Omega\times[0,t]}$ for starting points in the BSDE and diffusion losses coincide, then we have*

I.
$$\frac{\mathcal{L}_{\textit{diff, int}}(\varphi)}{\mathrm{t}^2} \to \mathcal{L}_{DGM,int}(\varphi) \tag{3.29}$$
as $\mathrm{t} \to 0$

II.
$$\mathcal{L}_{\textit{diff, int}}(\varphi) \to \mathcal{L}_{BSDE}(\varphi) \tag{3.30}$$
as $\mathrm{t} \to \infty$

Proof. For I) note that the interior part of the diffusion loss can be written as

$$\mathcal{L}^{\mathrm{t}}_{\mathrm{diff,int}}(\varphi) = \mathbb{E}\left[\left(\int_{t_0}^{\mathcal{T}}\left(\frac{\partial}{\partial s} + \mathcal{L}\right)\varphi(s, X_s)ds + \int_{t_0}^{\mathcal{T}} f\left(s, X_s, \varphi(s, X_s), \sigma^\top \nabla\varphi(s, X_s)\right)\mathrm{d}s\right)^2\right], \tag{3.31}$$

and that $\mathcal{T} \to t_0$ as $\mathrm{t} \to 0$ almost surely, then by the dominated convergence theorem I follow. Moreover, note that $\mathcal{T} \to T$ as $\mathrm{t} \to \infty$ almost surely, thus II) holds. ■

In practice, we use the same strategy as in the DGM algorithm. We simulate paths of the X_t process to estimate the interior diffusion loss using Monte-Carlo method. We approximate the integral by using, for example, the Euler-Maruyama scheme. Furthermore, we sample points according to ν_2, ν_3, ν_4 to estimate the losses associated to boundary and terminal conditions, using automatic differentiation when necessary.

With this estimation of the loss, we take gradient descent steps to reduce sequentially the total error and obtain a good approximation of the solution. However, there is almost no theory about what are the optimal hyperparameters, as the weights in the loss, the learning rate, length of the sample paths or network architecture to be chosen in the algorithm to converge optimally, so empirical experience is needed to obtain approximations efficiently.

3.2.3 An example

In our implementation of these methods for boundary problems we saw that they were significantly harder than unbounded problems. The convergence of those algorithms was very dependent on hyperparameters such as learning rate, sample points distribution, weigths in the losses and network structure. Even for simple domains it was difficult to find parameters that worked. What is worse, as these methods parametrize directly the solution and not its gradient, they cannot be used to calculate accurately controls dependent on the space derivative, because it is generally not well approximated.

Thus, just for illustration of the methods for a very simple case, consider a modification of the preceding example. We wish to find the control function for a single particle that would take them to the exit door of an empty room in the least amount of time. In Figure 3.8 is depicted the domain considered, with some sample paths used for the training process. Note that some of them are reflected in the boundary, as we saw that this reduced the training time. However, reflecting the paths is more costly than stopping them at the boundary.

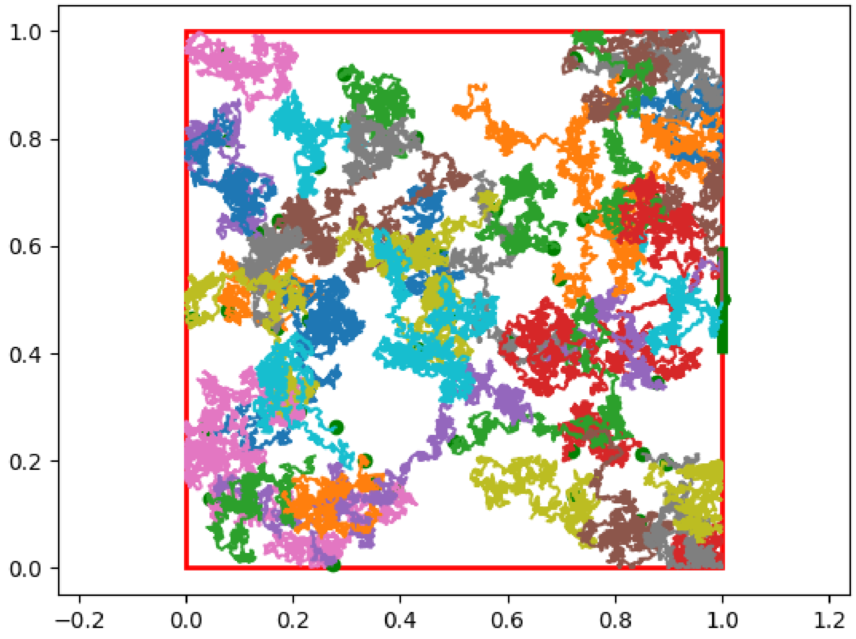


Figure 3.8: Empty room with sample paths

In this case, we would want to solve the HJB equation

$$\frac{\partial V}{\partial t} + \nu \Delta V - \lambda |\nabla V|^2 + 1 = 0, \tag{3.32}$$

subject to the Dirichlet and Neumann conditions

$$V(t, x) = 0 \quad \text{for } x \in \partial\Omega_d, \tag{3.33}$$

$$\frac{\partial V}{\partial \vec{n}}(t, x) = 0 \quad \text{for } x \in \partial\Omega_n, \tag{3.34}$$

and the terminal condition

$$V(1.0, x) = 0 \quad \text{for } x \in \Omega. \tag{3.35}$$

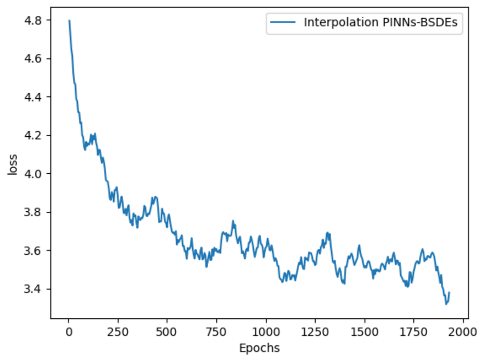
The constant 1 in the equation (3.32) models a steady running cost that will impulse the particle to exit the domain faster. The Dirichlet condition at the door implies that a process that reaches this part of the boundary will stay there without increasing its running cost. The Neumann boundary conditions represent the condition that the process is bounded in the room, which means that it is reflected along those boundary points when touched. Finally, the terminal condition implies that there is no cost in ending at every other part of the room.

We solve this equation using the DGM algorithm using batches of 512 interior points (t, x) uniformly distributed in $[0, T] \times \Omega$, 512 points uniformly distributed on the Neumann boundary, 200 points uniformly distributed on the Dirichlet boundary and 512 points uniformly distributed in Ω. The tunning of the weight losses was a critical task for the convergence of the algorithm. We found that it worked using $\alpha_{int} = 2.0$ and $\alpha_d = \alpha_n = \alpha_T = 1.0$. We use the DGM architecture for the neural network.

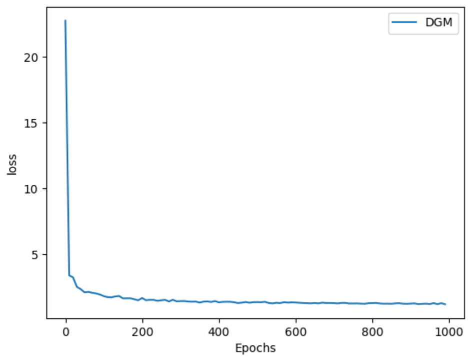
Similarly, we solve the same problem using the interpolation algorithm with the diffusion loss. We use batches of data of the same size as in the DGM algorithm. We use time discretization steps of $\Delta t = 0.01$ and a maximum path length of 20 steps. We use a DenseNet architecture for the neural network.

All networks were trained using the ADAM optimizer with learning rate of $\eta = 0.01$.

In Figure 3.9 we show the losses achieved during the training proccess of both algorithms, with respect to a fixed initial sample. Note they are not in the same graph as the losses are different in each case. In Figure 3.10 we plot our computed solution as a surface over the room domain at different time stages.

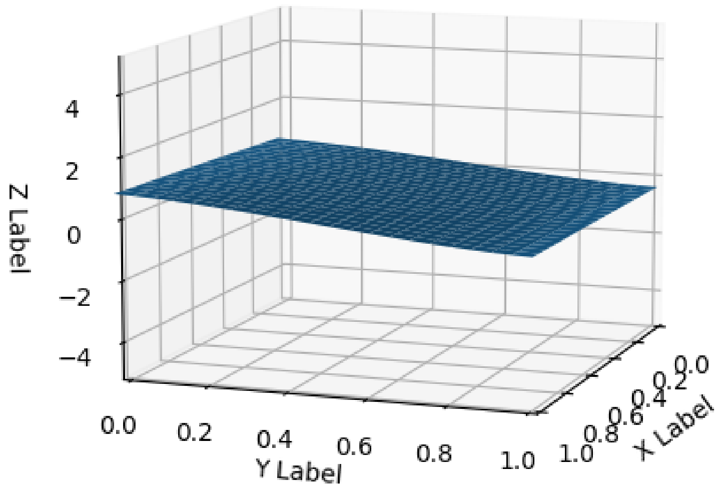


(a) Training loss interpolation algorithm

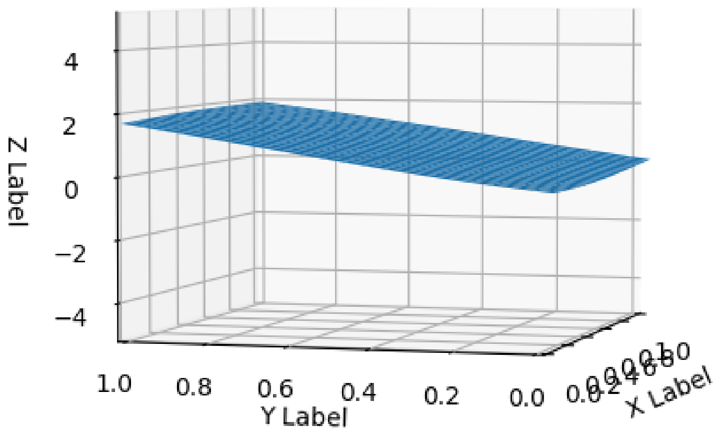


(b) Training loss DGM algorithm

Figure 3.9: Training losses for boundary problems algorithms



(a) Computed solution interpolation algorithm



(b) Computed solution DGM algorithm

What did not work

Our original idea to solve problems with boundary conditions was to use reflecting process to emulate Neumann conditions and stopping times to emulate Dirichlet conditions, as suggested by the linear and non linear Feynmann-Kac formulas Theorem 2.2.3 and Theorem 2.2.7.

For example, we considered the global control problem with three particles as in (3.16) but now in a room as in (3.32). In this case, boundary condition need to be stated in a high dimension space, which led us to confusion when choosing the ones that are adequate. The boundary of the 6-dimensional space is $\partial\Omega = \{(\vec{X_1}, \vec{X_2}, \vec{X_3}) \in \mathbb{R}^6 | X_i \in \Gamma_d \cup \Gamma_n\}$, where $\vec{X_i} \in \mathbb{R}^2$ represents the 2D coordinates of particle i and $\Gamma_{d,n}$ are the Dirichlet and Neumann parts of the boundary of the 2D room.

Our first option was to add the following boundary conditions to (3.16)

$$V(t,x) = 0 \quad \text{for } x \in \Xi_d, \tag{3.36}$$

where $\Xi_d = \{(\vec{X_1}, \vec{X_2}, \vec{X_3}) \in \mathbb{R}^6 | X_1, X_2, X_3 \in \Gamma_d\}$, which tries to model that all particles want reach the Dirichlet boundary, and

$$\frac{\partial V}{\partial n}(t,x) = 0 \quad \text{for } x \in \Xi_n, \tag{3.37}$$

where $\Xi_n = \{(\vec{X_1}, \vec{X_2}, \vec{X_3}) \in \mathbb{R}^6 | X_1 \text{ or } X_2 \text{ or } X_3 \in \Gamma_n\}$, which tries to model that particles reflect their paths at the boundary. Also, we modified the terminal condition to

$$V(T,x) = 0 \text{ for } x \in \Omega, \tag{3.38}$$

expecting that it is not a problem as the desire of the particles to exit quickly the room is modeled by it wanting to reach the Dirichlet boundary to avoid accumulating running cost.

To solve this equation, we simulated processes that are reflected in the Neumann part of the room and are stopped when all particles hit the Dirichlet boundary. However, we encountered many practical problems that did not allow us to verify that this approach effectively solved the equation. The first difficulty we encountered was that simulating reflected paths is not a vectorizable process, because in every time step the next position for each where is it at the moment and is not equal for all other particles, therefore, Python was not a great tool to calculate such paths, as the become very computationally expensive to simulate. To calculate efficiently the reflected paths we need some compiled of JIT-compiled tool, for which we used the Julia language. However, creating an interface for both languages was a painful process that we did not accomplish.

However, letting this code run long enough, we observed that there is a problem with this approach concerning to the Dirichlet boundary condition. If we sample Brownian paths for each particle starting at random positions in the room, it is a very rare event that all particles reach the Dirichlet boundary, therefore we are not capturing effectively this condition and our solution do reflect that.

We had two ideas to solve this problem. The first one is to use some modification of the *walking on spheres* algorithm to simulate first exist times from a high dimensional domain, and the second one was to modify the Dirichlet condition to decouple the problems for each particle, and stopped the process when any of the reached the Dirichlet boundary. Neither of them led us to an acceptable solution, and we have not and explanation for it, but probably it is due to implementation issues rather than a lack of theoretical justification.

Chapter 4

Application to N-player games

We have seen how deep learning tool can be used to solve partial differential equations, even in the high dimensional setting. However, we have not solved any problem for which we do not know an exact solution or an approximate one by means of classical numerical methods. Hence, in this section we will apply the exposed methods to solve a larger problem whose solution cannot be approximated by classical numerical methods. That is the case for N-agent games, which is the main topic of this section.

4.1 N-agent games

Classical optimal stochastic control deals with the problem of optimizing the optimal choice that a single agent should take to minimize a cost function in a random environment, see Appendix A. What happens when there are many agents taking decisions to minimize their own cost that may depend on other's states and strategies? How should an agent choose its strategy to minimize its own cost knowing that other agents will try to do the same? Such questions can be answered in the framework of stochastic differential games, for which here we give an introduction following [29, 30].

Consider a system that consist of N agents, also called players, whose states are represented by a continuous stochastic process $X_t^i \in \mathbb{R}^d$, with $i \in \mathcal{I} := \{1, \dots, N\}$, that continuously take actions α_t^i in a control set $\mathcal{A}^i \subset \mathbb{R}^k$. The dynamics in the time interval $[0,T]$ of the controlled state process X^i follows the stochastic differential equation

$$\begin{aligned} dX_t^i &= \mu^i(t, \mathbf{X}_t, \boldsymbol{\alpha}_t)dt + \sigma^i(t, \mathbf{X}_t, \boldsymbol{\alpha}_t)dW_t^i + \sigma^0(t, \mathbf{X}_t, \boldsymbol{\alpha}_t)dW_t^0 \\ X_0^i &= x_0^i \quad \text{ for } i \in \mathcal{I}, \end{aligned} \tag{4.1}$$

where $\mathbf{W} := (W^0, W^1, \dots W^N)$ are $N+1$ m-dimensional independent Brownian motions, W^i are individual noises and W^0 is common noise for all agents, and $\mathbf{X}_t = [X_t^1, \dots, X_t^N]$ is the joint vector for the N agent dynamics with their corresponding controls $\boldsymbol{\alpha}_t = (\alpha_t^1, \dots, \alpha_t^N)$. The individual drift and volatility (μ^i, σ^i) are deterministic functions $b^i : [0,T] \times \mathbb{R}^{dN} \times \mathcal{A}^N \to \mathbb{R}^d \times \mathbb{R}$ and $\sigma^i : [0,T] \times \mathbb{R}^{dN} \times \mathcal{A}^N \to \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{d \times m}$, which are dependent on the states and controls of every other agent.

Given a set of strategies $(\boldsymbol{\alpha}_t)_{t\in[0,T]}$, we associate a cost/value function to player i of the form

$$v^i(t,x,\boldsymbol{\alpha}_t) := \mathbb{E}\left[\int_t^T f^i(t,\mathbf{X}_t,\boldsymbol{\alpha}_t)dt + g^i(\mathbf{X}_T)\middle| X_t^i = x\right], \tag{4.2}$$

where the running cost $f^i : [0,T]\times\mathbb{R}^{dN}\times\mathcal{A}^N \to \mathbb{R}$ and the terminal cost $g^i : \mathbb{R}^{dN}\to\mathbb{R}$ are deterministic measurable functions.

Each player will try to minimize its repective total cost

$$J_0^i(\boldsymbol{\alpha}_t) := v(0,x_0^i,\boldsymbol{\alpha}_t), \tag{4.3}$$

choosing adequately $(\alpha_t^i)_{t\in[0,T]}$ within the set of admissible strategies $\mathbb{A}^i$. The choice of this set describes measurability and integrability of α_t^i. For example, if we choose $\mathbb{A}^i = \mathbb{A} = \mathbb{H}_T^2(\mathcal{A})$, the space of square integrable $\boldsymbol{W}$-progressively measurable $\mathcal{A}$-valued processes, the α is said to be a *open loop control*, because it only uses the information of the noise that has occurred. If instead we choose $\mathbb{A}^i$ as the set $\mathbb{H}_{\mathbf{X}}^2(\mathcal{A})$, the set of square integrable $\mathbf{X}$-measurable processes, the α_t is a *closed loop markovian control*, as it uses information of the current state $\mathbf{X}_t$.

In a noncooperative game, there is an important notion of optimality termed *Nash equilibrium*, which refers to a set of strategies for all agents such that no one has an incentive to deviate in order to reduce its own cost. Explicitly, we have the following definition

Definition 4.1.1. *A tuple $\boldsymbol{\alpha}^* = (\alpha^{1,*},\dots,\alpha^{N,*}) \in \mathcal{A}^1\times\cdots\times\mathcal{A}^N$ is said to be a Nash equilibrium if for all $i\in\mathcal{I}$ and any $\beta^i\in\mathcal{A}^i$ we have that*

$$J^i(\boldsymbol{\alpha}) \le J^i(\alpha^{1,*},\dots,\alpha^{i-1,*},\beta^i,\alpha^{i+1,*},\dots,\alpha^{N,*})$$

, where on the rigth-hand side the strategy $(\alpha^{1,*},\dots,\alpha^{i-1,*},\beta^i,\alpha^{i+1,*},\dots,\alpha^{N,*})$ is used to solve for $\mathbf{X}$ in the dynamics equation (4.2). Particularly, if we search for a markovian Nash equilibrium, we require the functions α_t^i to be of the form $\alpha_t^i = \alpha^i(t,\mathbf{X}_t)$, for α^i a measurable function.

. Finding such equilibria is an important, yet difficult task to acomplish. In the markovian setting, the equilibrium is related to solving N coupled Hamilton-Jacobi-Bellman equations. Consider the dynamics for the joint vector

$$\begin{aligned} &d\mathbf{X}_t = \mu(t,\mathbf{X}_t,\boldsymbol{\alpha}(t,\mathbf{X}))dt + \Sigma(t,\mathbf{X}_t,\boldsymbol{\alpha}(t,X_t))d\boldsymbol{W}_t \\ &\mathbf{X}_0 = \boldsymbol{x}_0, \end{aligned} \tag{4.4}$$

where we used the vector notation stated before and the joint noise matrix and drift given by

$$\mu = \begin{bmatrix}\mu^1\\ \mu^2\\ \vdots\\ \mu^N\end{bmatrix} \qquad \Sigma = \begin{bmatrix}\sigma^0 & \sigma^1\\ \sigma^0 & \sigma^2\\ \vdots & \vdots\\ \sigma^0 & \sigma^N\end{bmatrix}. \tag{4.5}$$

Then, the optimal value function for agent i satisfies the Hamilton-Jacobi-Bellman equation given by

$$\begin{aligned}&\frac{\partial u^i}{\partial t} + H^i(t, \boldsymbol{x}, \boldsymbol{\alpha}, D_x v^i, D_{xx} v^i) = 0\\&v^i(T, x) = g^i(x),\end{aligned} \tag{4.6}$$

where H^i is the Hamiltonian function

$$\begin{aligned}H^i(t, \boldsymbol{x}, \boldsymbol{\alpha}, p, q) &= \inf_{\alpha^i \in \mathcal{A}^i} \{\mu(t, \boldsymbol{x}, \boldsymbol{\alpha}) \cdot p + f^i(t, \boldsymbol{x}, \boldsymbol{\alpha}) + \frac{1}{2} \operatorname{Tr}(\Sigma\Sigma'(t, \boldsymbol{x}, \boldsymbol{\alpha})q)\}\\&:= \inf_{\alpha^i \in \mathcal{A}^i} \{G^i(t, \boldsymbol{x}, \boldsymbol{\alpha}, p, q)\}.\end{aligned} \tag{4.7}$$

The minimization over α^i should be carried while taking $(\alpha^1, \ldots, \alpha^{i-1}, \alpha^{i+1}, \ldots, \alpha^N)$ given and fixed. Note that this system of equations for $i \in \mathcal{I}$ is implicitly coupled through the joint control $\boldsymbol{\alpha}$ because it depends directly on every v^i from which we derive each α^i.

4.2 Deep Fictitious Play

If we were able to solve the latter coupled system of partial differential equations, we would be able to find the strategy that each player must follow to minimize its cost if every other player behaves as expected. However, this is not an achievable due to the complexity of the system.

Hence, the idea of *fictitius play* was introduced by Brown [31] to decouple this N-player game into N individual decision problems where opponents' strategies are fixed and assumed to follow their past play. If we solve these problems iteratively using the opponents' strategy at $(m-1)'s$ stage to compute the optimal strategy at stage m, we may find the Nash equilibrium of the system, if it exists. In [30], each of the individual decision problems is solved using the HJB equation and the deep learning methods we have studied.

To describe this process mathematically, let's denote by $\mathcal{A}^i \subset \mathbb{R}^k$ the range player i's strategy α^i, and $\mathcal{A} := \bigotimes_{i=1}^N \mathcal{A}^i$ the control space for all agents. Also, denote $\mathcal{A}^{-i} := \bigotimes_{j \neq i} \mathcal{A}^j$ the control space for all agents by i. A similar notation applies for $\mathbb{A}^i$,$\mathbb{A}, \mathbb{A}^{-1}$, the sets of measurable functions in which we search for controls.

Likewise, denote $\boldsymbol{\alpha}^{-1} = (\alpha^1, \ldots, \alpha^{i-1}, \alpha^{i+1}, \ldots, \alpha^N)$ the set of all agents' strategies excluding player i's. A superscript m in $\boldsymbol{\alpha}^m$ denotes the set of strategies of all players at stage m. When both superscripts are present, $\boldsymbol{\alpha}^{-i,m}$ it denotes the set of strategies of all players but i's at stage m. Finally, we use the notation $(\alpha^i, \boldsymbol{\alpha}^{-i,m}) =: (\alpha^{1,m}, \ldots, \alpha^{i-1,m}, \alpha^i, \alpha^{i+1,m}, \ldots, \alpha^{N,m})$

To start the iterations, assume we have an initial guess set of strategies $\boldsymbol{\alpha}^0$. At stage $m+1$, all players observe $\boldsymbol{\alpha}^m$, and the decision problem for player i is

$$\inf_{\alpha^i \in \mathbb{A}^i} J_0^i((\alpha^i, \boldsymbol{\alpha}^{-i,m})) \tag{4.8}$$

, where J_0^i is defined in equation (4.3) and $\mathbf{X}_t$ satisfies (4.4) with $\boldsymbol{\alpha}$ being replaced by $(\alpha^i, \boldsymbol{\alpha}^{-i,m})$. The optimal strategy for player i, if it exists, is denoted by $\alpha^{i,m+1}$. The set of all optimal strategies for $i \in \mathcal{I}$ together form $\boldsymbol{\alpha}^{m+1}$.

Due to the markovianity associated to these problems, each can be translated in a Hamilton-Jacobi-Bellman equation for the value function $V^{i,m+1}(t, \boldsymbol{x})$ of the form

$$\begin{aligned}&\frac{\partial V^{i,m+1}}{\partial t} + H^i(t, \boldsymbol{x}, (\alpha^i, \boldsymbol{\alpha}^{-i,m}), D_x V^{i,m+1}, D_{xx} V^{i,m+1}) = 0\\&V^i(T, \boldsymbol{x}) = g^i(\boldsymbol{x}),\end{aligned} \tag{4.9}$$

where H^i is defined in (4.7).

If this equation has a solution, the optimal strategy for player i at stage $m+1$ is given by the minimizer of the Hamiltonian function

$$\alpha^{i,m+1} = \arg\min_{\alpha^i \in \mathcal{A}^i} G^i(t, \boldsymbol{x}, (\alpha^i, \boldsymbol{\alpha}^{-i,m}), D_x V^{i,m+1}, D_{xx} V^{i,m+1}). \tag{4.10}$$

Solving these N uncoupled equations completes one stage in the fictitious play. If we solve them using deep learning methods, we call this algorithm *deep fictitious play*, see [30, 29].

As today, we have not found general necessary conditions for this algorithm to converge to a Nash equilibrium. Indeed, we may face many problems, for example, equation (4.9) may not have a solution for some not regular enough $\boldsymbol{\alpha}^0$. Even more there is nothing that guarantees that $\boldsymbol{\alpha}^m$ will converge, much less that it does so to a Nash equilibrium. In fact, there are numerous examples where it converges and where it does not, and there is not a clear pattern to evidence when it should work.

4.3 An example

Nevertheless, we will not worry too much about convergences issues and will test it numerically using the PDE solvers we have studied. Consider again the unbounded LQR problem in subsection 3.1.3. Suppose now that we can't control all players at the same time while trying to minimize a global cost function, i.e, there is not a central planner. However, every particle is able to choose its own control that minimizes it's own cost function. As before, the i-th particle state $X^i \in \mathbb{R}^2$ is described by the dynamics

$$\begin{aligned}dX_t^i &= 2\sqrt{\lambda}\alpha_t^i dt + \sqrt{2\nu} dW_t^i\\X_0^i &= x_0^i,\end{aligned} \tag{4.11}$$

where α_t^i is the respective i-th particle control function taking values in $\mathbb{R}^2$. The joint state vector is $\mathbf{X} = (X^1, \ldots, X^N)$ and the joint control is $\boldsymbol{\alpha} = (\alpha^1, \ldots, \alpha^N)$.

Each particle is trying to minimize individually the cost

$$J^i(\alpha_t^i) = \mathbb{E}\left[\int_0^T (|\alpha_t^i|^2 + F^i(t, \mathbf{X}_t))dt + g^i(\mathbf{X}_T)\right], \tag{4.12}$$

irrespective of what every other particle does. Here we denoted by

$$F^i(t, \mathbf{X}_t) = C \sum_{j \neq i} e^{-\frac{|X^i - X^j|^2}{\sigma}} \tag{4.13}$$

and

$$g^i(\mathbf{X}_T) = |X_i|^2. \tag{4.14}$$

Hence, if we try to apply the deep fictitious play algorithm to find a Nash equilibrium for this problem, at stage $m+1$ we would need to solve all the Hamilton-Jacobi-Bellman equation for the player i's value function $V^{i,m+1}$ given all other's player strategies $\boldsymbol{\alpha}^{-i,m}$

$$\frac{\partial V^{i,m+1}}{\partial t} + \inf_{\alpha^i} \left\{ 2\sqrt{\lambda}(\alpha^i, \boldsymbol{\alpha}^{-i,m}) \cdot \nabla V^{i,m+1} + |\alpha^i|^2 + F^i(t, \mathbf{X}_t) + \nu \Delta V^{i,m+1} \right\} = 0, \tag{4.15}$$

subject to the terminal condition

$$V^{i,m+1}(T, \mathbf{X}) = g^i(\mathbf{X}_T). \tag{4.16}$$

The infimum of this equation can be obtained analytically, being achieved at $\alpha^i = -\sqrt{\lambda}\nabla_i V$, where ∇_i denotes the operator $\nabla_i V = (\partial_{x^i} V, \partial_{y_i} V)$. Then, inserting this infimum in (4.15) we obtain

$$\frac{\partial V^{i,m+1}}{\partial t} - \lambda |\nabla_i V^{i,m+1}|^2 + 2\sqrt{\lambda}\boldsymbol{\alpha}^{-i,m} \cdot \nabla_{-i} V^{i,m+1} + F(t, \mathbf{X}) + \nu \Delta V^{i,m+1} = 0, \tag{4.17}$$

where we use the notation $\nabla_{-i} V = (\partial_{x_1} V, \partial_{y_1} V, \ldots, \partial_{x_{i-1}} V, \partial_{y_{i-1}} V, \partial_{x_{i+1}} V, \partial_{y_{i+1}} V, \ldots, \partial_{x_N} V, \partial_{y_N} V)$.

Now, to use the deep learning solvers we have seen, we recast this equation in the form (3.1) by choosing

$$\mu^i(t, \mathbf{X}) = 2\sqrt{\lambda} \begin{bmatrix} \alpha^1(t, \mathbf{X}) \\ \vdots \\ \alpha^{i-1}(t, \mathbf{X}) \\ 0 \\ \alpha^{i+1}(t, \mathbf{X}) \\ \vdots \\ \alpha^N(t, \mathbf{X}) \end{bmatrix} \qquad \sigma^i(t, \mathbf{X}) = \sqrt{2\nu}\mathbb{I}_{2N \times 2N} \tag{4.18}$$

and

$$f^i(t, \mathbf{X}, V(t, \mathbf{X}), \nabla V(t, \mathbf{X})) = -\lambda |\nabla_i V|^2 + F^i(t, \mathbf{X}), \tag{4.19}$$

where we dropped the m superscripts for simplicity.

Note that, in the case we solved before, all f^i and g^i are symmetric when interchanging players' names. Therefore, in each stage of the algorithm we need to calculate only one control that would work for every other player in the next stage of the algorithm.

We apply this method to solve the mentioned above problem. We do not found a good form to measure accuracy of solutions in each stage of the algorithm, then, the only empirical evidence we have that the iterations converged is that the simulated trajectories were similar to those in the centralized problem.

We ran the simulation using the same parameters as in the centralized case, and effectuated 30 iterations of the deep fictitious play. A plot of a trajectory path obtained using the optimized controls is shown in Figure 4.1. Note that it resembles the optimal centralized path we found earlier, however, we do not have a systematic way to perform comparison of both solutions, as we have not carried an error analysis involved in soling the equation in each stage.

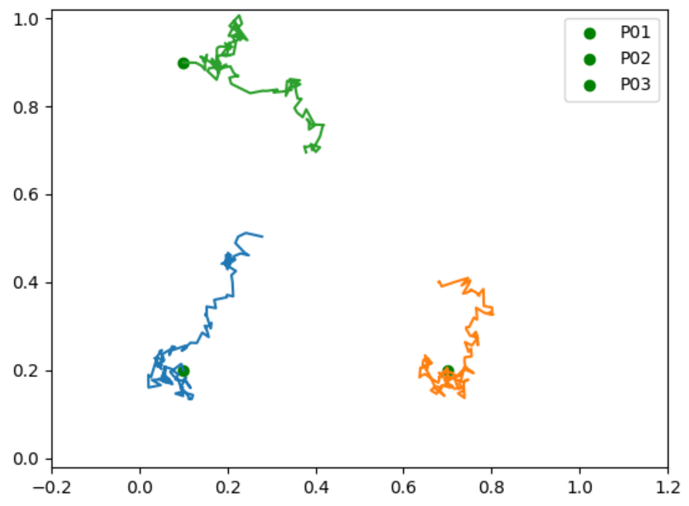


Figure 4.1: Optimal path for exiting the room

Chapter 5

Conclusion

In this work we evidenced the usefulness of various deep learning methods to solve partial differential equations. We applied the to solve optimal control problems using the Hamilton-Jacobi-Bellman equation in dimension 6, which is not accessible through traditional discretization methods. We implemented them in the Python programming language using the Pytorch library for automatic differentiation and neural networks tools.

These methods are very prone to error, as there are many hyperparameters that we need to tune to achieve optimal convergence. When we select them wrong, the process could even diverge. There is not a general formula to make them work in every case.

There is further work we did not accomplish due to time limitations. Our original objective was to solve an N-player game with a bigger N ($N > 100$) to compare with solutions provided by the mean field approach, see for example[32]. However, boundary conditions supposed a hard problem we have not solved yet, but we expect to figure out in a near future. Also, we may solve big centralized optimal control problems to compare them with the equivalent in the limit McKean-Vlasov control problem, see [33]. We may be able to study the degeneration of optimal state when there is not a centralized control, and instead every player is trying to optimize its own function.

We have many ideas to be implemented. Here we list some of them

1. To take into account boundary conditions while being able to access accurately derivatives of the solution to calculate controls, we plan to try reflecting and stopping the process upon contact with a certain part of the boundary, adjusting the Deep BSDE method to include these new conditions. Preliminary results suggest that it may work, but further analysis should be done to justify this procedure theoretically.

2. The DGM was not useful to solve bigger control problems with interaction between the particles. Our hypothesis is that this difficulty is due to the sampling step of interior points that not captures sufficient information for the solution to be accurate. Our idea is to device a method to sample points in important regions of the N-dimensional domain, where the solution is not well approximated by uniform

sampling.

3. In a similar spirit, maybe using a different way to produce sample paths driving them to certain regions of the domain that may be problematical would benefit convergence speed, see for example [34].

Appendix A

Stochastic Control

In this appendix we review, without proofs, the basics of stochastic optimal control leading to the Hamilton-Jacobi-Bellman equation used in this work, and give the linear-quadratic regulator as an example of this theory. We follow [16].

The Hamilton-Jacobi-Bellman Equation

Suppose that we want to control a process $X_t \in \mathbb{R}^n$ that satisfies a stochastic differential equation driven by d-dimensional Brownian motion of the form

$$\begin{aligned} dX_t &= \mu(X_t, \alpha_t)dt + \sigma(X_t, \alpha_t)dW_t \\ X_0 &= x, \end{aligned} \tag{A.1}$$

with a control function α_t taking values in some admissible space A. From now on we assume μ and σ satisfy the standard Lipschitz conditions required for a solution to this equation exist. We want to choose such control so that the total benefit functional given by

$$J(\alpha_t) = \mathbb{E}\left[\int_0^T f(s, X_s, \alpha_s)ds + g(X_T)\right]. \tag{A.2}$$

is maximum over all possible control functions. Here, the function f is called running cost and g is called terminal cost. At any time t, we can choose the controller using only information observed before t, as we are unable to foretell the future due to the system's randomness. Therefore, we require α_t to be $\mathcal{F}_t$-adapted and define the set of feasible controls as $\mathcal{A}([0,T]) = \{\alpha : [0,T] \times \Omega \to A\}$.

A stochastic control problem consists on finding $\hat{\alpha} \in \mathcal{A}([0,T])$ such that

$$J(\hat{\alpha}) = \sup_{\alpha_t \in \mathcal{A}([0,T])} J(\alpha). \tag{A.3}$$

We need the following definitions. For all $(t,x) \in \mathbb{R}^+ \times \mathbb{R}^n$ and $\alpha \in \mathcal{A}([0,T])$, we denote by $X_s^{t,x,\alpha}$ the solution to the SDE

$$\begin{aligned} dX_s^{t,x,\alpha} &= \mu(X_s^{t,x,\alpha}, \alpha_s)ds + \sigma(X_s^{t,x,\alpha}, \alpha_s)dW_s \\ X_t^{t,x,\alpha} &= x. \end{aligned} \tag{A.4}$$

Now, we define the value functional starting at time t and position x as

$$J(t, x, \alpha) = \mathbb{E}\left[\int_t^T f(s, X_s^{t,x,\alpha}, \alpha_s)ds + g(X_T^{t,x,\alpha})\right]. \tag{A.5}$$

and the *value function* $V(t, x)$ as

$$V(t, x) = \sup_{\alpha \in \mathcal{A}[t,T]} J(t, x, \alpha), \tag{A.6}$$

which is the expected optimal reward starting the process at time t and point x.

To solve the stochastic control problem, we follow the approach based on the *dynamic programming principle*, which states informally that

”An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision” Richard Bellman

and can be translated in the following theorem

Theorem A.1 (Stochastic dynamic programming [16]). *For all* $0 \leq t \leq s \leq T$ *and* $x \in \mathbb{R}^n$ *we have that the value function* $V(t, x)$ *satisfies*

$$V(t, x) = \sup_{\alpha \in \mathcal{A}[t,s]} \mathbb{E}\Bigg[\int_t^s f(r, X_r^{t,x,\alpha}, \alpha_r)dr + V(s, V_s^{t,x,\alpha})\Bigg], \tag{A.7}$$

from which a infinitesimal version can be derived, named the Hamilton-Jacobi-Bellman equation

$$\begin{aligned} &\frac{\partial V}{\partial t} + \sup_{a \in A}\{\mathcal{L}^a[V](t, x) + f(t, x, a)\} = 0 \\ &V(T, x) = g(x), \end{aligned} \tag{A.8}$$

where $\mathcal{L}$ *is the infinitesimal generator of the controlled process* X_t *given by*

$$\mathcal{L}^a[V](t, x) = \mu(x, a) \cdot D_xV(t, x) + \frac{1}{2}\operatorname{Tr}(\sigma(x, a)\sigma(x, a)' D_{xx}V(t, x)). \tag{A.9}$$

We can also write the Hamilton-Jacobi-Bellman equation as

$$\begin{aligned} &\frac{\partial V}{\partial t} + H(t, x, D_xV, D_{xx}V) = 0 \\ &V(T, x) = g(x), \end{aligned} \tag{A.10}$$

where the function $H(t, x, p, M)$ is the *hamiltonian* defined as

$$H(t, x, p, M) = \sup_{a \in A}\{\mu(x, a) \cdot p + \frac{1}{2}\operatorname{Tr}(\sigma\sigma'(x, a)M) + f(t, x, a)\}. \tag{A.11}$$

Note that we assume implicitly that the supremums appearing in these equations exists, but this condition is not necessary as pointed in [16].

Solving this equation the Hamilton-Jacobi-Bellman equation for the function $V(t,x)$ can be used to construct optimal controls for the original problem as will be shown below with the linear-quadratic regulator. However, we need a result stating that a solution to such equation is in fact the desired value function

Theorem A.2 (Verification theorem [16])**.** *Let w be a function in $C^{1,2}\left([0,T)\times\mathbb{R}^n\right)\cap C^0\left([0,T]\times\mathbb{R}^n\right)$, and satisfying a quadratic growth condition, i.e. there exists a constant C such that*

$$|w(t,x)|\leq C\left(1+|x|^2\right),\quad \forall(t,x)\in[0,T]\times\mathbb{R}^n$$

(i) Suppose that

$$\begin{aligned} -\frac{\partial w}{\partial t}(t,x)-\sup_{a\in A}\left[\mathcal{L}^a w(t,x)+f(t,x,a)\right]\geq 0,\quad & (t,x)\in[0,T)\times\mathbb{R}^n,\\ w(T,x)\geq g(x),\quad & x\in R^n. \end{aligned}$$

Then $w\geq v$ on $[0,T]\times\mathbb{R}^n$.

(ii) Suppose further that $w(T,x)=g(x)$, and there exists a measurable function $\hat{\alpha}(t,x)$, $(t,x)\in[0,T)\times\mathbb{R}^n$, valued in A such that

$$\begin{aligned} -\frac{\partial w}{\partial t}(t,x)-\sup_{a\in A}\left[\mathcal{L}^a w(t,x)+f(t,x,a)\right] &= -\frac{\partial w}{\partial t}(t,x)-\mathcal{L}^{\hat{\alpha}(t,x)}w(t,x)-f(t,x,\hat{\alpha}(t,x))\\ &=0 \end{aligned}$$

the SDE

$$dX_s=\mu\left(X_s,\hat{\alpha}\left(s,X_s\right)\right)ds+\sigma\left(X_s,\hat{\alpha}\left(s,X_s\right)\right)dW_s$$

admits a unique solution, denoted by $\hat{X}_s^{t,x}$, given an initial condition $X_t=x$, and the process $\left\{\hat{\alpha}\left(s,\hat{X}_s^{t,x}\right)t\leq s\leq T\right\}$ lies in $\mathcal{A}(t,x)$. Then

$$w=v\qquad on\ [0,T]\times\mathbb{R}^n,$$

and $\hat{\alpha}$ is an optimal Markovian control.

Note that this framework also works trivially for minimizing cost instead of maximaxing rewards by replacing sup by inf in all the preceding calculations.

The linear-quadratic regulator (LQR)

To exemplify how to use the stochastic dynamic programming approach to solve optimal control problems, we will solve the linear-quadratic regulator. This problem models a particle whose dynamics is described by the SDE

$$\begin{aligned} &dX_t=2\sqrt{\lambda}\alpha_t dt+\sqrt{2\nu}dW_t\\ &X_0=x, \end{aligned}\tag{A.12}$$

where α_t is the control process and λ, ν are positive constants representing the strength of the control and noise respectively.

We want to minimize the cost functional

$$J(\alpha_t) = \mathbb{E}\left[\int_0^T (|\alpha_t|^2 + F)dt + g(X_T)\right], \tag{A.13}$$

which models the cost of the particle to reach a desired state, whose distance is modeled by $g(x)$, using the least amount of fuel as possible, which is represented by α_t, and also penalizing the required time to reach the desired final state through a positive function $F(t,x)$. For example, if we want the particle to reach the point z_0, we should choose $g(x) = |x - z_0|^2$.

To derive the Hamilton-Jacobi-Bellman satisfied by the value function associated with this problems, note that the generator of the X process is given by

$$\mathcal{L}^a V(t,x) = 2\sqrt{\lambda}a \cdot D_x V + \nu \operatorname{Tr}(D^2_{xx}V) = 2\sqrt{\lambda}a \cdot D_x V + \nu\Delta V, \tag{A.14}$$

hence the Hamiltonian for this problem is

$$H(t,x,D_xV,D^2_{xx}V) = \inf_{a\in A}\{2\sqrt{\lambda}a \cdot D_x V + \nu\Delta V + |a|^2 + F\}, \tag{A.15}$$

where we can calculate this inf analytically by taking the derivative with respect to a and equating to cero, as the function inside is convex in a. Therefore, the minimum is attained at $a = -\sqrt{\lambda}D_xV$ and the Hamiltonian is

$$H(t,x,D_xV,D^2_{xx}V) = -2\lambda|D_xV|^2 + \lambda|D_xV|^2 + \nu\Delta V + F = \nu\Delta V + F - \lambda|D_xV|^2. \tag{A.16}$$

Thus, the associated HJB equation is

$$\frac{\partial V}{\partial t} + \nu\Delta V - \lambda|\nabla V|^2 + F = 0 \tag{A.17}$$

subject to the terminal condition

$$V(T,x) = g(x). \tag{A.18}$$

If we solve this equation for $V(t,x)$, then we can use the verification theorem to obtain the optimal control process as $\hat{\alpha} = -\sqrt{\lambda}D_xV(t,x)$.

Fortunately, we can solve this equation explicitly using the Hopf-Cole transformation, $u(t,x) = e^{-\frac{\lambda}{\nu}V(t,x)}$. For such $u(t,x)$ we have

$$\nabla u(t,x) = -\frac{\lambda}{\nu}e^{-\frac{\lambda}{\nu}V}\nabla V \tag{A.19}$$

and

$$\begin{aligned}\Delta u(t,x) &= \frac{\lambda^2}{\nu^2}e^{-\frac{\lambda}{\nu}V}|\nabla V|^2 - \frac{\lambda}{\nu}e^{-\frac{\lambda}{\nu}V}\Delta V \\ &= -\frac{\lambda}{\nu}e^{-\frac{\lambda}{\nu}V}\left(\Delta V - \frac{\lambda}{\nu}|\nabla V|^2\right) \\ &= -\frac{\lambda}{\nu^2}e^{-\frac{\lambda}{\nu}V}\left(\nu\Delta V - \lambda|\nabla V|^2\right)\end{aligned} \tag{A.20}$$

and so

$$\nu\Delta u(t,x) = -\frac{\lambda}{\nu}e^{-\frac{\lambda}{\nu}V}\left(\nu\Delta V - \lambda|\nabla V|^2\right). \tag{A.21}$$

We also have

$$\frac{\partial u}{\partial t}(t,x) = -\frac{\lambda}{\nu}e^{-\frac{\lambda}{\nu}V}\frac{\partial V}{\partial t}. \tag{A.22}$$

Therefore, adding these derivatives for $u(t,x)$ we obtain

$$\frac{\partial u}{\partial t} + \nu\Delta u = -\frac{\lambda}{\nu}e^{-\frac{\lambda}{\nu}V}\left(\frac{\partial V}{\partial t} + \nu\Delta V - \lambda|\nabla V|^2\right) = \frac{\lambda F}{\nu}u. \tag{A.23}$$

Which means that the HJB equation for u is

$$\frac{\partial u}{\partial t} + \nu\Delta u - \frac{\lambda F}{\nu}u = 0 \tag{A.24}$$

subject to the final condition

$$u(T,x) = e^{-\frac{\lambda}{\nu}g(x)}. \tag{A.25}$$

Hence, using the linear Feynman-Kac formula 2.2.2, we can give a probabilistic explicit representation of the solution as

$$u(t,x) = \mathbb{E}\left[\exp\left(-\frac{\lambda}{\nu}g(x+\sqrt{2\nu}W_{T-t}) - \frac{\lambda}{\nu}\int_t^T F(s,x+\sqrt{2\nu}W_{s-t})ds\right)\right] \tag{A.26}$$

and solving for $V(t,x)$ we obtain

$$V(t,x) = -\frac{\nu}{\lambda}\ln\left(\mathbb{E}\left[\exp\left(-\frac{\lambda}{\nu}g(x+\sqrt{2\nu}W_{T-t}) - \frac{\lambda}{\nu}\int_t^T F(s,x+\sqrt{2\nu}W_{s-t})ds\right)\right]\right). \tag{A.27}$$

Bibliography

[1] Catherine F. Higham and Desmond J. Higham. “Deep Learning: An Introduction for Applied Mathematicians”. In: *SIAM Review* 61.3 (Jan. 2019), pp. 860–891. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/18M1165748. URL: https://epubs.siam.org/doi/10.1137/18M1165748 (visited on 09/26/2022).

[2] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (Feb. 1, 2019), pp. 686–707. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125 (visited on 05/30/2023).

[3] Justin Sirignano and Konstantinos Spiliopoulos. “DGM: A deep learning algorithm for solving partial differential equations”. In: *Journal of Computational Physics* 375 (Dec. 2018), pp. 1339–1364. ISSN: 00219991. DOI: 10.1016/j.jcp.2018.08.029. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999118305527 (visited on 10/10/2022).

[4] Zongyi Li et al. *Fourier Neural Operator for Parametric Partial Differential Equations*. May 16, 2021. arXiv: 2010.08895[cs,math]. URL: http://arxiv.org/abs/2010.08895 (visited on 10/10/2022).

[5] Salvatore Cuomo et al. *Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What’s next*. June 7, 2022. arXiv: 2201.05624[physics]. URL: http://arxiv.org/abs/2201.05624 (visited on 10/05/2022).

[6] Jan Blechschmidt and Oliver G. Ernst. “Three ways to solve partial differential equations with neural networks — A review”. In: *GAMM-Mitteilungen* 44.2 (June 2021). ISSN: 0936-7195, 1522-2608. DOI: 10.1002/gamm.202100006. URL: https://onlinelibrary.wiley.com/doi/10.1002/gamm.202100006 (visited on 09/26/2022).

[7] Jiequn Han, Arnulf Jentzen, and Weinan E. “Solving high-dimensional partial differential equations using deep learning”. In: *Proceedings of the National Academy of Sciences* 115.34 (Aug. 21, 2018), pp. 8505–8510. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1718942115. URL: https://pnas.org/doi/full/10.1073/pnas.1718942115 (visited on 09/26/2022).

[8] Jean-Michel Bismut. “Conjugate convex functions in optimal stochastic control”. In: *Journal of Mathematical Analysis and Applications* 44.2 (Nov. 1, 1973), pp. 384–

404. ISSN: 0022-247X. DOI: 10.1016/0022-247X(73)90066-8. URL: https://www.sciencedirect.com/science/article/pii/0022247X73900668 (visited on 02/21/2023).

[9] E. Pardoux and S. G. Peng. “Adapted solution of a backward stochastic differential equation”. In: *Systems & Control Letters* 14.1 (Jan. 1, 1990), pp. 55–61. ISSN: 0167-6911. DOI: 10.1016/0167-6911(90)90082-6. URL: https://www.sciencedirect.com/science/article/pii/016769119090082 6 (visited on 02/21/2023).

[10] Jianfeng Zhang. *Backward Stochastic Differential Equations*. Vol. 86. Probability Theory and Stochastic Modelling. New York, NY: Springer New York, 2017. ISBN: 978-1-4939-7254-8 978-1-4939-7256-2. DOI: 10.1007/978-1-4939-7256-2. URL: http://link.springer.com/10.1007/978-1-4939-7256-2 (visited on 02/15/2023).

[11] Etienne Pardoux and Aurel R şcanu. *Stochastic Differential Equations, Backward SDEs, Partial Differential Equations*. Vol. 69. Stochastic Modelling and Applied Probability. Cham: Springer International Publishing, 2014. ISBN: 978-3-319-05713-2 978-3-319-05714-9. DOI: 10.1007/978-3-319-05714-9. URL: https://link.springer.com/10.1007/978-3-319-05714-9 (visited on 02/15/2023).

[12] Ricardo Romo Romero. “Maestro en ciencias con especialidad en probabilidad y estadística”. In: ().

[13] Nizar Touzi. *Optimal Stochastic Control, Stochastic Target Problems, and Backward SDE*. Vol. 29. Fields Institute Monographs. New York, NY: Springer New York, 2013. ISBN: 978-1-4614-4285-1 978-1-4614-4286-8. DOI: 10.1007/978-1-4614-4286-8. URL: https://link.springer.com/10.1007/978-1-4614-4286-8 (visited on 02/15/2023).

[14] Xuerong Mao. *Stochastic differential equations and applications*. 2nd ed. OCLC: ocn176925635. Chichester: Horwood Pub, 2008. 422 pp. ISBN: 978-1-904275-34-3.

[15] Jared Chessari et al. *Numerical Methods for Backward Stochastic Differential Equations: A Survey*. Mar. 10, 2022. arXiv: 2101.08936[cs,math]. URL: http://arxiv.org/abs/2101.08936 (visited on 01/24/2023).

[16] Huyên Pham. *Continuous-time Stochastic Control and Optimization with Financial Applications*. Vol. 61. Stochastic Modelling and Applied Probability. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. ISBN: 978-3-540-89499-5 978-3-540-89500-8. DOI: 10.1007/978-3-540-89500-8. URL: https://link.springer.com/10.1007/978-3-540-89500-8 (visited on 02/28/2023).

[17] N. El Karoui, S. Peng, and M. C. Quenez. “Backward Stochastic Differential Equations in Finance”. In: *Mathematical Finance* 7.1 (1997). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9965.00022, pp. 1–71. ISSN: 1467-9965. DOI: 10.1111/1467-9965.00022. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-9965.00022 (visited on 02/27/2023).

[18] René Carmona. *Lectures on BSDEs, Stochastic Control, and Stochastic Differential Games with Financial Applications*. Philadelphia, PA: Society for Industrial and Applied Mathematics, Feb. 18, 2016. ISBN: 978-1-61197-423-2 978-1-61197-424-9. DOI: 10.1137/1.9781611974249. URL: http://epubs.siam.org/doi/book/10.1137/1.9781611974249 (visited on 02/21/2023).

[19] J. Yong and Xun Yu Zhou. *Stochastic controls: Hamiltonian systems and HJB equations*. Applications of mathematics 43. New York: Springer, 1999. 438 pp. ISBN: 978-0-387-98723-1.

[20] Jared Chessari et al. “Numerical methods for backward stochastic differential equations: A survey”. In: ().

[21] Weinan E, Jiequn Han, and Arnulf Jentzen. “Algorithms for solving high dimensional PDEs: from nonlinear Monte Carlo to machine learning”. In: *Nonlinearity* 35.1 (Jan. 6, 2022), pp. 278–310. ISSN: 0951-7715, 1361-6544. DOI: 10.1088/1361-6544/ac337f. URL: https://iopscience.iop.org/article/10.1088/1361-6544/ac337f (visited on 02/14/2023).

[22] 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: *Journal of Nonlinear Science* 29.4 (Aug. 2019), pp. 1563–1619. ISSN: 0938-8974, 1432-1467. DOI: 10.1007/s00332-018-9525-3. arXiv: 1709.05963[cs,math,stat]. URL: http://arxiv.org/abs/1709.05963 (visited on 03/05/2023).

[23] 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: *Communications in Mathematics and Statistics* 5.4 (Dec. 1, 2017), pp. 349–380. ISSN: 2194-671X. DOI: 10.1007/s40304-017-0117-6. URL: https://doi.org/10.1007/s40304-017-0117-6 (visited on 04/10/2023).

[24] Quentin Chan-Wai-Nam, Joseph Mikael, and Xavier Warin. *Machine Learning for semi linear PDEs*. Dec. 10, 2018. arXiv: 1809.07609[cs,math,stat]. URL: http://arxiv.org/abs/1809.07609 (visited on 03/26/2023).

[25] Jiequn Han and Jihao Long. “Convergence of the Deep BSDE Method for Coupled FBSDEs”. In: *Probability, Uncertainty and Quantitative Risk* 5.1 (Dec. 2020), p. 5. ISSN: 2367-0126. DOI: 10.1186/s41546-020-00047-w. arXiv: 1811.01165[cs,math]. URL: http://arxiv.org/abs/1811.01165 (visited on 04/10/2023).

[26] Côme Huré, Huyên Pham, and Xavier Warin. “Deep backward schemes for high-dimensional nonlinear PDEs”. In: *Mathematics of Computation* 89.324 (Jan. 31, 2020), pp. 1547–1579. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/mcom/3514. URL: https://www.ams.org/mcom/2020-89-324/S0025-5718-2020-03514-5/ (visited on 02/28/2023).

[27] Maziar Raissi. *Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations*. Apr. 19, 2018. arXiv: 1804.07010[cs,math,stat]. URL: http://arxiv.org/abs/1804.07010 (visited on 03/22/2023).

[28] Nikolas Nüsken and Lorenz Richter. *Interpolating between BSDEs and PINNs: deep learning for elliptic and parabolic boundary value problems*. Jan. 29, 2023. arXiv: 2112.03749[cs,math,stat]. URL: http://arxiv.org/abs/2112.03749 (visited on 03/26/2023).

[29] Ruimeng Hu and Mathieu Laurière. “Recent Developments in Machine Learning Methods for Stochastic Control and Games”. In: ().

[30] Jiequn Han and Ruimeng Hu. “Deep Fictitious Play for Finding Markovian Nash Equilibrium in Multi-Agent Games”. In: *Proceedings of The First Mathematical and Scientific Machine Learning Conference*. Mathematical and Scientific Machine Learning. ISSN: 2640-3498. PMLR, Aug. 16, 2020, pp. 221–245. URL: https://proceedings.mlr.press/v107/han20a.html (visited on 02/28/2023).

[31] George W. Brown. *Some Notes on Computation of Games Solutions.* RAND Corporation, Jan. 1, 1949. URL: https://www.rand.org/pubs/research_memoranda/RM125.html (visited on 05/29/2023).

[32] Yves Achdou and Mathieu Laurière. "Mean Field Games and Applications: Numerical Aspects". In: Yves Achdou et al. *Mean Field Games.* Ed. by Pierre Cardaliaguet and Alessio Porretta. Vol. 2281. Series Title: Lecture Notes in Mathematics. Cham: Springer International Publishing, 2020, pp. 249–307. ISBN: 978-3-030-59836-5 978-3-030-59837-2. DOI: 10.1007/978-3-030-59837-2_4. URL: http://link.springer.com/10.1007/978-3-030-59837-2_4 (visited on 02/17/2023).

[33] René Carmona, François Delarue, and Aimé Lachapelle. "Control of McKean–Vlasov dynamics versus mean field games". In: *Mathematics and Financial Economics* 7.2 (Mar. 2013), pp. 131–166. ISSN: 1862-9679, 1862-9660. DOI: 10.1007/s11579-012-0089-y. URL: http://link.springer.com/10.1007/s11579-012-0089-y (visited on 05/30/2023).

[34] Nikolas Nüsken and Lorenz Richter. *Solving high-dimensional Hamilton-Jacobi-Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space.* Jan. 29, 2023. arXiv: 2005.05409[cs,math,stat]. URL: http://arxiv.org/abs/2005.05409 (visited on 03/27/2023).