2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Machine Learning architectures for price formation models with common noise

Diogo Gomes[1], Julian Gutierrez[1], and Mathieu Laurière[2]

***Abstract*— We propose a machine learning method to solve a mean-field game price formation model with common noise. This involves determining the price of a commodity traded among rational agents subject to a market clearing condition imposed by random supply, which presents additional challenges compared to the deterministic counterpart. Our approach uses a dual recurrent neural network encoding noise dependence and a particle approximation of the mean-field model with a single loss function optimized by adversarial training. We provide a posteriori estimates for convergence and illustrate our method through numerical experiments.**

I. Introduction

This work extends machine learning (ML) techniques for numerical solutions of mean-field games (MFGs) price formation models ([11]) to the common noise model from [12] (also [13]). Our goal is to determine the price ϖ of a commodity with a noisy supply Q, traded among rational agents within a finite time horizon $T > 0$, under a market-clearing condition. More precisely, we assume the supply, Q, satisfies the following stochastic differential equation (SDE)

$$\begin{cases} \mathrm{d}Q(t) = b^S(Q(t), t)\mathrm{d}t + \sigma^S(Q(t), t)\mathrm{d}W(t), \\ Q(0) = q_0 \end{cases} \tag{1}$$

where $q_0 \in \mathbb{R}$ and W, the common noise, is a one-dimensional Brownian motion. The coefficients b^S and σ^S satisfy Lipschitz conditions for existence and uniqueness of solutions (see [10]). Because of (1), our model explains the price formation for commodities with continuous fluctuations, such as stocks, bonds, currencies, and continuously produced or consumed goods, such as oil or natural gas. Additional noise sources, such as sudden and discontinuous fluctuations, could be modeled by adding Poisson jumps.

Let $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$ be a complete filtered probability space supporting W. Progressive measurability refers to the measurability with respect to this filtration, which we require for all stochastic processes. Setting a commodity's price based on market interactions defines the price formation problem. MFGs with common noise characterize the price as the solution of the following.

Problem 1: Suppose that $H : \mathbb{R}^2 \to \mathbb{R}$ is uniformly convex and differentiable in the second argument, m_0 is a probability measure on $\mathbb{R}$, and $u_T : \mathbb{R} \to \mathbb{R}$ is uniformly convex and differentiable. Find $m : [0,T] \times \mathbb{R} \to \mathbb{R}$, $u, Z : [0,T] \times \mathbb{R} \times \Omega \to \mathbb{R}$, and $\varpi : [0,T] \times \Omega \to \mathbb{R}$ progressively measurable, satisfying $m \geqslant 0$ and

$$\begin{cases} -\mathrm{d}u + H(x, \varpi + u_x)\mathrm{d}t = Z(t,x)\mathrm{d}W(t), \\ u(T,x) = u_T(x), \\ m_t - \left(H_p(x, \varpi + u_x)m\right)_x = 0, \\ m(0,x) = m_0(x), \\ -\int_{\mathbb{R}} H_p(x, \varpi + u_x)m\mathrm{d}x = Q(t). \end{cases} \tag{2}$$

The previous problem generalizes the one in [14], which corresponds to $\sigma^S = 0$. (1) determines Q in (2). As detailed in Section II, the price formed in (2) guarantees a minimal cost for all traders under a market-clearing condition. The numerical solution of (2) presents additional challenges compared to the deterministic counterpart due to infinite-dimensional state space. [13] computed a semi-explicit solution for (2) when b^S and σ^S are linear and H is quadratic. Section II presents the derivation of (2).

Numerical schemes for MFGs without common noise include Fourier series [21], semi-Lagrangian schemes [5], fictitious play [15], and variational methods [3]. [9] proposes an ML-based approach to solve bi-level Stackelberg problems between a principal and a mean field of agents by reformulating it as a single-level mean-field optimal control problem. [18] and [8] survey deep learning and reinforcement learning methods for MFGs and mean-field control. However, these methods fail to handle general forms of common noise, as the state space becomes infinite-dimensional. Recent works address this issue. [2] reduces continuous-time mean field games with finitely many states and common noise to a system of forward-backward systems of (random) ordinary differential equations. [20] used rough path theory and deep learning techniques. However, the common noise price formation problem's coupling is an integral constraint in an infinite dimensional space, which is beyond what standard methods can handle. In [1], the price formation model with common noise was converted into a convex variational problem and solved using ML, enforcing constraints by penalization. However, this approach introduces numerical instabilities. In contrast, our method implements the balance constraint in the loss functional through a Lagrange multiplier.

We use recurrent neural networks (RNNs) to approximate ϖ and the optimal vector field. We use adversarial training based on a particle approximation and an augmented Lagrangian-derived loss function. [22] used an augmented Lagrangian for constrained empirical risk minimization. [23],

Keywords: Mean Field Games; Price formation; Neural Networks

[1]King Abdullah University of Science and Technology (KAUST), CEMSE Division, Thuwal 23955-6900. Saudi Arabia diogo.gomes@kaust.edu.sa julian.gutierrezpineda@kaust.edu.sa

[2]New York University Shanghai (NYU Shanghai), Shanghai 200122. China ml5197@nyu.edu

[19], [4], and [20] explored ML methods in MFGs and relied on numerical tests for validation. In contrast, we offer analytical convergence guarantees. Although ML convergence for MFGs has been studied in [6], [7], direct application to our problem is hindered by the integral constraint. Convergence proofs of ML algorithms are difficult due to the complexity of models and optimization problems involved. Therefore, we develop a posteriori estimates to confirm the convergence. Given $f : [0,T] \times \Omega$, we write $\|f\| = \left(\mathbb{E}\left[\|f\|^2_{L^2([0,T]}\right]\right)^{1/2}$. Section II introduces the N-player problem corresponding to Problem 1 with state and adjoint variables $\mathbf{X}$ and $\mathbf{P}$. Section III presents additional notation and proves the following main result using the N-player problem.

Theorem 1: Suppose that H is uniformly concave-convex in (x,p), separable, with Lipschitz continuous derivatives, and u_T is convex with Lipschitz continuous derivative. Let $(\mathbf{X},\mathbf{P})$ and ϖ^N solve the N-player price formation problem with common noise, and let $(\tilde{\mathbf{X}},\tilde{\mathbf{P}})$ and $\tilde{\varpi}^N$ be an approximate solution to the N-player problem up to the error terms ϵ_H and ϵ_B. Then, there exists $C > 0$, depending on problem data, such that $\|\varpi^N - \tilde{\varpi}^N\| \leqslant C\Big(\|\epsilon_H\| + \|\epsilon_B\|\Big)$.

Here, ϵ_H and ϵ_B correspond to the deviation of $(\tilde{\mathbf{X}},\tilde{\mathbf{P}})$ from a first-order optimality condition and are given in Section III. Section IV presents our algorithm and Section V outlines the numerical results for the linear-quadratic setting. We stress, however, that our method can handle non linear-quadratic models. Moreover, the ML is well suited for higher-dimensional state spaces, where, for instance, several commodities are priced simultaneously. Section VI includes concluding remarks and future research directions.

II. The MFG price problem with common noise

Price formation is crucial in economics. For instance, in smart grids, load-adaptive pricing encourages consumers to modify energy use according to electricity price fluctuations. MFGs offer a mathematical model to examine interactions among market agents, such as buyers and producers. These interactions are formalized in Problem 1.

A representative player with an initial quantity $x_0 \in \mathbb{R}$ at time $t = 0$ selects a progressively measurable trading rate $v : [0,T] \times \Omega \to \mathbb{R}$ to minimize the cost functional

$$\mathbb{E}\left[\int_0^T \left(L(X(t),v(t)) + \varpi(t)v(t)\right)\mathrm{d}t + u_T\left(X(T)\right)\right], \tag{3}$$

where X solves

$$\mathrm{d}X(t) = v(t)\mathrm{d}t, \quad X(0) = x_0, \tag{4}$$

with $x_0 \sim m_0$. The Hamiltonian H in (2) is the Legendre transform of the Lagrangian L in (3): $H(x,p) = \sup_{v\in\mathbb{R}}\{-pv - L(x,v)\}, \quad (x,p) \in \mathbb{R}^2$. Here, we assume that H satisfies the assumptions of Theorem 1, and that $L(x,v)$ is uniformly convex in v for all $x \in \mathbb{R}$. The value function for representative agent is

$$u(t,x) = \mathbb{E}\left[\int_t^T \left(L(X(s),v(s)) + \varpi(s)v(s)\right)\mathrm{d}s | W(t)\right] + \mathbb{E}\left[u_T\left(X(T)\right) | W(t)\right].$$

Then, the first equation in (2) holds. The initial distribution is $m_0 \in \mathcal{P}(\mathbb{R})$ and it evolves according to the stochastic flow given by $v^*(t,x) = -H_p(x,\varpi(t) + u_x(t,x))$, the third equation in (2). The last equation is the market-clearing condition $\int_{\mathbb{R}} v(t,x)m\mathrm{d}x = Q(t)$, ensuring that all available supply is traded. The MFG's approach gives $v^* = -H_p(x,\varpi + u_x)$, which is the optimal control for all players to minimize (3) and simultaneously guarantee that the aggregated trading equals the supply, which is precisely the balance condition in (2). [1] discusses further details. In our method, we approximate ϖ, which allows the decoupling of the equations in (2).

The particle approximation of the model described by (2) involves Q in (1) and a finite population of N players with independent, identically distributed initial positions $x_0^n \in \mathbb{R}$, $n = 1,\dots,N$, with distribution m_0. Each player selects $v^n : [0,T] \times \mathbb{R} \times \Omega \to \mathbb{R}$, $1 \leqslant n \leqslant N$, determining its trajectory X^n according to (4) and aiming at minimizing the functional

$$\mathbb{E}\left[\int_0^T \left(L(X^n(t),v^n(t)) + \varpi^N(t)\left(v^n(t) - Q(t)\right)\right)\mathrm{d}t\right] + \mathbb{E}\left[u_T\left(X^n(T)\right)\right]. \tag{5}$$

The existence of v^{*n} minimizing (5) for $1 \leqslant n \leqslant N$ corresponds to the existence of a minimizer $\mathbf{v}^* = (v^{*1},\dots,v^{*N})$ minimizing the functional

$$\mathbb{E}\left[\frac{1}{N}\sum_{n=1}^N \int_0^T L(X^n(t),v^n(t))\mathrm{d}t + u_T\left(X^n(T)\right)\right] \tag{6}$$

subject to the market-clearing constraint

$$\frac{1}{N}\sum_{n=1}^N v^n(t) - Q(t) = 0. \tag{7}$$

The impact of one player's strategy on another's payoff is given solely by the price in (5). The same applies to (6), where the effects become implicit because any unilateral change in (7) immediately influences the choices of other players and, consequently, their payoff. The linear dependence of (7) on the player's controls is essential for the analysis in Section III, and it corresponds to the particle approximation of the balance condition in (2). The supply function implicitly affects the cost criteria of the agents. The cost functionals in (5) and (6) encode the supply effect on an agent that interacts with the market using the price instead of its supply, which is often the case in markets. As $N \to \infty$, the solution of (5) converges to that of (2).

We rely on the existence and uniqueness result for the N-player price formation model, presented in [13], which determines the price $\varpi^{*N} : [0,T] \times \Omega \to \mathbb{R}$ through the Lagrange multiplier associated with the market-clearing

constraint. ϖ^{*N} is the unique price allowing agents to minimize (5) while satisfying (7). Our goal is to extend the ML algorithm introduced in [11] to cover the case of common noise, providing a solution to the price formation problem in random environments. Relying on the particle approximation of the model to approximate the price solving (2), we approximate stationary points of the functional

$$\mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N}\int_0^T \big(L(X^n(t),v^n(t)) + \varpi^N(t)\left(v^n(t)-Q(t)\right)\big)dt + u_T\left(X^n(T)\right)\right] \quad (8)$$

by minimizing w.r.t. $\mathbf{v}$ and maximizing w.r.t. ϖ^N. The min-max problem corresponds to considering (8) as the augmented-Lagrangian of (6) under the constraint (7), ϖ^N begin the Lagrange multiplier of such constraint. In Section IV-B, adversarial training involves two RNNs–one maximizing, and the other minimizing the same objective functional. The approximation is done in the ML framework, and we guarantee its accuracy using a posteriori estimates of the N-player model, which we discuss next.

III. A posteriori estimates

In this section, we use optimality conditions for the N-player game to obtain a posteriori estimates to verify our approximation's convergence. We extend the proof presented in [11] to the common noise setting with minor modifications.

The optimality conditions for (5) give rise to a Hamiltonian system comprising the following backward-forward stochastic differential equation

$$\begin{cases} \mathrm{d}P^n(t) = H_x(X^n(t),P^n(t)+\varpi^N(t))\mathrm{d}t \\ \qquad\qquad +Z^n(t)\mathrm{d}W(t), \\ \mathrm{d}X^n(t) = -H_p(X^n(t),P^n(t)+\varpi^N(t))\mathrm{d}t, \end{cases} \quad (9)$$

$P^n(T) = u_T'(X^n(T))$, and $X^n(0) = x_0^n$ for $1 \leqslant n \leqslant N$.$X^n$ corresponds to the state variable of agent n, while P^n is its associated adjoint variable. Z^n is part of the unknowns and guarantees that P^n is progressively measurable despite evolving backward in time. $\mathbf{v}^*$ and ϖ^{*N} solving the N-player price formation problem define a solution of (9) by

$$P^{*n}(t) + L_v(X^{*n}(t),v^{*n}(t)) + \varpi^{*N}(t) = 0 \quad (10)$$

for $1 \leqslant n \leqslant N$, defining a saddle point of (8) that satisfies the market-clearing constraint (7). Let $\tilde{P}^n$, $\tilde{Z}^n$, $\tilde{X}^n$, and $\tilde{\varpi}^N$ satisfy

$$\begin{cases} \mathrm{d}\tilde{P}^n(t) = \left(H_x(\tilde{X}^n(t),\tilde{P}^n(t)+\varpi^N(t)) + \epsilon^n(t)\right)\mathrm{d}t \\ \qquad\qquad +\tilde{Z}^n(t)\mathrm{d}W(t), \\ \mathrm{d}\tilde{X}^n(t) = -H_p(\tilde{X}^n(t),\tilde{P}^n(t)+\tilde{\varpi}^N(t))\mathrm{d}t, \\ \frac{1}{N}\sum\limits_{n=1}^{N} -H_p(\tilde{X}^n(t),\tilde{P}^n(t)+\tilde{\varpi}^N(t)) = Q(t)+\epsilon_B(t), \end{cases} \quad (11)$$

where $\tilde{P}^n(T) = u_T'(\tilde{X}^n(T)) - \epsilon_T^n$, $\tilde{X}^n(0) = x_0^n$, $\epsilon^n, \epsilon_B : [0,T]\times\Omega \to \mathbb{R}$, $\epsilon_T^n : \Omega \to \mathbb{R}$, for $1 \leqslant n \leqslant N$. We write $\mathbf{X} = (X^1,\dots,X^N)$ and, similarly, for all indexed processes. We let $\epsilon_H = (\epsilon^1,\dots,\epsilon^N,\epsilon_T^1,\dots,\epsilon_T^N)$ and $\mathbb{1} = (1,\dots,1) \in \mathbb{R}^N$.

Proposition 2: Under the assumptions of Theorem 1, let $(\mathbf{X},\mathbf{P},\mathbf{Z})$ and ϖ^N solve (9), and let $(\tilde{\mathbf{X}},\tilde{\mathbf{P}},\tilde{\mathbf{Z}})$, $\tilde{\varpi}^N$, ϵ_H, and ϵ_B satisfy (11). Then,

$$\|P^n-\tilde{P}^n\|^2 \leqslant C\left(\|X^n(T)-\tilde{X}^n(T)\|^2 + \|X^n-\tilde{X}^n\|^2 + \|\epsilon_T^n\|^2 + \|\epsilon^n\|^2\right),$$

for $1 \leqslant n \leqslant N$, where $C > 0$ depends on problem data.

Proof: Integrating on $[t,T]$ the first equations in (9) and (11), taking expectations, and using the martingale property of the processes Z^n and $\tilde{Z}^n$, we have the estimate

$$\mathbb{E}\left[(P^n-\tilde{P}^n)^2\right] \leqslant C\mathbb{E}\left[(X^n(T)-\tilde{X}^n(T))^2 + (X^n-\tilde{X}^n)^2 + (\epsilon_T^n)^2 + (\epsilon^n)^2\right]$$

for $1 \leqslant n \leqslant N$. Integrating the previous inequality over $[0,T]$ we obtain the result. ∎

Proposition 3: Under the assumptions of Theorem 1, let $(\mathbf{X},\mathbf{P},\mathbf{Z})$ and ϖ^N solve (9), and let $(\tilde{\mathbf{X}},\tilde{\mathbf{P}},\tilde{\mathbf{Z}})$, $\tilde{\varpi}^N$, ϵ_H, and ϵ_B satisfy (11). Then, $\|\mathbf{P}+\mathbb{1}\varpi^N-(\tilde{\mathbf{P}}+\mathbb{1}\tilde{\varpi}^N)\|^2 + \|\mathbf{X}-\tilde{\mathbf{X}}\|^2 \leqslant C\left(\|\epsilon_H\|^2 + +\|\epsilon_B\|^2\right)$, where $C > 0$ depends on problem data.

Proof: We write $\|\cdot\|_2 = \|\cdot\|_{L^2([0,T])}$. The uniform concavity-convexity assumption on H and the equations in (9) and (11) give

$$\begin{aligned} &\tfrac{\gamma_p}{2}\|P^n+\varpi^N-(\tilde{P}^n+\tilde{\varpi}^N)\|_2^2 + \tfrac{\gamma_x}{2}\|X^n-\tilde{X}^n\|_2^2 \\ &\leqslant \int_0^T \Big(\mathrm{d}\left(\tilde{X}^n-X^n\right)(P^n-\tilde{P}^n) \\ &\qquad -\mathrm{d}\left(X^n-\tilde{X}^n\right)(\varpi^N-\tilde{\varpi}^N) \\ &\qquad -\left(\mathrm{d}(P^n-\tilde{P}^n)+\epsilon^n\mathrm{d}t\right)(X^n-\tilde{X}^n) \\ &\qquad +(Z^n-\tilde{Z}^n)\mathrm{d}W(t)\Big) \end{aligned}$$

for $1 \leqslant n \leqslant N$, for some $\gamma_p,\gamma_x > 0$. Using Itô product rule, the initial and terminal conditions in (9) and (11), and the convexity of u_T, the previous inequality gives

$$\begin{aligned} &\tfrac{\gamma_p}{2}\|P^n+\varpi^N-(\tilde{P}^n+\tilde{\varpi}^N)\|_2^2 + \tfrac{\gamma_x}{2}\|X^n-\tilde{X}^n\|_2^2 \\ &\leqslant \left(-\tfrac{\gamma_T}{2}+\tfrac{1}{4\delta_1}\right)(X^n(T)-\tilde{X}^n(T))^2 + \delta_1(\epsilon_T^n)^2 \\ &\quad -\int_0^T \Big(\mathrm{d}\left(X^n-\tilde{X}^n\right)(\varpi^N-\tilde{\varpi}^N) + \epsilon^n(X^n-\tilde{X}^n)\mathrm{d}t \\ &\quad +(Z^n-\tilde{Z}^n)\mathrm{d}W(t)\Big) \end{aligned}$$

for some $\gamma_T > 0$ and $\delta_1 > 0$ to be selected. Adding the previous inequality over n and using the third equation in (9) and (11), we get

$$\begin{aligned} &\tfrac{\gamma_p}{2}\|\mathbf{P}+\mathbb{1}\varpi^N-(\tilde{\mathbf{P}}+\mathbb{1}\tilde{\varpi}^N)\|_2^2 + \tfrac{\gamma_x}{2}\|\mathbf{X}-\tilde{\mathbf{X}}\|_2^2 \\ &\leqslant \left(-\tfrac{\gamma_T}{2}+\tfrac{1}{4\delta_1}\right)|\mathbf{X}(T)-\tilde{\mathbf{X}}(T)|^2 + \delta_1|\epsilon_T|^2 \\ &\quad +\int_0^T \Big(N\epsilon_B(\varpi^N-\tilde{\varpi}^N)\mathrm{d}t + (\mathbf{Z}-\tilde{\mathbf{Z}})\mathrm{d}W(t)\Big) \\ &\quad +\delta_2\|\boldsymbol{\epsilon}\|_2^2 + \tfrac{1}{4\delta_2}\|\mathbf{X}-\tilde{\mathbf{X}}\|_2^2 \end{aligned} \quad (12)$$

for some $\delta_2 > 0$ to be selected. By the triangle inequality,

$$\frac{N}{2}\|\varpi^N-\tilde{\varpi}^N\|^2 \leqslant \|\mathbf{P}+\mathbb{1}\varpi^N-(\tilde{\mathbf{P}}+\mathbb{1}\tilde{\varpi}^N)\|^2 + \|\mathbf{P}-\tilde{\mathbf{P}}\|^2. \quad (13)$$

Using (13) on the RHS of (12), taking expectations, and using Proposition 2, we obtain

$$\begin{aligned}
&\tfrac{\gamma_p}{2}\|\mathbf{P}+\mathbb{1}\varpi^N-(\tilde{\mathbf{P}}+\mathbb{1}\tilde{\varpi}^N)\|^2+\tfrac{\gamma_x}{2}\|\mathbf{X}-\tilde{\mathbf{X}}\|^2\\
&\leqslant\left(-\tfrac{\gamma_T}{2}+\tfrac{1}{4\delta_1}+\tfrac{C}{\delta_4}\right)\|\mathbf{X}(T)-\tilde{\mathbf{X}}(T)\|^2\\
&\quad+\left(\delta_1+\tfrac{C}{\delta_4}\right)\|\epsilon_T\|^2+(N\delta_3+N\delta_4)\|\epsilon_B\|^2\\
&\quad+\tfrac{1}{4\delta_3}\|\mathbf{P}+\mathbb{1}\varpi^N-(\tilde{P}+\mathbb{1}\tilde{\varpi}^N)\|^2\\
&\quad+\left(\delta_2+\tfrac{C}{\delta_4}\right)\|\boldsymbol{\epsilon}\|^2+\left(\tfrac{1}{4\delta_2}+\tfrac{C}{\delta_4}\right)\|\mathbf{X}-\tilde{\mathbf{X}}\|^2
\end{aligned}$$

for some $\delta_3,\delta_4>0$ to be selected. Selecting δ_i, $i=1,\dots,4$ conveniently, the previous expression provides the result. ■

Proof: [Proof of Theorem 1] Using (13), Proposition 2, and Proposition 3, we get

$$\|\varpi^N-\tilde{\varpi}^N\|^2\leqslant\frac{C}{N}\left(\|\epsilon_H\|^2+\|\epsilon_B\|^2+\|\mathbf{X}(T)-\tilde{\mathbf{X}}(T)\|^2\right). \tag{14}$$

The Lipstichz continuity of H_p and Proposition 2 give

$$\|\mathbf{X}(T)-\tilde{\mathbf{X}}(T)\|^2\leqslant C\left(\|\epsilon_H\|^2+\|\epsilon_B\|^2\right). \tag{15}$$

Using (15) in (14), we obtain the result. ■

IV. Neural Networks for progressively measurable processes

This section explains the RNN used for estimating v^* and ϖ, and describes how the ideas in Section III are implemented. Section V presents numerical experiments. RNNs, popular in natural language processing, produce sequentially dependent outputs. This architecture has a cell that iterates through input sequences and has a hidden state tracking historical dependencies; see [17] for details. RNNs have also been used in the context of control problems with delay in [16]. Here our motivation comes from the impact of the common noise on the mean-field.

In our architecture, the RNN takes as inputs an ordered sequence, such as a discrete realization of the supply $\mathrm{Q}=\left(\mathrm{Q}^{\langle 0\rangle},\dots,\mathrm{Q}^{\langle K\rangle}\right)$. The RNN features a hidden state h, initialized as zero, that captures the temporal dependence. Inside the RNN, a weight matrix $\mathrm{W}^{[l]}$ and a bias vector $\mathrm{b}^{[l]}$ determine layer l, where $1\leqslant l\leqslant L$. Their dimensions depend on the number of neurons $n^{[l]}$ per layer. The activation function of layer j is denoted by $\sigma^{[l]}$. The cell parameters (weight matrices and bias vectors) are denoted by Θ.

RNNs parametrize progressively measurable processes. We use two RNNs for approximating the control variable v and the price ϖ. A trade-off must be made between computational cost and accuracy. Deep-RNN uses several layers and neurons in their cell. After multiple numerical experiments, we select $L=5$ layers, with $n^{[1]}=16$, $n^{[2]},n^{[3]},n^{[4]}=32$, and $n^{[5]}=1$ for the RNN approximating $\mathrm{v}^{\langle k\rangle}$, and $n^{[1]},n^{[2]},n^{[3]},n^{[4]}=16$, and $n^{[5]}=1$ for the RNN approximating ϖ. Further research is needed to determine which RNN architecture excels for specific modeling parameters. As a common practice for RNN, the activation functions are hyperbolic tangent for the first layer, which computes the hidden state, and sigmoid for layers two to four. The last layer has the identity as activation function. Although we do not address it, an interesting research question is how sensitive the results are to the choice of parameters of the RNN. Moreover, a comparison regarding the accuracy and computational efficiency against other methods, such as forward-backward SDEs methods, can be formulated based on the adaptability of those methods to the price formation MFG problem with common noise.

Let $\Delta t=T/K$ be the time-step size and discretize (4) as

$$X^{\langle k+1\rangle}=X^{\langle k\rangle}+\mathrm{v}^{\langle k\rangle}(\Theta_v)\Delta t,\quad X^{\langle 0\rangle}=x_0 \tag{16}$$

for $k=0,\dots,K$, where Θ_v is the parameter of the RNN approximating v. The second RNN, with parameter Θ_ϖ, computes $\varpi^{\langle k\rangle}$. More precisely, the inputs and outputs of the two RNNs are as follows. For the RNN computing $\varpi(\Theta_\varpi)$, the input consists of a supply realization and the time; that is, $\left((Q^{\langle k\rangle})_{k=0}^K,\quad (t^{\langle k\rangle})_{k=0}^K\right)$. The output is $(\varpi^{\langle k\rangle})_{k=0}^K$. For the RNN computing $v(\Theta_v)$, the input consists of the time, the state variables (which the RNN updates according to (16) as it iterates in the temporal direction), and the current price approximation; that is, $\left((t^{\langle k\rangle})_{k=0}^K,(X^{\langle k\rangle})_{k=0}^K,(\varpi^{\langle k\rangle})_{k=0}^K,\right)$. The output is $(\mathrm{v}^{\langle k\rangle})_{k=0}^K$. Because we consider a population of N agents, we add the superscript (n) to denote the position and control sequence of the agent being considered; that is, $(X^{(n)\langle k\rangle})_{k=0}^K$, and $(\mathrm{v}^{(n)\langle k\rangle})_{k=0}^K$, for $1\leqslant n\leqslant N$.

A. Numerical implementation of a posteriori estimates

To implement the results of Section III, we discretize (11) as follows. Let $\Delta\tilde{P}^{\langle k\rangle}=\tilde{P}^{\langle k+1\rangle}-\tilde{P}^{\langle k\rangle}$ for $k=0,\dots,K-1$. At the discrete level, (11) is equivalent to

$$\begin{cases}
\Delta\tilde{P}^{(n)\langle k\rangle}=\left(H_x(\tilde{X}^{(n)\langle k\rangle},\tilde{P}^{(n)\langle k\rangle}+\tilde{\varpi}^{N\langle k\rangle})+\epsilon^{(n)\langle k\rangle}\right)\Delta t\\
\quad+Z^{(n)\langle k\rangle}\Delta W^{\langle k\rangle},\\
\Delta\tilde{X}^{(n)\langle k\rangle}=-H_p(\tilde{X}^{(n)\langle k\rangle},\tilde{P}^{(n)\langle k\rangle}+\tilde{\varpi}^{N\langle k\rangle})\Delta t,\\
\frac{1}{N}\sum_{n=1}^N-H_p(\tilde{X}^{(n)\langle k\rangle},\tilde{P}^{(n)\langle k\rangle}+\tilde{\varpi}^{N\langle k\rangle})=Q^{\langle k\rangle}+\epsilon_B^{\langle k\rangle}
\end{cases}$$

$\tilde{P}^{(n)\langle K\rangle}=u_T'(\tilde{X}^{(n)\langle K\rangle})-\epsilon_T^{(n)}$, $\tilde{X}^{(n)\langle 0\rangle}=x_0^n$ for $1\leqslant n\leqslant N$. $H_x(x,p)=-L_x(x,v)$ at the point v where the supremum is achieved. Thus, taking expectations on both sides of the equation in the prior system, and using the martingale property for $\tilde{Z}^n$, for $1\leqslant n\leqslant N$, we get

$$\begin{cases}
\mathbb{E}\left[\Delta\tilde{P}^{(n)\langle k\rangle}\right]=\mathbb{E}\left[-L_x(\tilde{X}^{(n)\langle k\rangle},\tilde{v}^{(n)\langle k\rangle})+\epsilon^{(n)\langle k\rangle}\right]\Delta t\\
\mathbb{E}\left[\Delta\tilde{X}^{(n)\langle k\rangle}\right]=\mathbb{E}\left[\tilde{v}^{(n)\langle k\rangle}\right]\Delta t,\\
\mathbb{E}\left[\frac{1}{N}\sum_{n=1}^N\tilde{v}^{(n)\langle k\rangle}\right]=\mathbb{E}\left[Q^{\langle k\rangle}+\epsilon_B^{\langle k\rangle}\right],
\end{cases} \tag{17}$$

$\mathbb{E}\left[\tilde{P}^{(n)\langle K\rangle}\right]=\mathbb{E}\left[u_T'(\tilde{X}^{(n)\langle K\rangle})-\epsilon_T^{(n)}\right]$, and $\tilde{X}^{(n)\langle 0\rangle}=x_0^n$, where $\tilde{v}^{(n)\langle k\rangle}=-H_p(\tilde{X}^{(n)\langle k\rangle},\tilde{P}^{(n)\langle k\rangle}+\tilde{\varpi}^{N\langle k\rangle})$ drives the process $\tilde{X}^{(n)\langle k\rangle}$ according to (16). While the initial condition $\tilde{X}^{(n)\langle 0\rangle}$ is deterministic, the terminal condition $\tilde{P}^{(n)\langle K\rangle}$ is random. We take a Monte Carlo (MC) approximation of (17) with J realizations; that is,

$$\begin{cases}
\frac{1}{J}\sum_{j=1}^J\Delta\tilde{P}_j^{(n)\langle k\rangle}=\frac{\Delta t}{J}\sum_{j=1}^J\left(-L_x(\tilde{X}_j^{(n)\langle k\rangle},\tilde{v}_j^{(n)\langle k\rangle})+\epsilon_j^{(n)\langle k\rangle}\right)\\
\frac{1}{J}\sum_{k=1}^J\Delta\tilde{X}_j^{(n)\langle k\rangle}=\frac{\Delta t}{J}\sum_{j=1}^J\tilde{v}_j^{(n)\langle k\rangle},\\
\frac{1}{JN}\sum_{k=1}^J\sum_{n=1}^N\tilde{v}_j^{(n)\langle k\rangle}=\frac{1}{J}\sum_{j=1}^J\left(Q_j^{\langle k\rangle}+{\epsilon_B}_j^{\langle k\rangle}\right),
\end{cases}$$

$\tilde{X}_j^{(n)\langle 0\rangle} = x_0^n$ and $\frac{1}{J}\sum_{k=1}^{J}\tilde{P}_j^{(n)\langle K\rangle} = \frac{1}{J}\sum_{k=1}^{J}\left(u_T'(\tilde{X}_j^{(n)\langle K\rangle}) - \epsilon_T{}_j^{(n)}\right)$. Thus, to implement the a posteriori estimate of Theorem 1 numerically, let $\tilde{v}_j^{(n)\langle k\rangle}$ and $\tilde{\varpi}^{N\langle k\rangle}$ be given. Define $\tilde{X}_j^{(n)\langle k\rangle}$ and $\tilde{P}_j^{(n)\langle k\rangle}$ according to (16) and (10), respectively, and compute the mean-square error (MSE) of ϵ_H and ϵ_B by

$$\text{MSE}(\epsilon_H) = \tfrac{1}{JNK}\sum_{j=1}^{J}\sum_{k=0}^{K}\sum_{n=1}^{N}\Bigg(\Big(\Delta\tilde{P}_j^{(n)\langle k\rangle} + \Delta t L_x(\tilde{X}_j^{(n)\langle k\rangle}, v_j^{(n)\langle k\rangle})\Big)^2 + \Big(u_T'(\tilde{X}_j^{(n)\langle K\rangle}) - \tilde{P}_j^{(n)\langle K\rangle}\Big)^2\Bigg),$$

$$\text{MSE}(\epsilon_B) = \tfrac{1}{JK}\sum_{j=1}^{J}\sum_{k=0}^{K}\left(\frac{1}{N}\sum_{n=1}^{N} v_j^{(n)\langle k\rangle} - Q_j^{\langle k\rangle}(t)\right)^2. \quad (18)$$

We measure (18) as we train the neural network with the algorithm we introduce next.

Remark 4: Theorem 1 addresses the convergence for finite populations. A complete analysis of the convergence in our method involves three steps we identify by writing

$$\|\varpi - \tilde{\varpi}_{\text{ML}}^N\| \leqslant \|\varpi - \varpi^N\| + \|\varpi^N - \varpi_{\text{MC}}^N\| + \|\varpi_{\text{MC}}^N - \tilde{\varpi}_{\text{ML}}^N\|.$$

First, the convergence of finite to continuum population games. Second, the convergence of the MC approximation to the finite population game, addressing the dependence of sample size w.r.t. population size. Third, the convergence of the ML to the MC approximation, involving the RNN parameters in the estimates. This is the error that our a posteriori estimate controls.

B. Training algorithm

In typical ML frameworks, a neural network is trained by minimizing a loss function $\mathcal{L}$. Within a fixed architecture, $\mathcal{L}$ assigns a real number $\mathcal{L}(\Theta)$ to a parameter Θ. The objective is to minimize $\mathcal{L}$ across all possible values of Θ. For a given realization Q_i of the supply, the loss function is

$$\mathcal{L}(\Theta_v, \Theta_\varpi) = \frac{1}{N}\sum_{n=1}^{N}\Bigg(\sum_{k=0}^{K-1}\Delta_t\Big(L(X^{(n)\langle k\rangle}, \mathrm{v}^{(n)\langle k\rangle}(\Theta_v)) + \varpi^{\langle k\rangle}(\Theta_\varpi)\left(\mathrm{v}^{(n)\langle k\rangle}(\Theta_v) - Q_i^{\langle k\rangle}\right)\Big) + u_T(X^{(n)\langle M\rangle})\Bigg), \quad (19)$$

obtained by time discretization and Monte-Carlo approximation of (8). Algorithm 1 describes the training algorithm. Our algorithm uses adversarial training, where one RNN maximizes and another minimizes the loss functional (19).

In contrast to the algorithm in [11], in Algorithm 1, the supply input changes between training steps. The algorithm trains two neural networks in an adversarial manner. At each step, we generate a sample of the probability distribution m_0. To minimize the agent's cost function, we update Θ_v in the direction of descent while Θ_ϖ is fixed. Conversely, to penalize deviation from the balance condition, we maximize the cost functional by updating Θ_ϖ in the direction of ascent while Θ_v is fixed. This process is repeated, approximating the saddle point corresponding to the control minimizing the cost functional and its Lagrange multiplier.

Algorithm 1: Training algorithm

Input : number of training iterations I, epoch size I_e, number of time steps K, training sample size N_{train}, test sample size N_{test}, MC sample size J, initial density m_0.

Initialize Θ_v^1, Θ_ϖ^1;

for $i = 1, \ldots, I$ **do**

- sample $(x_0^n)_{n=1}^{N_{\text{train}}}$ according to m_0;
- sample $(Q_i^{\langle k\rangle})_{k=0}^{K}$ according to (1);
- compute $(\varpi^{\langle k\rangle}(\Theta_\varpi^i))_{k=0}^{K}$ and $((\mathrm{v}^{(n)\langle k\rangle}(\Theta_v^i))_{n=1}^{N_{\text{train}}})_{k=0}^{K}$;
- compute $\mathcal{L}(\Theta_v^i, \Theta_\varpi^i)$ according to (19);
- compute Θ_v^{i+1} by updating Θ_v^i in the descent direction $\mathcal{L}_{\Theta_v}(\Theta_v^i, \Theta_\varpi^i)$;
- compute $((\mathrm{v}^{(n)\langle k\rangle}(\Theta_v^{i+1}))_{n=1}^{N_{\text{train}}})_{k=0}^{K}$;
- compute $\mathcal{L}(\Theta_v^{i+1}, \Theta_\varpi^i)$ according to (19);
- compute Θ_ϖ^{i+1} by updating Θ_ϖ^i in the ascent direction $\mathcal{L}_{\Theta_\varpi}(\Theta_v^{i+1}, \Theta_\varpi^i)$;
- **if** $i \mod I_e = 0$ *(epoch is completed)* **then**
  - sample $(x_0^n)_{n=1}^{N_{\text{test}}}$ according to m_0;
  - sample $((Q_j^{\langle k\rangle})_{k=0}^{K})_{j=1}^{J}$ according to (1);
  - compute $(\varpi^{\langle k\rangle}(\Theta_\varpi^{i+1}))_{k=0}^{K}$ and $((\mathrm{v}^{(n)\langle k\rangle}(\Theta_v^{i+1}))_{n=1}^{N_{\text{test}}})_{k=0}^{K}$;
  - compute $\text{MSE}(\epsilon_H)$ and $\text{MSE}(\epsilon_B)$ according to (18).
- **end**

end

Output: Θ_ϖ^I, Θ_v^I

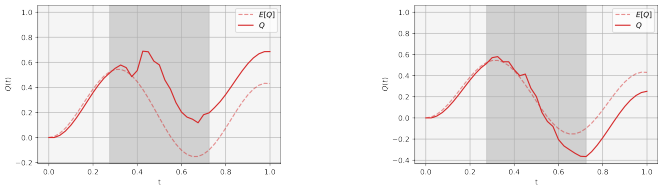
V. Numerical results

Here, we show how the a posteriori estimate (Theorem 1) ensures that our method has accurate approximations. We validate our findings with a linear-quadratic model as a benchmark. We use the Tensorflow library for the implementation.

We set $T = 1$ and $K = 40$ and assume the supply follows

$$\mathrm{d}Q(t) = \big(b^S(t) - Q(t)\big)\,\mathrm{d}t + \sigma^S(t)\mathrm{d}W(t), \quad (20)$$

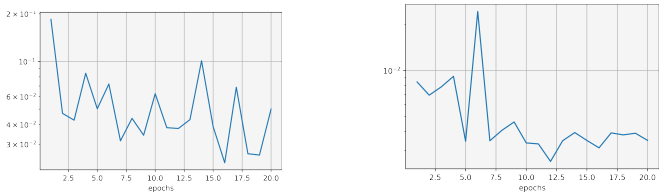
where $b^S(t) = 3\sin(3\pi t)$, $\sigma^S(t) = \max\{0.5\sin(2\pi(t - 0.25)), 0\}$, and $Q(0) = 0$. The Brownian noise affects the time interval $[0.25, 0.75]$ and causes deviations from the expected value, as illustrated in Figure 1 with two sample paths of the supply. The initial distribution m_0 is a normal distribution with mean $\overline{m}_0 = -\frac{1}{4}$ and standard deviation 0.2. The sample size for the training is $N_{\text{train}} = 30$. We train for 20 epochs, an epoch consisting of 500 training steps. We compute the MC estimate of the a posteriori estimate at the end of each epoch using $J = 60$ supply samples and a population size of $N_{\text{test}} = 30$. Empirically, the previous training parameters solved the trade-off between computational cost and accuracy.



(a) Supply realization. (b) Supply realization.

Fig. 1: Two supply realizations of (20). The grey window highlights the times where noise operates.

We select $L(x, v) = \frac{1}{2}(x-1)^2 + \frac{1}{2}v^2$ and $u_T(x) = \frac{1}{2e}(x-1)^2$. Figure 2 shows the evolution of the a posteriori estimate in Theorem 1. The balance error achieves enough accuracy and slightly oscillates with a decreasing trend. The optimality error also has a decreasing trend, but accuracy does not improve, suggesting other combinations of training and discretization parameters may be needed.



(a) MSE(ϵ_B) (balance error).

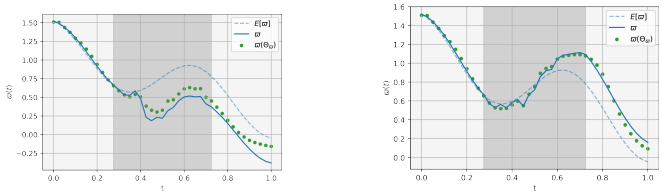
(b) MSE(ϵ_H) (optimality error).

Fig. 2: Evolution of the a posteriori estimates during training.

Furthermore, we use the analytic solution derived in [13] to verify the price approximation's accuracy. In the linear-quadratic framework, the price follows from the SDE system

$$\begin{cases} \mathrm{d}Q(t) = (b^S(t) - Q(t))\mathrm{d}t + \sigma^S(t)\mathrm{d}W(t), \\ \mathrm{d}\overline{X}(t) = Q(t)\mathrm{d}t, \\ \mathrm{d}\varpi(t) = \left(\overline{X}(t) - b^S(t) + Q(t) - 1\right)\mathrm{d}t - \frac{a_2^3(t)+1}{a_2^4(t)+1}\sigma^S(t)\mathrm{d}W(t), \end{cases} \tag{21}$$

with $Q(0) = 0$, $\overline{X}(0) = \overline{m}_0$, and $\varpi(0) = w_0$. The value w_0 and a_2^3 and a_2^4 are determined by a system of ordinary differential explicitly solvable. Figure 3 shows the corresponding price approximation and exact price (obtained from (21)) for the two supply realizations of Figure 1. The decreasing trend in the errors observed Figure 2 is reflected in the precise approximation observed in Figure 3. The effect of the error in the time window $[0.25, 0.75]$ decreases, as expected, the accuracy of the approximation compared to the time region $[0, 0.25]$, where no noise is present.



(a) Price realization for Figure 1a.

(b) Price realization for Figure 1b.

Fig. 3: Exact price and RNN approximation for Figure 1. The grey window highlights the times where noise operates.

As the figures show, our method has excellent performance in approximating solutions in various noise regimes.

VI. Conclusions and Further Directions

We extend the ML approach from [11], applying it to the common noise scenario via RNN architectures for non-anticipating controls, demonstrating accuracy and performance similar to the deterministic case. Future research could explore the method's robustness against RNN and discretization parameter changes, simultaneously addressing computational cost versus accuracy. Comprehensive experiments may identify optimal RNN layer and neuron number. Improved coding strategies could further reduce computational costs while maintaining or enhancing accuracy. Extensions could also accommodate additional noise sources, such as jump processes, and include the dependence of the supply on the price.

Acknowledgment

D. Gomes and J. Gutierrez were supported by King Abdullah University of Science and Technology (KAUST) baseline funds and KAUST OSR-CRG2021-4674.

References

[1] Y. Ashrafyan, T. Bakaryan, D. Gomes, and J. Gutierrez. The potential method for price-formation models. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 7565–7570, 2022.

[2] C. Belak, D. Hoffmann, and F. T. Seifried. Continuous-time mean field games with finite state space and common noise. *Applied Mathematics & Optimization*, 84(3):3173–3216, 2021.

[3] J.-D. Benamou and G. Carlier. Augmented lagrangian methods for transport optimization, mean field games and degenerate elliptic equations. *Journal of Optimization Theory and Applications*, 167(1):1–26, 2015.

[4] H. Cao, X. Guo, and M. Laurière. Connecting GANs, MFGs, and OT, 2021.

[5] E. Carlini and F. J. Silva. A fully discrete semi-lagrangian scheme for a first order mean field game problem. *SIAM Journal on Numerical Analysis*, 52(1):45–67, 2014.

[6] R. Carmona and M. Laurière. Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games I: the ergodic case. *SIAM Journal on Numerical Analysis*, 59(3):1455–1485, 2021.

[7] R. Carmona and M. Laurière. Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games II: The finite horizon case. *To appear in Annals of Applied Probability*, 2021.

[8] R. Carmona and M. Laurière. Deep learning for mean field games and mean field control with applications to finance. *To appear in Machine Learning And Data Sciences For Financial Markets (arXiv preprint arXiv:2107.04568)*, 2021.

[9] G. Dayanikli and M. Lauriere. A machine learning method for stackelberg mean field games, 2023.

[10] N. El Karoui, S. Peng, and M. C. Quenez. Backward stochastic differential equations in finance. *Mathematical Finance*, 7(1):1–71, 1997.

[11] D. Gomes, J. Gutiérrez, and M. Laurière. Machine learning architectures for price formation models, 2022.

[12] D. Gomes, J. Gutierrez, and R. Ribeiro. A mean field game price model with noise. *Math. Eng.*, 3(4):Paper No. 028, 14, 2021.

[13] D. Gomes, J. Gutierrez, and R. Ribeiro. A random-supply mean field game price model, 2021.

[14] D. Gomes and J. a. Saúde. A Mean-Field Game Approach to Price Formation. *Dyn. Games Appl.*, 11(1):29–53, 2021.

[15] S. Hadikhanloo and F. J. Silva. Finite mean field games: Fictitious play and convergence to a first order continuous mean field game. *Journal de Mathématiques Pures et Appliquées*, 132:369–397, 2019.

[16] J. Han and R. Hu. Recurrent neural networks for stochastic control problems with delay. *Mathematics of Control, Signals, and Systems*, 33:775–795, 2021.

[17] C. Higham and D. Higham. Deep learning: an introduction for applied mathematicians. *SIAM Review*, 61(4):860–891, Nov. 2019.

[18] M. Laurière, S. Perrin, M. Geist, and O. Pietquin. Learning mean field games: A survey, 2022.

[19] A. T. Lin, S. W. Fung, W. Li, L. Nurbekyan, and S. J. Osher. Alternating the population and control neural networks to solve high-dimensional stochastic mean-field games. *Proceedings of the National Academy of Sciences*, 118(31), 2021.

[20] M. Min and R. Hu. Signatured deep fictitious play for mean field games with common noise, 2021.

[21] C. Mou, X. Yang, and C. Zhou. Numerical methods for mean field games based on gaussian processes and fourier features, 2021.

[22] S. Park and P. V. Hentenryck. Self-supervised primal-dual learning for constrained optimization, 2022.

[23] L. Ruthotto, S. Osher, W. Li, L. Nurbekyan, and S. W. Fung. A machine learning framework for solving high-dimensional mean field game and mean field control problems, 2020.