Master's degree MASEF, University Paris Dauphine–PSL

Research Memoir

Mean Field Optimal Execution and Deep Learning Methods

Ryan Timeus

Supervisor:

Julien Claisse

October 15, 2025

Contents

Introduction

In modern financial markets, high-frequency trading (HFT) has transformed the way liquidity is provided and large transactions are executed. A central issue in this context is the optimal execution problem, where a trader seeks to execute a large order over a finite horizon while minimizing trading costs and mitigating the adverse impact of her own activity on market prices. The challenge arises from the interplay between two opposing forces, trading too quickly increases transaction costs through price impact, whereas trading too slowly exposes the trader to price volatility and inventory risk. Over the past two decades, this problem has been extensively studied within the framework of stochastic control. In particular, when multiple strategic traders interact, the problem naturally falls within the scope of Mean Field Games (MFGs), a theory introduced by (Lasry and Lions, 2007). In such a framework, the market is modeled as a large population of agents, each optimizing their own execution strategy while collectively influencing the dynamics of asset prices. The mean field formulation overcomes the curse of dimensionality inherent in the $N-$player stochastic game by replacing the empirical distribution of controls with its limiting law, thereby providing a tractable approximation of a Nash equilibrium.

This memoir is structured around two main objectives. First, we provide a rigorous analysis of optimal execution with price impact in the MFG framework introduced by (Carmona and Lacker, 2015). We review the literature in order to deepen our understanding of the model, emphasize issues of model interpretability, and present solutions to the optimal execution problem via both the partial differential equation (PDE) and forward-backward stochastic differential equation (FBSDE) approaches. Second, we explore modern deep learning methods for solving mean field problems, following the approaches presented in (Carmona and Laurière, 2021c), motivated by the growing interest in machine learning within quantitative finance and enabled by the computational power of modern GPUs. In particular, we place a strong emphasis on reinforcement learning (RL) methods, which are especially well-suited to address the complex challenges of quantitative finance, given that financial markets are inherently dynamic and continuously evolving. In this context, we try to extend a recent infinite horizon RL algorithm proposed in (Andrea Angiuli et al., 2025) within our finite horizon setting. For simplicity and consistency, we mainly focus our attention on the MFG formulation over the Mean Field Control (MFC) formulation. Nevertheless, all the results presented here can, with some additional work, be adapted to the MFC setting in a relatively straightforward manner. Finally, we make our best effort to obtain meaningful numerical results with these different methods and to compare them with the true solution derived in the first chapter.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Julien Claisse, for his constant guidance and support throughout this work. His valuable insights, constructive feedback, and the many references he provided were essential to the completion of this memoir.

Chapter 1

Optimal Execution and Price Impact in a Mean Field Game

1.1 A Mean Field Optimal Execution problem

In high frequency trading, the optimal execution problem refers to the strategic challenge faced by traders who must execute large volumes of trades rapidly, while minimizing transaction costs and mitigating adverse price movements caused by their own trading activity. Traders face a fundamental trade-off: executing too quickly amplifies market impact, while executing too slowly increases exposure to market volatility and inventory risk. Accordingly, optimal execution is typically formulated as a stochastic control problem, in which the objective is to minimize expected total execution costs by balancing immediate market impact with inventory related risks. Concretely, the investor must choose the speed at which she submits market orders to liquidate ξ shares over a fixed trading horizon T. Here we focus on the liquidation problem, the acquisition problem is treated in a similar way. In the MFG framework adopted here, this optimization is studied in a setting with numerous interacting traders, each influencing the market collectively while minimizing their individual execution costs.

1.1.1 The $N-$player stochastic game

We formulate the optimal liquidation problem as an $N-$player stochastic differential game. To this end, we consider a trading horizon $T > 0$ and a filtered probability space $(\Omega, \mathscr{F}, \mathbb{F} = (\mathscr{F}_t)_{t\in[0,T]}, \mathbb{P})$ supporting $N+1$ independent standard Brownian motions $W^0, \dots, W^N$ and N independent initial conditions $\xi^1, \dots, \xi^N$ belonging to $L^2(\Omega, \mathscr{F}_0, \mathbb{P})$. At each time $t \in [0, T]$, player $i \in \{1, \dots, N\}$ trades at rate α_t^i and incurs an instantaneous cost $c(\alpha_t^i)$, where $c : \mathbb{R} \to \mathbb{R}_+$ is convex and satisfies $c(0) = 0$. This function models the *temporary price impact* of trading at speed α_t^i, representing the premium that liquidity takers must pay to liquidity providers. As explained in (Carmona and Webster, 2013, Section 6), the temporary price impact function can be interpreted as the Legendre transform of the order book shape. Throughout the paper, following (Almgren and Chriss, 2000), we assume a flat order book. While this assumption is not realistic, it is standard in the literature and leads to a quadratic transaction cost: $c(\alpha) = \frac{c_\alpha}{2}\alpha^2$, with $c_\alpha > 0$.

All players trade the same asset, whose mid-price S_t is exogenous to individual orders. For each player $i \in \{1, \dots, N\}$, we denote by X_t^i its inventory at time $t \in [0, T]$, which follows the controlled dynamics

$$dX_t^i = \alpha_t^i dt + \sigma dW_t^i, \quad X_0^i = \xi^i, \quad \sigma > 0. \tag{1.1}$$

If a trader submits a market order of size α_t^i when the mid-price is S_t, the transaction cost him $\alpha_t^i S_t + \frac{c_\alpha}{2}(\alpha_t^i)^2$.

Consequently, its cash position K_t^i follows naturally the dynamics

$$dK_t^i = -\alpha_t^i \left(S_t + \tfrac{c_\alpha}{2}\alpha_t^i \right) dt. \tag{1.2}$$

The process α^i controls the trading rate of player i, while the noise term captures the random demand that a broker may receive from clients. In much of the related literature, the volatility parameter σ is set to zero. However, the choice $\sigma > 0$ is supported by (Carmona and Webster, 2013) and (Carmona and Leal, 2021), who performed statistical tests on high frequency market data and concluded that the inventory process should include a Brownian component.

In line with (Carmona and Lacker, 2015), we choose a linear permanent impact, so that the mid-price evolves as

$$dS_t = \gamma \left(\int_{\mathbb{R}} a\bar{\nu}_t^N(da) \right) dt + \sigma_0 dW_t^0, \quad \gamma > 0, \ \sigma_0 > 0, \tag{1.3}$$

where $\bar{\nu}_t^N := \frac{1}{N}\sum_{i=1}^N \delta_{\alpha_t^i}$ is the empirical distribution of trading rates. The drift term encodes the *permanent price impact*, meaning that the mid-price is affected by the average trading rate of all players, so that sustained buying (resp. selling) pressure drives the price upward (resp. downward) proportionally to γ. The volatility term σ_0 captures exogenous fluctuations of the asset price that are independent of the players' trading activity. Finally, the wealth of player i is given by the marked-to-market value of their position,

$$V_t^i = K_t^i + X_t^i S_t, \tag{1.4}$$

from which one can derive its dynamics, recalling that $\langle W^i, W^0 \rangle \equiv 0$

$$\begin{aligned} dV_t^i &= dK_t^i + S_t dX_t^i + X_t^i dS_t, \\ &= \left(\gamma X_t^i \int_{\mathbb{R}} a\bar{\nu}_t^N(da) - \tfrac{c_\alpha}{2}(\alpha_t^i)^2 \right) dt + \sigma_0 X_t^i dW_t^0 + \sigma S_t dW_t^i. \end{aligned} \tag{1.5}$$

We denote by $\mathscr{A}$ the set of admissible controls, defined as the set of $\mathbb{F}$-progressively measurable processes α such that $\mathbb{E}[\int_0^T |\alpha_t|^2 dt]$ is finite. Assuming that $S_0 \in L^2(\Omega, \mathscr{F}_0, \mathbb{P})$, we recall that for every $\alpha \in \mathscr{A}$, one has

$$\mathbb{E}\left[\sup_{0\le t\le T} |X_t^i|^2 \right] < \infty, \quad \mathbb{E}\left[\sup_{0\le t\le T} |S_t|^2 \right] < \infty, \tag{1.6}$$

which ensures that both local martingale terms in (1.5) are true martingales. Finally, each player i specifies a cost functional as a performance criterion, which serves as the basis for finding a Nash equilibrium. The ith trader seeks to solve

$$V^i(\alpha^{-i}) = \inf_{\alpha^i \in \mathscr{A}} J^{i,N}(\alpha^1, \dots, \alpha^N) = \inf_{\alpha^i \in \mathscr{A}} \mathbb{E}\left[\int_0^T c_X(X_t^i) dt + g(X_T^i) - V_T^i \right], \tag{1.7}$$

$$= \inf_{\alpha^i \in \mathscr{A}} \mathbb{E}\left[\int_0^T f(X_t^i, \alpha_t^i, \bar{\nu}_t^N) dt + g(X_T^i) - V_0^i \right], \tag{1.8}$$

where (1.8) is obtained from (1.5). Following (Almgren and Chriss, 2000), both c_X and g are chosen to be quadratic

$$c_X(x) = \frac{c_X}{2}x^2, \ g(x) = \frac{c_g}{2}x^2, \quad c_X > 0, \ c_g > 0, \qquad f(x, \alpha, \nu) = \frac{c_X}{2}x^2 + \frac{c_\alpha}{2}\alpha^2 - \gamma x \int a\nu(da).$$

A symmetric strategy $(\tilde{\alpha}^1, \dots, \tilde{\alpha}^N)$ forms a Nash equilibrium if no player can reduce her cost by unilaterally deviating from her strategy, namely

$$J^{i,N}(\tilde{\alpha}^1,\dots,\tilde{\alpha}^N) \leq J^{i,N}((\tilde{\alpha}^j)_{j\neq i},\alpha), \qquad \forall \alpha \in \mathscr{A}, \quad \forall i \in \{1,\dots,N\}.$$

The quadratic specification ensures tractability in the Linear–Quadratic (LQ) mean field game framework. Importantly, the quadratic structure is not only a technical convenience leading to explicit solutions, but also admits a clear economic interpretation. In the spirit of (Cardaliaguet and Lehalle, 2017), we can equivalently rewrite the cost functional of player i as

$$J^{i,N}(\alpha^1,\dots,\alpha^N) = \mathbb{E}\left[\int_0^T \frac{c_X}{2}(X_t^i)^2 dt - X_T^i\left(S_T - \frac{c_g}{2}X_T^i\right) - K_T^i\right].$$

This objective consists of three components, each with a natural financial interpretation. The first term, the running penalty on inventory, captures the risk of carrying open positions over time. Large inventories expose the trader to adverse price moves, and the quadratic penalty discourages holding such positions. In particular, a higher inventory penalty parameter c_X leads to faster liquidation at the beginning of the trading horizon, so c_X may be viewed as an urgency parameter. Moreover, the inclusion of this running penalty is also motivated by settings with model uncertainty, where the agent is ambiguity averse, a larger value of c_X reflects lower confidence in the trend of the mid-price. We refer to (Cartea et al., 2013; Cartea and Jaimungal, 2016) for a detailed discussion of this interpretation. The second term penalizes residual inventory left at maturity, reflecting the cost of liquidating a block position at time T. The parameter c_g quantifies this penalty, representing the price concession required to offload the remaining inventory. More generally, as suggested by (Carmona and Leal, 2021), this term can be extended to $c_g(x) = \frac{c_g}{2}(x - x_T)^2$, where x_T is a target final inventory. In many practical execution problems, the agent aims to reach a prescribed terminal inventory, here we take $x_T = 0$. The third term corresponds to the player's final cash holdings K_T^i.

As emphasized in (Cardaliaguet and Lehalle, 2017), this cost specification is chosen deliberately, as the literature on optimal execution (Cartea and Jaimungal, 2016; Cartea et al., 2015) shows that, with suitable modifications, it can approximate many of the execution cost functions used by brokers in practice. In its current form, it corresponds to an *Implementation Shortfall* (IS) algorithm, but it can be adapted to replicate other benchmarks such as *Volume-Weighted Average Price* (VWAP). Such extensions would alter the analysis substantially and lie beyond the scope of this work.

Finally, since we assume linear utility for V_T^i, the common noise W^0 cancels out of the optimization problem. This simplification eliminates aggregate randomness from the players' objectives and greatly reduces the complexity of the analysis.

1.1.2 The mean field game formulation

We have described a model with N interacting traders. Solving this multi-agent problem requires analyzing a system of N coupled Hamilton–Jacobi–Bellman (HJB) equations, which quickly becomes intractable as N grows, due to the well known *curse of dimensionality.*

This difficulty motivates the use of the MFG approach. In the mean field limit, we consider the problem of a single representative player facing an infinite population of opponents. By propagation of chaos, players become independent as $N \to \infty$, and the representative player is influenced only through the law of the population's controls rather than the empirical distribution. The dynamics of the representative player's inventory and cash remain the same as in the N-player model, except that the index i is dropped. Similarly, in the mid-price dynamics, the empirical distribution of controls is replaced by the law of the representative player's control, which is now denoted by $\nu = (\nu_t)_{t\in[0,T]}$. The representative player now seeks to find a *mean field equilibrium*, that is, a pair $(\alpha^*, \nu^*) \in \mathscr{A} \times \mathcal{C}([0,T], \mathcal{P}_2(\mathbb{R}))$ satisfying

1. α^* is an optimal control for the stochastic control problem parameterized by the flow ν, when $\nu = \nu^*$

$$\inf_{\alpha \in \mathscr{A}} J_\nu^{\mathrm{MFG}}(\alpha) = \inf_{\alpha \in \mathscr{A}} \mathbb{E}\left[\int_0^T f(X_t, \alpha_t, \nu_t)\, dt + g(X_T) - V_0\right]. \tag{1.9}$$

2. At each time $t \in [0,T]$, the law of the optimal control coincides with the prescribed flow: $\mathcal{L}(\alpha_t^*) = \nu_t^*$ for every $t \in [0,T]$.

We emphasize that in the optimization problem (1.9), the flow ν is treated as fixed. The resulting optimal control, which we denote by $\alpha^{\nu,*}$, therefore depends implicitly on ν. A mean field equilibrium is then obtained by finding a flow ν such that the consistency condition $\mathcal{L}(\alpha_t^{\nu,*}) = \nu_t$ holds for every $t \in [0,T]$. In other words, the problem reduces to finding a fixed point.

1.2 Solution to the Optimal Execution problem

There are two main approaches to solving mean field game problems. The first, often referred to as the analytic approach, was pioneered by (Lasry and Lions, 2007). It relies on the dynamic programming principle and leads to a system of coupled PDEs: a backward HJB equation describing the optimal control of the representative agent, and a forward Kolmogorov–Fokker–Planck (KFP) equation governing the evolution of the distribution of states. These two PDEs are coupled through the flow of measures, and their joint solution characterizes the mean field equilibrium. The second approach, often called the probabilistic approach, was later developed systematically in the monographs (Carmona and Delarue, 2018a,b). Instead of PDEs, it relies on the stochastic maximum principle and characterizes the equilibrium by a system of McKean–Vlasov forward–backward stochastic differential equations (MKV FBSDEs). The forward equation describes the state dynamics, while the backward component corresponds to the adjoint process associated with the control problem.

1.2.1 The PDE approach

Using the formulation given in (1.9), we define the value function $V_\nu : [0,T] \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

$$V_\nu(t,x,s,q) = \inf_{\alpha \in \mathscr{A}} \mathbb{E}\left[\int_t^T \left(\frac{c_X}{2} X_u^2 + \frac{c_\alpha}{2}\alpha_u^2 - \gamma X_u \int_{\mathbb{R}} a \nu_u(da)\right) du + \frac{c_g}{2} X_T^2 - V_t \,\middle|\, X_t = x, S_t = s, K_t = q\right].$$

Recalling that $V_t = K_t + X_t S_t$, the value function can be rewritten as

$$V_\nu(t,x,s,q) = \inf_{\alpha \in \mathscr{A}} \mathbb{E}\left[\int_t^T \left(\frac{c_X}{2} X_u^2 + \frac{c_\alpha}{2}\alpha_u^2 - \gamma X_u \int_{\mathbb{R}} a \nu_u(da)\right) du + \frac{c_g}{2} X_T^2 \,\middle|\, X_t = x\right] - (q + xs). \tag{1.10}$$

We observe that the optimization problem depends only on the trader's initial inventory and is independent of both the initial cash position and the initial asset price. This allows us to write

$$V_\nu(t,x,s,q) = v_\nu(t,x) - (q + xs).$$

The function v_ν satisfies, under suitable assumtions, the HJB equation

$$\begin{cases} \partial_t v_\nu(t,x) = \gamma x \displaystyle\int_{\mathbb{R}} a' \nu_s(da') - \frac{c_X}{2}x^2 - \frac{\sigma^2}{2}\partial_{xx}^2 v_\nu(t,x) - \inf_{a \in \mathbb{R}} \left\{\frac{c_\alpha}{2}a^2 + a\partial_x v_\nu(t,x)\right\}, \\ v_\nu(T,x) = \dfrac{c_g}{2}x^2. \end{cases} \tag{1.11}$$

The optimal feedback control associated with (1.11) is given by $\alpha^*(t,x) = -\frac{\partial_x v_\nu(t,x)}{c_\alpha}$. In parallel, for every $t \in [0,T]$, let $m(t,\cdot)$ denote the density of the law of the inventory process X_t. It satisfies, under suitable assumtions, the KFP equation

$$\begin{cases} \partial_t m(t,x) = \dfrac{\sigma^2}{2}\partial^2_{xx} m(t,x) + \dfrac{1}{c_\alpha}\partial_x\left(m\partial_x v_\nu\right)(t,x), \\ \quad m(0,x) = m_0(x). \end{cases} \tag{1.12}$$

where m_0 denotes the density of the initial distribution of ξ. Moreover, for every $t \in [0,T]$, we should have $\nu_t = m_t \circ \alpha^*(t,\cdot)^{-1}$, that is, the image measure of m_t by the optimal feedback control $\alpha^*(t,\cdot)$ is ν_t. Putting all components together, a mean field equilibrium corresponds to a solution of the coupled system

$$\begin{cases} \partial_t v_\nu(t,x) = \gamma x \int_{\mathbb{R}} a'\nu_s(da') - \frac{c_X}{2}x^2 - \frac{\sigma^2}{2}\partial^2_{xx} v_\nu(t,x) + \frac{\partial_x v_\nu(t,x)^2}{2c_\alpha}, \\ \partial_t m(t,x) = \frac{\sigma^2}{2}\partial^2_{xx} m(t,x) + \frac{1}{c_\alpha}\partial_x\left(m\partial_x v_\nu\right)(t,x), \\ v_\nu(T,x) = \frac{c_g}{2}x^2,\ m(0,x) = m_0(x), \\ \nu_t = m_t \circ \alpha^*(t,\cdot)^{-1}. \end{cases} \tag{1.13}$$

This system is closely related to the one presented in (Cardaliaguet and Lehalle, 2017). The novelty in our setting lies in the fact that we allow $\sigma > 0$, which introduces second–order derivative terms in both the HJB and the KFP equations. Although this modification alters the system to be solved, the resulting solution remains very similar to the one obtained in (Cardaliaguet and Lehalle, 2017).

To solve for v_ν, we exploit the structure of the PDE and make the ansatz $v_\nu(t,x) = \frac{A(t)}{2}x^2 + B(t)x + C(t)$, where A, B, and C are continuously differentiable functions to be determined. We introduce the mean inventory $E(t) := \mathbb{E}\left[X_t\right] = \int_{\mathbb{R}} x m(t,x)dx$. Assuming that the density $m(t,\cdot)$ decays sufficiently fast at infinity (for instance, Gaussian tails) for every $t \in [0,T]$, one may use (1.12) together with integration by parts to compute

$$\begin{aligned} E'(t) &= -\int_{\mathbb{R}} \frac{\partial_x v_\nu(t,x)}{c_\alpha} m(t,x)dx = \int_{\mathbb{R}} \alpha^*(t,x)m(t,x)dx = \int_{\mathbb{R}} a\nu_t(da) \\ &= -\int_{\mathbb{R}} \frac{A(t)x + B(t)}{c_\alpha} m(t,x)dx = -\frac{1}{c_\alpha}\left(A(t)E(t) + B(t)\right). \end{aligned} \tag{1.14}$$

Combining (1.11) with (1.14) yields the following system of ODEs

$$\begin{cases} c_\alpha A'(t) = A(t)^2 - c_\alpha c_X \\ c_\alpha B'(t) = B(t)(A(t) - \gamma) - \gamma A(t)E(t) \\ 2c_\alpha C'(t) = B(t)^2 - c_\alpha\sigma^2 A(t) \\ c_\alpha E''(t) = c_X E(t) - \gamma E'(t) \end{cases} \tag{1.15}$$

With boundary conditions given by $A(T) = c_g$, $B(T) = C(T) = 0$, $E(0) = \mathbb{E}[\xi]$ and $c_g E(T) + c_\alpha E'(T) = 0$. The first equation is a Riccati equation, whose explicit solution is

$$A(t) = -\sqrt{c_X c_\alpha}\frac{(\sqrt{c_X c_\alpha} - c_g) - (\sqrt{c_X c_\alpha} + c_g)\exp(2\sqrt{c_X/c_\alpha}(T-t))}{(\sqrt{c_X c_\alpha} - c_g) + (\sqrt{c_X c_\alpha} + c_g)\exp(2\sqrt{c_X/c_\alpha}(T-t))}.$$

The last equation yields an explicit expression for the mean inventory

$$E(t) = E_0\frac{(c_g + c_\alpha r_-)e^{-r_+(T-t)} - (c_g + c_\alpha r_+)e^{-r_-(T-t)}}{(c_g + c_\alpha r_-)e^{-r_+T} - (c_g + c_\alpha r_+)e^{-r_-T}}, \quad r_\pm = \frac{-\gamma \pm \sqrt{\gamma^2 + 4c_X c_\alpha}}{2c_\alpha}.$$

Using again (1.14), one can recover the feedback coefficient $B(t)$ via $B(t) = -(A(t)E(t) + c_\alpha E'(t))$. The inventory process X then evolves as an Ornstein–Uhlenbeck like diffusion

$$dX_t = -\frac{1}{c_\alpha}\big(A(t)X_t + B(t)\big)dt + \sigma dW_t,$$

whose unique strong solution is

$$X_t = \Phi(t,0)X_0 - \frac{1}{c_\alpha}\int_0^t \Phi(t,s)B(s)ds + \sigma\int_0^t \Phi(t,s)dW_s, \quad \Phi(t,s) = e^{-\frac{1}{c_\alpha}\int_s^t A(u)du}.$$

Therefore, if the initial inventory is Gaussian, $X_0 \sim \mathcal{N}(\mu_X(0), \Sigma_X(0))$, then X_t remains Gaussian for all $t \in [0,T]$, with $X_t \sim \mathcal{N}(\mu_X(t), \Sigma_X(t))$. Consequently, the optimal control is also Gaussian, $\alpha^*(t, X_t) \sim \mathcal{N}(\mu_\alpha(t), \Sigma_\alpha(t))$, with

$$\mu_X(t) = \Phi(t,0)\mu_X(0) - \frac{1}{c_\alpha}\int_0^t \Phi(t,s)B(s)ds, \quad \mu_\alpha(t) = -\frac{A(t)\mu_X(t) + B(t)}{c_\alpha},$$
$$\Sigma_X(t) = \Phi(t,0)^2\Sigma_X(0) + \sigma^2\int_0^t \Phi(t,s)^2 ds, \qquad \Sigma_\alpha(t) = \frac{A(t)^2}{c_\alpha^2}\Sigma_X(t).$$

1.2.2 The FBSDE approach via the stochastic maximum principle

We define the reduced Hamiltonian $H : \mathbb{R} \times \mathbb{R} \times \mathcal{P}_2(\mathbb{R}) \times \mathbb{R} \to \mathbb{R}$ by

$$H(x, \alpha, \nu, y) = \frac{c_X}{2}x^2 + \frac{c_\alpha}{2}\alpha^2 - \gamma x\int_{\mathbb{R}} a\nu(da) + y\alpha.$$

As explained in (Carmona and Delarue, 2018a), a mean field game equilibrium can be characterized as the solution of a MKV FBSDE, derived from the stochastic maximum principle adapted to the MFG setting. A rigorous proof of the stochastic maximum principle for MFC (necessary and sufficient conditions) is provided in (Acciaio et al., 2018). The extension to MFGs can be easily derived from this framework. See also (Carmona and Delarue, 2018a, Chapter 4), where the MKV FBSDE formulation of MFGs is developed in detail, together with proofs of existence and uniqueness under convexity and regularity conditions.

Our model satisfies the standing assumptions required for the stochastic maximum principle to hold (convexity of the Hamiltonian in the control, differentiability of the coefficients, and square integrability), and thus the following derivation is valid. Formally, the stochastic maximum principle yields the MKV FBSDE

$$\begin{cases} dX_t = \partial_y H\big(X_t, \alpha^*(t, X_t, Y_t), \mathcal{L}(\alpha^*(t, X_t, Y_t)), Y_t\big)dt + \sigma dW_t, & X_0 = \xi, \\ dY_t = -\partial_x H\big(X_t, \alpha^*(t, X_t, Y_t), \mathcal{L}(\alpha^*(t, X_t, Y_t)), Y_t\big)dt + Z_t dW_t, & Y_T = \partial_x g(X_T), \end{cases} \tag{1.16}$$

where $\alpha^*(t,x,y)$ is the control that minimizes the Hamiltonian, which read

$$\alpha^*(t,x,y) = -\frac{y}{c_\alpha}. \tag{1.17}$$

Substituting into (1.16), and recalling that the consistency condition enforces $\int_{\mathbb{R}} a\nu_t(da) = -\frac{1}{c_\alpha}\mathbb{E}[Y_t]$, we obtain the explicit MKV FBSDE

$$\begin{cases} dX_t = -\dfrac{1}{c_\alpha}Y_t dt + \sigma dW_t, & X_0 = \xi, \\ dY_t = -\left(c_X X_t - \dfrac{\gamma}{c_\alpha}\mathbb{E}\left[Y_t\right]\right)dt + Z_t dW_t, & Y_T = c_g X_T. \end{cases} \tag{1.18}$$

We do not provide the full solution here, as it is developed in detail in (Carmona and Delarue, 2018a, Section 4.7.1). Instead, we emphasize that the affine ansatz $Y_t = \eta_t X_t + \chi_t$ used there follows naturally from the structure of the problem. Indeed, from the stochastic maximum principle one has

$$Y_t = \partial_x v_{\mathcal{L}(\alpha^*)}(t, X_t).$$

In the PDE approach developed above, we showed that $v_{\mathcal{L}(\alpha^*)}(t, x)$ is quadratic in x, which in turn justifies the affine representation of Y_t. Moreover, by comparing the two formulations, one can immediately identify $\eta_t = A(t)$ and with more work that $\chi_t = B(t)$.

Chapter 2

Deep Learning methods for Mean Field problems

In this chapter, we review several deep learning methods designed to solve the mean field problem introduced in the previous chapter. The first three methods follow the presentation in (Carmona and Laurière, 2021c), while the last one is a finite horizon adaptation of the algorithm proposed in (Andrea Angiuli et al., 2025). For consistency, all algorithms are presented within the framework of our price impact model, although they readily extend to more general mean field formulations. We present numerical results and compare them with the explicit solutions.

Before presenting the algorithms, we introduce some useful notation and recall the neural network framework employed throughout this chapter. We begin by discretizing time, fix $N_T \in \mathbb{N}^*$, set $\Delta t = T/N_T$, and define $t_n = n\Delta t$ for $n \in \{0, \ldots, N_T\}$. Let $(\Delta \bar{W}_{t_n})_{n\in\{0,\ldots,N_T\}}$ denote Brownian increments and $\bar{X}$ the Euler-Maruyama discretization of the inventory process X

$$\bar{X}_{t_{n+1}} = \bar{X}_{t_n} + \alpha(t_i, \bar{X}_{t_n})\,\Delta t + \sigma\,\Delta \bar{W}_{t_n}, \quad \bar{X}_0 = \xi, \qquad n \in \{0, \ldots, N_T - 1\}. \tag{2.1}$$

Throughout this chapter, we use fully connected feedforward neural networks as function approximators. We denote by $\Phi_\theta : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1}$ a neural network with $\ell > 0$ hidden layers, each containing m neurons, and activation function ϱ, except for the output layer, which uses the identity activation. Here, $\theta \in \Theta \subset \mathbb{R}^{N_m}$ is the parameter vector of the network, with N_m the total number of trainable parameters. We denote by $\mathcal{NN}^{\varrho}_{d_0,d_1,\ell,m}$ the set of all such neural networks.

2.1 Direct method for McKean–Vlasov Control problems

In this section, we present a method for the MFC counterpart of our MFG problem. Unlike the MFG setting, where the equilibrium is characterized by a fixed point condition, the MFC framework corresponds to a *centralized* stochastic control problem. Intuitively, in MFC agents act cooperatively rather than competitively, one can view the system as governed by a social planner who directly chooses the controls of all agents in order to minimize a global cost functional depending on their joint distribution. In our framework, this problem takes the form

$$\inf_{\alpha\in\mathscr{A}} J^{\text{MFC}}(\alpha) = \inf_{\alpha\in\mathscr{A}} \mathbb{E}\left[\int_0^T f(X_t, \alpha_t, \mathcal{L}(\alpha_t))dt + g(X_T) - V_0\right]. \tag{2.2}$$

We emphasize that in contrast to the MFG case, the flow $\nu_t = \mathcal{L}(\alpha_t)$ is not fixed a priori but evolves during the optimization process. If $\hat{\alpha}$ denotes an optimal control for (2.2) with associated law $\hat{\nu} = \mathcal{L}(\hat{\alpha})$, and (α^*, ν^*)

is an MFG equilibrium, then one has

$$J^{\mathrm{MFC}}(\hat{\alpha}) = J^{\mathrm{MFG}}_{\hat{\nu}}(\hat{\alpha}) \leq J^{\mathrm{MFG}}_{\nu^*}(\alpha^*),$$

and $(\alpha^*, \nu^*) \neq (\hat{\alpha}, \hat{\nu})$ in general. The direct method we propose consists in approximating the control by a neural network $\alpha_\theta \in \mathcal{NN}^{\varrho}_{2,1,\ell,m}$ and minimizing, over Θ, the cost functional (omitting V_0, which is independent of the control)

$$J_{\mathrm{MFC}}(\theta) = \mathbb{E}\left[\int_0^T f(X_t, \alpha_\theta(t, X_t), \mathcal{L}(\alpha_\theta(t, X_t)))dt + g(X_T)\right].$$

To make this problem numerically tractable, we introduce two approximations. First, motivated by the principle of propagation of chaos, we approximate the law of the control by the empirical distribution of a system of $N > 0$ interacting particles. This yields

$$J_{\mathrm{MFC}}(\theta) \approx \frac{1}{N}\sum_{i=0}^{N} \mathbb{E}\left[\int_0^T f(X^i_t, \alpha_\theta(t, X^i_t), \bar{\nu}^N_t)dt + g(X^i_T)\right], \quad \bar{\nu}^N_t = \frac{1}{N}\sum_{i=1}^{N} \delta_{\alpha_\theta(t, X^i_t)}.$$

Second, we discretize time, replace the integral with a Riemann sum, and approximate X^i by its Euler scheme $\bar{X}^i$ for every $i \in \{1, \ldots, N\}$. This leads to the empirical loss function

$$L_{\mathrm{MFC}}(\theta) = \frac{1}{N}\sum_{i=0}^{N}\left[\sum_{n=0}^{N_T-1} f(\bar{X}^i_{t_n}, \alpha_\theta(t_n, \bar{X}^i_{t_n}), \check{\nu}^N_{t_n})dt + g(\bar{X}^i_T)\right], \quad \check{\nu}^N_{t_n} = \frac{1}{N}\sum_{i=1}^{N} \delta_{\alpha_\theta(t_n, \bar{X}^i_{t_n})}.$$

The optimization of $L_{\mathrm{MFC}}(\theta)$ is carried out by stochastic gradient descent (SGD) type algorithm. The procedure is summarized in Algorithm 1. For a complete discussion on this method, see (Carmona and Laurière, 2021c) and convergence analysis can be found in (Carmona and Laurière, 2021b).

Algorithm 1 Direct Method for MFC

```
Let $\alpha_{\theta_0} \in \mathcal{NN}^{\varrho}_{2,1,\ell,m}$; number of step $K$; a sequence of learning rates $(\beta_k)_{k\in\{0,\ldots,K-1\}}$.
for $k = 0, \ldots, K-1$ do
    Sample $S = (\bar{X}^i_0, (\Delta\bar{W}^i_{t_n})_n)_{i\in\{1,\ldots,N\}}$
    Compute the loss $L_{\mathrm{MFC}}(\theta)$
    Set $\theta_{k+1} = \theta_k - \beta_k \nabla_\theta L_{\mathrm{MFC}}(\theta)$
end for
```

We present results obtained with the deep learning method described above and compare them with the true solution reported in (Angiuli et al., 2021). The implementation relies on mini-batches and the ADAM optimizer to improve convergence. To evaluate performance during training, we consider the error metric

$$e_\alpha(k) = \mathbb{E}\left[\int_0^T |\alpha^*(t, X_t) - \alpha_{\theta_k}(t, X_t)|^2\, dt\right]. \tag{2.3}$$

For our numerical experiments, we use the model parameters reported in Table 2.1. We set $N = 10{,}000$ in order to obtain a reasonable approximation of the mean field, and choose $N_T = 100$ time steps. For the neural network α_θ, we take $\ell = 3$ hidden layers, $m = 128$ neurons per layer, and employ the ReLU activation function.

T	σ	γ	c_α	c_X	c_g	X^i_0
1.0	0.5	1.75	1.0	2.0	0.3	$\mathcal{N}(0.5, 0.3^2)$

Table 2.1: Model parameters used in all numerical experiments.

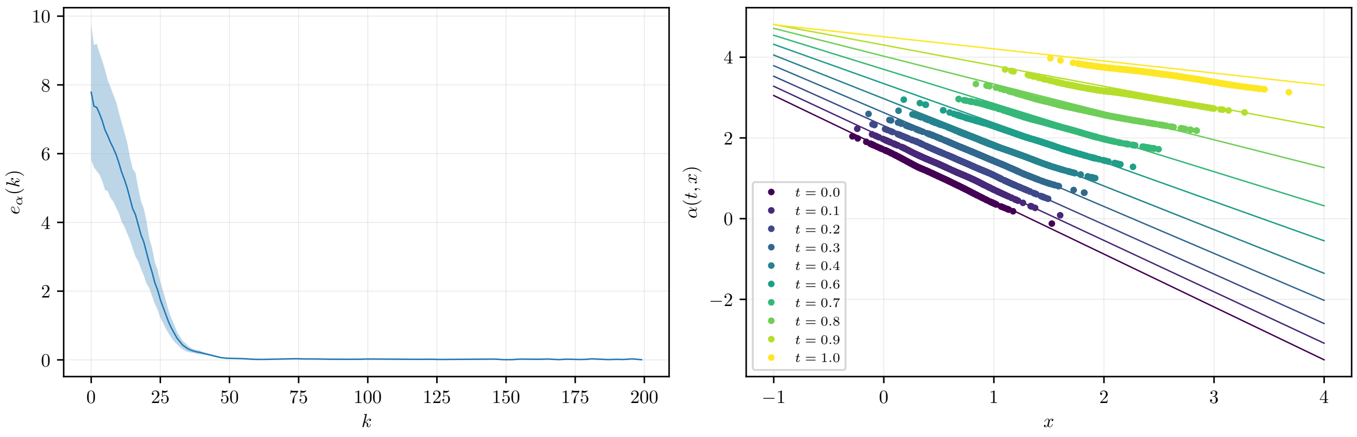

Figure 2.1: Results for the direct MFC method with parameters in Table 2.1. *Left:* evolution of the empirical error $e_\alpha(k)$ over epochs, with the shaded region representing the standard deviation across trajectories (as opposed to the expectation in (2.3)). *Right:* Control learnt (dot) and exact solution (lines) at different time point.

The empirical error $e_\alpha(k)$ exhibits rapid decay within the first 50 epochs before stabilizing near zero, thereby demonstrating both efficiency and stability of the training procedure. The comparison on the right confirms that the learned control aligns closely with the analytical solution across the time. Nevertheless, a loss of precision is observed near maturity ($t = 1.0$), which can plausibly be attributed to discretization effects, in particular those arising from the Euler scheme. Overall, the direct MFC approach yields a reliable approximation with fast convergence properties.

2.2 Deep BSDE method for McKean–Vlasov FBSDEs

We now present an extension of the deep BSDE method introduced in (E et al., 2017), adapted to the mean field setting through the *Global Direct Solver* algorithm developed in (Germain et al., 2022), and apply it to our MKV FBSDE (1.18). The main idea is to approximate the initial condition Y_0 by $\mathcal{Y}_{\theta^y}(X_0) \in \mathcal{NN}^{\varrho}_{1,1,\ell,m}$ and the process Z_t by $\mathcal{Z}_{\theta^z}(t, X_t) \in \mathcal{NN}^{\varrho}_{2,1,\ell,m}$. We then consider the Euler–Maruyama discretization of the MKV FBSDE (1.18)

$$\begin{cases} \bar{X}_{t_{n+1}} = \bar{X}_{t_n} - \dfrac{1}{c_\alpha}\bar{Y}_{t_n}\Delta t + \sigma\Delta\bar{W}_{t_n}, & \bar{X}_0 = \xi, \\ \bar{Y}_{t_{n+1}} = \bar{Y}_{t_n} - \left(c_X\bar{X}_{t_n} - \dfrac{\gamma}{c_\alpha}\mathbb{E}[\bar{Y}_{t_n}]\right)\Delta t + \mathcal{Z}_{\theta^z}(t_n, \bar{X}_{t_n})\Delta\bar{W}_{t_n}, & \bar{Y}_0 = \mathcal{Y}_{\theta^y}(\bar{X}_0). \end{cases} \tag{2.4}$$

The key observation is that the backward component is rewritten as a forward recursion. The training objective is therefore to find parameters $\theta = (\theta^y, \theta^z)$ such that the terminal condition $\bar{Y}_T = c_g\bar{X}_T$ is satisfied. As in the previous method, we approximate expectations by empirical averages over an $N-$particle system, which leads to the mean-squared error loss

$$L_{\text{FBSDE}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left|\bar{Y}^i_T - c_g\bar{X}^i_T\right|^2 \tag{2.5}$$

Note that both $\bar{Y}^i_T$ and $\bar{X}^i_T$ depend on θ through their construction in (2.4). The intuition is that a solution to (1.18) is minimizer of the following optimization problem

$$\inf_{Y_0,\,(Z_t)_{t\in[0,T]}} \mathbb{E}\left[\left|Y_T - c_gX_T\right|^2\right],$$

subject to the dynamics

$$\begin{cases} X_t = \xi - \dfrac{1}{c_\alpha}\displaystyle\int_0^t Y_s ds + \sigma W_t, \\ Y_t = Y_0 - \displaystyle\int_0^t \left(c_X X_s - \tfrac{\gamma}{c_\alpha}\mathbb{E}[Y_s] \right) ds + \displaystyle\int_0^t Z_s dW_s. \end{cases}$$

Since existence and uniqueness of the above equations are guaranteed, minimizing (2.5) under the dynamics given in (2.4) provides a consistent approximation of the true solution. This lead to the following algorithm.

Algorithm 2 Deep MKV FBSDE

```
Let $\mathcal{Y}_{\theta_0^y} \in \mathcal{NN}^{\varrho}_{1,1,\ell,m}$ and let $\mathcal{Z}_{\theta_0^z} \in \times\mathcal{NN}^{\varrho}_{2,1,\ell',m'}$; number of step $K$; a sequence of learning rates $(\beta_k)_{k\in\{0,\dots,K-1\}}$.
for $k = 0, \dots, K-1$ do
    Sample $S = (\bar{X}_0^i, (\Delta\bar{W}_{t_n}^i)_n)_{i\in\{1,\dots,N\}}$
    Build the trajectories $(\bar{X}_{t_n}^i, \bar{Y}_{t_n}^i)_{n\in\{0,\dots,N_T\}, i\in\{1,\dots,N\}}$ using dynamics (2.4)
    Compute the MSE loss $L_{\mathrm{FBSDE}}(\theta_k)$, where $\theta_k = (\theta_k^y, \theta_k^z)$
    Set $\theta_{k+1} = \theta_k - \beta_k \nabla_\theta L_{\mathrm{FBSDE}}(\theta_k)$
end for
```

Convergence for coupled FBSDEs is studied in (Han and Long, 2020). Again, we take $N = 10{,}000$ in order to obtain a reasonable approximation of the mean field and take $N_T = 100$. The network $\mathcal{Y}_{\theta^y}$ has $\ell = 2$ hidden layers with $m = 64$ neurons per layer, while $\mathcal{Z}_{\theta^z}$ has $\ell' = 2$ hidden layers with $m' = 128$ neurons per layer. Both networks use the ReLU activation function.

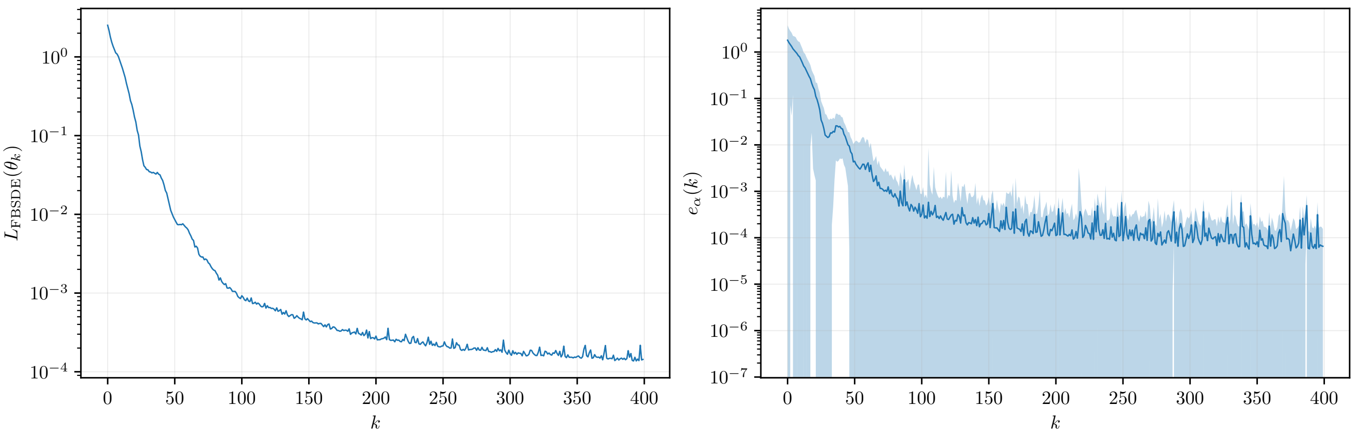

Figure 2.2: Training results using Algorithm 2 with parameters in Table 2.1. *Left:* evolution of the loss $L_{\mathrm{FBSDE}}(\theta_k)$ over epochs in log-scale. *Right:* We plot in log-scale the evolution of the empirical error $e_\alpha(k)$ over epochs, with the shaded region representing the standard deviation across trajectories. Recalling (1.17), the approximated optimal control is $-Y_t/c_\alpha$ here.

The Deep MKV BSDE method demonstrate both efficiency and robustness. As shown in Figure 2.2, both loss function and empirical error converges relatively quickly. Figure 2.3 further confirms the accuracy of the method, the learned feedback control and the population distribution both match the analytical solutions very closely throughout the time horizon. A slight increase in variance can be observed near maturity, likely due to discretization effects. Among the model-based approaches, this method is the most effective overall, as it provides fast and accurate convergence while being applicable to both MFG and MFC problems (even though we do not display the MFC results here).

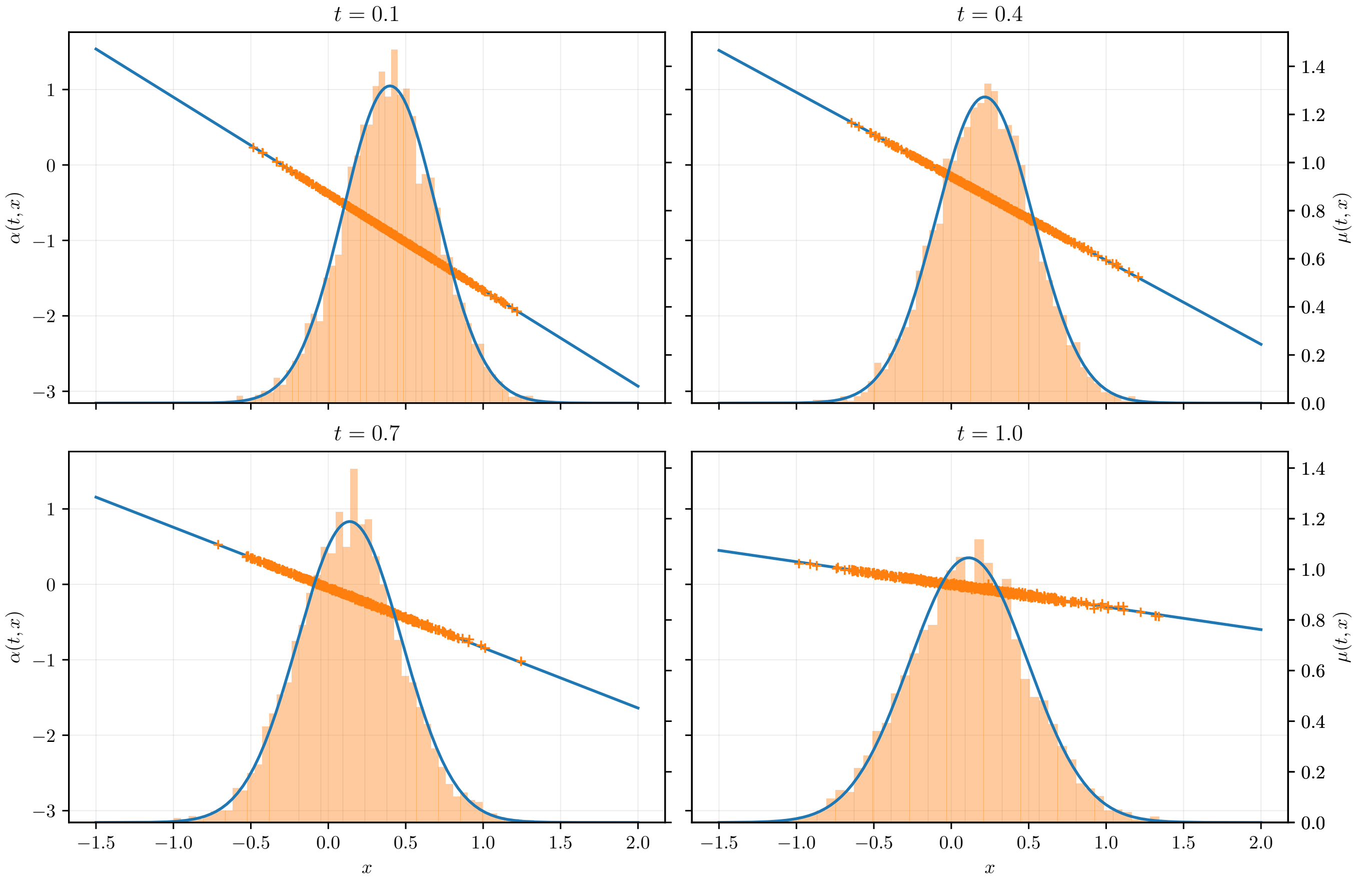

Figure 2.3: Results using Algorithm 2 with parameters in Table 2.1. At selected times $t \in \{0.1, 0.4, 0.7, 1.0\}$, we show the empirical distribution of the inventory process (histogram) together with the true density represented by the right axis. The learned feedback control is represented by scatter points, while the true optimal control is plotted as a solid line on the left axis.

2.3 Deep Galerkin method for Mean Field PDEs

The Deep Galerkin Method (DGM), introduced in (Sirignano and Spiliopoulos, 2018), is a neural network approach designed to solve (high-dimensional) PDEs. It is particularly well suited to mean field problems, since in our case we must solve simultaneously a HJB and KFP equations. Specifically, we aim to solve (1.11) and (1.12). The key idea is to approximate the unknown solutions of these PDEs by neural networks and train them to satisfy the PDE residuals together with the initial/terminal conditions. For simplicity, we let $v(t,x) = v_\nu(t,x)$ and make the approximation $v_{\theta^1}(t,x) \approx v(t,x)$ and $m_{\theta^2}(t,x) \approx m(t,x)$. The objective is thus to find $\theta = (\theta^1, \theta^2)$ such that the following PDE system holds

$$\begin{cases} \partial_t v_{\theta^1}(t,x) = -\gamma x \int_{\mathbb{R}} \frac{\partial_x v_{\theta^1}(t,y)}{c_\alpha} m(t,y)dy - \frac{c_X}{2}x^2 - \frac{\sigma^2}{2}\partial^2_{xx} v_{\theta^1}(t,x) + \frac{\partial_x v_{\theta^1}(t,x)^2}{2c_\alpha}, \\ \partial_t m_{\theta^2}(t,x) = \frac{\sigma^2}{2}\partial^2_{xx} m_{\theta^2}(t,x) + \frac{1}{c_\alpha}\partial_x \left(m_{\theta^2}\partial_x v_{\theta^1}\right)(t,x), \\ v_{\theta^1}(T,x) = \frac{c_g}{2}x^2, \quad m_{\theta^2}(0,x) = m_0(x). \end{cases} \tag{2.6}$$

Instead of discretizing the PDEs on a fixed grid, we sample training points (t,x) randomly from a compact subset of $[0,T] \times \mathbb{R}$. We denote by $\mathbf{S} = (S, S_0, S_T)$ the training sets, where S is a finite subset of $[0,T] \times \mathbb{R}$, S_0 and S_T a finite subset of $\mathbb{R}$ for the initial and terminal condition respectively. The total loss is defined as

$$L_{\text{DGM}}(\theta) = L_{\text{HJB}}(\theta) + L_{\text{KFP}}(\theta),$$

with

$$
\begin{aligned}
&L_{\mathrm{HJB}}(\theta) \\
&= C^{\mathrm{HJB}} \left(\frac{1}{|S|} \sum_{(t,x)\in S} \left| \partial_t v_{\theta^1}(t,x) + \gamma x \int_{\mathbb{R}} \frac{\partial_x v_{\theta^1}(t,y)}{c_\alpha} m(t,y)dy + \frac{c_X}{2}x^2 + \frac{\sigma^2}{2}\partial^2_{xx} v_{\theta^1}(t,x) - \frac{\partial_x v_{\theta^1}(t,x)^2}{2c_\alpha} \right|^2 \right)^{1/2} \\
&+ C_T^{\mathrm{HJB}} \left(\frac{1}{|S_T|} \sum_{x\in S_T} \left| v_{\theta^1}(T,x) - \frac{c_g}{2}x^2 \right|^2 \right)^{1/2},
\end{aligned}
$$

and

$$
\begin{aligned}
L_{\mathrm{KFP}}(\theta) = C^{\mathrm{KFP}} &\left(\frac{1}{|S|} \sum_{(t,x)\in S} \left| \partial_t m_{\theta^2}(t,x) - \frac{\sigma^2}{2}\partial^2_{xx} m_{\theta^2}(t,x) - \frac{1}{c_\alpha}\partial_x \left(m_{\theta^2} \partial_x v_{\theta^1} \right)(t,x) \right|^2 \right)^{1/2} \\
&+ C_0^{\mathrm{KFP}} \left(\frac{1}{|S_0|} \sum_{x\in S_0} \left| m_{\theta^2}(0,x) - m_0(x) \right|^2 \right)^{1/2}.
\end{aligned}
$$

Here $C^{\mathrm{HJB}}, C_T^{\mathrm{HJB}}, C^{\mathrm{KFP}}, C_0^{\mathrm{KFP}}$ are positive hyperparameters that balance the different contributions of the loss. Note that $L_{\mathrm{DGM}}(\theta) = 0$ whenever $(v_{\theta^1}, m_{\theta^2})$ is an exact solution of (2.6) on the training set $\mathbf{S}$, so minimizing the loss provides an approximation of the true solution (at least around $\mathbf{S}$). We summarized this method in Algorithm 3.

Algorithm 3 DGM for MFG PDEs

```
Let $(v_{\theta_0^1}, m_{\theta_0^2}) \in \mathcal{NN}^{\varrho}_{2,1,\ell,m} \times \mathcal{NN}^{\varrho}_{2,1,\ell',m'}$; number of step $K$; learning rates $(\beta_k)_{k\in\{0,\dots,K-1\}}$.
for $k = 0, \dots, K-1$ do
    Sample points in $\mathbf{S} = (S, S_0, S_T)$
    Compute the loss $L_{\mathrm{DGM}}(\theta_k)$, where $\theta_k = (\theta_k^1, \theta_k^2)$
    Set $\theta_{k+1} = \theta_k - \beta_k \nabla_\theta L_{\mathrm{DGM}}(\theta_k)$
end for
```

One of the main challenges in the KFP equation is to ensure that m remains positive and integrates to one. A solution proposed in (Al-Aradi et al., 2018) is to perform the change of variable $m(t,x) = e^{-u(t,x)}/c(t)$, where $c(t) = \int_{\mathbb{R}} e^{-u(t,x)}dx$ which guarantees positivity and normalization. The function u then satisfies the PDE

$$
\partial_t u = \frac{\sigma^2}{2}\left(\partial_{xx} u - \partial_x u^2 \right) + \frac{1}{c_\alpha}\left(\partial_x v \partial_x u - \partial_{xx} v \right) - \frac{c'}{c}.
$$

Since the initial density is Gaussian, the corresponding initial condition for u is $u(0,x) = \frac{(x-\mu_X(0))^2}{2\Sigma_X(0)}$. The integral term $c'(t)/c(t)$ can be approximated by uniformly sampling N_x points x_i in $[x_{\min}, x_{\max}]$

$$
-\frac{c'(t)}{c(t)} = \frac{\int_{\mathbb{R}} \partial_t u(t,x) e^{-u(t,x)}dx}{\int_{\mathbb{R}} e^{-u(t,x)}dx} \approx \frac{\sum_{i=1}^{N_x} \partial_t u(t,x_i) e^{-u(t,x_i)}}{\sum_{i=1}^{N_x} e^{-u(t,x_i)}}.
$$

Similarly, the mean field interaction term becomes

$$
\int_{\mathbb{R}} \partial_x v(t,x) m(t,x) dx = \frac{\int_{\mathbb{R}} \partial_x v(t,x) e^{-u(t,x)}dx}{\int_{\mathbb{R}} e^{-u(t,x)}dx} \approx \frac{\sum_{i=1}^{N_x} \partial_x v(t,x_i) e^{-u(t,x_i)}}{\sum_{i=1}^{N_x} e^{-u(t,x_i)}}.
$$

In practice, we approximate u with a neural network u_{θ^2}, and replace in L_{KFP} the residual and initial condition terms for m by their counterparts expressed in terms of u. As emphasized in (Sirignano and Spiliopoulos, 2018), the architecture of the neural networks plays a crucial role in the performance of the method. For this reason, instead of using feedforward networks, we adopt the DGM architecture introduced in (Sirignano and Spiliopoulos, 2018). This architecture is inspired by long short-term memory (LSTM) networks and is specifically designed to better capture temporal dependencies in PDE solutions. The recursive structure of the DGM layers is given by

$$\begin{aligned}
S^1 &= \varrho(W^1\mathbf{x} + b^1),\\
Z^\ell &= \varrho(U^{z,\ell}\mathbf{x} + W^{z,\ell}S^\ell + b^{z,\ell}), \quad \ell = 1,\dots,L,\\
G^\ell &= \varrho(U^{g,\ell}\mathbf{x} + W^{g,\ell}S^\ell + b^{g,\ell}), \quad \ell = 1,\dots,L,\\
R^\ell &= \varrho(U^{r,\ell}\mathbf{x} + W^{r,\ell}S^\ell + b^{r,\ell}), \quad \ell = 1,\dots,L,\\
H^\ell &= \varrho(U^{h,\ell}\mathbf{x} + W^{h,\ell}(S^\ell \odot R^\ell) + b^{h,\ell}), \quad \ell = 1,\dots,L,\\
S^{\ell+1} &= (1 - G^\ell) \odot H^\ell + Z^\ell \odot S^\ell, \quad \ell = 1,\dots,L,\\
\Psi_\theta(t,x) &= WS^{L+1} + b,
\end{aligned}$$

where $\odot$ denotes the element-wise product. Here $\mathbf{x} = (t,x)$, the network has $L+1$ hidden layers, and ϱ is an element-wise nonlinear activation function. The set of trainable parameters is

$$\theta = \left\{W^1, b^1, \left(U^{i,\ell}, W^{i,\ell}, b^{i,\ell}\right)_{i\in\{z,g,r,h\},\ell\in\{1,\dots,L\}}, W, b\right\},$$

with dimensions specified as

$$W^{i,\ell} \in \mathbb{R}^{m\times m}, \quad W^1, U^{i,\ell} \in \mathbb{R}^{m\times(d+1)}, \quad b^1, b^{i,\ell} \in \mathbb{R}^m, \quad W \in \mathbb{R}^{1\times m}, b \in \mathbb{R}, \quad i = z,g,r,h,$$

where d is the dimension of the state variable x, and m is the width of each hidden layer. For our numerical results, we take $L = 3$, $m = 50$ and $\varrho = \tanh$ and sample 10,000 points in $\mathbf{S}$. One of the main challenges of this method lies in the choice of the relative weights in the loss function. If these weights are not properly balanced, the SGD can easily become trapped in local minima, which may lead to significantly longer training times compared to the two previous methods, see (Carmona and Laurière, 2021a) for a discussion on this subject.

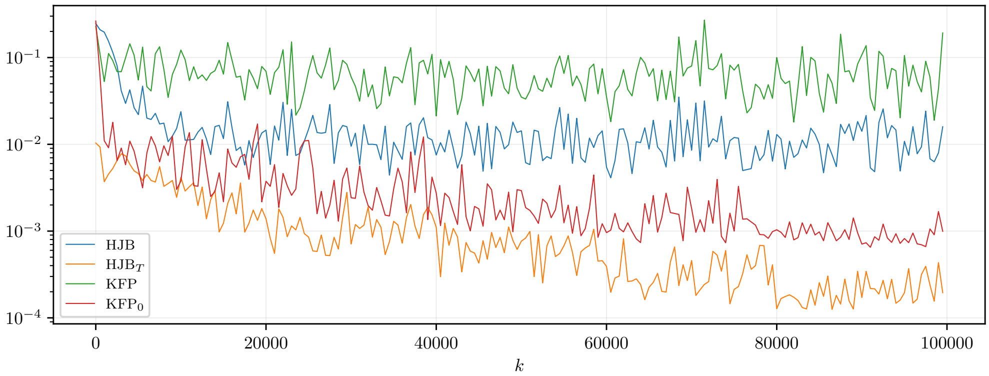

Figure 2.4: Training results using Algorithm 3 with parameters in Table 2.1. We plot the evolution of the different contributions to the loss $L_{\mathrm{DGM}}(\theta_k)$ over epochs, HJB residuals (blue), HJB terminal condition (orange), KFP residuals (green) and KFP initial condition (red).

In our experiments, we set $C^{\mathrm{HJB}} = C^{\mathrm{KFP}} = 2, C_T^{\mathrm{HJB}} = 4$, and $C_0^{\mathrm{KFP}} = 1$. As shown in Figure 2.4, both boundary condition terms converge toward the true solution. However, with this choice of weights, both HJB and KFP residual losses do not decay significantly across epochs, suggesting that the optimization may be trapped in local minima. This likely explains the poor approximation near $x = 1$ observed in Figure 2.5. Overall, this method proves to be the least effective among those tested, as the sensitivity to weight selection

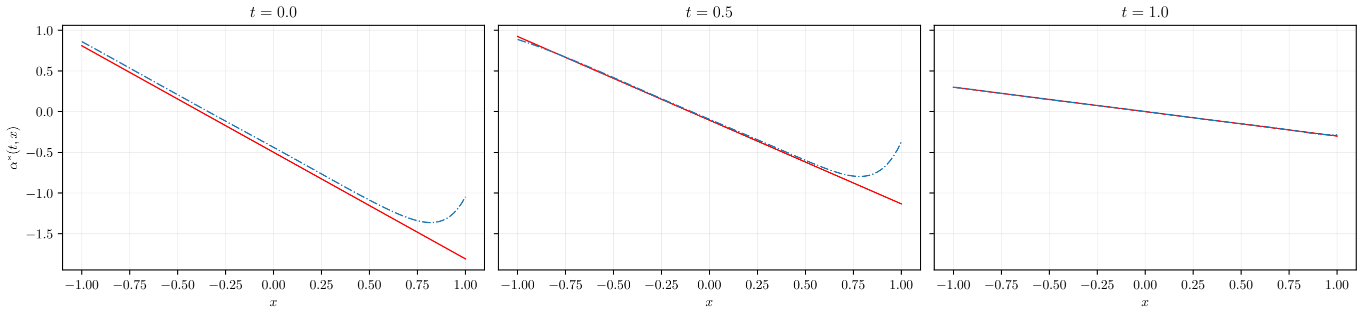

Figure 2.5: Results using Algorithm 3 with parameters in Table 2.1. We plot the learned optimal control (dashed lines) and the exact solution (solid lines) at different time points $t \in \{0.0, 0.5, 1.0\}$.

makes training unstable and considerably slower compared to the other approaches.

2.4 Reinforcement Learning methods

In this section, we focus on two reinforcement learning (RL) methods. Over the past decade, RL has made substantial progress in both theory and practice, generating significant interest in quantitative finance. In contrast to the methods studied in the previous sections, which rely on explicit knowledge of the model dynamics, RL methods are typically *model–free*, the agent interacts with the environment and learns optimal strategies directly from observations, effectively treating the environment as a "black box". This feature is particularly appealing in financial contexts, where the true market dynamics are often only partially known or subject to model misspecification, see (Pippas et al., 2025) for a recent survey.

2.4.1 Mean field Q–learning method

In this section, we present the only method that do not use deep learning but an adaptation to mean field setting of the so called Q–learning method in RL presented in (Angiuli et al., 2021). While most of the RL literature considers infinite horizon discounted problems, optimal execution and related applications are naturally finite horizon. This setting is more numerically demanding as the value function and policy is time dependend, but it is crucial for practical relevance. At each time step t_n, the agent observes the state X_{t_n}, selects an action A_{t_n} according to a policy $\pi : [0,T] \times \mathbb{R} \to \mathcal{P}(\mathbb{R})$, and receives an instantaneous cost $F(X_{t_n}, A_{t_n}, \nu_{t_n})$, at maturity T, the agent incurs a terminal cost $G(X_T)$[1]. We recall that in our framework $F = f\Delta t$ and $G = g$. The goal is to learn an optimal policy π^* minimizing the expected cumulative cost. The value function associated with a policy π is

$$v^\pi(t_n, x_n) = \mathbb{E}^\pi \left[\sum_{k=n}^{N_T - 1} F(X_{t_k}, A_{t_k}, \nu_{t_k}) + G(X_T) \,\middle|\, X_{t_n} = x_n \right]. \tag{2.7}$$

Similarly, the action–value function, or *Q-function*, is given by

$$q^\pi(t_n, x_n, a_n) = \mathbb{E}^\pi \left[\sum_{k=n}^{N_T - 1} F(X_{t_k}, A_{t_k}, \nu_{t_k}) + G(X_T) \,\middle|\, X_{t_n} = x_n, A_{t_n} = a_n \right]. \tag{2.8}$$

The optimal value function is $v^*(t,x) = \inf_\pi v^\pi(t,x)$, and the optimal Q-function is $q^*(t,x,a) = \inf_\pi q^\pi(t,x,a)$. The optimal policy π^* is then characterized as the minimizer, and one has the relation $v^*(t,x) = \inf_a q^*(t,x,a)$. This formulation is particularly convenient in the discrete setting, where the optimal policy can be expressed

[1] In the model-free setting, the agent does not know the exact dynamics of X nor the functions F and G, these are accessible only through interaction with the environment.

as $\pi^*(\cdot \mid t, x) = \mathrm{unif}(\arg\min_a(q^*(t,x,a)))$. The value function satisfies the well-known *Bellman equation*

$$v^\pi(t_n, x_n) = \mathbb{E}^\pi\left[F(X_{t_n}, A_{t_n}, \nu_{t_n}) + v^\pi(t_{n+1}, X_{t_{n+1}}) \,\middle|\, X_{t_n} = x_n\right], \tag{2.9}$$

For the mean field Q-learning method, we consider a discrete state space $\mathcal{X} \subset \mathbb{R}$ and a discrete action space $\mathcal{A} \subset \mathbb{R}$. By dynamic programming, the optimal Q-function satisfies in $\mathcal{X} \times \mathcal{A}$

$$\begin{cases} q^*(n,x,a) = F(x,a,\nu_{t_n}) + \sum_{x' \in \mathcal{X}} p(x' \mid x, a, \theta_{t_n}) \min_{a' \in \mathcal{A}} q^*(n+1, x', a'), \quad n < N_T, \\ q^*(N_T, x, a) = G(x). \end{cases} \tag{2.10}$$

where $p(x' \mid x, a, \theta_{t_n})$ denotes the transition probability to state x' given that the agent is currently in state x, chooses action a, while the state–action distribution is θ_{t_n}. We denote by μ and ν the state and action marginals of θ, respectively. Here $q^* = q^*_\theta$ depends implicitly on the population distribution $\theta = (\theta_{t_n})_n$. The Q-function represents, at each time step t_n, the minimal expected cost-to-go for an agent starting in state x, applying action a at time t_n, and acting optimally thereafter, while the population evolves according to θ.

In order to characterize a Nash equilibrium, one must compute $q^*_{\theta^*}$, where θ^* is the population distribution induced by the optimal control derived from $q^*_{\theta^*}$. The algorithm proceeds by iteratively updating the Q-function and the state–action distribution $\theta \in \mathcal{P}_2(\mathbb{R}^2)$. Starting from an initial guess θ^0, we repeat the following two steps

1. We solve the backward equation for $q^{k+1} := q^*_{\theta^k}$ given θ^k

$$\begin{cases} q^{k+1}(n,x,a) = F(x,a,\nu^k_{t_n}) + \sum_{x' \in \mathcal{X}} p(x' \mid x, a, \theta_{t_n}) \min_{a' \in \mathcal{A}} q^{k+1}(n+1, x', a'), \quad n < N_T, \\ q^{k+1}(N_T, x, a) = G(x). \end{cases} \tag{2.11}$$

2. We solve the forward equation for θ^{k+1} given q^{k+1} which characterizes the evolution of the state-action distribution if everyone uses the optimal control $\alpha^{k+1}_{t_n}(x) := \arg\min_a q^{k+1}(n,x,a)$

$$\begin{cases} \mu^{k+1}_0(x) = \mu_0(x), \\ \theta^{k+1}_0(x,a) = \mu_0(x)\mathbb{1}_{a = \alpha^{k+1}_0(x)}, \\ \mu^{k+1}_{t_{n+1}}(x) = \sum_{x' \in \mathcal{X}} p(x' \mid x, \alpha^{k+1}_{t_n}(x'), \theta^{k+1}_{t_n}), \quad 0 \leq n < N_T \\ \theta^{k+1}_{t_{n+1}}(x,a) = \mu^{k+1}_{t_{n+1}}(x)\mathbb{1}_{a = \alpha^{k+1}_{t_{n+1}}(x)}, \quad 0 \leq n < N_T. \end{cases} \tag{2.12}$$

Here, the evolution of the joint state–action distribution is simply given by the product of the state distribution and a Dirac mass, $\theta^{k+1}_{t_n} = \mu^{k+1}_{t_n} \otimes \delta_{\alpha^{k+1}_{t_n}}$. The updates (2.11) and (2.12) can be summarized as

$$q^{k+1} = \widetilde{\mathcal{T}}(\theta^k), \qquad \theta^{k+1} = \widetilde{\mathcal{P}}(q^{k+1}),$$

where $\widetilde{\mathcal{T}}$ and $\widetilde{\mathcal{P}}$ denote, respectively, the backward and forward operators. We expect this iteration to converge to a pair $(q^\infty, \theta^\infty)$ such that

$$q^\infty = \widetilde{\mathcal{T}}(\theta^\infty), \qquad \theta^\infty = \widetilde{\mathcal{P}}(q^\infty),$$

which implies that θ^∞ is the equilibrium state–action distribution of the MFG, and that the associated optimal control is

$$\alpha^\infty_{t_n}(x) = \arg\min_{a \in \mathcal{A}} q^\infty(n,x,a), \qquad \forall n \in \{0, \ldots, N_T\}.$$

In practice, however, the scheme does not guarantee convergence due to the absence of a strict contraction property. To address this issue, we introduce a damping mechanism through two learning-rate sequences

$(\rho^q_k)_{k\geq 0}$ and $(\rho^\theta_k)_{k\geq 0}$, and perform the updates

$$q^{k+1} = (1-\rho^q_k)q^k + \rho^q_k \widetilde{\mathcal{T}}(\theta^k), \qquad \theta^{k+1} = (1-\rho^\theta_k)\theta^k + \rho^\theta_k \widetilde{\mathcal{P}}(q^{k+1}).$$

This corresponds to a *two-timescale approach*, in which the Q-function and the state–action distribution are updated at different speeds. To obtain convergence toward the MFG solution (and not the MFC solution), we should have $\rho^\theta_k < \rho^q_k$. The intuition is that the state–action distribution θ is seen as quasi-frozen while the Q-function q is updated. We refer to (Angiuli et al., 2021) for a detailed discussion on this approach and the choice of learning rates. The resulting procedure is summarized in Algorithm 2.4.1.

Algorithm 4 Finite Horizon Mean Field Q–Learning

```
Let $\mathcal{X} = \{x_0, \dots, x_{|\mathcal{X}|-1}\}$ and $\mathcal{A} = \{a_0, \dots, a_{|\mathcal{A}|-1}\}$.
Let $\varepsilon > 0$ related to the $\varepsilon$-greedy policy; number of episode $K$.
Initialize $q^0(n,x,a) = 0$ for all $(n,x,a) \in \{0,\dots,N_T\} \times \mathcal{X} \times \mathcal{A}$.
Initialize $\theta^0_{t_n}(x,a) = \frac{1}{|\mathcal{A}|\times|\mathcal{X}|}$ for all $(n,x,a) \in \{0,\dots,N_T\} \times \mathcal{X} \times \mathcal{A}$.
for $k = 1, \dots, K$ do
    Agent receives initial state $X_0 \sim \xi$ from the environment
    for $n = 0, \dots, N_T$ do
        Choose action $A_{t_n}$ using $\varepsilon$-greedy policy derived from $q^{k-1}(x, X_{t_n}, \cdot)$.
        Update $\theta^k$
            $\theta^k_{t_n} = \theta^{k-1}_{t_n} + \rho^\theta_k(\boldsymbol{\delta}(X_{t_n}, A_{t_n}) - \theta^{k-1}_{t_n}),$
        where $\boldsymbol{\delta}(X_{t_n}, A_{t_n}) = (\mathbb{1}_{x,a}(X_{t_n}, A_{t_n}))_{x\in\mathcal{X},a\in\mathcal{A}}$.
        Observe the cost $F_{t_{n+1}} = F(X_{t_n}, A_{t_n}, \nu^k_{t_n})$ and state $X_{t_{n+1}}$ from the environment.
        Update $q^k$
            $q^k(n,x,a) = \begin{cases} q^{k-1}(n,x,a) + \rho^q_k(\mathcal{B} - q^{k-1}(n,x,a)) & \text{if } (X_{t_n}, A_{t_n}) = (x,a), \\ q^{k-1}(n,x,a) & \text{o.w.} \end{cases}$
        where
            $\mathcal{B} = \begin{cases} F_{t_{n+1}} + \min_{a'\in\mathcal{A}} q^{k-1}(n+1, X_{t_{n+1}}, a') & \text{if } t_n < T, \\ F_{t_{n+1}} & \text{o.w.} \end{cases}$
    end for
end for
```

For the numerical experiments, we consider the discrete state space $\mathcal{X} = \{x_0 = -1.5, \dots, x_{|\mathcal{X}|-1} = 1.75\}$ and the discrete action space $\mathcal{A} = \{a_0 = -2.5, \dots, a_{|\mathcal{A}|-1} = 1\}$, with discretization steps $\Delta_x = \Delta_a = \sqrt{\Delta_t} = 1/4$. The learning rates are defined as

$$\rho^q_k = \frac{1}{(1 + \#(x,a,t_n,k))^{\omega^q}}, \qquad \rho^\theta_k = \frac{1}{(1+k)^{\omega^\theta}},$$

where $\#(x,a,t_n,k)$ denotes the number of visits of the state–action pair (x,a) at time t_n up to episode k. We set $(\omega^q, \omega^\theta) = (0.55, 0.85)$. The exploitation–exploration trade-off is handled at each episode through an ε-greedy policy, if the agent is in state x, the algorithm selects the action that is currently estimated as optimal with probability $1-\varepsilon$, and a random action from $\mathcal{A}$ with probability ε. In our implementation, we fix $\varepsilon = 0.1$.

As shown in Figure 2.6, the learned optimal control does not align well with the true solution. In particular, at maturity $t = 1$, the method completly fails to recover the correct solution. This discrepancy is likely due to the coarse discretization of the state, action, and time spaces. Indeed, refining these grids would improve accuracy, but at the cost of an explosion in the size of the Q-matrix, a clear manifestation of the curse of dimensionality. This fundamental limitation makes the tabular Q-learning approach impractical beyond very

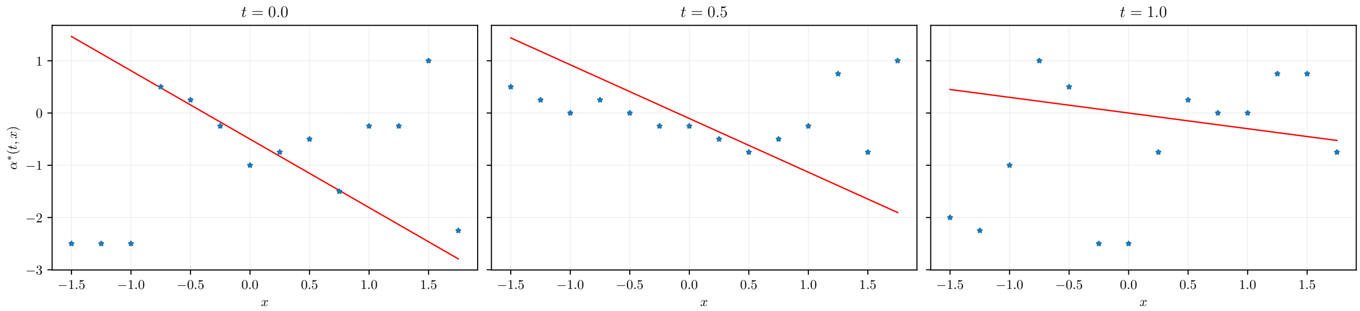

Figure 2.6: Results using Algorithm 2.4.1 with parameters in Table 2.1. We plot the learned optimal control (dots) and the exact solution (solid lines) at different time points $t \in \{0.0, 0.5, 1.0\}$.

low dimensions. For this reason, we now turn to more advanced RL techniques, such as actor–critic methods, which leverage function approximation to overcome these issues and scale more effectively to continuous and higher-dimensional settings.

2.4.2 Deep Reinforcement Learning Actor–Critic method

We present an adaptation of the actor–critic (AC) algorithm proposed in (Andrea Angiuli et al., 2025) to our finite horizon MFG framework. Recalling the definition of the value function (2.7), we approximate the value function (the *critic*) by a neural network V_θ and the policy (the *actor*) by a neural network Π_ψ. We know that the value function satisfies the Bellman equation (2.9) which provides the foundation for updating the critic via *temporal difference* (TD) learning. The TD error is defined as

$$\delta_n = F(X_{t_n}, A_{t_n}, \nu_{t_n}) + V_\theta(t_{n+1}, X_{t_{n+1}}) - V_\theta(t_n, X_{t_n}), \tag{2.13}$$

which in expectation, should vanish when V_θ is an accurate approximation of v^π. This motivates the critic loss function

$$L_V^{(n)}(\theta) = \delta_n^2. \tag{2.14}$$

Following the discussion in (Andrea Angiuli et al., 2025), only the term $V_\theta(t_n, X_{t_n})$ in (2.13) is treated as depending on θ, while the remaining terms are considered fixed during differentiation. This convention accelerates convergence in practice. Updating the actor is less direct. Minimizing V_θ with respect to ψ would require computing $\nabla_\psi V_\theta$, which is generally intractable. Instead, we use the *policy gradient theorem*, which provides an estimate of the policy gradient when combined with the TD error as a baseline. This estimate is obtained using the log-derivative trick, one has

$$\nabla_\psi v^{\Pi_\psi}(0, x_0) = \mathbb{E}^{\Pi_\psi}\left[\sum_{n=0}^{N_T - 1} \delta_n \nabla_\psi \log \Pi_\psi(A_{t_n} \,|\, t_n, X_{t_n}) \,\middle|\, X_0 = x_0\right]. \tag{2.15}$$

This suggests the per-step actor loss

$$L_\Pi^{(n)}(\psi) = \delta_n \log \Pi_\psi(A_{t_n} \,|\, t_n, X_{t_n}). \tag{2.16}$$

A key step in the MFG setting is to represent the mean field distribution ν_t. Following (Andrea Angiuli et al., 2025), we adopt a score-matching approach. Assume that ν_t admits a density p_t, we define the Stein score function as

$$s_t(x) := \nabla \log p_t(x).$$

This function enables us to sample from ν_t without explicit knowledge of p_t, using a Langevin Monte Carlo scheme. Given a step size $\varepsilon > 0$ and an arbitrary initialization a_0, the sequence $(a_k)_{k\geq 0}$ is defined by

$$a_{k+1} = a_k + \frac{\varepsilon}{2} s_t(a_k) + \sqrt{\varepsilon} Z_k, \qquad Z_k \sim \mathcal{N}(0,1), \tag{2.17}$$

which converges in law to ν_t as $k \to \infty$ and $\varepsilon \to 0$. We approximate the score function s by introducing a third neural network Σ_φ. The objective is then to minimize over φ, the L^2 error

$$\mathbb{E}\left[\int_0^T |s_t(A_t) - \Sigma_\varphi(t, A_t)|^2 \, dt\right]. \tag{2.18}$$

Expanding the square and applying Fubini's theorem, minimizing (2.18) is equivalent to minimizing

$$\int_0^T \mathbb{E}[|\Sigma_\varphi(t, A_t)|^2] - 2\mathbb{E}\left[s_t(A_t)\Sigma_\varphi(t, A_t)\right] dt. \tag{2.19}$$

Assuming that $p_t(x)\Sigma_\varphi(x) \to 0$ as $|x| \to \infty$, recalling the definition of s_t, integration by parts yields

$$\mathbb{E}\left[s_t(A_t)\Sigma_\varphi(t, A_t)\right] = -\mathbb{E}\left[\operatorname{div}_x \Sigma_\varphi(t, A_t)\right].$$

Plugin this identity into (2.19) and dividing by 2, suggests the following per-step loss

$$L_\Sigma^{(n)}(\varphi) = \tfrac{1}{2} \left|\Sigma_\varphi(t_n, A_{t_n})\right|^2 + \operatorname{div}_x \Sigma_\varphi(t_n, A_{t_n}). \tag{2.20}$$

We use Σ_φ to sample from ν_t, whose empirical distribution provides an approximation of the mean field. The learning rates for the critic, actor, and score network must be chosen carefully. Inspired by (Angiuli et al., 2021), we employ a three-timescale update scheme. First, we set $\rho^\Pi < \rho^V$ so that the critic learns faster than the actor, this ensures that the critic provides an accurate estimate of the value function for the current policy, while the actor can be regarded as nearly frozen from the critic's perspective. Next, we enforce $\rho^\Sigma < \rho^\Pi < \rho^V$ to guarantee convergence to the MFG solution rather than the MFC one. The intuition is that the mean field should evolve on the slowest timescale, so that both actor and critic adapt to a quasi static environment.

In summary, the finite horizon actor–critic algorithm alternates between updating the critic to satisfy the Bellman equation, updating the actor to improve the policy via the policy gradient, and updating the score network to represent the mean field. The three–timescale learning scheme is constructed so that each component of the algorithm evolves at a different rate, which intuitively promotes stability of the overall learning dynamics. On this basis, we expect the procedure to converge toward a mean field equilibrium. Nevertheless, establishing rigorous convergence guarantees for such actor–critic schemes in the mean field setting remains an open question. The full procedure is presented in Algorithm 2.4.2.

In our implementation, both the critic and the score networks are standard feedforward architectures with $\ell = 2$ hidden layers, $m = 128$ neurons per layer, and tanh activation functions. The actor network is designed to output the mean and standard deviation of a Gaussian distribution. It consists of a shared hidden layer with 128 neurons and tanh activation, followed by two separate branches, each with two hidden layers of 128 neurons and tanh activations. A final `softplus` layer is added to the standard deviation branch to ensure positivity of the output. For the Langevin Monte Carlo iterations, we use a step size of $\varepsilon = 5 \times 10^{-2}$, with 200 iterations at each step and $k = 1{,}000$ samples. The learning rates are set as follows

$$\rho^V = 10^{-5}, \qquad \rho^\Pi = 5 \times 10^{-6}, \qquad \rho^\Sigma = 10^{-6}.$$

Algorithm 5 Finite Horizon Mean Field Actor–Critic

```
Let $V_{\theta_0} \in \mathcal{NN}^{\varrho}_{2,1,\ell,m}$, $\Pi_{\psi_0} \in \mathcal{NN}^{\varrho}_{2,1,\ell,m}$ and $\Sigma_{\varphi_0} \in \mathcal{NN}^{\varrho}_{2,1,\ell,m}$; number of mean field sample $m$; number of episode $K$; learning rates $\rho^V$, $\rho^\Pi$ and $\rho^\Sigma$.
for $k = 0, \dots, K-1$ do
    Agent receives initial state $X_0 \sim \xi$ from the environment
    for $n = 0, \dots, N_T - 1$ do
        Agent sample action $A_{t_n} \sim \Pi_\psi(\cdot \,|\, t_n, X_{t_n})$
        Environment compute score loss $L_\Sigma^{(n)}(\varphi_k)$
        Environment set $\varphi_{k+1} = \varphi_k - \rho^\Sigma \nabla_\varphi L_\Sigma$
        Environment generate mean field samples $U_{t_n} = \left(U_{t_n}^{(1)}, \dots, U_{t_n}^{(m)}\right)$ from $\Sigma_{\varphi_k}$ using Langevin Monte Carlo and set $\bar{\nu}_{t_n} = \frac{1}{m}\sum_{i=1}^m \delta_{U_{t_n}^{(i)}}$
        Agent observes next state $X_{t_{n+1}}$ and cost $F(X_{t_n}, A_{t_n}, \bar{\nu}_{t_n})$ generated from the environment based on its knowledge of the dynamics and of $\bar{\nu}_{t_n}$.
        Agent compute critic loss $L_V^{(n)}(\theta_k)$
        Agent set $\theta_{k+1} = \theta_k - \rho^V \nabla_\theta L_V$
        Agent compute actor loss $L_\Pi^{(n)}(\psi_k)$
        Agent set $\psi_{k+1} = \psi_k - \rho^\Pi \nabla_\psi L_\Pi$
    end for
end for
```

Our results, illustrated in Figures 2.7 and 2.8, highlight significant limitations of the actor–critic method in this finite-horizon setting. First, the training curves for the actor, critic, and score exhibit high variance across epochs and do not display a clear decay. In particular, the score loss stabilizes at a relatively large value, suggesting that the approximation of the score function remains inaccurate. Second, when examining the learned value function and the associated optimal control, the results are clearly unsatisfactory. The value function departs noticeably from the true solution, while the learned policy fails almost completely to approximate the optimal control. In particular, the control appears nearly flat instead of linear, as would be expected in this framework. Moreover, the training process itself is very long, further limiting the practicality of this approach.

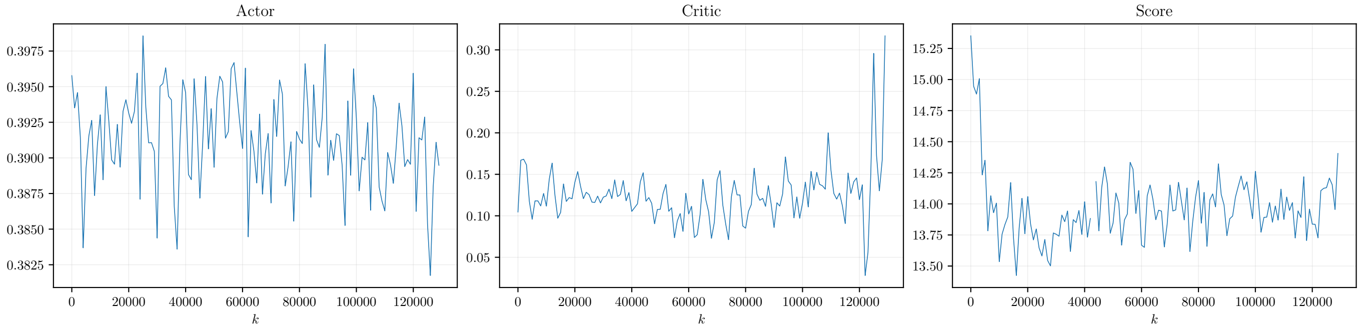

Figure 2.7: Training results using Algorithm 2.4.2 with parameters in Table 2.1. We plot the evolution of the L^2 errors of both the actor, the critic and the score.

By contrast, the model–based methods presented earlier, such as the direct MFC approach or the Deep BSDE method, demonstrated both faster convergence and more accurate approximations of the true solution. This comparison emphasizes the current weakness of actor–critic methods in finite-horizon mean field problems, where instability of training and poor control approximation remain major obstacles. Overall, these experiments suggest that further refinements whether in the choice of network architectures, in the tuning of learning rates, or in the design of the overall training procedure are necessary before actor–critic methods can compete with more established model–based approaches in this context.

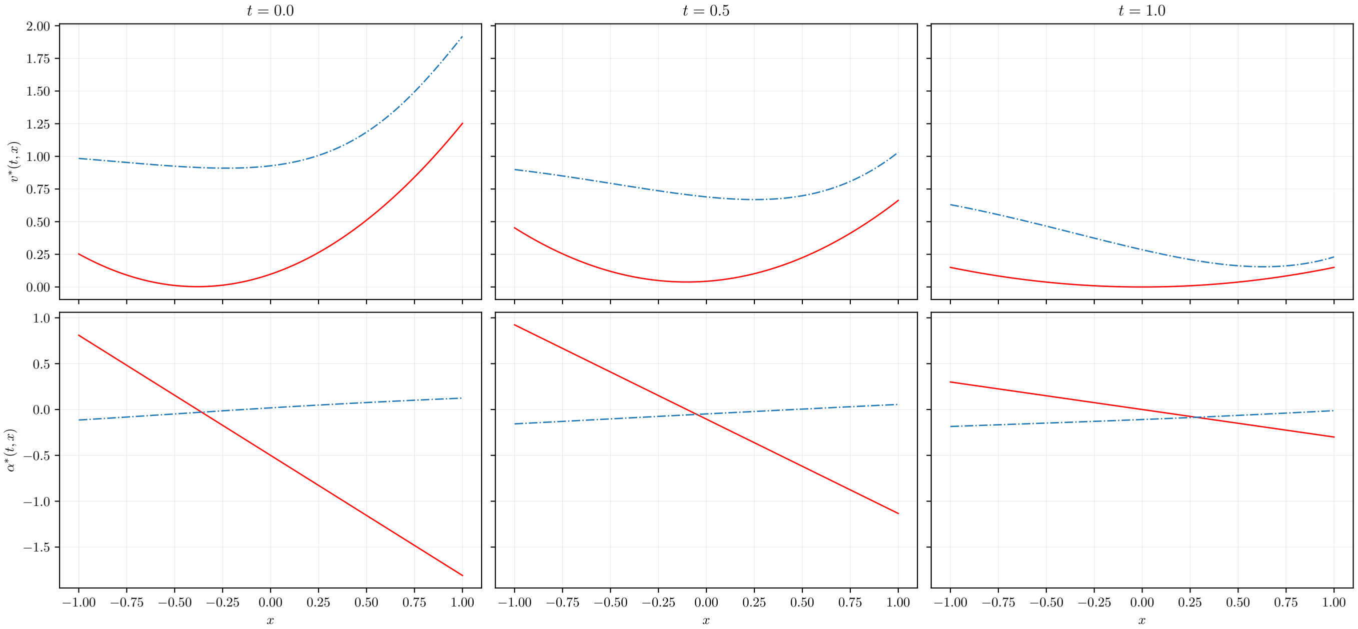

Figure 2.8: Results using Algorithm 2.4.2 with parameters in Table 2.1. At different time points $t \in \{0.0, 0.5, 1.0\}$, we plot the learned optimal value function (dotted lines) and the exact solution (solid lines) on top. At the bottom we plot the learned optimal control function (dotted lines) and the exact solution (solid lines).

Conclusion & Discussions

In this memoir, we studied an optimal execution problem within the framework of Mean Field Games. Our analysis combined both mathematical and economic interpretation, providing a comprehensive perspective on the model. We examined the problem through two complementary approaches: an analytic one based on PDE methods and a probabilistic one based on FBSDEs. On the computational side, we implemented and analyzed several numerical methods based on deep learning for solving mean field problems, and applied them to the optimal execution setting. Among the model–based approaches, the first two probabilistic methods yielded reliable results, whereas the Deep Galerkin Method proved more delicate to train, with convergence often hindered by local minima. We also extended the mean field actor–critic algorithm, originally developed for infinite horizon settings, to the finite horizon case. This adaptation introduces significant challenges due to the explicit time dependence of the problem, making the learning dynamics considerably more complex.

Future research could investigate several directions. From a modeling perspective, natural extensions include incorporating common noise or studying Mean Field Control Games, which combine competitive and cooperative features across groups of agents and offer meaningful interpretations in quantitative finance. From a numerical perspective, further work is required to better understand and improve the convergence properties of deep learning methods in the mean field setting, particularly reinforcement learning approaches, for which rigorous guarantees are still lacking.

References

Acciaio, B., Backhoff-Veraguas, J., and Carmona, R. (2018). Extended mean field control problems: stochastic maximum principle and transport perspective.

Al-Aradi, A., Correia, A., Naiff, D., Jardim, G., and Saporito, Y. (2018). Solving nonlinear and high-dimensional partial differential equations via deep learning.

Almgren, R. and Chriss, N. A. (2000). Optimal execution of portfolio trans-actions.

Andrea Angiuli, A. A., Jean-Pierre Fouque, J.-P. F., Ruimeng Hu, R. H., and Alan Raydan, A. R. (2025). Deep reinforcement learning for infinite horizon mean field problems in continuous spaces. *Journal of Machine Learning*, 4(1):11–47.

Angiuli, A., Fouque, J.-P., and Lauriere, M. (2021). Reinforcement learning for mean field games, with applications to economics.

Cardaliaguet, P. and Lehalle, C.-A. (2017). Mean field game of controls and an application to trade crowding.

Carmona, R. and Delarue, F. (2018a). *Probabilistic Theory of Mean Field Games with Applications I.* Springer.

Carmona, R. and Delarue, F. (2018b). *Probabilistic Theory of Mean Field Games with Applications II.* Springer.

Carmona, R. and Lacker, D. (2015). A probabilistic weak formulation of mean field games and applications. *The Annals of Applied Probability*, 25(3).

Carmona, R. and Laurière, M. (2021a). Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games: I – the ergodic case.

Carmona, R. and Laurière, M. (2021b). Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games: II – the finite horizon case.

Carmona, R. and Laurière, M. (2021c). Deep learning for mean field games and mean field control with applications to finance.

Carmona, R. and Leal, L. (2021). Optimal execution with quadratic variation inventories.

Carmona, R. and Webster, K. (2013). The self-financing equation in high frequency markets.

Cartea, Á., Donnelly, R. F., and Jaimungal, S. (2013). Algorithmic trading with model uncertainty. *SIAM Journal on Financial Mathematics.*

Cartea, Á. and Jaimungal, S. (2016). Incorporating order-flow into optimal execution. *SSRN Electronic Journal.*

Cartea, Á., Jaimungal, S., and Penalva, J. S. (2015). *Algorithmic and High-Frequency Trading.* Cambridge University Press.

E, W., Han, J., and Jentzen, A. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380.

Germain, M., Mikael, J., and Warin, X. (2022). Numerical resolution of mckean-vlasov fbsdes using neural networks.

Han, J. and Long, J. (2020). Convergence of the deep bsde method for coupled fbsdes. *Probability, Uncertainty and Quantitative Risk*, 5(1).

Lasry, J.-M. and Lions, P.-L. (2007). Mean field games. *Japanese Journal of Mathematics*, 2:229–260.

Pippas, N., Ludvig, E. A., and Turkay, C. (2025). The evolution of reinforcement learning in quantitative finance: A survey. *ACM Computing Surveys*, 57(11):1–51.

Sirignano, J. and Spiliopoulos, K. (2018). Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364.