Proceedings of Machine Learning Research vol 107:221–245, 2020

Deep Fictitious Play for Finding Markovian Nash Equilibrium in Multi-Agent Games

Jiequn Han JIEQUNH@PRINCETON.COM
Department of Mathematics, Princeton University, Princeton, NJ 08544-1000, USA

Ruimeng Hu RH2937@COLUMBIA.EDU
Department of Statistics, Columbia University, New York, NY 10027-4690, USA

Abstract

We propose a deep neural network-based algorithm to identify the Markovian Nash equilibrium of general large N-player stochastic differential games. Following the idea of fictitious play, we recast the N-player game into N decoupled decision problems (one for each player) and solve them iteratively. The individual decision problem is characterized by a semilinear Hamilton-Jacobi-Bellman equation, to solve which we employ the recently developed deep BSDE method. The resulted algorithm can solve large N-player games for which conventional numerical methods would suffer from the curse of dimensionality. Multiple numerical examples involving identical or heterogeneous agents, with risk-neutral or risk-sensitive objectives, are tested to validate the accuracy of the proposed algorithm in large group games. Even for a fifty-player game with the presence of common noise, the proposed algorithm still finds the approximate Nash equilibrium accurately, which, to our best knowledge, is difficult to achieve by other numerical algorithms.

Keywords: stochastic differential game, Markovian Nash equilibrium, fictitious play, deep learning

1. Introduction

The development of communication technologies makes the world more interactive than ever, and crucial decisions are generally made in a common environment nowadays where multiple players/agents[1] act reciprocally. The stochastic differential game provides an elegant framework for studying such scenarios and has been widely used in the areas of financial mathematics, economics, management science, engineering, etc. For instance, under this framework, Carmona et al. (2015) analyze systemic risk caused by inter-bank lending and borrowing, Achdou et al. (2017) study the income and wealth distribution in macroeconomics, and Bensoussan et al. (2014) model the reinsurance games between insurance companies, just to list a few.

Among all applications of stochastic differential games, a core problem is to find the Nash equilibrium. A Nash equilibrium refers to a set of all players' strategies in which nobody has the incentive to deviate, given the others' strategies fixed, and a Markovian strategy means the decision is made based only on the current situation of all players. Under the Markovian setting in continuous time, the Nash equilibrium is characterized by N-coupled Hamilton-Jacobi-Bellman (HJB) equations, each of which is usually high-dimensional when N is large. In some special settings, one can find the closed-form Nash equilibrium. However, an analytic solution is generally unavailable, and one has to resort to numerical solutions. The notorious computational difficulty "curse of dimensionality" is often encountered in this situation due to the high dimensionality of

1. Hereafter, we shall use *player* and *agent* interchangeably.

state variables. So it is crucial but challenging to develop numerical algorithms to accurately and efficiently find the Nash equilibrium in large N-player stochastic differential games.

In recent years the deep neural network (DNN) has shown its remarkable ability in representing and approximating high-dimensional functions in numerical computation, ranging from artificial intelligence (cf. Bengio (2009); LeCun et al. (2015)) such as computer vision and speech recognition, to scientific computing (see e.g., Carleo and Troyer (2017); Gao and Duan (2017); Han et al. (2018); Zhang et al. (2018b); Han et al. (2019)). Considering the astonishing performance of the DNN, we shall leverage it to help identify the Nash equilibrium through solving high-dimensional coupled HJB equations. To achieve this, we design a scalable algorithm, which we refer to as *deep fictitious play*, based on a combination of DNN and the idea of fictitious play. Fictitious play is an iterative scheme in which, at each stage, the whole coupled game is decomposed to independent decision problems for each player by considering her opponents' strategies are fixed and following their past play. Such a scheme enables the algorithm to be decoupled and parallel between players. The individual decision problem is then translated into a single HJB equation and solved via the deep BSDE method (E et al. (2017); Han et al. (2018)). Since decision problems are solved individually for all players through general HJB equations, the algorithm can accommodate games with arbitrary heterogeneity, and with risk-neutral or risk-sensitive cost functional. Note that the idea of deep fictitious play is recently proposed in Hu (2019) as well for solving open-loop Nash equilibrium. However, due to the entirely different strategy set, the algorithm developed in Hu (2019) is inapplicable for Markovian Nash equilibrium (see Remark 3.2 for details).

An alternative approach to studying large N-player game is mean-field game theory, introduced by Lasry and Lions (2006a,b, 2007) and by Huang et al. (2007, 2006) around the same time. In mean-field game theory, the limiting solution as $N \to \infty$ is used to approximate the game of finite population. Under mild assumptions, Carmona and Delarue (2013) have shown that the approximation error is of order $N^{-1/(d+4)}$, where d is the dimension of each player's state variable. The limitation of the mean-field game is threefold. First, the approximation error is not well controlled for a moderate N. Second, it requires lots of symmetry. To make the mean-field approximation work, the players need to be indistinguishable, i.e., they control their states in the same way and have identical objectives. Third, the numerical computation of the mean-filed game is usually infeasible when there is common noise in addition to idiosyncratic noise. In contrast to these limitations, as shown in the description of the algorithm and numerical results, the proposed algorithm can solve general N-player stochastic differential games with the presence of heterogeneity and common noise.

Another line of research relevant to this work is the so-called multi-agent reinforcement learning (MARL) (see Busoniu et al. (2008) for an earlier review), which recently has gained increasing research attention (see e.g. Foerster et al. (2016); Zhang et al. (2018a)), given the remarkable success of reinforcement learning in many applications, especially accompanied with the powerful function approximation based on DNN. The main difference is that MARL is usually model-free and in discrete time while our work considers model-based games in continuous time, for which we can leverage the HJB equation and BSDE reformulation to develop efficient algorithms.

In short, our algorithm focuses on the moderate number of the agents, i.e., $10 \leq N \leq 100$, in which case conventional numerical methods lose their efficiencies while the mean-field theory has not yet been accurate in approximating the finite population game. The strength of the proposed algorithm is the ability to deal with heterogeneous agents, to which scenarios mean-field games are not good at in general, and the ease of dealing with common noise. The rest of the paper is organized as follows. Section 2 gives the mathematical formulation of general N-player stochastic

differential games in continuous time. The algorithm based on fictitious play and DNN is detailed in Section 3, followed by numerical examples in Section 4. We conclude in Section 5 and provide technical details in appendices.

2. Mathematical Formulation

We consider N-player non-cooperative stochastic differential games described by the following dynamics,

$$\mathrm{d}\boldsymbol{X}_t^{\boldsymbol{\alpha}} = b(t, \boldsymbol{X}_t^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}(t, \boldsymbol{X}_t^{\boldsymbol{\alpha}}))\,\mathrm{d}t + \Sigma(t, \boldsymbol{X}_t^{\boldsymbol{\alpha}})\,\mathrm{d}\boldsymbol{W}_t, \quad \boldsymbol{X}_0 = \boldsymbol{x}_0, \tag{1}$$

where $\boldsymbol{X}_t^{\boldsymbol{\alpha}}$ is an $\mathbb{R}^n$-valued *common* state process influenced by Markovian controls $\boldsymbol{\alpha} = (\alpha^1, \dots, \alpha^N)$, a collection of all players' strategies. Each α^i, as the control of player i, is a Borel measurable function $[0,T] \times \mathbb{R}^n \hookrightarrow \mathcal{A}^i \subset \mathbb{R}^{n_i}$. b and Σ are deterministic functions denoting the drift and diffusion coefficients of the common state, $b\colon [0,T] \times \mathbb{R}^n \times \mathcal{A} \hookrightarrow \mathbb{R}^n$, $\Sigma\colon [0,T] \times \mathbb{R}^n \hookrightarrow \mathbb{R}^{n\times k}$, where $\mathcal{A} = \otimes_{i=1}^N \mathcal{A}^i$ is the space for joint control $\boldsymbol{\alpha}$, and $\boldsymbol{W}$ is a k-dimensional standard Brownian motion on a filtered probability space $(\Omega, \mathbb{F}, \{\mathcal{F}_t\}_{0\le t\le T}, \mathbb{P})$.

Denote by $\mathbb{A}^i$ the set of admissible $\mathcal{A}^i$-valued strategy for player $i \in \mathcal{I} := \{1, 2, \dots, N\}$, and by $\mathbb{A} = \otimes_{i=1}^N \mathbb{A}^i$ the product space of $\mathbb{A}^i$. Given the other's strategy $\alpha^j \in \mathbb{A}^j$, $j \neq i$, player i chooses α^i to minimize the cumulative cost J_0^i,

$$J_t^i(\boldsymbol{\alpha}) := \mathbb{E}\left[\int_t^T f^i(s, \boldsymbol{X}_s^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}(s, \boldsymbol{X}_s^{\boldsymbol{\alpha}}))\,\mathrm{d}s + g^i(\boldsymbol{X}_T^{\boldsymbol{\alpha}})\right], \tag{2}$$

or its risk-sensitive version

$$J_t^i(\boldsymbol{\alpha}) := \mathbb{E}\left[\theta_i \exp\left\{\theta_i\left(\int_t^T f^i(s, \boldsymbol{X}_s^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}(s, \boldsymbol{X}_s^{\boldsymbol{\alpha}})\,\mathrm{d}t + g^i(\boldsymbol{X}_T^{\boldsymbol{\alpha}})\right)\right\}\right], \tag{3}$$

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

We are interested in solving the above game in terms of finding the Nash equilibrium, in particular, the Markovian Nash equilibrium.

Definition 2.1 *A Markovian Nash equilibrium is a tuple* $\boldsymbol{\alpha}^* = (\alpha^{1,*}, \dots, \alpha^{N,*}) \in \mathbb{A}$ *such that*

$$\forall i \in \mathcal{I}, \text{ and } \alpha^i \in \mathbb{A}^i, \quad J_0^i(\boldsymbol{\alpha}^*) \le J_0^i(\alpha^{1,*}, \dots, \alpha^{i-1,*}, \alpha^i, \alpha^{i+1,*}, \dots, \alpha^{N,*}).$$

The above problem (1)–(3) is more general than the usual finite-player stochastic differential games or mean-field games in two aspects. Firstly, in addition to the usual (risk-neutral) cost functional (2), we also consider a risk-sensitive version (3), where θ_i is a parameter characterizing how risk-averse/seeking player i is. This flexibility allows us to model much broader classes of games that accommodate the players' attitudes to risk. Secondly and more importantly, $\boldsymbol{X}_t$ in (1) is a common state process that is influenced by all players, as opposed to the traditional case that player i can only control her private state. The former feature is common in the literature of economics (see e.g., Dockner et al. (2000); Prasad and Sethi (2004); Van Long (2011)), thus, we think it is important to include it in (1). However, this leads to a stronger coupling problem, which is unsurprisingly harder to deal with both theoretically and numerically. The difficulty even persists in the limiting problem as $N \to \infty$ with indistinguishable players, when allowing α^i entering into others' states.

This is called the extended mean-field game and it has attracted lots of attention recently (see e.g., Gomes and Voskanyan (2013); Gomes et al. (2014); Gomes and Voskanyan (2016); Cardaliaguet and Lehalle (2018)).

Note that by choosing b and Σ in (1) properly, one can reduce the formulation (1) to the simpler case where each player controls her *private* state through α^i. We highlight this relation in the following remark.

Remark 2.2 *Let $n = dN$ and $b^\ell \equiv b^\ell(t, \boldsymbol{x}, \alpha^i)$ for $\ell = (i-1)d+1, \dots, id$, then the problem (1)–(3) is the usual modeling in financial mathematics literature, where the i^{th} player's d-dimensional private state $(X_t^{(i-1)d+1}, \dots, X_t^{id})$ is controlled by α^i only. A benchmark example we shall study in Section 4.1 is a model of inter-bank borrowing and lending proposed in Carmona et al. (2015), where $d = 1$, and the dynamics for each player is*

$$\mathrm{d}X_t^i = [a(\overline{X}_t - X_t^i) + \alpha_t^i]\,\mathrm{d}t + \sigma\left(\rho\,\mathrm{d}W_t^0 + \sqrt{1-\rho^2}W_t^i\right), \quad \overline{X}_t = \frac{1}{N}\sum_{i=1}^N X_t^i.$$

In the Markovian setting, finding a Nash equilibrium is related to solving N-coupled HJB equations. To this end, we define the value function of player i by

$$V^i(t, \boldsymbol{x}) = \inf_{\alpha^i \in \mathbb{A}^i} \mathbb{E}[J_t^i(\boldsymbol{\alpha}) | \boldsymbol{X}_t = \boldsymbol{x}].$$

Using the dynamic programming principle, the HJB system reads

$$\begin{cases} V_t^i + \inf\limits_{\alpha^i \in \mathcal{A}^i} G^i(t, \boldsymbol{x}, \boldsymbol{\alpha}, \nabla_{\boldsymbol{x}} V^i, V^i) + \dfrac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}} \mathrm{Hess}_{\boldsymbol{x}} V^i \Sigma) = 0, \\ V^i(T, \boldsymbol{x}) = g^i(\boldsymbol{x}), \quad i \in \mathcal{I}, \end{cases} \tag{4}$$

where G^i is given by

$$\begin{aligned} &\text{under objective (2)}, \quad G^i = G^i(t, \boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{p}) = b(t, \boldsymbol{x}, \boldsymbol{\alpha}) \cdot \boldsymbol{p} + f^i(t, \boldsymbol{x}, \boldsymbol{\alpha}), \\ &\text{under objective (3)}, \quad G^i = G^i(t, \boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{p}, s) = b(t, \boldsymbol{x}, \boldsymbol{\alpha}) \cdot \boldsymbol{p} + \theta_i s f^i(t, \boldsymbol{x}, \boldsymbol{\alpha}). \end{aligned}$$

Here $\nabla_{\boldsymbol{x}} V$, $\mathrm{Hess}_{\boldsymbol{x}} V$ denote the gradient and the Hessian of function V respect to $\boldsymbol{x}$ and Tr denotes the trace of a matrix. In the risk-neutral case (2), G^i is in fact the Hamiltonian and is usually denoted by H^i. Nevertheless, to unify the notation between two objectives (2) and (3), we shall stick to $G^i(t, \boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{p}, s)$ for the rest of the paper, although it does not depend on s under objective (2). Note that while minimizing α^i in the equation for V^i in (4), the policies $(\alpha^1, \dots, \alpha^{i-1}, \alpha^{i+1}, \dots, \alpha^N)$ are given and fixed. In other words, V^i implicitly depends on the other players' strategies, thus on V^j.

Throughout the paper, we assume that there is a unique classical solution to the HJB system (4). Moreover, we require that the minimizer $\arg\min_{\alpha^i \in \mathcal{A}^i} G^i(t, \boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{p}, s)$ exists, is unique and explicit in other arguments, $\forall\, i \in \mathcal{I}$, $(t, \boldsymbol{x}, \boldsymbol{p}, s)$ and $\alpha^j \in \mathcal{A}^j$ with $j \neq i$.

3. Methodology

3.1. Fictitious Play

Fictitious play was firstly introduced by Brown in the static game Brown (1949, 1951), and was recently adapted to the mean-field setting by Cardaliaguet and Hadikhanloo (2017); Briani and

Cardaliaguet (2018). It is a simple yet important learning idea in game theory. The key is to decouple the N-player game into N individual decision problems where opponents' strategies are fixed and assumed to follow their past play. These N individual problems are solved iteratively, starting from stage 1. At stage m, we assume that the opponents' strategies are their stage $(m-1)$'s best responses. To better describe this procedure mathematically, we first summarize the notations that shall be used below.

- $\mathcal{A}^i \subset \mathbb{R}^{n_i}$, the range of player i's strategy α^i. $\mathcal{A} = \otimes_{i=1}^N \mathcal{A}^i$, the control space for all players, and $\mathcal{A}^{-i} = \otimes_{j\neq i}\mathcal{A}^j$, the control space for all players but i. The same applies to the admissible spaces of measurable functions $\mathbb{A}^i$, $\mathbb{A}$ and $\mathbb{A}^{-i}$. Note that, with these notations, $\inf_{\alpha^i\in\mathcal{A}^i}$ and $\inf_{\alpha^i\in\mathbb{A}^i}$ means seeking for an optimal vector in $\mathcal{A}^i$ and $\mathcal{A}^i$-valued function (strategy), respectively.
- $\boldsymbol{\alpha} = (\alpha^1, \alpha^2, \ldots, \alpha^N)$, a collection of all players' strategy profiles. With a negative superscript, $\boldsymbol{\alpha}^{-i} = (\alpha^1, \ldots, \alpha^{i-1}, \alpha^{i+1}, \ldots, \alpha^N)$ means the strategy profiles excluding player i's. If a non-negative superscript m appears, $\boldsymbol{\alpha}^m$ is a N-tuple standing for the strategies of all players at stage m. When both exist, $\boldsymbol{\alpha}^{-i,m} = (\alpha^{1,m}, \ldots, \alpha^{i-1,m}, \alpha^{i+1,m}, \ldots, \alpha^{N,m})$ is a $(N-1)$-tuple representing strategies excluding player i at stage m.

Assume that we start with a guess of the solution $\boldsymbol{\alpha}^0 \in \mathbb{A}$. The idea of fictitious play motivates us to consider an iterative algorithm according to the following rules. At stage $m+1$, $\boldsymbol{\alpha}^m$ is observed by all players, and player i's decoupled decision problem is

$$\inf_{\alpha^i\in\mathbb{A}^i} J_0^i(\alpha^i; \boldsymbol{\alpha}^{-i,m}), \tag{5}$$

where J_0^i is defined in (2) or (3), and the state process $\boldsymbol{X}_t$ is given in (1) with $\boldsymbol{\alpha}$ being replaced by $(\alpha^i, \boldsymbol{\alpha}^{-i,m})$. The optimal strategy, if ever exists, is denoted by $\alpha^{i,m+1}$. The problem (5) for all $i \in \mathcal{I}$ are solved simultaneously using $\boldsymbol{\alpha}^{-i,m}$, and the optimal responses together form $\boldsymbol{\alpha}^{m+1}$. Due to the Markovian structure, the problem (5) is translated into a HJB equation

$$V_t^{i,m+1} + \inf_{\alpha^i\in\mathcal{A}^i} G^i(t, \boldsymbol{x}, (\alpha^i, \boldsymbol{\alpha}^{-i,m}(t,\boldsymbol{x})), \nabla_{\boldsymbol{x}} V^{i,m+1}, V^{i,m+1}) + \frac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}}\mathrm{Hess}_{\boldsymbol{x}} V^{i,m+1}\Sigma) = 0, \tag{6}$$

with the terminal condition $V^{i,m+1}(T, \boldsymbol{x}) = g^i(\boldsymbol{x})$. If the classical solution ever exists, the optimal strategy at stage $m+1$ for player i is given by

$$\alpha^{i,m+1}(t,\boldsymbol{x}) = \operatorname*{arg\,min}_{\alpha^i\in\mathcal{A}^i} G^i(t, \boldsymbol{x}, (\alpha^i, \boldsymbol{\alpha}^{-i,m}(t,\boldsymbol{x})), \nabla_{\boldsymbol{x}} V^{i,m+1}(t,\boldsymbol{x}), V^{i,m+1}(t,\boldsymbol{x})). \tag{7}$$

Solving (6) for all $i \in \mathcal{I}$ completes one stage in the loop of fictitious play.

Given the iterative procedure described above, one can naturally ask: (1) does $\alpha^{i,m}$ always exist; (2) if yes, does $\boldsymbol{\alpha}^m$ converge; and (3) if yes, is the limiting strategy $\boldsymbol{\alpha}^\infty$ admissible and does it form a Nash equilibrium. The first question closely depends on the choice of the initial belief $\boldsymbol{\alpha}^0$. Usually a regular enough $\boldsymbol{\alpha}^0$ (plus regular b, Σ, f^i, g^i) will ensure the unique classical solution $V^{i,1}$, thus ensure the existence of $\boldsymbol{\alpha}^1$. Then it is likely that the regularity of $\boldsymbol{\alpha}$ persists from stage to stage. The remaining two questions are tough in general. For the second question, there is no universal criterion that can guarantee the convergence. Even in the static finite-action games,

there are numerous examples when it converges (Robinson (1951); Miyasawa (1961); Milgrom and Roberts (1991); Monderer and Shapley (1996b,a); Berger (2005); Hofbauer and Sandholm (2002)) and when it does not (Shapley (1964); Jordan (1993); Monderer and Sela (1996); Foster and Young (1998); Krishna and Sjöström (1998)). The third question is related to the stability of the game, and has to be analyzed case by case.

Instead of answering the theoretical questions raised above, this paper focuses on a practical numerical scheme for finding the Markovian Nash equilibrium, especially when N is large. As proof of methodology, we shall present four examples to show the performance of the proposed algorithm. These examples cover a large variety of stochastic differential games, including homogeneous/heterogeneous, risk-neutral/risk-sensitive ones. We remark that the first three examples are chosen so that they have closed-form solutions or can be solved via other numerical approaches in low dimensions, such that we can benchmark numerical solutions. The algorithm in fact can be applied in much more general games, as demonstrated in the last example.

Remark 3.1 *The learning procedure described above is slightly different from the original version proposed by Brown (1949, 1951) where two-player normal-form games are studied. Targeting at pure or mixed Nash equilibrium, the player therein chooses the best response against the empirical distribution of her opponent's past play. That is, if her opponent uses strategies $\alpha_1, \dots, \alpha_k$ during the first k stages, where α_i is from a finite strategy set, then at stage $k+1$, her response will be a pure strategy that maximizes her expected payoff with respect to her opponent's mixed strategy $\frac{1}{k}\sum_{i=1}^{k} \delta_{\alpha_i}$.*

In this paper, we prefer to use the last stage information, instead of the average of the past. The reason will be explained after we introduce our deep learning algorithm, see Remark 3.2.

3.2. A Deep Learning Algorithm Based on Fictitious Play

In order to find the Markovian Nash equilibrium through fictitious play, the main challenge is numerically solving the PDE (6) at each stage. When the dimension d is large, conventional numerical algorithms soon lose their efficiency. Here we employ the deep BSDE method in E et al. (2017); Han et al. (2018) to deal with the high-dimensionality.

Instead of solving the parabolic PDE (6), the deep BSDE method recasts the problem into an optimization problem based on the associated BSDE. Under the standing assumptions, the function $\alpha^{i,m+1}(t, \boldsymbol{x}, \boldsymbol{p}, s; \boldsymbol{\alpha}^{-i,m}) = \arg\min_{\alpha^i \in \mathcal{A}^i} G^i(t, \boldsymbol{x}, (\alpha^i, \boldsymbol{\alpha}^{-i,m}(t, \boldsymbol{x})), \boldsymbol{p}, s)$ is unique and explicit. We plug it into (6) and rewrite the equation as a semilinear parabolic PDE:

$$V_t^{i,m+1} + \frac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}}\mathrm{Hess}_{\boldsymbol{x}} V^{i,m+1}\Sigma) + \mu^i(t, \boldsymbol{x}; \boldsymbol{\alpha}^{-i,m}) \cdot \nabla_{\boldsymbol{x}} V^{i,m+1} + h^i(t, \boldsymbol{x}, V^{i,m+1}, \Sigma^{\mathrm{T}}\nabla_{\boldsymbol{x}} V^{i,m+1}; \boldsymbol{\alpha}^{-i,m}) = 0.$$

Note that we replace $\nabla_{\boldsymbol{x}} V^{i,m+1}$ by $\Sigma^{\mathrm{T}}\nabla_{\boldsymbol{x}} V^{i,m+1}$ in the term h^i for the sake of simplicity when describing the algorithm, and that we treat $\boldsymbol{\alpha}^{-i,m}$ as known functions of the PDE due to its exogeneity. The concrete equations of the considered numerical examples are presented in the associated subsections or Appendix A. For less cumbersome notations, we drop the superscript m that denotes the index of fictitious play and consider the PDE

$$\begin{cases} V_t^i + \frac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}}\mathrm{Hess}_{\boldsymbol{x}} V^i\Sigma) + \mu^i(t, \boldsymbol{x}; \boldsymbol{\alpha}^{-i}) \cdot \nabla_{\boldsymbol{x}} V^i + h^i(t, \boldsymbol{x}, V^i, \Sigma^{\mathrm{T}}\nabla_{\boldsymbol{x}} V^i; \boldsymbol{\alpha}^{-i}) = 0, \\ V^i(T, \boldsymbol{x}) = g^i(\boldsymbol{x}). \end{cases} \tag{8}$$

This PDE is intimately related to the following BSDE:

$$\begin{cases} \boldsymbol{X}_t^i = \boldsymbol{x}_0 + \int_0^t \mu^i(s, \boldsymbol{X}_s^i; \boldsymbol{\alpha}^{-i}(s, \boldsymbol{X}_s^i))\,\mathrm{d}s + \int_0^t \Sigma(s, \boldsymbol{X}_s^i)\,\mathrm{d}\boldsymbol{W}_s, & (9) \\ Y_t^i = g^i(\boldsymbol{X}_T^i) + \int_t^T h^i(s, \boldsymbol{X}_s^i, Y_s^i, \boldsymbol{Z}_s^i; \boldsymbol{\alpha}^{-i}(s, \boldsymbol{X}_s^i))\,\mathrm{d}s - \int_t^T (\boldsymbol{Z}_s^i)^{\mathrm{T}}\,\mathrm{d}\boldsymbol{W}_s, & (10) \end{cases}$$

where $\boldsymbol{W}_t$ is a k-dimensional Brownian motion and $\boldsymbol{x}_0$ is a square-integrable random variable independent of $\boldsymbol{W}_t$. Specifically, the nonlinear Feynman-Kac formula (cf. Pardoux and Peng (1992); El Karoui et al. (1997); Pardoux and Tang (1999)) states that, under certain regularity conditions,

$$Y_t^i = V^i(t, \boldsymbol{X}_t^i) \quad \text{and} \quad \boldsymbol{Z}_t^i = \Sigma(t, \boldsymbol{X}_t^i)^{\mathrm{T}} \nabla_{\boldsymbol{x}} V^i(t, \boldsymbol{X}_t^i) \tag{11}$$

defines the unique solution to the BSDE (9)–(10). Accordingly, we consider the variational problem

$$\begin{aligned} &\inf_{Y_0^i, \{\boldsymbol{Z}_t^i\}_{0\le t\le T}} \mathbb{E}|g^i(\boldsymbol{X}_T^i) - Y_T^i|^2, \qquad (12) \\ &s.t.\ \ \boldsymbol{X}_t^i = \boldsymbol{x}_0 + \int_0^t \mu^i(s, \boldsymbol{X}_s^i; \boldsymbol{\alpha}^{-i}(s, \boldsymbol{X}_s^i))\,\mathrm{d}s + \int_0^t \Sigma(s, \boldsymbol{X}_s^i)\,\mathrm{d}\boldsymbol{W}_s, \\ &\qquad Y_t^i = Y_0^i - \int_0^t h^i(s, \boldsymbol{X}_s^i, Y_s^i, \boldsymbol{Z}_s^i; \boldsymbol{\alpha}^{-i}(s, \boldsymbol{X}_s^i))\,\mathrm{d}s + \int_0^t (\boldsymbol{Z}_s^i)^{\mathrm{T}}\,\mathrm{d}\boldsymbol{W}_s, \end{aligned}$$

where Y_0^i is $\mathcal{F}_0$-measurable and square-integrable, and $\boldsymbol{Z}_t^i$ is a $\mathcal{F}_t$-adapted square-integrable process. Eq. (11) gives a minimizer of the above problem since the loss function attains zero when it is evaluated according to (11). In addition, if the BSDE (9)–(10) is wellposed (under some regularity conditions), the minimizer must exist and is unique.

The deep BSDE method builds on the temporal discretization of (12). Given a partition π of size N_T on the time interval $[0, T], 0 = t_0 < t_1 < \ldots < t_{N_T} = T$, we consider the discretized version of (12) based on the Euler scheme (the subscript t_k in $\boldsymbol{X}, Y, \boldsymbol{Z}$ has been replaced by k for simplicity):

$$\begin{aligned} &\inf_{\psi_0 \in \mathcal{N}_0^{i'},\ \{\phi_k \in \mathcal{N}_k^i\}_{k=0}^{N_T-1}} \mathbb{E}|g^i(\boldsymbol{X}_{N_T}^{i,\pi}) - Y_{N_T}^{i,\pi}|^2, \qquad (13) \\ &s.t.\ \ \boldsymbol{X}_0^{i,\pi} = \boldsymbol{x}_0, \quad Y_0^{i,\pi} = \psi_0(\boldsymbol{X}_0^{i,\pi}), \quad \boldsymbol{Z}_k^{i,\pi} = \phi_k(\boldsymbol{X}_k^{i,\pi}), \quad k = 0, \ldots, N_T - 1 \\ &\qquad \boldsymbol{X}_{k+1}^{i,\pi} = \boldsymbol{X}_k^{i,\pi} + \mu^i(t_k, \boldsymbol{X}_k^{i,\pi}; \boldsymbol{\alpha}^{-i}(t_k, \boldsymbol{X}_k^{i,\pi}))\Delta t_k + \Sigma(t_k, \boldsymbol{X}_k^{i,\pi})\Delta \boldsymbol{W}_k, \qquad (14) \\ &\qquad Y_{k+1}^{i,\pi} = Y_k^{i,\pi} - h^i(t_k, \boldsymbol{X}_k^{i,\pi}, Y_k^{i,\pi}, \boldsymbol{Z}_k^{i,\pi}; \boldsymbol{\alpha}^{-i}(t_k, \boldsymbol{X}_k^{i,\pi}))\Delta t_k + (\boldsymbol{Z}_k^{i,\pi})^{\mathrm{T}} \Delta \boldsymbol{W}_k, \qquad (15) \end{aligned}$$

where $\Delta t_k = t_{k+1} - t_k$, $\Delta \boldsymbol{W}_k = \boldsymbol{W}_{t_{k+1}} - \boldsymbol{W}_{t_k}$. Here $\mathcal{N}_0^{i'}$ and $\{\mathcal{N}_k^i\}_{k=0}^{N_T-1}$ are hypothesis spaces of player i related to deep neural networks. The goal of the optimization is to find optimal deterministic maps $\psi_0^{i,*}, \{\phi_k^{i,*}\}_{k=0}^{N_T-1}$ such that the loss function is small. Intuitively, we expect that (13) defines a benign optimization problem close to (12) and $\psi_0^{i,*}, \{\phi_k^{i,*}\}_{k=0}^{N_T-1}$ provide us good approximations to $V^i(0, \cdot), \{\nabla_{\boldsymbol{x}} V^i(t_k, \cdot)\}_{k=0}^{N_T-1}$, the solution of the original PDE (8).

In practice, the expectation in (13) is approximated by the standard Monte Carlo sampling of (14)–(15). Given sample paths of $\{\boldsymbol{X}_k^{i,\pi}\}_{k=0}^{N_T}$ and the associated white noises $\{\boldsymbol{W}_{t_k}\}_{k=0}^{N_T}$, one notes that the loss function (13) can be interpreted as the final output of a very deep network after stacking

all the subnetworks $\psi_0^i, \{\phi_k^i\}_{k=0}^{N_T-1}$ in sequence according to (15). This means that we can use backpropagation to derive the gradient of the loss function with respect to all the parameters in the neural networks and use stochastic gradient descent (SGD) to optimize all the parameters. We refer the interested readers to E et al. (2017); Han et al. (2018); Han and Long (2018) for more detailed description and theoretical justification of the deep BSDE method.

Remark 3.2 *According to the deep BSDE method, the best responses at stage m, $\boldsymbol{\alpha}^m(t,\boldsymbol{x})$, are defined through the outputs of neural networks. Due to the direct feedback nature, $\boldsymbol{X}_t$ changes as strategies vary from stage to stage, so do the function evaluations $\boldsymbol{\alpha}^m(t,\boldsymbol{X}_t)$. Therefore, if one were interested in using $\overline{\boldsymbol{\alpha}}^{-i,m} := \frac{1}{m}\sum_{\ell=1}^m \boldsymbol{\alpha}^{-i,\ell}$ to replace $\boldsymbol{\alpha}^{-i,m}$ in (5), the parameters of all neural networks from stage 1 to m need to be saved, which is infeasible for the sake of computational memory. This is indeed the very reason that we use $\boldsymbol{\alpha}^{-i,m}$ in objective (5).*

Note that this is not the case for searching open-loop Nash equilibrium. There, $\boldsymbol{\alpha}^m$ are adapted processes to the filtration generated by Brownian motions, i.e., one can intuitively think $\boldsymbol{\alpha}^m$ as functions of $\boldsymbol{W}_{[0,T]}$, thus do not change as strategies/states vary, as long as the training paths of $\boldsymbol{W}$ are fixed. Therefore, using $\overline{\boldsymbol{\alpha}}^{-i,m}$ only requires constant memory, since one can update it by a weighted sum of $\overline{\boldsymbol{\alpha}}^{-i,m-1}$ and $\boldsymbol{\alpha}^{-i,m}$.

For further discussion on this discrepancy, we refer to a closely related work by Hu (2019) where a deep learning scheme is designed for open-loop Nash equilibrium based on fictitious play. Indeed, it is empirically observed in Hu (2019) that using $\overline{\boldsymbol{\alpha}}^{-i,m}$ leads to faster convergence in the open-loop case. We would expect the similar phenomenon in the Markovian case, if we were able to record all the best responses $\boldsymbol{\alpha}^m(t,\boldsymbol{x})$. However, as discussed above, due to the direct feedback nature, merely adapting the algorithm in Hu (2019) will not be efficient nor parallelizable for finding Markovian Nash equilibrium, and a completely different approach is necessary.

3.3. Implementation

A few details should be specified regarding the methodology described in Section 3.2.

First, the hypothesis spaces $\mathcal{N}_0^{i'}$ and $\{\mathcal{N}_k^i\}_{k=0}^{N_T-1}$ need to be specified. Note that, at each stage m, the optimal policy is defined through

$$\alpha^{i,m}(t,\boldsymbol{x}) = \arg\min_{\alpha^i\in\mathcal{A}^i} G^i(t,\boldsymbol{x},(\alpha^i,\boldsymbol{\alpha}^{-i,m-1}(t,\boldsymbol{x})),\nabla_{\boldsymbol{x}}V^{i,m}(t,\boldsymbol{x}),V^{i,m}(t,\boldsymbol{x})),$$

one wishes to have direct access to the solved $V^{i,m}(t,\boldsymbol{x})$. Therefore we parametrize $V^i(t,\boldsymbol{x})$ (the superscript m is dropped again for simplicity) directly with a neural network, denoted by $\mathrm{Net}(t,\boldsymbol{x})$. Accordingly, $\mathrm{Net}(0,\boldsymbol{x})$ becomes a hypothesis function in $\mathcal{N}_0^{i'}$. For $\mathcal{N}_k^i$, we know from the nonlinear Feynman-Kac formula (11) that $\Sigma(t,\boldsymbol{x})\nabla_{\boldsymbol{x}}V^i(t,\boldsymbol{x})$ is the optimal map defining $\boldsymbol{Z}_t^i$ in the variational problem (12). Hence we choose $\Sigma(t_k,\boldsymbol{x})\nabla_{\boldsymbol{x}}\mathrm{Net}(t_k,\boldsymbol{x})$ to be a hypothesis function in $\mathcal{N}_k^i$. In other words, the hypothesis functions in $\mathcal{N}_0^{i'}$ and $\{\mathcal{N}_k^i\}_{k=0}^{N_T-1}$ all share the same set of parameters. In this work, we use a fully-connected feedforward network with three hidden layers to instantiate $\mathrm{Net}(t,\boldsymbol{x})$. The detailed architecture of $\mathrm{Net}(t,\boldsymbol{x})$ are provided in Appendix B.

Second, at each stage m, we seek for an approximate solution to the PDE (8) for each player i, by iteratively updating the parameters of the neural networks. Note that the neural networks for N players are decoupled and can be optimized in parallel. In our practice, the Adam method (Kingma and Ba (2015)), a variant of SGD, is used to optimize the parameters. A relevant question is how many SGD iteration steps should be used per stage. After testing with different choices, we decide

to use a moderate number as the iteration steps (see the discussion in Section 4.1 for details). In fact, it is unwise to solve (8) accurately at each stage with a lot of SGD updates, especially at early stages, as the opponents' strategies used for computing the best response are not even close to the Nash equilibrium. A similar idea of solving individual control problems not so accurately at each stage has also been used in Seale and Burnett (2006). On the other hand, since the parameters are continued to be updated incrementally from the previous stage without re-initialization, it still suffices to expect the algorithm to converge with a moderate number of SGD updates per stage.

With the implementation details explained above, the pseudo-code of the proposed deep fictitious play algorithm is summarized in Algorithm 1.

Algorithm 1 Deep Fictitious Play for Finding Markovian Nash Equilibrium

Require: N = # of players, N_T = # of subintervals on $[0,T]$, M = # of total stages in fictitious play, N_{sample} = # of sample paths generated for each player at each stage of fictitious play, $N_{\text{SGD\_per\_stage}}$ = # of SGD steps for each player at each stage, N_{batch} = batch size per SGD update, $\boldsymbol{\alpha}^0$: the initial policies that are smooth enough

Initialize N deep neural networks to represent $V^{i,0}, i \in \mathcal{I}$
for $m \leftarrow 1$ to M **do**
  for all $i \in \mathcal{I}$ **do in parallel**
    Generate N_{sample} sample paths $\{\boldsymbol{X}_k^{i,\pi}\}_{k=0}^{N_T}$ according to (14) and the realized optimal policies $\boldsymbol{\alpha}^{-i,m-1}(t_k, \boldsymbol{X}_k^{i,\pi})$
    for $\ell \leftarrow 1$ to $N_{\text{SGD\_per\_stage}}$ **do**
      Update the parameters of the i^{th} neural network one step with N_{batch} paths using the SGD algorithm (or its variant), based on the loss function (13)
    end for
    Obtain the approximate optimal policy $\alpha^{i,m}$ according to (7)
  end for
  Collect the optimal policies at stage m: $\boldsymbol{\alpha}^m \leftarrow (\alpha^{1,m}, \ldots, \alpha^{N,m})$
end for
return The optimal policy $\boldsymbol{\alpha}^M$

4. Numerical Examples

In this section, we illustrate our algorithm on four examples, including games with identical or heterogeneous agents, and risk-neutral or risk-sensitive cost. In the main text we shall mainly focus on introducing the game setups and presenting numerical results. Technical details, such as concrete PDEs we aim to solve and the associated ground truth solutions V^i, will be kept to the minimal level. They, together with the hyperparameters and runtime of learning, are deferred to Appendices A and B.

4.1. An Inter-Bank Borrowing and Lending Game

Our first example models an inter-bank game concerning the systemic risk (Carmona et al. (2015)). Consider an inter-bank market with N banks, and let $X_t^i \in \mathbb{R}$ be the log-monetary reserves of bank

i at time t. We model its dynamics as the following diffusion processes,

$$\mathrm{d}X_t^i = [a(\overline{X}_t - X_t^i) + \alpha_t^i]\,\mathrm{d}t + \sigma\left(\rho\,\mathrm{d}W_t^0 + \sqrt{1-\rho^2}W_t^i\right), \quad \overline{X}_t = \frac{1}{N}\sum_{i=1}^N X_t^i, \quad i \in \mathcal{I}. \tag{16}$$

Here $a(\overline{X}_t - X_t^i)$ represents the rate at which bank i borrows from or lends to other banks in the lending market, while α_t^i denotes its control rate of cash flows to a central bank. The standard Brownian motions $\{W_t^i\}_{i=0}^N$ are independent, in which $\{W_t^i, i \geq 1\}$ stands for the idiosyncratic noises and W_t^0 denotes the systemic shock, or so-called common noise in the general context. To describe the model in the form of (1), we concatenate the log-monetary reserves X_t^i of N banks to form $\boldsymbol{X}_t^{\boldsymbol{\alpha}} = [X_t^1, \ldots, X_t^N]^{\mathrm{T}}$. The associated drift term and diffusion term are defined as

$$b(t, \boldsymbol{x}, \boldsymbol{\alpha}) = [a(\bar{x} - x^1) + \alpha^1, \ldots, a(\bar{x} - x^N) + \alpha^N]^{\mathrm{T}} \in \mathbb{R}^{N\times 1}, \quad \bar{x} = \frac{1}{N}\sum_{i=1}^N x^i, \tag{17}$$

$$\Sigma(t, \boldsymbol{x}) = \begin{bmatrix} \sigma\rho & \sigma\sqrt{1-\rho^2} & 0 & \cdots & 0 \\ \sigma\rho & 0 & \sigma\sqrt{1-\rho^2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma\rho & 0 & 0 & \cdots & \sigma\sqrt{1-\rho^2} \end{bmatrix} \in \mathbb{R}^{N\times(N+1)}, \tag{18}$$

and $\boldsymbol{W}_t = (W_t^0, \ldots, W_t^N)$ is $(N+1)$-dimensional. The cost functional (2) that player i wishes to minimize has the form

$$f^i(t, \boldsymbol{x}, \boldsymbol{\alpha}) = \frac{1}{2}(\alpha^i)^2 - q\alpha^i(\bar{x} - x^i) + \frac{\epsilon}{2}(\bar{x} - x^i)^2, \quad g^i(\boldsymbol{x}) = \frac{c}{2}(\bar{x} - x^i)^2.$$

All the aforementioned parameters $a, \sigma, q, \epsilon, c$ are non-negative with $|\rho| \leq 1$ and $q^2 \leq \epsilon$. We direct the interested readers to Carmona et al. (2015) for the detailed interpretation of this model.

The coupled HJB system corresponding to this game reads

$$\begin{cases} \partial_t V^i + \inf\limits_{\alpha^i}\left\{\sum\limits_{j=1}^N [a(\bar{x} - x^j) + \alpha^j]\partial_{x^j} V^i + \dfrac{(\alpha^i)^2}{2} - q\alpha^i(\bar{x} - x^i) + \dfrac{\epsilon}{2}(\bar{x} - x^i)^2\right\} \\ \qquad\qquad + \dfrac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}}\mathrm{Hess}_{\boldsymbol{x}} V^i \Sigma) = 0, \\ V^i(T, \boldsymbol{x}) = \dfrac{c}{2}(\bar{x} - x^i)^2, \quad i \in \mathcal{I}. \end{cases} \tag{19}$$

The minimizer in the infimum gives a candidate of the optimal control for player i: $\alpha^i(t, \boldsymbol{x}) = q(\bar{x} - x^i) - \partial_{x^i} V^i(t, \boldsymbol{x})$. Plugging it back into the i^{th} equation yields a PDE of form (8):

$$\begin{aligned} \partial_t V^i + \frac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}}\mathrm{Hess}_{\boldsymbol{x}} V^i \Sigma) + a(\bar{x} - x^i)\partial_{x^i} V^i + \sum_{j\neq i}[a(\bar{x} - x^j) + \alpha^j(t, \boldsymbol{x})]\partial_{x^j} V^i \\ + \frac{\epsilon}{2}(\bar{x} - x^i)^2 - \frac{1}{2}(q(\bar{x} - x^i) - \partial_{x^i} V^i)^2 = 0, \end{aligned}$$

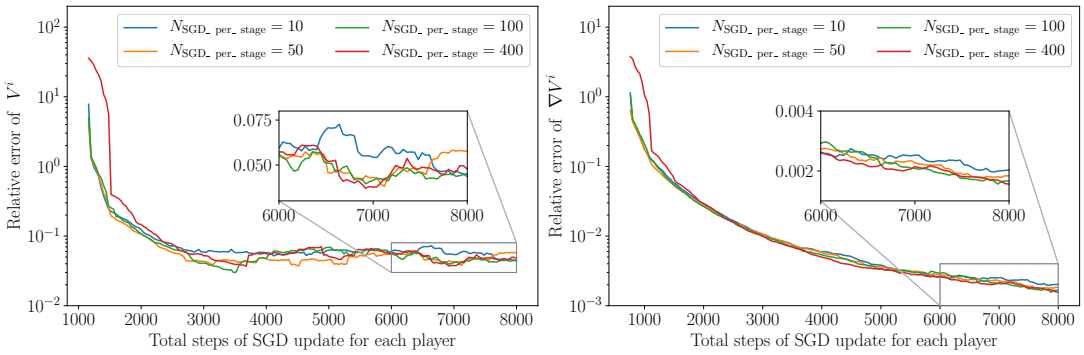

Figure 1: The relative squared errors of V^i (left) and ∇V^i (right) along the training process of deep fictitious play for the inter-bank game in Section 4.1. The relative squared errors of $V^i(0, \boldsymbol{X}_0^{i,\pi})$ and $\{\nabla V^i(t_k, \boldsymbol{X}_k^{i,\pi})\}_{k=0}^{N_T-1}$ are evaluated. The error is computed every 400 SGD updates, averaged over all the players. A smoothed moving average with window size 3 is applied in the final plots.

where α^j with $j \neq i$ are considered exogenous for player i's problem, and are given by the best responses of the other players from the previous stage. To be precise, μ^i and h^i in (8) are defined as:

$$\mu^i(t, \boldsymbol{x}; \boldsymbol{\alpha}^{-i}) = [a(\bar{x} - x^1) + \alpha^1, \ldots, a(\bar{x} - x^i), \ldots, a(\bar{x} - x^N) + \alpha^N]^{\mathrm{T}},$$
$$h^i(t, \boldsymbol{x}, y, \boldsymbol{z}; \boldsymbol{\alpha}^{-i}) = \frac{\epsilon}{2}(\bar{x} - x^i)^2 - \frac{1}{2}(q(\bar{x} - x^i) - \frac{z^i}{\sigma\sqrt{1-\rho^2}})^2,$$

in which $\boldsymbol{z} = (z^0, z^1, \ldots, z^N) \in \mathbb{R}^{N+1}$.

Figures 1–2 show the performance of our algorithm on a 10-player game, using the parameter:

$$a = 0.1, \quad q = 0.1, \quad c = 0.5, \quad \epsilon = 0.5, \quad \rho = 0.2, \quad \sigma = 1, \quad T = 1. \tag{20}$$

We define the relative squared error by

$$\text{RSE} = \frac{\sum_{\substack{i\in\mathcal{I}\\1\le j\le J}} \left(V^i(0, \boldsymbol{x}_{t_0}^{(j)}) - \widehat{V}^i(0, \boldsymbol{x}_{t_0}^{(j)})\right)^2}{\sum_{\substack{i\in\mathcal{I}\\1\le j\le J}} \left(V^i(0, \boldsymbol{x}_{t_0}^{(j)}) - \bar{V}^i\right)^2}, \text{ or } \text{RSE} = \frac{\sum_{\substack{i\in\mathcal{I}\\0\le k\le N_T-1\\1\le j\le J}} \left(\nabla_{\boldsymbol{x}} V^i(t_k, \boldsymbol{x}_{t_k}^{(j)}) - \nabla_{\boldsymbol{x}} \widehat{V}^i(t_k, \boldsymbol{x}_{t_k}^{(j)})\right)^2}{\sum_{\substack{i\in\mathcal{I}\\0\le k\le N_T-1\\1\le j\le J}} \left(\nabla_{\boldsymbol{x}} V^i(t_k, \boldsymbol{x}_{t_k}^{(j)}) - \overline{\nabla_{\boldsymbol{x}} V^i}\right)^2},$$

where V^i is given by the explicit formula provided in Appendix A.1, $\hat{V}^i$ is the prediction from the neural networks, and $\bar{V}^i$ (*resp.* $\overline{\nabla_{\boldsymbol{x}} V^i}$) is the average of V^i (*resp.* $\nabla_{\boldsymbol{x}} V^i$) evaluated at all the indices j, k. To compute the relative error, we generate $J = 256$ ground truth sample paths $\{\boldsymbol{x}_{t_k}^{(j)}\}_{k=0}^{N_T-1}$ using Euler scheme based on (1)(17)(18) and the true optimal strategy (provided in Appendix A.1). Note that the superscript (j) here does not mean the player index, but the j^{th} path for all players. The relative errors reported in Sections 4.2 and 4.3 are defined in the same way. Figure 1 in particular compares the relative squared error as $N_{\text{SGD\_per\_stage}}$ varies from 10 to 400. The convergence of the learning curves with small $N_{\text{SGD\_per\_stage}}$ asserts that each individual problem does not need to

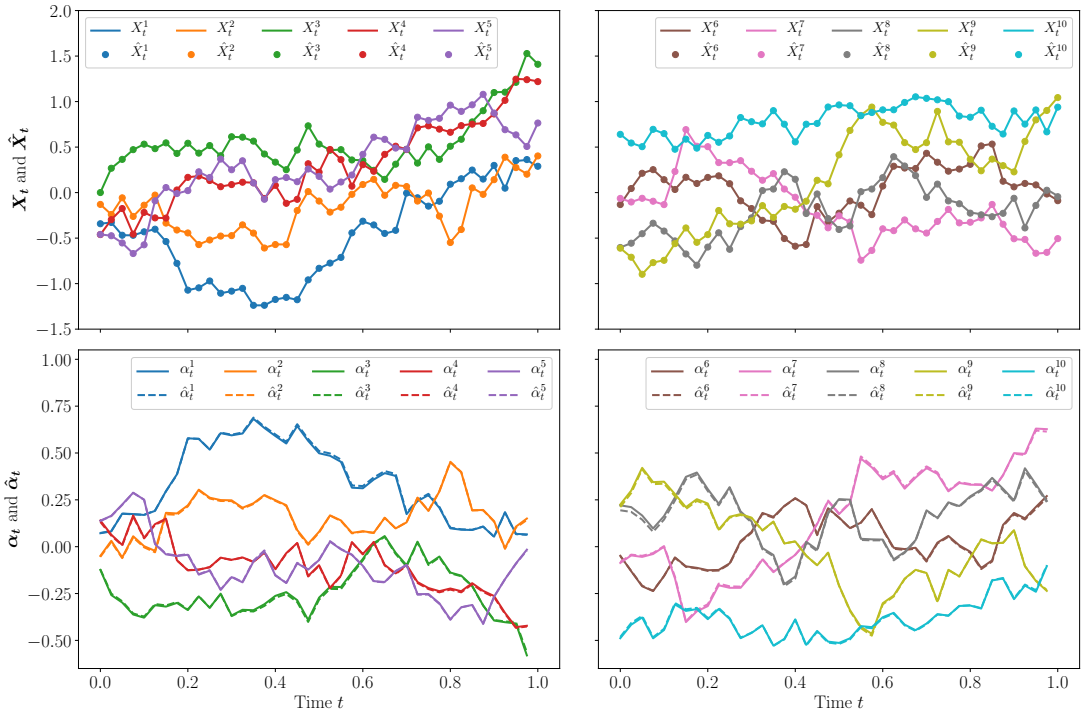

Figure 2: A sample path for each player of the inter-bank game in Section 4.1 with $N = 10$. Top: the optimal state process X_t^i (solid lines) and its approximation $\hat{X}_t^i$ (circles) provided by the optimized neural networks, under the same realized path of Brownian motion. Bottom: comparisons of the strategies α_t^i and $\hat{\alpha}_t^i$ (dashed lines).

be solved so accurately. Furthermore, the similar performances under different $N_{\text{SGD\_per\_stage}}$ with the same total budget of SGD updates suggests that the algorithm is insensitive to the choice of this hyperparameter. In all the following numerical experiments, we fix $N_{\text{SGD\_per\_stage}} = 100$. The final relative squared errors of V and ∇V averaged from three independent runs of deep fictitious play are 4.6% and 0.2%, respectively. Figure 2 presents one sample path for each player of the optimal state process X_t^i and the optimal control α_t^i *vs.* their approximations $\hat{X}_t^i, \hat{\alpha}_t^i$ provided by the optimized neural networks.

One concern is that how sensitive the numerical result is to the parameters chosen in (20). For instance, in solving mean-field games, the numerical algorithms may produce bifurcations (cf. Angiuli et al. (2018); Chassagneux et al. (2019)). To show the robustness (within a certain range of parameters) of our algorithm, we conduct two experiments by increasing the game length T, and the coupling between players described by a. The results are provided in Appendix C.

4.2. A Risk-Sensitive Version of the Inter-Bank Game in Section 4.1

Next we consider the same inter-bank game as in Section 4.1, but under risk-sensitive utility (3). The cases $\theta_i < 0$, $\theta_i > 0$ correspond respectively to what an economist would term *risk-preferring* and *risk-averse* attitudes on the expected utility. Finding the Nash equilibrium of this game is reduced

to solving N-coupled matrix Riccati equations. Its derivation, as well as the concrete form of PDE (8) is presented in Appendix A.2.

For numerical illustration, we study a game of 10 heterogeneous players, with risk-sensitivity $\theta_i = 0.6 + 0.02i$. Other parameters follow (20) except $c = 0.3, \epsilon = 0.3$. As before, Figure 3 compares the sample paths of the true optimal states/control *vs.* the approximated ones produced by the proposed deep fictitious play algorithm. The final relative squared errors of V and ∇V averaged from three independent runs are 1.4% and 0.1%, respectively.

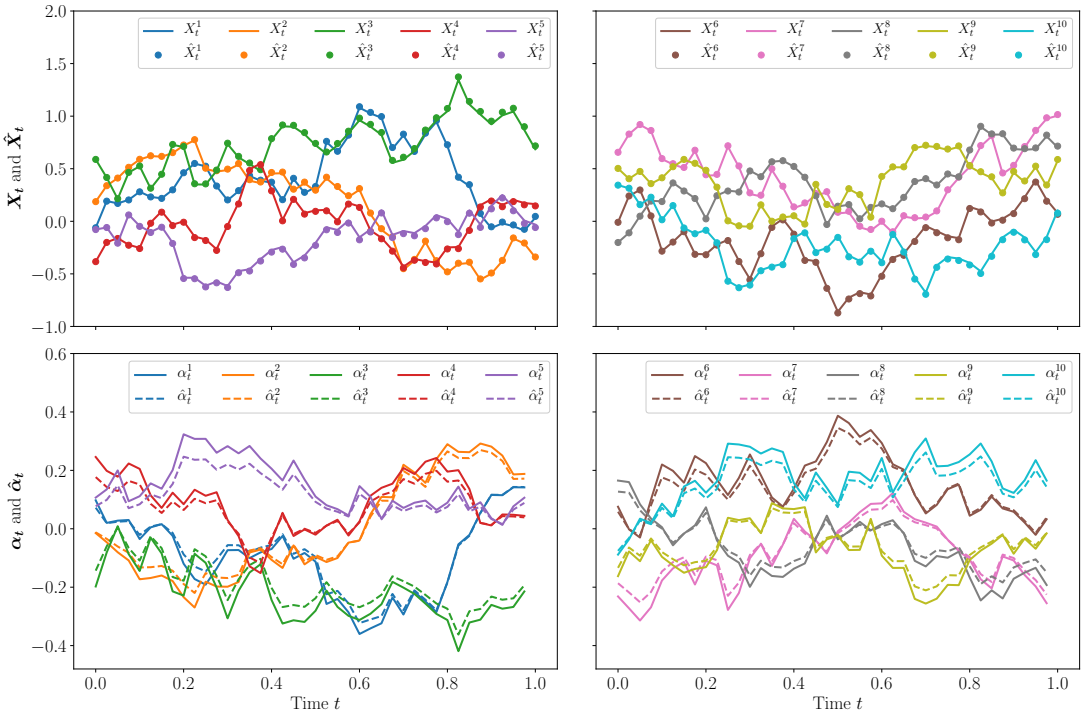

Figure 3: A sample path for each player of the risk-sensitive inter-bank game in Section 4.2 with $N = 10$. Top: the optimal state process X_t^i (solid lines) and its approximation $\hat{X}_t^i$ (circles) provided by the optimized neural networks, under the same realized path of Brownian motion. Bottom: comparisons of the strategies α_t^i and $\hat{\alpha}_t^i$ (dashed lines).

4.3. A General Linear-Quadratic Risk-Sensitive Dynamic Game

In this section, we consider a general linear-exponential-quadratic game of the form (1):

$$b(t, \boldsymbol{x}, \boldsymbol{\alpha}) = A\boldsymbol{x} + \sum_{i=1}^{N} B_i \alpha^i(t, \boldsymbol{x}), \quad \Sigma(t, \boldsymbol{x}) = \Sigma,$$

where $A \in \mathbb{R}^{n\times n}$, $B_i \in \mathbb{R}^{n\times n_i}$, $\Sigma \in \mathbb{R}^{n\times n}$ are constant matrices, Σ is of full rank, $\boldsymbol{x} \in \mathbb{R}^n, \alpha^i \colon [0,T] \times \mathbb{R}^n \to \mathbb{R}^{n_i}$. In contrast to the first two examples in which each player i can only control her own state X^i, here the control α^i of each player i contributes to the dynamics of

the common state $\boldsymbol{X}$ through a general linear relationship. The cost functional for each player is risk-sensitive of the form (3) with:

$$f^i(t,\boldsymbol{x},\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}Q_i\boldsymbol{x} + \frac{1}{2}(\alpha^i)^{\mathrm{T}}R_i\alpha^i, \quad g^i(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}M_i\boldsymbol{x},$$

where $Q_i \in \mathbb{R}^{n\times n}$, $R_i \in \mathbb{R}^{n_i\times n_i}$ and $M_i \in \mathbb{R}^{n\times n}$ are symmetric, positive definite constant matrices.

As before, one can solve N-coupled matrix Riccati equations to obtain the Nash equilibrium. The technical details are deferred to Appendix A.3, and we shall focus on the numerics. We present a 10-player heterogeneous game in which the dimensions of the state $\boldsymbol{X}_t$ and the controls α^i are all 10, i.e., $n = n_i = 10$. The risk-sensitive parameters are the same as in Section 4.2, i.e., $\theta_i = 0.6 + 0.02i$. Denote by $I_{10} \in \mathbb{R}^{10\times 10}$ the identity matrix, and let $B_i = R_i = 0.4I_{10}, \Sigma = I_{10}$. The matrices A, Q_i, M_i are defined by

$$A = 0.1I_{10} + 0.05(\tilde{A} + \tilde{A}^{\mathrm{T}}), \quad Q_i = 0.1I_{10} + 0.05(\tilde{Q}_i + \tilde{Q}_i^{\mathrm{T}}), \quad M_i = 0.2I_{10} + 0.1(\tilde{M}_i + \tilde{M}_i^{\mathrm{T}}).$$

Here $\tilde{A}, \tilde{Q}_i, \tilde{M}_i$ are all random matrices whose diagonal entries are 0 and off-diagonal entries are independently sampled from the uniform distribution on $[-1, 1]$. Hence the game is entirely heterogeneous due to different θ_i, Q_i, M_i. As before, we illustrate the sample paths of the true optimal state process/policy *vs.* the approximated ones in Figure 4. The final relative squared errors of V and ∇V averaged from three independent trials are 6.5% and 0.4%, respectively.

4.4. A Variant of the Inter-Bank Game in Section 4.1 with $N = 50$

In this section we first recompute the example in Section 4.1 with all the same parameters as in (20) but $N = 50$ to test the performance of the proposed deep fictitious play with even more players. Since this game is totally symmetric for all the players, the fictitious play in fact reduces it into N identical decision problems. Due to this, at each stage of the fictitious play, if all the players use the same strategy, then the updated strategies for them are still essentially the same. Therefore we can leverage such symmetry of the game to reduce the computational cost when N is large. Specifically, we impose that all the players always share the same strategy at each stage by using a single network to derive each player's strategy. Consequently, at each stage, only one player's decision problem needs to be solved by the deep BSDE method to update the neural network-based strategy. Algorithm 1 is accordingly simplified, whose details are provided in the Appendix B. After solving (19) with the deep fictitious play, we simulate 10,000 paths following the optimized neural network, and we plot the histograms of $\hat{\boldsymbol{X}}_T$ and $\hat{\boldsymbol{\alpha}}_T$ in Figure 5 by considering all 50 components together, i.e., the histograms are generated by 500,000 data points. Note that due to all the symmetry of this problem, the distribution of each component is identical. Therefore considering them together empirically does not change the underlying distribution but reduces the variance. Figure 5 shows great consistency between the histograms obtained from the optimized strategies/paths (blue dashed lines) and true optimal strategies/paths (black solid lines).

Convinced by the reliability of the algorithm with a large number of players, we further consider a variant of the example in Section 4.1 with $N = 50$ again. In contrast to (16), the drift term becomes nonlinear

$$\mathrm{d}X_t^i = [a(\overline{X}_t - X_t^i)^3 + \alpha_t^i]\,\mathrm{d}t + \sigma\left(\rho\,\mathrm{d}W_t^0 + \sqrt{1-\rho^2}W_t^i\right). \tag{21}$$

In this case, as far as the authors are aware, there is no analytic solution or simple characterization of the Nash equilibrium suitable for numerical computation. For the corresponding mean-field game,

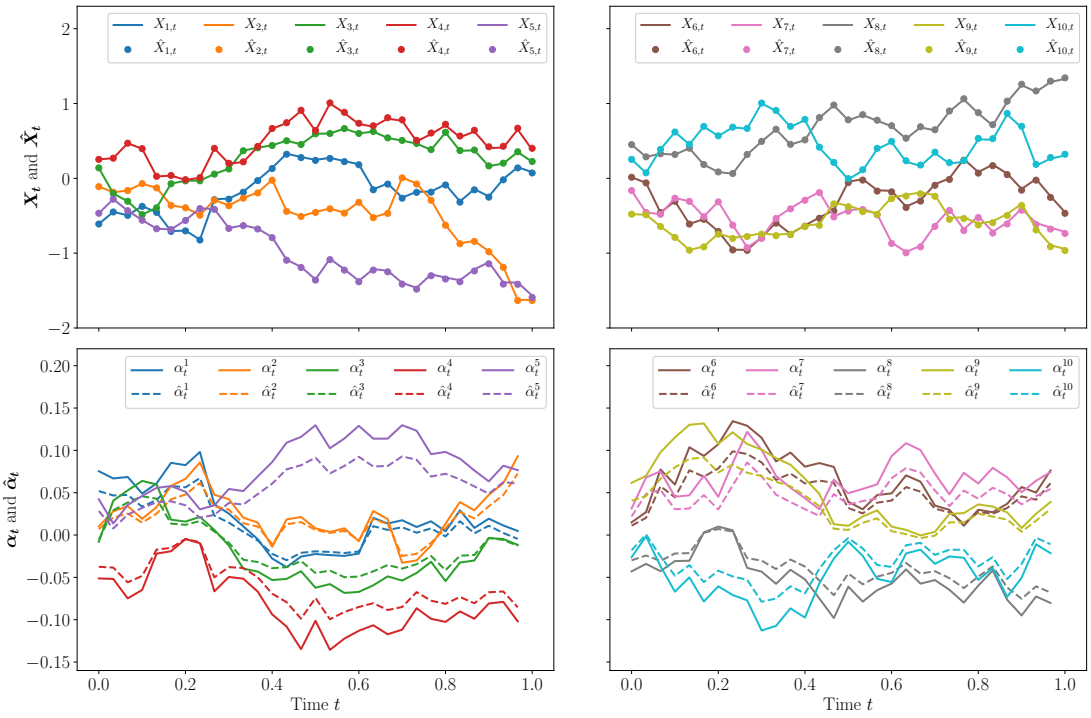

Figure 4: A sample path for each player of the general linear-exponential-quadratic game in Section 4.3 with $N = 10$. Top: the optimal state process $X_{i,t}$ (solid lines) and its approximation $\hat{X}_{i,t}$ (circles) provided by the optimized neural networks, under the same realized path of Brownian motion. Bottom: comparisons of the strategies α_t^i and $\hat{\alpha}_t^i$ (dashed lines).

due to the presence of common noise, the Nash equilibrium is characterized by a coupled system of stochastic partial differential equations, which is also very difficult, if not impossible, to solve numerically. In computation of the deep fictitious play, we set $a = 10$ in order to compensate the smaller drift near 0 and all other parameters the same as in (20). It turns out that the algorithm still finds reasonable equilibrium for this problem, as shown in Figure 5 as well. For X_T^i, since the drift term $a(\overline{X}_t - X_t^i)^3$ is superlinear, the final distribution of X_T^i is expected to be more concentrated than the one under linear drift, which is normal distributed with kurtosis 3. It is confirmed in the left panel of Figure 5, in which the kurtosis of the orange dashed line is $2.72 < 3$. For α_T^i, if X_t^i is far away from the average, the superlinear term $a(\overline{X}_t - X_t^i)^3$ will push it back quickly, saving some effort of bank i and reducing α_t^i. Therefore the tail of α_T^i is much lighter than the Gaussian distribution (with kurtosis $2.36 < 3$), as shown in the right panel of Figure 5.

5. Conclusion

In this paper, we propose a deep fictitious play algorithm to compute the Markovian Nash equilibrium of large N-player stochastic differential games. The game is firstly decoupled into N individual decision problems by the idea of fictitious play, and then each is solved iteratively. Due to the

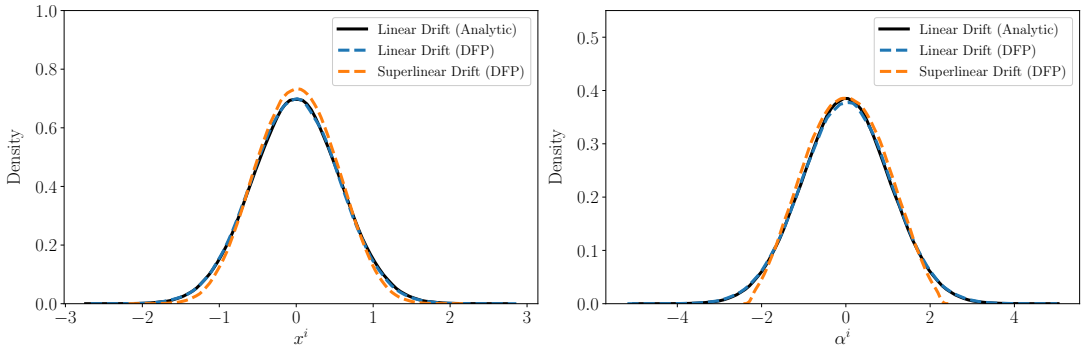

Figure 5: The densities of X_T^i (left) and α_T^i (right) of the inter-bank game with $N = 50$. We plot the true distribution of the linear dynamics (16) (black solid lines), and the distributions obtained from the deep fictitious play with the linear dynamics (16) (blue dashed lines) and superlinear dynamics (21) (orange dashed lines).

feedback nature of Markovian Nash equilibrium, it is inefficient to directly parameterize the optimal policy as in Hu (2019). Instead, we rely on the HJB approach and solve it through the deep BSDE method. Three examples with closed-form solutions are carefully examined, and the algorithm performs unanimously well. The fourth example without tractability is also presented evidencing the algorithm's applicability to general games without the linear-(exponential)-quadratic structure.

Future work includes the convergence analysis on the proposed algorithm and more practical applications in other disciplines such as operation research and economics. For stochastic differential games involving controlling volatility, it is also promising to use the deep fictitious play to find the Nash equilibrium, by combining a second-order BSDE formulation.

Acknowledgments

Part of this work was done during the visit of JH and RH to the Beijing Institute of Big Data Research, China. They really appreciate the hospitality of the institute, and thank Professor Weinan E for hosting and useful discussions.

References

Y. Achdou, J. Han, J.-M. Lasry, P.-L. Lions, and B. Moll. Income and wealth distribution in macroeconomics: A continuous-time approach. Working Paper 23732, National Bureau of Economic Research, August 2017.

A. Angiuli, C. V. Graves, H. Li, J.-F. Chassagneux, F. Delarue, and R. Carmona. Numerical probabilistic approach to mfg. *arXiv preprint arXiv:1805.02406*, 2018.

Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2 (1):1–127, 2009.

A. Bensoussan, C. C. Siu, S. C. P. Yam, and H. Yang. A class of non-zero-sum stochastic differential investment and reinsurance games. *Automatica*, 50(8):2025–2037, 2014.

U. Berger. Fictitious play in $2\times$ n games. *Journal of Economic Theory*, 120(2):139–154, 2005.

A. Briani and P. Cardaliaguet. Stable solutions in potential mean field game systems. *Nonlinear Differential Equations and Applications*, 25(1):1, 2018.

G. W. Brown. Some notes on computation of games solutions. Technical report, Rand Corp Santa Monica CA, 1949.

G. W. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 13(1):374–376, 1951.

L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

P. Cardaliaguet and S. Hadikhanloo. Learning in mean field games: the fictitious play. *ESAIM: Control, Optimisation and Calculus of Variations*, 23(2):569–591, 2017.

P. Cardaliaguet and C.-A. Lehalle. Mean field game of controls and an application to trade crowding. *Mathematics and Financial Economics*, 12(3):335–363, 2018.

G. Carleo and M. Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

R. Carmona and F. Delarue. Probabilistic analysis of mean-field games. *SIAM Journal on Control and Optimization*, 51(4):2705–2734, 2013.

R. Carmona, J.-P. Fouque, and L.-H. Sun. Mean field games and systemic risk. *Communications in Mathematical Sciences*, 13(4):911–933, 2015.

J.-F. Chassagneux, D. Crisan, and F. Delarue. Numerical method for fbsdes of mckean–vlasov type. *The Annals of Applied Probability*, 29(3):1640–1684, 2019.

E. J. Dockner, S. Jorgensen, N. Van Long, and G. Sorger. *Differential Games in Economics and Management Science*. Cambridge University Press, 2000.

W. E, J. Han, and A. Jentzen. 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, 2017.

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

J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

D. P. Foster and H. P. Young. On the nonconvergence of fictitious play in coordination games. *Games and Economic Behavior*, 25(1):79–96, 1998.

X. Gao and L.-M. Duan. Efficient representation of quantum many-body states with deep neural networks. *Nature Communications*, 8(1):662, 2017.

D. A. Gomes and V. K. Voskanyan. Extended deterministic mean-field games. *arXiv preprint arXiv:1305.2600*, 2013.

D. A. Gomes and V. K. Voskanyan. Extended deterministic mean-field games. *SIAM Journal on Control and Optimization*, 54(2):1030–1055, 2016.

D. A. Gomes, S. Patrizi, and V. Voskanyan. On the existence of classical solutions for stationary extended mean field games. *Nonlinear Analysis: Theory, Methods & Applications*, 99:49–79, 2014.

J. Han and J. Long. Convergence of the deep BSDE method for coupled FBSDEs. *arXiv:1811.01165*, 2018.

J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

J. Han, C. Ma, Z. Ma, and W. E. Uniformly accurate machine learning-based hydrodynamic models for kinetic equations. *Proceedings of the National Academy of Sciences*, 116(44):21983–21991, 2019.

J. Hofbauer and W. H. Sandholm. On the global convergence of stochastic fictitious play. *Econometrica*, 70(6):2265–2294, 2002.

R. Hu. Deep fictitious play for stochastic differential games. *arXiv preprint arXiv:1903.09376*, 2019.

M. Huang, R. P. Malhamé, and P. E. Caines. Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Communications in Information and Systems*, 6(3):221–252, 2006.

M. Huang, P. E. Caines, and R. P. Malhamé. Large-population cost-coupled LQG problems with nonuniform agents: individual-mass behavior and decentralized ϵ-Nash equilibria. *IEEE Transactions on Automatic Control*, 52(9):1560–1571, 2007.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

J. S. Jordan. Three problems in learning mixed-strategy Nash equilibria. *Games and Economic Behavior*, 5(3):368–386, 1993.

D. Kingma and J. Ba. Adam: a method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

V. Krishna and T. Sjöström. On the convergence of fictitious play. *Mathematics of Operations Research*, 23(2):479–511, 1998.

J.-M. Lasry and P.-L. Lions. Jeux à champ moyen. I. Le cas stationnaire. *C. R. Math. Acad. Sci. Paris*, 9:619–625, 2006a.

J.-M. Lasry and P.-L. Lions. Jeux à champ moyen. II. Horizon fini et contrôle optimal. *C. R. Math. Acad. Sci. Paris*, 10:679–684, 2006b.

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

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

P. Milgrom and J. Roberts. Adaptive and sophisticated learning in normal form games. *Games and Economic Behavior*, 3(1):82–100, 1991.

K. Miyasawa. On the convergence of the learning process in a 2×2 non-zero-sum two-person game. Technical report, Princeton University NJ, 1961.

D. Monderer and A. Sela. A $2\times$ 2 game without the fictitious play property. *Games and Economic Behavior*, 14(1):144–148, 1996.

D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996a.

D. Monderer and L. S. Shapley. Fictitious play property for games with identical interests. *Journal of Economic Theory*, 68(1):258–265, 1996b.

E. Pardoux and S. Peng. Backward stochastic differential equations and quasilinear parabolic partial differential equations. In *Stochastic Partial Differential Equations and Their Applications*, pages 200–217. Springer, 1992.

E. Pardoux and S. Tang. Forward-backward stochastic differential equations and quasilinear parabolic PDEs. *Probability Theory and Related Fields*, 114(2):123–150, 1999.

A. Prasad and S. P. Sethi. Competitive advertising under uncertainty: A stochastic differential game approach. *Journal of Optimization Theory and Applications*, 123(1):163–185, 2004.

J. Robinson. An iterative method of solving a game. *Annals of Mathematics*, pages 296–301, 1951.

A. Sannai, Y. Takai, and M. Cordonnier. Universal approximations of permutation invariant/equivariant functions by deep neural networks. *arXiv preprint arXiv:1903.01939*, 2019.

D. A. Seale and J. E. Burnett. Solving large games with simulated fictitious play. *International Game Theory Review*, 8(03):437–467, 2006.

L. S. Shapley. Some topics in two-person games. *Advances in Game Theory*, 52:1–29, 1964.

N. Van Long. Dynamic games in the economics of natural resources: a survey. *Dynamic Games and Applications*, 1(1):115–148, 2011.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017.

K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5872–5881, 2018a.

L. Zhang, J. Han, H. Wang, R. Car, and W. E. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical Review Letters*, 120(14):143001, 2018b.

L. Zhang, J. Han, H. Wang, W. Saidi, R. Car, and W. E. End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems. In *Advances in Neural Information Processing Systems*, pages 4436–4446, 2018c.

Appendix A. Technical Details to Numerical Examples

A.1. The Analytic Solution in Section 4.1

The results in this section is firstly derived in (Carmona et al., 2015, Section 3), and we repeat them here for completeness. Assume the following ansatz for the HJB system (19):

$$V^i(t,\boldsymbol{x}) = \frac{\eta(t)}{2}(\bar{x}-x^i)^2 + \mu(t), \quad i \in \mathcal{I},$$

where $\eta(t), \mu(t)$ are two scalar functions to be determined. Under this ansatz the optimal feedback control becomes

$$\alpha^{i,*}(t,\boldsymbol{x}) = \left[q + \eta(t)(1-\frac{1}{N})\right](\bar{x}-x^i).$$

Plugging the ansatz into (19) and collecting the coefficients of the squared and constant terms, we find that $\eta(t)$ solves a Riccati equation

$$\dot{\eta}(t) = 2(a+q)\eta(t) + (1-1/N^2)\eta^2(t) - (\epsilon - q^2), \quad \eta(T) = c,$$

and $\mu(t)$ depends on $\eta(t)$ through:

$$\dot{\mu}(t) = -\frac{1}{2}\sigma^2(1-\rho^2)(1-1/N)\eta(t), \quad \mu(T) = 0.$$

The solution to the Riccati equation is

$$\eta(t) = \frac{-(\epsilon-q^2)(e^{(\delta^+-\delta^-)(T-t)}-1) - c(\delta^+ e^{(\delta^+-\delta^-)(T-t)} - \delta^-)}{(\delta^- e^{(\delta^+-\delta^-)(T-t)} - \delta^+) - c(1-1/N^2)(e^{(\delta^+-\delta^-)(T-t)}-1)},$$

where $\delta^\pm = -(a+q) \pm \sqrt{R}$, and $R = (a+q)^2 + (1-1/N^2)(\epsilon - q^2)$.

A.2. Technical Details in Section 4.2

In this risk-sensitive dynamic game, the Nash equilibrium is related to the following system:

$$\begin{cases} \partial_t V^i + \inf\limits_{\alpha^i}\left\{\sum\limits_{j=1}^N [a(\bar{x}-x^j)+\alpha^j]\partial_{x^j}V^i + V^i\theta_i\left(\frac{(\alpha^i)^2}{2} - q\alpha^i(\bar{x}-x^i) + \frac{\epsilon}{2}(\bar{x}-x^i)^2\right)\right\} \\ \qquad + \frac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}}\mathrm{Hess}_{\boldsymbol{x}}V^i\Sigma) = 0, \\ V^i(T,\boldsymbol{x}) = \theta_i \exp\left\{\theta_i\frac{c}{2}(\bar{x}-x^i)^2\right\}, \end{cases} \tag{22}$$

and the candidate of player i's optimal strategy is given by

$$\alpha^i(t,\boldsymbol{x}) = q(\bar{x}-x^i) - \frac{\partial_{x^i}V^i(t,\boldsymbol{x})}{\theta_i V^i(t,\boldsymbol{x})}. \tag{23}$$

Observing the exponential quadratic form in the terminal condition and in the cost functional, we assume the following ansatz

$$V^i(t,\boldsymbol{x}) = \theta_i \exp\left\{\theta_i\left(\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}P^i(t)\boldsymbol{x} + p^i(t)\right)\right\}, \quad i \in \mathcal{I},$$

where $P^i(t) \in \mathbb{R}^{n\times n}, p^i(t) \in \mathbb{R}$ are two (matrix) functions to be determined. Under this ansatz the optimal feedback control becomes

$$\alpha^{i,*}(t,\boldsymbol{x}) = q(\bar{x} - x^i) - P^i(t)\boldsymbol{x}.$$

Plugging the ansatz into (22) and collecting the coefficients of the squared and constant terms, we find that $P^i(t)$ and $p^i(t)$ satisfy

$$\dot{P}^i + 2(a+q)AP^i - 2\sum_{j=1}^{N}(P^j)^{\mathrm{T}}\Delta_{j,j}P^i + (P^i)^{\mathrm{T}}(\Delta_{i,i} + \theta_i\Sigma\Sigma^{\mathrm{T}})P^i + (\epsilon - q^2)e_ie_i^{\mathrm{T}} = 0, \quad P^i(T) = ce_ie_i^T,$$

and

$$\dot{p}^i(t) + \frac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}}P^i(t)\Sigma) = 0, \quad p^i(T) = 0.$$

Here we have used the notation

$$A = (A_{i,j}) = (\frac{1}{N} - \delta_{i,j}), \quad e_i = \Big[\frac{1}{N}, \ldots, \frac{1}{N}, \underset{\substack{\uparrow \\ i^{th}}}{\frac{1}{N} - 1}, \frac{1}{N}, \ldots, \frac{1}{N}\Big]^{\mathrm{T}},$$

$$B_i = [0, \ldots, 0, \underset{\substack{\uparrow \\ i^{th}}}{1}, 0, \ldots, 0]^{\mathrm{T}}, \quad \Delta_{i,i} = B_iB_i^{\mathrm{T}},$$

and Σ is defined as in (17). The solutions of $P^i(t)$ and $p^i(t)$ then can be obtained through high-precision numerical integration. Note that these matrix Riccati equations for $\{P^i(t), i \in \mathcal{I}\}$ are coupled, while $p^i(t)$ depends solely on $P^i(t)$.

To obtain the HJB equation for the individual decision problem decoupled by fictitious play, we plugin (23) into (22) and deduce

$$\begin{aligned}\partial_t V^i + \frac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}}\mathrm{Hess}_{\boldsymbol{x}}V^i\Sigma) + a(\bar{x} - x^i)\partial_{x^i}V^i + \sum_{j\neq i}[a(\bar{x} - x^j) + \alpha^j(t,\boldsymbol{x})]\partial_{x^j}V^i \\ + \theta_iV^i\left(\frac{\epsilon}{2}(\bar{x} - x^i)^2 - \frac{1}{2}\left(q(\bar{x} - x^i) - \frac{\partial_{x^i}V^i}{\theta_iV^i}\right)^2\right) = 0.\end{aligned}$$

This is indeed of the form (8), with $\mu^i(t,\boldsymbol{x};\boldsymbol{\alpha}^{-i})$ following the definition in Section 4.1, and h^i given by:

$$h^i(t,\boldsymbol{x},y,\boldsymbol{z};\boldsymbol{\alpha}^{-i}) = \theta_iy\left(\frac{\epsilon}{2}(\bar{x} - x^i)^2 - \frac{1}{2}(q(\bar{x} - x^i) - \frac{z^i}{\theta_iy\sigma\sqrt{1-\rho^2}})^2\right).$$

A.3. Technical Details in Section 4.3

In this general linear-exponential-quadratic game, the coupled HJB system can be written as

$$\begin{cases} V_t^i + \frac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}}\mathrm{Hess}_{\boldsymbol{x}}V^i\Sigma) + \inf\limits_{\alpha^i\in\mathbb{R}^{n_i}}\left\{\nabla_{\boldsymbol{x}}V^i\cdot(A\boldsymbol{x} + \sum\limits_{j=1}^{N}B_j\alpha^j) + \frac{1}{2}\theta_iV^i\left(\boldsymbol{x}^{\mathrm{T}}Q_i\boldsymbol{x} + (\alpha^i)^{\mathrm{T}}R_i\alpha^i\right)\right\} = 0, \\ V^i(T,\boldsymbol{x}) = \theta_i\exp\left\{\frac{\theta_i}{2}\boldsymbol{x}^{\mathrm{T}}M_i\boldsymbol{x}\right\}, \quad i\in\mathcal{I}. \end{cases} \tag{24}$$

The optimal control satisfies

$$\alpha^i(t,\boldsymbol{x}) = -\frac{R_i^{-1}B_i^{\mathrm{T}}\nabla_{\boldsymbol{x}}V^i(t,\boldsymbol{x})}{\theta_i V_i(t,\boldsymbol{x})}. \tag{25}$$

In order to solve the system (24) directly for the Nash equilibrium, like in the second example, we assume the following ansatz with the exponential form

$$V^i(t,\boldsymbol{x}) = \theta_i \exp\left\{\theta_i\left(\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}P^i(t)\boldsymbol{x} + p^i(t)\right)\right\}, \quad i \in \mathcal{I},$$

where $P^i(t) \in \mathbb{R}^{n\times n}, p^i(t) \in \mathbb{R}$ are two functions of t to be determined. Using this ansatz, the optimal control becomes

$$\alpha^{i,*}(t,\boldsymbol{x}) = -R_i^{-1}B_i^{\mathrm{T}}P^i(t)\boldsymbol{x}.$$

Plugging the ansatz into (24), we deduce that $P^i(t)$ solves a matrix Riccati equation

$$\begin{aligned}\dot{P}^i + (P^i)^{\mathrm{T}}A + A^{\mathrm{T}}P^i + Q_i + (P^i)^{\mathrm{T}}(B_iR_i^{-1}B_i^{\mathrm{T}} + \theta_i\Sigma\Sigma^{\mathrm{T}})P^i \\ - (P^i)^{\mathrm{T}}\sum_{j=1}^N B_jR_j^{-1}B_j^{\mathrm{T}}P^j - (\sum_{j=1}^N B_jR_j^{-1}B_j^{\mathrm{T}}P^j)^{\mathrm{T}}P^i = 0, \quad P^i(T) = M_i,\end{aligned}$$

and $p^i(t)$ solves the ODE

$$\dot{p^i}(t) + \frac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}}P^i(t)\Sigma) = 0, \quad p^i(T) = 0.$$

We remark that these Riccati equations are coupled as well, and can be solved by high-precision numerical integration.

To obtain the HJB equation for the individual decision problem decoupled by fictitious play, we plugin(25) into (24) and deduce the simplified PDE

$$\begin{aligned}V_t^i + \frac{1}{2}\mathrm{Tr}(\Sigma^{\mathrm{T}}\mathrm{Hess}_{\boldsymbol{x}}V^i\Sigma) + \nabla_{\boldsymbol{x}}V^i\cdot(A\boldsymbol{x} + \sum_{j\neq i}B_j\alpha^j) + \frac{1}{2}\theta_iV^i\boldsymbol{x}^{\mathrm{T}}Q_i\boldsymbol{x} \\ - \frac{1}{2}\frac{(\nabla_{\boldsymbol{x}}V^i)^{\mathrm{T}}B_iR_i^{-1}B_i^{\mathrm{T}}\nabla_{\boldsymbol{x}}V^i}{\theta_iV^i} = 0.\end{aligned}$$

This PDE has the same form of (8), with

$$\begin{aligned}\mu^i(t,\boldsymbol{x};\boldsymbol{\alpha}^{-i}) &= A\boldsymbol{x} + \sum_{j\neq i}B_j\alpha^j(t,\boldsymbol{x}), \\ h^i(t,\boldsymbol{x},y,\boldsymbol{z};\boldsymbol{\alpha}^{-i}) &= \frac{1}{2}\theta_iy\boldsymbol{x}^{\mathrm{T}}Q_i\boldsymbol{x} - \frac{1}{2}\frac{\boldsymbol{z}^{\mathrm{T}}\Sigma^{-1}B_iR_i^{-1}B_i^{\mathrm{T}}(\Sigma^{-1})^{\mathrm{T}}\boldsymbol{z}}{\theta_iy}.\end{aligned}$$

Appendix B. Hyperparameters and Details of Deep Fictitious Play

This appendix reports the hyperparameters used in the numerical examples of Section 4. We fix $N_{\text{SGD\_per\_stage}} = 100$ and set $N_{\text{sample}} = N_{\text{SGD\_per\_stage}}N_{\text{batch}}$. This means that the samples are never reused in the learning. In all the numerical examples, the value functions V^i are approximated by

feedforward neural networks $\text{Net}(t, \boldsymbol{x})$ with 3 hidden layers whose widths are all the same. We use *tanh* as the activation function and adopt batch normalization (Ioffe and Szegedy (2015)) right after each affine transformation and before activation. Each component of $\boldsymbol{x}_0$, the initial state $\boldsymbol{X}_0^i$, is sampled independently from the uniform distribution on $[-\delta_0, \delta_0]$. δ_0 is chosen such that in the following process driven by the optimal policy $\boldsymbol{\alpha}^*$

$$\mathrm{d}\boldsymbol{X}_t^{\boldsymbol{\alpha}^*} = b(t, \boldsymbol{X}_t^{\boldsymbol{\alpha}^*}, \boldsymbol{\alpha}^*(t, \boldsymbol{X}_t^{\boldsymbol{\alpha}^*}))\,\mathrm{d}t + \Sigma(t, \boldsymbol{X}_t^{\boldsymbol{\alpha}^*})\,\mathrm{d}\boldsymbol{W}_t, \quad \boldsymbol{X}_0 = \boldsymbol{x}_0,$$

the standard deviation of $\{\boldsymbol{X}_t\}_{t=0}^T$ is approximately δ_0. In other words, δ_0 is determined as a fixed-point. The rationale for such a procedure is to make sure the data generated for the learning is representative enough in the whole state space. We also test other reasonable choices of $\boldsymbol{x}_0$ and find the final accuracy is insensitive to it. For the inter-bank game in Section 4.4 with $N = 50$ and the superlinear dynamics (21), the optimal policy is unavailable for determining δ_0. We instead use δ_0 defined for the game with $N = 50$ and the linear dynamics (16).

Table 1 reports the values of some other hyperparameters used in the numerical examples.

Table 1: Hyperparameters and runtime for the numerical examples presented in Section 4.

Parameters / Problem	Section 4.1	Section 4.2	Section 4.3	Section 4.4
N_T	40	40	30	40
width of hidden layers	40	40	40	60
M (# of total stages)	80	100	200	400
N_{batch}	256	512	512	256
learning rate	5e-4	5e-4	(5e-3, 5e-4)*	5e-4
runtime (hours) †	7	13	20.5	35

*The learning rate is piecewise constant in the example of Section 4.3. It equals 5e-3 for the first half of SGD updates and 5e-4 for the second half.

† The numerical experiments were conducted on an NVIDIA Tesla P100 GPU. The runtime is subject to further reduction with a multi-GPU system.

In Section 4.4 we solve the large scale multi-agent game with $N = 50$. Due to the intrinsic symmetry of the game, we assume all the players essentially share the same strategies during each stage of the fictitious play. Without loss of generality, we only keep track of player 1's strategy and denote its neural network approximation by

$$\alpha^{1,m}(t, x_1, x_2, \ldots, x_N) = \text{Net}(t, x_1, x_2, \ldots, x_N; \boldsymbol{w}), \tag{26}$$

where Net is a neural network and $\boldsymbol{w}$ denotes all the trainable parameters. Then player i's strategy $(i \neq 1)$ is defined by swapping the components x_1 and x_i in the arguments, i.e.,

$$\alpha^{i,m}(t, x_1, x_2, \ldots, x_N) = \text{Net}(t, x_i, \ldots, x_{i-1}, x_1, x_{i+1}, \ldots, x_N; \boldsymbol{w}). \tag{27}$$

This completes the simplification of Algorithm 1, whose pseudo-code is summarized in Algorithm 2 below.

Note that there are other ways besides (27) to define the others' strategies based on (26), for instance,

$$\alpha^{i,m}(t, x_1, x_2, \ldots, x_N) = \text{Net}(t, x_i, \ldots, x_{i+1}, x_1, x_{i-1}, \ldots, x_N; \boldsymbol{w}). \tag{28}$$

Due to the total symmetry of this game, the optimal strategy $\alpha^{1,*}$ should be permutation invariant with respect to the arguments $x_2, \ldots, x_N$, which implies that (27) and (28) are the same if $\text{Net}(\cdot; \boldsymbol{w})$ represents $\alpha^{1,*}$ exactly. In the current algorithm, the plain feedforward neural network does not guarantee this property, but we find different choices (like (27) and (28)) do not make a substantial difference in final accuracy. On the hand, designing specific neural network architectures to guarantee permutation invariance/equivariance exactly has been discussed in depth in the recent literature (Zaheer et al. (2017); Zhang et al. (2018c); Sannai et al. (2019)). It will be of interest to test in future work whether such networks can improve the performance in learning the Nash equilibrium of totally symmetric games.

Algorithm 2 Deep Fictitious Play for Finding Markovian Nash Equilibrium of Symmetric Game

Require: N = # of players, N_T = # of subintervals on $[0, T]$, M = # of total stages in fictitious play, N_{sample} = # of sample paths generated for each player at each stage of fictitious play, $N_{\text{SGD\_per\_stage}}$ = # of SGD steps for each player at each stage, N_{batch} = batch size per SGD update, $\alpha^{1,0}$: the initial smooth policy for player 1

Initialize one deep neural network to represent $V^{1,0}$ for player 1
Define $\alpha^{2,0}, \ldots, \alpha^{N,0}$ according to (27) and collect $\boldsymbol{\alpha}^0 \leftarrow (\alpha^{1,0}, \ldots, \alpha^{N,0})$
for $m \leftarrow 1$ to M **do**
  Generate N_{sample} sample paths $\{\boldsymbol{X}_k^{1,\pi}\}_{k=0}^{N_T}$ according to (14) and the realized optimal policies $\boldsymbol{\alpha}^{-1,m-1}(t_k, \boldsymbol{X}_k^{1,\pi})$
  for $\ell \leftarrow 1$ to $N_{\text{SGD\_per\_stage}}$ **do**
    Update the parameters of the neural network one step with N_{batch} paths using the SGD algorithm (or its variant), based on the loss function (13)
  end for
  Obtain the approximate optimal policy $\alpha^{1,m}$ according to (7)
  Define $\alpha^{2,m}, \ldots, \alpha^{N,m}$ according to (27) and collect the optimal policies at stage m: $\boldsymbol{\alpha}^m \leftarrow (\alpha^{1,m}, \ldots, \alpha^{N,m})$
end for
return The optimal policy $\boldsymbol{\alpha}^M$

Appendix C. Robustness of the Deep Fictitious Play

Regarding robustness of the deep fictitious play, we did two more tests on the inter-bank game in Section 4.1. Compared to the original parameter (see Eq. (20)), we solve two modified cases: (1) increasing the game length $T = 2$ (with N increased to 100) instead of $T = 1$, and (2) increasing the coupling $a = 0.3$ instead of $a = 0.1$. Figures 6 and 7 present the sample paths for each player of the optimal state process X_t^i and the optimal control α_t^i *vs.* their approximations $\hat{X}_t^i, \hat{\alpha}_t^i$ provided by the optimized neural networks. In these two cases, the final RSE of V and ∇V are (1) 4.1% and 0.2%, (2) 4.9% and 0.1%. We have not identified bifurcation in these experiments, which exists in algorithms of solving mean field games (cf. Angiuli et al. (2018)). In other words, the performance of deep fictitious play should be robust at least to a certain range of parameters.

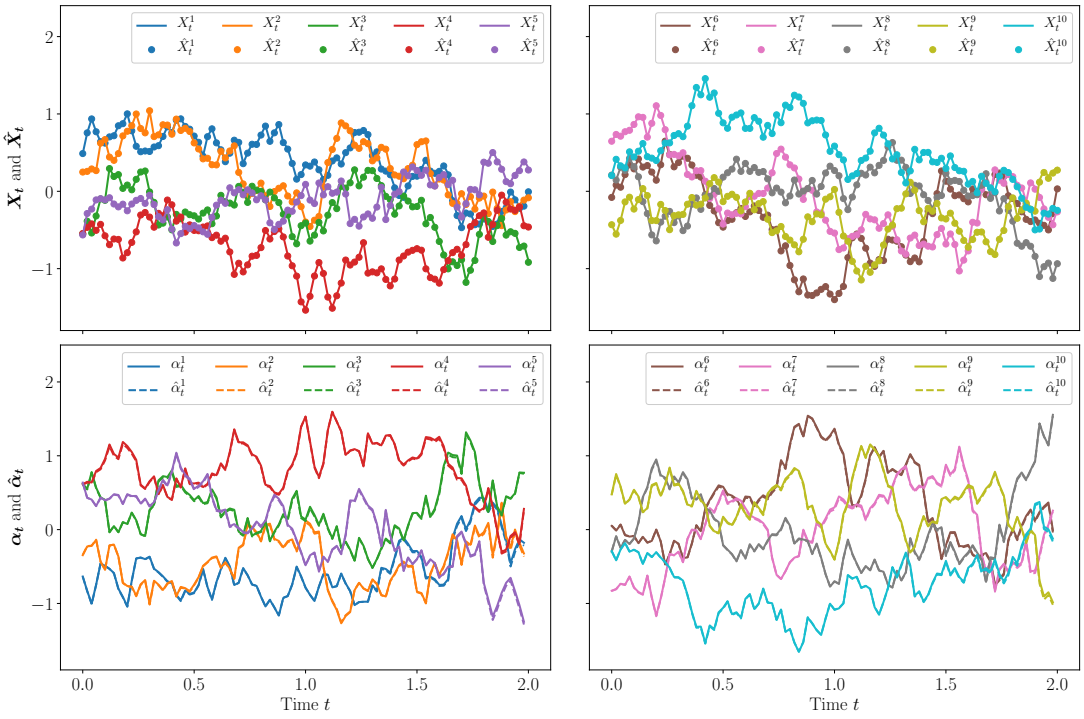

Figure 6: A sample path for each player of the inter-bank game in Section 4.1 with $N = 10$ and the parameters in Eq. (20) except $T = 2$. Top: the optimal state process X_t^i (solid lines) and its approximation $\hat{X}_t^i$ (circles) provided by the optimized neural networks, under the same realized path of Brownian motion. Bottom: comparisons of the strategies α_t^i and $\hat{\alpha}_t^i$ (dashed lines).

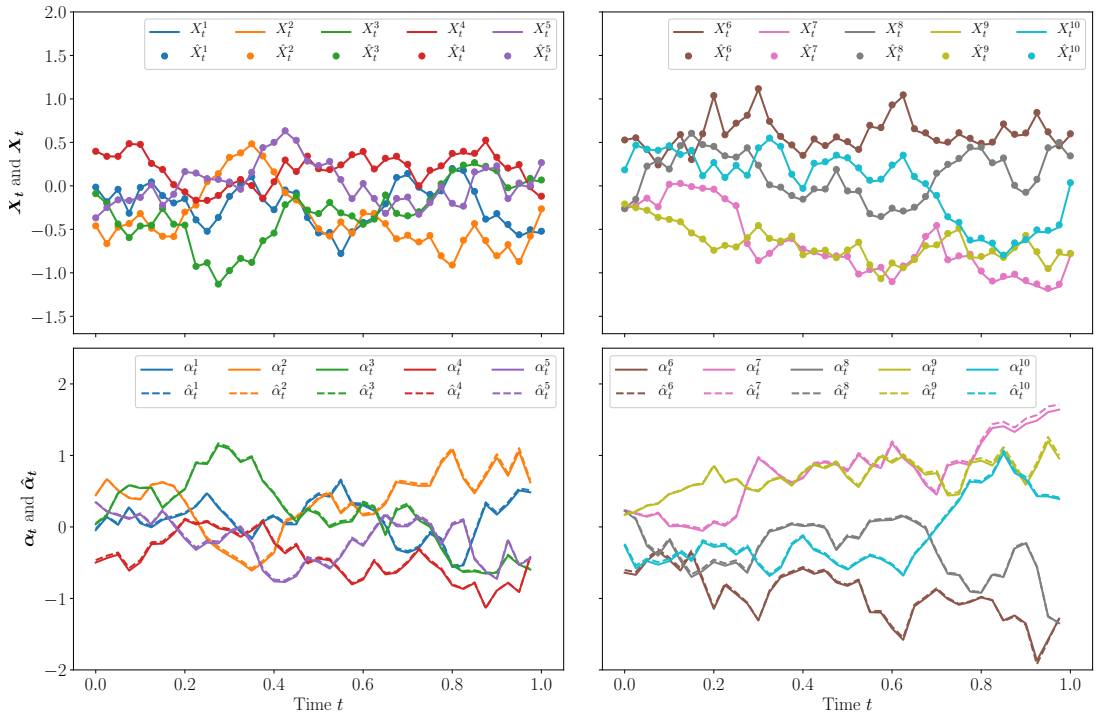

Figure 7: A sample path for each player of the inter-bank game in Section 4.1 with $N = 10$ and the parameters in Eq. (20) except $a = 0.3$. Top: the optimal state process X_t^i (solid lines) and its approximation $\hat{X}_t^i$ (circles) provided by the optimized neural networks, under the same realized path of Brownian motion. Bottom: comparisons of the strategies α_t^i and $\hat{\alpha}_t^i$ (dashed lines).