COMMUN. MATH. SCI.
Vol. 19, No. 2, pp. 325–353

DEEP FICTITIOUS PLAY FOR STOCHASTIC DIFFERENTIAL GAMES*

RUIMENG HU†

Abstract. In this paper, we apply the idea of fictitious play to design deep neural networks (DNNs), and develop deep learning theory and algorithms for computing the Nash equilibrium of asymmetric N-player non-zero-sum stochastic differential games, for which we refer as *deep fictitious play*, a multi-stage learning process. Specifically at each stage, we propose the strategy of letting individual player optimize her own payoff subject to the other players' previous actions, equivalent to solving N decoupled stochastic control optimization problems, which are approximated by DNNs. Therefore, the fictitious play strategy leads to a structure consisting of N DNNs, which only communicate at the end of each stage. The resulting deep learning algorithm based on fictitious play is scalable, parallel and model-free, *i.e.*, using GPU parallelization, it can be applied to any N-player stochastic differential game with different symmetries and heterogeneities (*e.g.*, existence of major players). We illustrate the performance of the deep learning algorithm by comparing to the closed-form solution of the linear quadratic game. Moreover, we prove the convergence of fictitious play under appropriate assumptions, and verify that the convergent limit forms an open-loop Nash equilibrium. We also discuss the extensions to other strategies designed upon fictitious play and closed-loop Nash equilibrium in the end.

Keywords. Stochastic differential game; fictitious play; deep learning; Nash equilibrium.

AMS subject classifications. 91A15; 91B50; 91A26; 68T20; 60G99.

1. Introduction

In stochastic differential games, a Nash equilibrium refers to strategies by which no player has an incentive to deviate. Finding a Nash equilibrium is one of the core problems in noncooperative game theory, however, due to the notorious intractability of N-player game, the computation of the Nash equilibrium has been shown extremely time-consuming and memory demanding, especially for large N [16]. On the other hand, a rich literature on game theory has been developed to study consequences of strategies on interactions between a large group of rational "agents", *e.g.*, system risk caused by inter-bank borrowing and lending, price impacts imposed by agents' optimal liquidation, and market price from monopolistic competition. This makes it crucial to develop efficient theory and fast algorithms for computing the Nash equilibrium of N-player stochastic differential games.

Deep neural networks with many layers have been recently shown to do a great job in artificial intelligence (*e.g.*, [2, 44]). The idea behind is to use compositions of simple functions to approximate complicated ones, and there are approximation theorems showing that a wide class of functions on compact subsets can be approximated by a single hidden layer neural network (*e.g.*, [59]). This brings a possibility of solving a high-dimensional system using deep neural networks, and in fact, these techniques have been successfully applied to solve stochastic control problems [1, 20, 33].

In this paper, we propose to build deep neural networks by using strategies of fictitious play, and develop parallelizable deep learning algorithms for computing the Nash equilibrium of asymmetric N-player non-zero-sum stochastic differential games.

*Received: January 10, 2020; Accepted (in revised form): August 30, 2020. Communicated by Arnulf Jentzen.

†Department of Statistics, Columbia University, New York, NY, 10027-4690, USA. Current position: Department of Mathematics and Department of Statistics and Applied Probability, University of California, Santa Barbara, CA, 93106-3080, USA (rhu@ucsb.edu).
RH was partially supported by the NSF grant DMS-1953035.

We consider a stochastic differential game with N players, and each player $i \in \mathcal{I} := \{1,2,\ldots,N\}$ has a state process $X_t^i \in \mathbb{R}^d$ and takes an action α_t^i in the control set $A \subset \mathbb{R}^k$. The dynamics of the controlled state process $X_\cdot^i$ on $[0,T]$ are given by

$$\mathrm{d}X_t^i = b^i(t, \boldsymbol{X}_t, \boldsymbol{\alpha}_t)\mathrm{d}t + \sigma^i(t, \boldsymbol{X}_t, \boldsymbol{\alpha}_t)\mathrm{d}W_t^i + \sigma^0(t, \boldsymbol{X}_t, \boldsymbol{\alpha}_t)\mathrm{d}W_t^0, \quad X_0^i = x^i, \quad i \in \mathcal{I}, \tag{1.1}$$

where $\boldsymbol{W} := [W^0, W^1, \ldots, W^N]$ are $N+1$ m-dimensional independent Brownian motions, (b^i, σ^i) are deterministic functions: $[0,T] \times \mathbb{R}^{d\times N} \times A^N \hookrightarrow \mathbb{R}^d \times \mathbb{R}^{d\times m}$. The N dynamics are coupled since all private states $\boldsymbol{X}_t = [X_t^1, \ldots, X_t^N]$ and all the controls[1] $\boldsymbol{\alpha}_t = [\alpha_t^1, \ldots, \alpha_t^N]$ affect the drifts b^i and diffusions σ^i.

Each player's control α_t^i lives in the space $\mathbb{A} = \mathbb{H}_T^2(A)$ of progressively measurable A-valued processes satisfying the integrability condition:

$$\mathbb{E}[\int_0^T |\alpha_t^i|^2 \mathrm{d}t] < \infty. \tag{1.2}$$

Using the strategy $\boldsymbol{\alpha} \in \mathbb{A}^N$, the cost associated to player i is of the form:

$$J^i(\boldsymbol{\alpha}) := \mathbb{E}\left[\int_0^T f^i(t, \boldsymbol{X}_t, \boldsymbol{\alpha}_t)\mathrm{d}t + g^i(\boldsymbol{X}_T)\right],$$

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

In solving stochastic differential games, the notion of optimality of common interest is the Nash equilibrium. A set of strategies $\boldsymbol{\alpha}^* = (\alpha^{1,*}, \ldots, \alpha^{N,*}) \in \mathbb{A}^N$ is called a Nash equilibrium if

$$\forall i \in \mathcal{I} \text{ and } \beta^i \in \mathbb{A}, \quad J^i(\boldsymbol{\alpha}^*) \le J^i(\beta^i, \boldsymbol{\alpha}^{-i,*}), \tag{1.3}$$

where $\boldsymbol{\alpha}^{-i,*}$ represents strategies of players other than the i-th one

$$\boldsymbol{\alpha}^{-i,*} := [\alpha^{1,*}, \ldots, \alpha^{i-1,*}, \alpha^{i+1,*}, \ldots, \alpha^{N,*}] \in \mathbb{A}^{N-1}.$$

In fact, depending on the space where one searches for actions (the information structure available to the players), the types of equilibria include open-loop ($\boldsymbol{W}_{[0,t]}$), closed-loop ($\boldsymbol{X}_{[0,t]}$), and closed-loop in feedback form ($\boldsymbol{X}_t$). We start with the setup (1.3) which corresponds to the open-loop case. Theoretically, it is more tractable, due to the indirect nature (*i.e.* player i will not change his strategy when player j's strategy changes because player i can not observe or feel the change). Practically, there are applications falling into this framework, for instance, the prisoner's dilemma from game theory. This is the scenario that when two people get arrested and investigated, they are in solitary confinements and can not communicate with each other, nor observe the other's choice. In this case, it is reasonable to assume that α_t^i does not depend on the past decisions $\boldsymbol{\alpha}_{[0,t)}$ nor the players' states $\boldsymbol{X}_{[0,t]}$ as this information is not available under this framework. The generalization of deep learning theory for closed-loop cases will be discussed in Section 5.4.

[1] Although in the literature of math finance, one usually models b^i and σ^i to only depend on player i's own action, but it is common in literature of economics that player i's private state is also influenced by others' actions, *e.g.*, α_t^i is a price set by companies and X_t^i is the production quantity. To be general, we include this feature in our model, which yields (1.1).

An alternative method of solving N-player stochastic differential games is via mean-field games, introduced by Lasry and Lions in [41–43] and by Huang, Malhamé and Caines in [31, 32]. The idea is to approximate the Nash equilibrium by the solution of mean field equilibrium (the formal limit of $N \to \infty$) under mild conditions [9], which leads to an approximation error of order $N^{-1/(d+4)}$ assuming that the players are indistinguishable, *i.e.*, all coefficients $(b^i, \sigma^i, f^i, g^i)$ are free of i. We refer to the books [10, 11] and the references therein for further background on mean-field games. However, beyond the case of a continuum of infinitesimal agents with or without major players, the mean-field equilibrium may not be a good approximation in general. In addition, the mean-field game often exhibits multiple equilibria, some of which do not correspond to the limit of N-player game as $N \to \infty$, *e.g.*, in the optimal stopping games [55]. Moreover, when the number of players is of middle size (*e.g.*, $N \sim 50$), the approximation error made by the mean-field equilibrium is large while direct solvers based on forward-backward stochastic differential equations (FBSDEs) or on partial differential equations (PDEs) are still computationally unaffordable. Therefore, it is demanding to develop new theory and algorithms for solving the N-player game.

The idea proposed in this paper is natural and motivated by the fictitious play, a learning process in game theory firstly introduced by Brown in the static case [6, 7] and recently adapted to the mean field case by Cardaliaguet [5, 8] and coauthors. In the fictitious play, after some arbitrary initial moves at the first stage, the players myopically choose their best responses against the empirical strategy distribution of others' action at every subsequent stage. It is hoped that such a learning process will converge and lead to a Nash equilibrium. In fact, Robinson [62] showed this holds for zero-sum games, and Miyazawa [49] extended it to 2×2 games. However, Shapley's famous 3×3 counter-example [63] indicates that this is not always true. Since then, many attempts are made to identify classes of games where the global convergence holds [3, 14, 27, 28, 48, 52, 53], and where the process breaks down [19, 35, 38, 50], to name a few.

Based on fictitious play, we propose a deep learning theory and algorithm for computing the open-loop Nash equilibria. Unlike closed-loop strategies of feedback form, which can be reformulated as the solution to N-coupled Hamilton-Jacobi-Bellman (HJB) equations by dynamic programming principle (DPP), open-loop strategies are usually identified through FBSDEs. The existence of explicit solutions to both equations highly depends on the symmetry of the problem, in particular, for most cases where explicit solutions are available, the players are statistically identical. Therefore, an efficient and accurate numerical scheme is crucial for solving such FBSDEs. Traditional ways run into the technical difficulty of the curse of dimensionality, thus are not feasible when the dimensionality goes beyond 5. Observing impressive results solved by deep learning on various challenging problems [2, 39, 44], we shall use deep neural networks to overcome the curse of dimensionality for moderately large N and asymmetric games. We first boil down the game into N stochastic control subproblems, which are conditionally independent given past play at each stage. Since we first focus on open-loop equilbria (as opposed to closed-loop ones) in each subproblem, the strategies are considered as general progressively measurable processes (as opposed to functions of $(t, \boldsymbol{X}_t)$). Therefore, without the feedback effects, one can design a deep neural network to solve stochastic control subproblems individually. The control at each time step is approximated by a feed-forward subnetwork, whose inputs are initial states $\boldsymbol{X}_0$ and noises $\boldsymbol{W}_{[0,t]}$ in lieu of the definition of open-loop equlibria. For player i's control problem, $\boldsymbol{X}^{-i}$ is generated using strategies from past, *i.e.*, considered as fixed while player i optimizes herself.

Main contribution. The contribution of deep fictitious play is three-fold. Firstly,

our algorithm is scalable: in each round of play, the N subproblems can be solved in parallel, which can be accelerated by the feature of multi-GPU. Secondly, we propose a deep neural network for solving general stochastic control problem where strategies are general processes instead of feed-back form. In lack of DPP, algorithms from reinforcement learning are no longer available. We approximate the optimal control directly in contrast to approximating value functions [60]. Thirdly, the algorithm can be applied to asymmetric games, as for each player, there is a corresponding neural network.

Related literature. Most literature in deep learning and reinforcement learning algorithms in stochastic control problems uses DPP with which, the problem can be solved backwardly, *i.e.*, to find the optimal control at the terminal time, and then decide the previous decision. Among them, let me mention the recent works [1,33], which approximate the optimal policy by neural networks in the spirit of deep reinforcement learning, and the approximated optimal policy is obtained in a backward manner. While in our algorithm, we stack these subnetworks together to form a deep network and train them simultaneously. In fact, our structure is inspired by Han and E [20]. The difference is that they feed the network with X_t seeking for feedback-form controls, while we feed the initial states X_0 and noises $W_{[0,t]}$ for each player's network, seeking for open-loop Nash equilibrium. In terms of using fictitious play to solve multi-agent problems, [26, 40, 47] design reinforcement learning algorithms assuming the system (1.1) is unknown; while our algorithm needs the knowledge of b^i, σ^i, f^i and g^i.

Organization of the paper. In Section 2, we systematically introduce the deep fictitious play theory, and implementation of deep learning algorithms using Keras with GPU acceleration. In Section 3, we apply deep fictitious play to linear quadratic games, and prove the convergence of fictitious play under proper assumptions on parameters, with the limit forming an open-loop Nash equilibrium. Performance of deep learning algorithms are presented in Section 4, where we simulate stochastic differential games with a large number of players (*e.g.*, $N=24$). We make conclusive remarks, and discuss the extensions to other strategies of fictitious play and closed-loop cases in Section 5.

2. Deep fictitious play

In this section, we describe the theory and algorithms of deep fictitious play, which by name, is known to build on fictitious play and deep learning. We first summarize all the notations that shall be used as below. Given a probability space $(\Omega,\mathcal{F},\mathbb{P})$, we consider

- $\boldsymbol{W}=[W^0,W^1,\dots,W^N]$, a $(N+1)$-vector of m-dimensional independent Brownian motions;
- $\mathbb{F}=\{\mathcal{F}_t,0\le t\le T\}$, the augmented filtration generated by $\boldsymbol{W}$;
  $\mathbb{H}^2_T(\mathbb{R}^d)$, the space of all progressively measurable $\mathbb{R}^d$-valued stochastic processes $\alpha:[0,T]\times\Omega\hookrightarrow\mathbb{R}^d$ such that $\|\alpha\|_2=\mathbb{E}[\int_0^T|\alpha_t|^2\,\mathrm{d}t]<\infty$.
- $\mathbb{A}=\mathbb{H}^2_T(A)$, the space of admissible strategies, *i.e.*, elements in $\mathbb{A}$ satisfy (1.2). $\mathbb{A}^N=\mathbb{A}\times\mathbb{A}\times\ldots\times\mathbb{A}$, a product of N copies of $\mathbb{A}$;
- $\boldsymbol{\alpha}=[\alpha^1,\alpha^2,\dots,\alpha^N]$, a collection of all players' strategy profiles. With a negative superscript, $\boldsymbol{\alpha}^{-i}=[\alpha^1,\dots,\alpha^{i-1},\alpha^{i+1},\dots,\alpha^N]$ means the strategy profiles excluding player i's. If a non-negative superscript n appears (*e.g.*, $\boldsymbol{\alpha}^n$), this N-tuple stands for the strategies from stage n. When both exist, $\boldsymbol{\alpha}^{-i,n}=[\alpha^{1,n},\dots,\alpha^{i-1,n},\alpha^{i+1,n},\dots,\alpha^{N,n}]$ is a $(N-1)$-tuple representing strategies excluding player i at stage n. We use the same notations for other stochastic processes (*e.g.*, $\boldsymbol{X}^{-i},\boldsymbol{X}^n$);

We assume that the players start with an initial smooth belief $\boldsymbol{\alpha}^0 \in \mathbb{A}^N$. At the beginning of stage $n+1$, $\boldsymbol{\alpha}^n$ is observable by all players. Player i then chooses best response to her beliefs about opponents described by their play at the previous stage $\boldsymbol{\alpha}^{-i,n}$. Then, player i faces an optimization problem:

$$\inf_{\beta^i \in \mathbb{A}} J^i(\beta^i; \boldsymbol{\alpha}^{-i,n}), \quad J^i(\beta^i; \boldsymbol{\alpha}^{-i,n}) = \mathbb{E}\left[\int_0^T f^i(t, \boldsymbol{X}_t^\alpha, (\beta_t^i, \boldsymbol{\alpha}_t^{-i,n}))\,\mathrm{d}t + g^i(\boldsymbol{X}_T^\alpha)\right], \tag{2.1}$$

where $\boldsymbol{X}_t^\alpha = [X_t^{1,\alpha}, X_t^{2,\alpha}, \ldots, X_t^{N,\alpha}]$ are state processes controlled by $(\beta^i, \boldsymbol{\alpha}^{-i,n})$:

$$\begin{aligned}\mathrm{d}X_t^{\ell,\alpha} = b^\ell(t, \boldsymbol{X}_t^\alpha, (\beta_t^i, \boldsymbol{\alpha}_t^{-i,n}))\,\mathrm{d}t + \sigma^\ell(t, \boldsymbol{X}_t^\alpha, (\beta_t^i, \boldsymbol{\alpha}_t^{-i,n}))\,\mathrm{d}W_t^\ell \\ + \sigma^0(t, \boldsymbol{X}_t^\alpha, (\beta_t^i, \boldsymbol{\alpha}_t^{-i,n}))\,\mathrm{d}W_t^0,\ X_0^{\ell,\alpha} = x^\ell,\end{aligned}$$

for all $\ell \in \mathcal{I}$. Denote by $\alpha^{i,n+1}$ the minimizer in (2.1):

$$\alpha^{i,n+1} := \operatorname*{arg\,min}_{\beta^i \in \mathbb{A}} J^i(\beta^i; \boldsymbol{\alpha}^{-i,n}), \quad \forall i \in \mathcal{I}, n \in \mathbb{N}, \tag{2.2}$$

we assume $\alpha^{i,n+1}$ exists throughout the paper. More precisely, $\alpha^{i,n+1}$ is the player i's optimal strategy at the stage $n+1$ when her opponents' dynamics (1.1) evolve according to $\alpha^{j,n}$, $j \neq i$. All players find their best responses simultaneously, which together form $\boldsymbol{\alpha}^{n+1}$.

REMARK 2.1. Note that the above learning process is slightly different than the usual simultaneous fictitious play, where the belief is described by the time average of past play: $\frac{1}{n}\sum_{k=1}^n \boldsymbol{\alpha}^{-i,k}$. We shall discuss this with more details in Section 5.2.

As discussed in the introduction, in general one can not expect that the player's actions always converge. However, if the sequence $\{\boldsymbol{\alpha}^n\}_{n=1}^\infty$ ever admits a limit, denoted by $\boldsymbol{\alpha}^\infty$, we expect it to form an open-loop Nash equilibrium under mild assumptions. Intuitively, in the limiting situation, when all other players are using strategies $\alpha_t^{j,\infty}$, $j \neq i$, by some stability argument, player i's optimal strategy to the control problem (2.1) should be $\alpha_t^{i,\infty}$, meaning that she will not deviate from $\alpha_t^{i,\infty}$, which makes $(\alpha_t^{i,\infty})_{i=1}^N$ an open loop equilibrium by definition. Therefore, finding an open-loop Nash equilibrium consists of iterating this play until it converges.

We here give an argument under general problem setup using Pontryagin stochastic maximum principle (SMP). For simplicity, we present the case of uncontrolled volatility without common noise: $\sigma^i(t, \boldsymbol{x}, \boldsymbol{\alpha}) \equiv \sigma^i(t, \boldsymbol{x})$, $\forall i \in \mathcal{I}$, $\sigma^0 \equiv 0$, and refer to [11, Chapter 1] for generalization. The Hamiltonian $H^{i,n+1} : [0,T] \times \Omega \times \mathbb{R}^{dN} \times \mathbb{R}^{dN} \times A \hookrightarrow \mathbb{R}$ for player i at stage $n+1$ is defined by:

$$H^{i,n+1}(t, \omega, \boldsymbol{x}, \boldsymbol{y}, \alpha) = \boldsymbol{b}(t, \boldsymbol{x}, (\alpha, \boldsymbol{\alpha}^{-i,n})) \cdot \boldsymbol{y} + f^i(t, \boldsymbol{x}, (\alpha, \boldsymbol{\alpha}^{-i,n})),$$

where the dependence on ω is introduced by $\boldsymbol{\alpha}^{-i,n}$. We assume all coefficients (b^i, σ^i, f^i) are continuously differentiable with respect to $(\boldsymbol{x}, \boldsymbol{\alpha}) \in \mathbb{R}^{dN} \times A^N$; g^i is convex and continuously differentiable with respect to $\boldsymbol{x} \in \mathbb{R}^{dN}$; $A \in \mathbb{R}^k$ is convex; the function $H^{i,n+1}$ is convex $\mathbb{P}$-almost surely in $(\boldsymbol{x}, \boldsymbol{\alpha})$. By the sufficient part of SMP, we look for a control $\hat{\alpha}^{i,n+1} \in A$ of the form:

$$\hat{\alpha}^{i,n+1}(t, \omega, \boldsymbol{x}, \boldsymbol{y}) \in \operatorname*{arg\,min}_{\alpha \in A} H^{i,n+1}(t, \omega, \boldsymbol{x}, \boldsymbol{y}, \alpha),$$

and solve the resulting forward-backward stochastic differential equations (FBSDEs):

$$\begin{cases} \mathrm{d}X_t^{\ell,n+1} = b^\ell(t,\boldsymbol{X}_t^{n+1},(\hat{\alpha}^{i,n+1}(t,\boldsymbol{X}_t^{n+1},\boldsymbol{Y}_t^{n+1}),\boldsymbol{\alpha}_t^{-i,n}))\,\mathrm{d}t+\sigma^\ell(t,\boldsymbol{X}_t^{n+1})\,\mathrm{d}W_t^\ell, \\ \mathrm{d}Y_t^{\ell,n+1} = -\partial_{x^\ell}H^{i,n+1}(t,\boldsymbol{X}_t^{n+1},\boldsymbol{Y}_t^{n+1},\hat{\alpha}^{i,n+1}(t,\boldsymbol{X}_t^{n+1},\boldsymbol{Y}_t^{n+1}))\,\mathrm{d}t+\sum_{j=1}^N Z_t^{\ell,j,n+1}\,\mathrm{d}W_t^j, \\ X_0^{\ell,n+1}=x_0^\ell, \quad Y_T^{\ell,n+1}=\partial_{x^\ell}g^i(\boldsymbol{X}_T^{n+1}), \quad \ell\in\mathcal{I}. \end{cases} \tag{2.3}$$

If there exists a solution $(\boldsymbol{X}^{n+1},\boldsymbol{Y}^{n+1},\boldsymbol{Z}^{n+1})\in H_T^2(\mathbb{R}^{dN}\times\mathbb{R}^{dN}\times\mathbb{R}^{dN\times mN})$, then an optimal control to problem (2.1) is given by plugging the solution into the function $\hat{\alpha}^{i,n+1}$:

$$\alpha_t^{i,n+1}=\hat{\alpha}^{i,n+1}(t,\boldsymbol{X}_t^{n+1},\boldsymbol{Y}_t^{n+1}). \tag{2.4}$$

Now suppose (2.3) is solvable, the sequence given in (2.4) converges to $\boldsymbol{\alpha}^\infty$ as $n\to\infty$. Denote by $(\boldsymbol{X}^\infty,\boldsymbol{Y}^\infty,\boldsymbol{Z}^\infty)$ the solution of (2.3) with $\boldsymbol{\alpha}^n$ being replaced by $\boldsymbol{\alpha}^\infty$. If the system possesses stability, then $(\boldsymbol{X}^\infty,\boldsymbol{Y}^\infty,\boldsymbol{Z}^\infty)$ is also the limit of $(\boldsymbol{X}^{n+1},\boldsymbol{Y}^{n+1},\boldsymbol{Z}^{n+1})$. In this case, given other players using $\boldsymbol{\alpha}^{-i,\infty}$, the optimal control of player i is

$$\alpha^{i,\infty}(t,\boldsymbol{X}_t^\infty,\boldsymbol{Y}_t^\infty)=\lim_{n\to\infty}\hat{\alpha}^{i,n}(t,\boldsymbol{X}_t^n,\boldsymbol{Y}_t^n)=\lim_{n\to\infty}\alpha^{i,n}=\alpha^{i,\infty},$$

where we have used the stability of (2.3) and the continuous dependence of H on the parameter $\boldsymbol{\alpha}^{-i,n}$ for the first identity, the solvability of (2.3) for the second identity, and the convergence of $\alpha^{i,n}$ for the last identity. Therefore, one can put appropriate conditions on (b^i,σ^i,f^i,g^i) to ensure these, and we refer to [45, 46, 57, 58] for detailed discussions. Remark that, all assumptions are satisfied for the case of linear-quadratic games, and thus all the above arguments can go through. We will give more details in Section 3.

In general, problem (2.2) is not analytically tractable, and one needs to solve it numerically. Next we present a novel architecture of DNN and a deep learning algorithm that has a parallelization feature. It starts with a brief introduction on deep learning, followed by the detailed deep fictitious play algorithm.

2.1. Preliminaries on deep learning. Inspired by neurons in human brains, a neural network (NN) is designed for computers to learn from observational data. It has become an effective tool in many fields including computer vision, speech recognition, social network filtering, image analysis, *etc.*, where results produced by NNs are comparable or even superior to human experts. An example of NNs performing well is image classification, where the task is to identify which of a set of categories a new observation belongs to, on the basis of a training set of data containing observations of known category membership. Denote by x the observations and z its category. This problem consists of efficient and accurate learning of the mapping from observations to categories $x\hookrightarrow z(x)$, which can be complicated and non-trivial. Thanks to the universal approximation theorem and the Kolmogorov-Arnold representation theorem [15, 29, 37], NNs are able to provide good approximations to non-trivial mapping.

Our goal is to use deep neural networks to solve the stochastic control problem (2.2). NNs are made by stacking layers one on top of another. Layers with different functions or neuron structures are called differently, including fully-connected layer, constitutional layer, pooling layer, recurrent layers, *etc.*. As our algorithm 1 will focus on fully-connected layers, we here give an example of feed-forward NN using fully-connected

layers in Figure 2.1. Nodes in the figure represent neurons and arrows represent the information flow. As shown, information is constantly "fed forward" from one layer to the next. The first layer (leftmost column) is called the input layer, and the last layer (rightmost column) is called the output layer. Layers in between are called hidden layers, as they have no connection with the external world. In this case, there is only one hidden layer with four neurons.

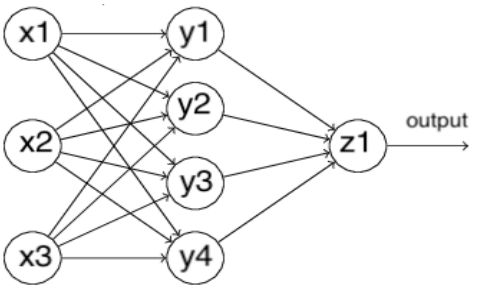

FIG. 2.1. *An illustration of a simple feedforward neural network.*

We now explain how information is processed in NNs. For fully-connected layers, every neuron consists of two kinds of parameters, the weights w and the bias b. Each layer can choose an activation function, then an input x goes through it gives $f(w\cdot x+b)$. In the above example of NN, the data $\boldsymbol{x}=[x_1,x_2,x_3]$ fed to neuron y_i outputs $f(\boldsymbol{w}_j\cdot\boldsymbol{x}+b_j)$, $j=1,\dots,4$, which yields $\boldsymbol{y}=[y_1,y_2,y_3,y_4]$ as the input of neuron z_1. The final output is $z_1=f(\boldsymbol{w}_z\cdot\boldsymbol{y}+b_z)$. In traditional classification problems, categorical information $z(\boldsymbol{x})$ associated to the input $\boldsymbol{x}$ is known, and the optimal weights and bias are chosen to minimize a loss function L:

$$c(w,b):=L(z,z(\boldsymbol{x})), \tag{2.5}$$

where z is the output of the NNs, as functions of (w,b), and $z(\boldsymbol{x})$ is given from the data. The process of finding optimal parameters is called the training of an NN.

The activation function f and loss function L are chosen at the user's preference, and common choices are sigmoid $\frac{1}{1+e^x}$, ReLU x^+ for f, and mean squared error $\sum|z-z(\boldsymbol{x})|^2$ or cross entropy $-\sum z(\boldsymbol{x})\log(z)$ for L in (2.5). In terms of finding the optimal parameters $(\boldsymbol{w},\boldsymbol{b})$ in (2.5), it is in general a high-dimensional optimization problem, and usually done by various stochastic gradient descent methods (e.g. Adam [36, 61], NADAM [17]). For further discussions, we refer to [30, Section 2.1] and [33, Section 2.2].

However, solving (2.2) is not in line with the above procedure, in the sense that there is no target category $z(\boldsymbol{x})$ assigned to each input $\boldsymbol{x}$, and consequently, the loss function is not a distance measured between the network output z and $z(\boldsymbol{x})$. We aim at approximating the optimal strategy at each stage by feedforward NNs. What we actually use from NN is its ability of approximating complex relations by composition of simple functions (by stacking fully connected layers) and finding the (sub-)optimizer with its well-developed built-in stochastic gradient descent (SGD) solvers. We shall explain further the structures of NNs in the following section.

2.2. Deep learning algorithms. We introduce the algorithms of deep learning based on fictitious play by describing two key parts as below.

2.2.1. Part I: solve a stochastic control problem using DNN. We in fact solve a time discretization version of problem (2.2). Partitioning $[0,T]$ into N_T equally-spaced intervals, with the time step $h=T/N_T$. Denote by $\tilde{\mathbb{F}}:=\{\tilde{\mathcal{F}}_k,0\le k\le N_T\}$ the "discretized" filtration with $\tilde{\mathcal{F}}_k=\sigma\{\boldsymbol{W}_{jh},0\le j\le k\}$. A discrete-time analogy of (2.2) is:

$$\tilde{\alpha}^{i,n+1}=\underset{\{\beta^i_{kh}\in\tilde{\mathcal{F}}_k\}_{k=0}^{N_T-1}}{\arg\min}\ \tilde{J}^i(\beta^i;\tilde{\boldsymbol{\alpha}}^{-i,n}), \tag{2.6}$$

where

$$\tilde{J}^i(\beta^i;\tilde{\boldsymbol{\alpha}}^{-i,n}):=\mathbb{E}\left[\sum_{k=0}^{N_T-1}f^i\left(kh,\boldsymbol{X}_{kh},(\beta^i_{kh},\tilde{\boldsymbol{\alpha}}^{-i,n}_{kh})\right)h+g^i(\boldsymbol{X}_T)\right], \tag{2.7}$$

and each entry X^ℓ_{kh} in $\boldsymbol{X}_{kh}$ follows the Euler scheme of (1.1) associated to the strategy β^ℓ if $\ell=i$, and to $\tilde{\alpha}^{\ell,n}$ if $\ell\neq i$:

$$\begin{aligned}X^\ell_{(k+1)h}=X^\ell_{kh}+b^\ell(kh,\boldsymbol{X}_{kh},(\beta^i_{kh},\tilde{\boldsymbol{\alpha}}^{-i,n}_{kh}))h+\sigma^\ell(kh,\boldsymbol{X}_{kh},(\beta^i_{kh},\tilde{\boldsymbol{\alpha}}^{-i,n}_{kh}))(W^\ell_{(k+1)h}-W^\ell_{kh})\\ +\sigma^0(kh,\boldsymbol{X}_{kh},(\beta^i_{kh},\tilde{\boldsymbol{\alpha}}^{-i,n}_{kh}))(W^0_{(k+1)h}-W^0_{kh}),\quad \ell\in\mathcal{I}.\end{aligned} \tag{2.8}$$

Remark that the above time discretization uses Euler scheme, and thus leads to a weak error of $\mathcal{O}(h)$ and a strong error of $\mathcal{O}(\sqrt{h})$.

In the discrete setting, $\beta^i_{kh}\in\tilde{\mathcal{F}}_k$ is interpreted as $\beta^i_{kh}=\beta^i_{kh}(\boldsymbol{X}_0,\boldsymbol{W}_h,\dots,\boldsymbol{W}_{kh})$. Our task is to approximate the functional dependence of the control on noises. Similar to the strategy used in [20], we implement this by a multilayer feedforward sub-network:

$$\beta^i_{kh}\sim\beta^i_{kh}(\boldsymbol{X}_0,\boldsymbol{W}_h,\dots,\boldsymbol{W}_{kh}|\theta^i_{kh}), \tag{2.9}$$

where θ^i_{kh} denotes the collection of all weights and biases in the k^{th} sub-network for player i. Then, at stage $n+1$, the optimization problem for player i becomes

$$\min_{\{\theta^i_{kh}\}_{k=0}^{N_T-1}}\mathbb{E}\left[\sum_{k=0}^{N_T-1}f^i\left(kh,\boldsymbol{X}_{kh},(\beta^i_{kh}(\theta^i_{kh}),\tilde{\boldsymbol{\alpha}}^{-i,n}_{kh})\right)h+g^i(\boldsymbol{X}_T)\right]. \tag{2.10}$$

Denote by $\theta^{i,n+1}_{kh}$ the minimizer of (2.10), then the approximated optimal strategy $\tilde{\alpha}^{i,n+1}$ is given by (2.9) evaluated at $\theta^{i,n+1}_{kh}$. Note that even though we only write explicitly the dependence of β^i's on θ^i, it affects all X^i's through interactions (2.8). In fact, X^ℓ_{kh} depends on $\{\theta^{i,n+1}_0,\dots,\theta^{i,n+1}_{(k-1)h}\}$, for all $\ell\in\mathcal{I}$. Therefore, finding the gradient in minimizing (2.10) is a non-trivial task. Thanks to the key feature of NNs, computation can be done via a forward-backward propagation algorithm derived from chain rule composition [54].

The architecture of the NN for finding $\tilde{\alpha}^{i,n+1}$ is presented in Figure 2.2: "Input-Layer" are inputs of this network; "Rcost" and "Tcost", representing running and terminal cost, contribute to the total cost J^i; "Sequential" is a multilayer feedforward subnetwork for control approximation at each time step; "Concatenate" is an auxiliary layer combining some of previous layers as inputs of "Sequential".

There are three main kinds of information flows in the network for each period $[kh,(k+1)h]$, $k=0,\dots N_T-1$:

(1) $\text{State}_{kh}:=(\boldsymbol{X}_0,\boldsymbol{W}_h,\dots,\boldsymbol{W}_{kh})\to\beta^i_{kh}$ given by "Sequential" layer. It is an L-layer feed-forward subnetwork to approximate the control of player i at time kh, containing parameters θ^i_{kh} to be optimized.

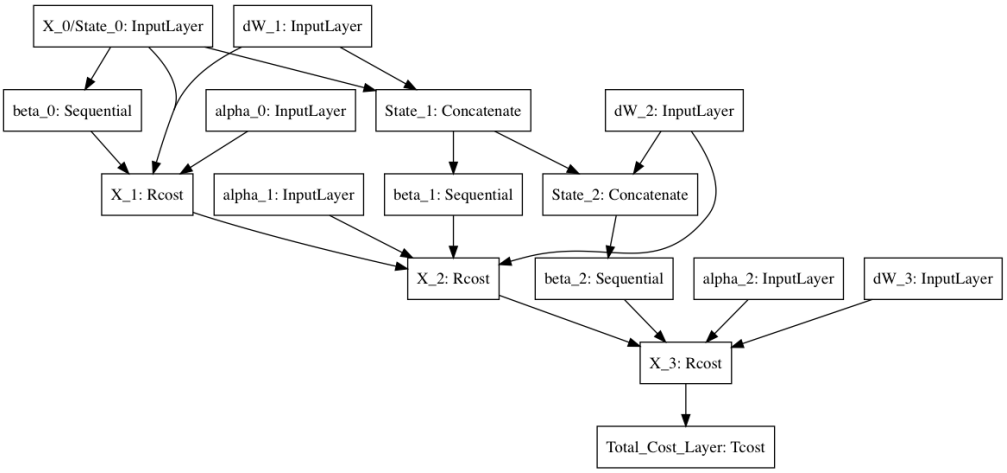

FIG. 2.2. *Illustration of the network architecture for problem* (2.10) *with* $N_T = T = 3$.

(2) $(\boldsymbol{X}_{kh}, \beta^i_{kh}, \boldsymbol{\alpha}^{-i,n}_{kh}, \mathrm{d}\boldsymbol{W}_{(k+1)h} := \boldsymbol{W}_{(k+1)h} - \boldsymbol{W}_{kh}) \to \boldsymbol{X}_{(k+1)h}$ given by "Rcost" layer. This layer possesses two functions. Firstly, it computes the running cost at time kh using $(\boldsymbol{X}_{kh}, \beta^i_{kh}, \tilde{\boldsymbol{\alpha}}^{-i,n}_{kh})$, where β^i_{kh} is produced from previous step. The cost is then added to the final output. Secondly, it updates states value $\boldsymbol{X}_{(k+1)h}$ via dynamics (2.8), using β^i_{kh} for player i and using $\boldsymbol{\alpha}^{-i,n}_{kh}$ for player $j \neq i$ which are inputs of the network. No parameter is minimized at this layer.

(3) $(\mathrm{State}_{kh}, \mathrm{d}\mathbf{W}_{(k+1)h}) \to \mathrm{State}_{(k+1)h}$ given by "Concatenate" layer. This layer combines two previous ones together, acting as a preparation for the input of "Sequential" layer. No parameter is minimized at this layer.

At time $T = N_T \times h$, the terminal cost is calculated using $\boldsymbol{X}_T$ and added to the final output via "Tcost" layer. With these preparations, we introduce the deep fictitious play as below.

2.2.2. Part II: find an equilibrium by fictitious play. Here we use a flowchart to describe the algorithm of deep fictitious play (see Algorithm 1).

2.3. Implementation.

Computing environment. The Algorithm 1 described in Section 2.2.2 is implemented in Python using the high-level neural network API Keras [13]. Numerical examples will be presented in Section 4. All experiments are performed using Amazon EC2 services, which provide a variety of instances for computing acceleration. All computations use NVIDIA K80 GPUs with 12GiB of GPU memory on Deep Learning Amazon Machine Image running on Ubuntu 16.04.

Parallelizability. As N going relatively large, to make computation manageable, one can distribute Step $5-9$ to several GPUs. That is, assigning each available GPU the task of training a subset of neural networks, where this subset is fixed from stage to stage. This will speed up the computation time significantly, as peer-to-peer GPU communications are not needed in the designed algorithm.

Input, output and parameters for neural networks. Before training, we sample $\boldsymbol{W} = \{W^i_{kh} \in \mathbb{R}^m, i \in \mathcal{I}\}_{k=1}^{N_T}$, which, together with the initial states $\boldsymbol{X}_0$ and initial be-

Algorithm 1 Deep Fictitious Play for Finding Nash Equilibrium

Require: $N = \#$ of players, $N_T = \#$ of subintervals on $[0,T]$, $M = \#$ of training paths, $M' = \#$ of out-of-sample paths for final evaluation, $\boldsymbol{\alpha}^0 = \{\alpha_{kh}^{i,0} \in A \subset \mathbb{R}^k, i \in \mathcal{I}\}_{k=0}^{N_T-1}$ = initial belief, $\boldsymbol{X}_0 = \{x_0^i \in \mathbb{R}^d, i \in \mathcal{I}\}$ = initial states

1: Create N separated deep neural networks as described in Section 2.2.1
2: Generate M sample paths of BM: $\boldsymbol{W} = \{W_{kh}^i \in \mathbb{R}^m, i \in \mathcal{I} \cup \{0\}\}_{k=1}^{N_T}$
3: $n \leftarrow 0$
4: **repeat**
5: $n \leftarrow n+1$
6: **for** $i \leftarrow 1$ to N **do**
7: (Continue to) Train i^{th} NN with data $\{\boldsymbol{X}_0, \boldsymbol{\alpha}^{-i,n-1} = \{\alpha_{kh}^{j,n-1}, j \in \mathcal{I} \setminus \{i\}\}_{k=0}^{N_T-1}, \boldsymbol{W}\}$
8: Obtain the approximated optimal strategy $\alpha^{i,n}$ and cost $J^i(\alpha^{i,n}; \boldsymbol{\alpha}^{-i,n-1})$
9: **end for**
10: Collect optimal policies at stage n: $\boldsymbol{\alpha}^n \leftarrow (\alpha^{1,n}, \ldots, \alpha^{N,n})$
11: Compute relative change of cost $err^n := \max_{i \in \mathcal{I}} \left\{ \frac{|J^i(\alpha^{i,n}; \boldsymbol{\alpha}^{-i,n-1}) - J^i(\alpha^{i,n-1}; \boldsymbol{\alpha}^{-i,n-2})|}{J^i(\alpha^{i,n-1}; \boldsymbol{\alpha}^{-i,n-2})} \right\}$
12: **until** err^n go below a threshold
13: Generate M' out-of-sample paths of BM for final evaluation
14: $n' \leftarrow 0$
15: **repeat**
16: $n' \leftarrow n'+1$
17: Evaluate i^{th} NN with $\{\boldsymbol{X}_0, \boldsymbol{\alpha}^{-i,n'-1}$, out-of-sample paths$\}$, $\forall i \in \mathcal{I}$
18: Obtain $\alpha^{i,n'}$ and $J^{i,n'} := J^i(\alpha^{i,n'}; \boldsymbol{\alpha}^{-i,n'-1})$ $\forall i \in \mathcal{I}$
19: **until** $J^{i,n'}$ converges in n', $\forall i \in \mathcal{I}$
20: **return** The optimal policy $\alpha^{i,n'}$, and the final cost for each player $J^{i,n'}$

lief $\boldsymbol{\alpha}^0 = \{\alpha_{kh}^{i,0} \in A \subset \mathbb{R}^k, i \in \mathcal{I}\}_{k=0}^{N_T-1}$, are the inputs of NNs. Adam, a variant of SGD that adaptively estimates lower-order moments, is chosen to optimize the parameters $\{\theta_{kh}^i\}_{k=0}^{N_T-1}$. The hyper-parameters set for Adam solver follows the original paper [36]. Regarding the architecture of "Sequential", it is a L-layered subnetwork. We set $L = 4$, with 1 input layer, 2 hidden layers, and 1 output layer containing k nodes. Rectified linear unit is chosen for hidden layers while no activation is applied to the output layer. We also add Batch Normalization [34] for hidden layers before activation. This method performs the normalization for each training mini-batch to eliminate internal covariate shift phenomenon, and thus frees us from delicate parameter initialization. It also acts as a regularizer, in some cases eliminating the need for Dropout. Note that the choice of L and size of $\{\theta_{kh}^i\}_{k=0}^{N_T-1}$ are empirical. For testing problems that have benchmark solutions, one can do grid-search method to select the one with the best performance in the validation set. However, for real problems there is no universal rule for all problem settings.

Parameters of the network are initialized at Step 1. In Step 7, training continues from previous stage without re-initialization. This is because, although opponents' policies change from stage to stage, they will not vary significantly and parameter values from previous stage should be better than a random initialization. For fixed computational budget, instead of using the stopping criteria in Step 12 one can terminate the loop until n reaches a predetermined upper bound $\bar{n}$. In Step 7, the number of epochs to train the model at every single stage does not need to be large (at the scale

of hundreds). This is because we are not aiming at a one-time accurate approximation of the optimal policy. Especially at the first few rounds when opponents' policies are far from optimal, pursuing accurate approximation is not meaningful. Instead, by using small budget to obtain moderate accuracy at each iteration, we are able to repeat the game for more times. In summary, for the two computational scheme: large $\bar{n}$ small epochs, and small $\bar{n}$ large epochs, the former one is better.

If opponents' policies stay the same from stage to stage, then the two schemes receive the same accuracy. This is justified by the following argument: Suppose opponents' policies stay the same, then player i essentially faces the same optimization problem from stage to stage. Since we do not re-initialize network parameters in Step 7, the difference between the two schemes is to train the same problem with small epochs and large rounds *vs.* large epochs and small rounds. This is the same in terms of SGD training, thus should lead to the same relative error. In reality, the opponents' policies are updated from time to time, and the former scheme enables us to obtain player i's reaction with more updated belief of his opponents. Step 15-19 are not computationally costly, and the value functions usually converge after several iterations in our numerical study.

3. Linear-quadratic games

Although the deep fictitious theory and algorithm can be applied for any N-player game, the proof of convergence is in general hard. Here we consider a special case of linear-quadratic symmetric N-player games, and analyze the convergence of $\boldsymbol{\alpha}^n$ defined in (2.2). The strategy analyzed here will provide an open-loop Nash equilibrium, as proved at the end of the section.

We follow the linear-quadratic model proposed in [12], where players's dynamics interact through their empirical mean:

$$\mathrm{d}X_t^i = [a(\overline{X}_t - X_t^i) + \alpha_t^i]\,\mathrm{d}t + \sigma\left(\rho\,\mathrm{d}W_t^0 + \sqrt{1-\rho^2}\,\mathrm{d}W_t^i\right), \quad X_0^i = x^i, \quad \overline{X}_t = \frac{1}{N}\sum_{i=1}^N X_t^i. \tag{3.1}$$

Here $\{W_t^i, 0 \le i \le N\}$ are independent standard Brownian motions (BMs). Each player $i \in \{1,2,\dots,N\}$ controls the drift by α_t^i in order to minimize the cost functional

$$J^i(\alpha^1,\dots,\alpha^N) = \mathbb{E}\left\{\int_0^T f^i(\boldsymbol{X}_t, \alpha_t^i)\,\mathrm{d}t + g^i(\boldsymbol{X}_T)\right\}, \tag{3.2}$$

with the running cost defined by

$$f^i(\boldsymbol{x}, \alpha) = \frac{1}{2}\alpha^2 - q\alpha(\bar{x} - x^i) + \frac{\epsilon}{2}(\bar{x} - x^i)^2, \quad \bar{x} = \frac{1}{N}\sum_{i=1}^N x^i,$$

and the terminal cost function g^i by

$$g^i(\boldsymbol{x}) = \frac{c}{2}(\bar{x} - x^i)^2.$$

All parameters a, ϵ, c, q are non-negative, and $q^2 \le \epsilon$ is imposed so that $f^i(\boldsymbol{x}, \alpha)$ is convex in $(\boldsymbol{x}, a)$. In [12], X_t^i is viewed as the log-monetary reserves of bank i at time t. For further interpretation, we refer to [12].

In the spirit of fictitious play, the N-player game is recasted into N individual optimal control problems played iteratively. The players start with a smooth belief

of their opponents' actions $\boldsymbol{\alpha}^0$. At stage $n+1$, the players have observed the same past controls $\alpha^{i,n}$'s, and then each player optimizes her control problem individually, assuming other players will follow their choice at state n. That is, for player i's problem, her dynamics are controlled through α^i_t, while other players' states evolve according to the past strategies $\boldsymbol{\alpha}^{-i,n}$:

$$\mathrm{d}X^{i,n+1}_t = [a(\overline{X}^{n+1}_t - X^{i,n+1}_t) + \alpha^i_t]\,\mathrm{d}t + \sigma(\rho\,\mathrm{d}W^0_t + \sqrt{1-\rho^2}\,\mathrm{d}W^i_t), \tag{3.3}$$

$$\mathrm{d}X^{j,n+1}_t = [a(\overline{X}^{n+1}_t - X^{j,n+1}_t) + \alpha^{j,n}_t]\,\mathrm{d}t + \sigma(\rho\,\mathrm{d}W^0_t + \sqrt{1-\rho^2}\,\mathrm{d}W^j_t), \quad j \neq i. \tag{3.4}$$

Player i faces an optimal control problem:

$$\inf_{\alpha^i \in \mathbb{A}} J^{i,n+1}(\alpha^i; \boldsymbol{\alpha}^{-i,n}), \text{ where}$$

$$J^{i,n+1}(\alpha^i; \boldsymbol{\alpha}^{-i,n}) := \mathbb{E}\left\{ \int_0^T \frac{1}{2}(\alpha^i_t)^2 - q\alpha^i_t(\overline{X}^{n+1}_t - X^{i,n+1}_t) + \frac{\epsilon}{2}(\overline{X}^{n+1}_t - X^{i,n+1}_t)^2\,\mathrm{d}t \right.$$
$$\left. + \frac{c}{2}(\overline{X}^{n+1}_T - X^{i,n+1}_T)^2 \right\}. \tag{3.5}$$

The space where we search for optimal α^i is the space of square-integrable progressively-measurable $\mathbb{R}$-valued processes on $\mathbb{A} := \mathbb{H}^2_T(\mathbb{R})$, to be consistent with open-loop equilibria. Denote by $\alpha^{i,n+1}$ the minimizer of this control problem at stage $n+1$:

$$\alpha^{i,n+1} := \operatorname*{arg\,min}_{\alpha^i \in \mathbb{A}} J^{i,n+1}(\alpha^i; \boldsymbol{\alpha}^{-i,n}). \tag{3.6}$$

In what follows, we shall show:

(a) $\alpha^{i,n+1}$ exists $\forall i \in \mathcal{I}, n \in \mathbb{N}$, that is, the minimal cost in (3.5) is always attainable;

(b) the family $\{\boldsymbol{\alpha}^n\}$ converges;

(c) the limit of $\boldsymbol{\alpha}^n$ forms an open-loop Nash equilibrium.

3.1. The probabilistic approach. Observing that the cost functional $J^{i,n+1}$ in (3.5) solely depends on the process $\widetilde{X}^{i,n+1} := \overline{X}^{n+1} - X^{i,n+1}$ and the control α^i, we make the following simplification. Notice that (3.3) and (3.4) imply

$$\mathrm{d}\widetilde{X}^{i,n+1}_t = \left[\frac{\sum_{j\neq i} \alpha^{j,n}_t}{N} - \frac{N-1}{N}\alpha^i_t - a\widetilde{X}^{i,n+1}_t \right] \mathrm{d}t + \sigma\sqrt{1-\rho^2}\left(\frac{1}{N}\sum_{j=1}^N \mathrm{d}W^j_t - \mathrm{d}W^i_t\right). \tag{3.7}$$

Then, player i's problem is equivalent to:

$$\inf_{\alpha^i \in \mathbb{A}} \mathbb{E}\left\{ \int_0^T \frac{1}{2}(\alpha^i_t)^2 - q\alpha^i_t\widetilde{X}^{i,n+1}_t + \frac{\epsilon}{2}(\widetilde{X}^{i,n+1}_t)^2\,\mathrm{d}t + \frac{c}{2}(\widetilde{X}^{i,n+1}_T)^2 \right\}.$$

In what follows, we show the existence of unique minimizer, denoted by $\alpha^{i,n+1}$, using SMP. The Hamiltonian for player i at stage $n+1$ reads as

$$H^{i,n+1}(t,\omega,x,y,\alpha) = \left(\frac{\sum_{j\neq i}\alpha^{j,n}_t}{N} - \frac{N-1}{N}\alpha - ax\right)y + \frac{1}{2}\alpha^2 - q\alpha x + \frac{\epsilon}{2}x^2.$$

For a given admissible control $\alpha^i \in \mathbb{A}$, the adjoint processes $(Y^{i,n+1}_t, Z^{i,j,n+1}_t, 0 \le j \le N)$ satisfy the backward stochastic differential equation (BSDE):

$$\mathrm{d}Y^{i,n+1}_t = -[-aY^{i,n+1}_t - q\alpha^i_t + \epsilon\widetilde{X}^{i,n+1}_t]\,\mathrm{d}t + \sum_{j=0}^N Z^{i,j,n+1}_t\,\mathrm{d}W^j_t, \tag{3.8}$$

with the terminal condition $Y_T^{i,n+1} = c\widetilde{X}_T^{i,n+1}$. Standard results on BSDE [56], together with the estimates on the controlled state $\widetilde{X}_t^{i,n+1}$, guarantee the existence and uniqueness of adjoint processes. Pontryagin SMP suggests the form of optimizer:

$$\partial_\alpha H^{i,n+1} = 0 \Longleftrightarrow \hat{\alpha} = qx + \frac{N-1}{N} y. \tag{3.9}$$

Plugging this candidate into the system (3.7)-(3.8) produces a system of affine FBSDEs:

$$\begin{cases} \mathrm{d}\widetilde{X}_t^{i,n+1} = \left[\dfrac{\sum_{j\neq i} \alpha_t^{j,n}}{N} - (a + (1-\frac{1}{N})q)\widetilde{X}_t^{i,n+1} - (1-\frac{1}{N})^2 Y_t^{i,n+1} \right] \mathrm{d}t \\ \qquad\qquad + \sigma\sqrt{1-\rho^2}\left(\frac{1}{N}\sum_{j=1}^{N} \mathrm{d}W_t^j - \mathrm{d}W_t^i\right), \\ \mathrm{d}Y_t^{i,n+1} = -[-(a + (1-\frac{1}{N})q)Y_t^{i,n+1} + (\epsilon - q^2)\widetilde{X}_t^{i,n+1}]\mathrm{d}t + \sum_{j=0}^{N} Z_t^{i,j,n+1}\mathrm{d}W_t^j, \\ \widetilde{X}_0^{i,n+1} = \overline{x}_0 - x_0^i, \quad Y_T^{i,n+1} = c\widetilde{X}_T^{i,n+1}. \end{cases} \tag{3.10}$$

The sufficient condition of SMP suggests that if we solve (3.10), we actually have obtained the optimal control by plugging its solution into Equation (3.9). In fact, the coefficients satisfy the G-monotone property in [58], thus the system is uniquely solved in $\mathbb{H}_T^2(\mathbb{R}\times\mathbb{R}\times\mathbb{R}^{N+1})$, and the resulting optimal control is indeed admissible. This answers question (a). For the other two questions, we need to further analyze (3.10).

Note that the system can be decoupled using:

$$Y_t^{i,n+1} = K_t\widetilde{X}_t^{i,n+1} - \psi_t^{i,n+1}, \tag{3.11}$$

where K_t satisfies the Riccati equation:

$$\dot{K}_t = 2(a + (1-\frac{1}{N})q)K_t + (\frac{N-1}{N})^2 K_t^2 - (\epsilon - q^2), \quad K_T = c, \tag{3.12}$$

and the decoupled processes $(\widetilde{X}_t^{i,n+1}, \psi_t^{i,n+1}, \phi_t^{i,j,n+1}, 0 \le j \le N)$ satisfy:

$$\begin{cases} \mathrm{d}\widetilde{X}_t^{i,n+1} = \left[\dfrac{\sum_{j\neq i} \alpha_t^{j,n}}{N} - \gamma_t\widetilde{X}_t^{i,n+1} + (1-\frac{1}{N})^2\psi_t^{i,n+1} \right] \mathrm{d}t \\ \qquad\qquad + \sigma\sqrt{1-\rho^2}\left(\frac{1}{N}\sum_{j=1}^{N} \mathrm{d}W_t^j - \mathrm{d}W_t^i\right), \\ \mathrm{d}\psi_t^{i,n+1} = -[-\gamma_t\psi_t^{i,n+1} - K_t\dfrac{\sum_{j\neq i}\alpha_t^{j,n}}{N}]\mathrm{d}t + \sum_{j=0}^{N}\phi_t^{i,j,n+1}\mathrm{d}W_t^j, \\ \widetilde{X}_0^{i,n+1} = \overline{x}_0 - x_0^i, \quad \psi_T^{i,n+1} = 0, \end{cases} \tag{3.13}$$

where γ_t is a deterministic function on $[0,T]$:

$$\gamma_t = a + (1-\frac{1}{N})q + (1-\frac{1}{N})^2 K_t, \tag{3.14}$$

and the optimal strategy is expressed as

$$\alpha_t^{i,n+1} = (q + (1 - \frac{1}{N})K_t)\widetilde{X}_t^{i,n+1} - (1 - \frac{1}{N})\psi_t^{i,n+1}. \tag{3.15}$$

Again, since $\boldsymbol{\alpha}^n \in \mathbb{H}_T^2(\mathbb{R}^N)$, existence and uniqueness of $(\psi^{i,n+1}, \phi^{i,j,n+1}, 0 \le j \le N) \in \mathbb{H}^2(\mathbb{R} \times \mathbb{R}^{N+1})$ is guaranteed $\forall i \in \mathcal{I}$, $n \in \mathbb{N}$, and the forward equation possesses a unique strong solution. Then the triple $(X^{i,n+1}, Y^{i,n+1}, Z^{i,j,n+1})$ solves the original FBSDEs (3.10) with $Y_t^{i,n+1}$ defined by (3.11) and $Z_t^{i,j,n+1}$ by

$$Z_t^{i,0,n+1} = -\phi_t^{i,0,n+1}, \quad Z_t^{i,j,n+1} = -\phi_t^{i,j,n+1} + K_t \sigma \sqrt{1-\rho^2}(\frac{1}{N} - \delta_{i,j}), \quad j \in \mathcal{I}.$$

To answer questions (b) and (c), we state the main theorem in this section, with the proofs presented in the next subsections.

THEOREM 3.1. *For linear-quadratic games, the family* $\{\boldsymbol{\alpha}^n\}_{n\in\mathbb{N}}$ *defined in* (3.5)-(3.6) *converges if*

$$\frac{1 - e^{-2T\underline{\gamma}}}{\underline{\gamma}} C < 1. \tag{3.16}$$

It forms an open-loop Nash equilibrium of the original problem (3.1)-(3.2). *Moreover, the limit, denoted by* $\boldsymbol{\alpha}^\infty$, *is independent from the choice of initial belief* $\boldsymbol{\alpha}^0$. *Here* $\underline{\gamma} = a + (1 - \frac{1}{N})q + (1 - \frac{1}{N})^2 \underline{K}$, $\overline{K}$ *and* $\underline{K}$ *are the maximum and minimum values of* K_t *on* $[0,T]$, *and the constant* C *is*

$$C = (1 - \frac{1}{N})^2 \left((1 - \frac{1}{N})^2 \overline{K}^2 + (q + (1 - \frac{1}{N})\overline{K})^2 \left(\frac{1 - e^{-2T\underline{\gamma}}}{\underline{\gamma}} (1 - \frac{1}{N})^4 \overline{K}^2 + 2 \right) \right). \tag{3.17}$$

REMARK 3.1. The condition (3.16) is sufficient but not necessary. The numerical performance of the proposed algorithm can do better. In Section 4, the parameters are chosen so that the condition is violated, but the algorithm still converges fast, in order to illustrate the sufficiency. By observing the form of C and $\underline{\gamma}$, we remark that the convergence rate decreases in the number of players N.

PROPOSITION 3.1. *The following three classes of parameters satisfy condition* (3.16):

(i) Small time duration, that is, T *is small.*

(ii) Strong mean-reversion rate, i.e., a *is large.*

(iii) Small terminal cost and small intensive to borrowing or landing, that is, c *and* q *are small. Also the "remaining" running cost of the state process*[2] *is small, i.e.,* $\epsilon - q^2$ *is small.*

Proof. We first notice that the solution to (3.12) is smooth and monotone on $[0,T]$, by computing its derivative:

$$\dot{K}_t \sim -(\epsilon - q^2) + c^2(1 - \frac{1}{N})^2 + 2c(a + (1 - \frac{1}{N})q).$$

[2]The running cost $f^i(\boldsymbol{x}, \alpha)$ can be rewritten as $f^i(\boldsymbol{x}, \alpha) = \frac{1}{2}(\alpha - q(\bar{x} - x^i))^2 + \frac{1}{2}(\epsilon - q^2)(\bar{x} - x^i)^2$, therefore, can be interpreted as penalizing the control from deviating $q(\bar{x} - x^i)$, borrowing or lending proportionally to the difference from average with a rate q, as well as penalizing the distance from average with weight $\epsilon - q^2$.

So $\overline{K}=\max\{c,K_0\}$ and $\underline{K}=\min\{c,K_0\}$. Also, when $\dot{K}_t>0$, K_0 is bounded below by $\frac{-(\epsilon-q^2)-c\delta^+}{\delta^- -c(1-\frac{1}{N})^2}$; otherwise when K_t is decreasing, K_0 is bounded above by $\frac{-(\epsilon-q^2)-c\delta^+}{\delta^- -c(1-\frac{1}{N})^2}$, where

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

Then case (i) follows by the fact that C has an upper bound that is free of T.

For a sufficiently large, K_t is increasing and $\overline{K}=c$. Then C has a upper bound (uniformly in a), and case (ii) follows $\frac{1-e^{-2T\underline{\gamma}}}{\underline{\gamma}}<\frac{1}{a}$. Under case (iii), $\overline{K}$ is sufficiently small, thus C is small and the factor is less than 1. □

3.2. Proof of convergence. This section proves Theorem 3.1. Define $\Delta\zeta_t^{i,n}:=\zeta_t^{i,n+1}-\zeta_t^{i,n}$ the difference from stage n to $n+1$ for the i^{th} player, with $\zeta=\alpha,\psi,\phi,\widetilde{X}$, respectively. Using Equation (3.13), the increment in ψ satisfies:

$$\mathrm{d}\Delta\psi_t^{i,n}=-[-\gamma_t\Delta\psi_t^{i,n}-\frac{K_t}{N}\sum_{j\neq i}\Delta\alpha_t^{j,n-1}]\,\mathrm{d}t+\sum_{j=0}^{N}\Delta\phi_t^{i,j,n}\,\mathrm{d}W_t^j,\quad \Delta\psi_T^{i,n}=0,$$

whose solution is:

$$\Delta\psi_t^{i,n}=\mathbb{E}\left[\int_t^T-\frac{K_s}{N}\sum_{j\neq i}\Delta\alpha_s^{j,n-1}e^{\int_s^t\gamma_u\,\mathrm{d}u}\,\mathrm{d}s\,\middle|\,\mathcal{F}_t\right].$$

By Jensen's inequality, one deduces:

$$\begin{aligned}
\left\|\Delta\psi^{i,n}\right\|_2^2&\leq\int_0^T\mathbb{E}\left[\int_t^T\frac{K_s^2}{N^2}\left(\sum_{j\neq i}\Delta\alpha_s^{j,n-1}\right)^2e^{2\int_s^t\gamma_u\,\mathrm{d}u}\,\mathrm{d}s\right]\mathrm{d}t\\
&\leq\frac{\overline{K}^2}{N^2}\int_0^T\int_t^T\mathbb{E}\left(\sum_{j\neq i}\Delta\alpha_s^{j,n-1}\right)^2e^{2(t-s)\underline{\gamma}}\,\mathrm{d}s\,\mathrm{d}t\\
&=\frac{\overline{K}^2}{N^2}\int_0^T\mathbb{E}\left(\sum_{j\neq i}\Delta\alpha_s^{j,n-1}\right)^2\frac{1-e^{-2s\underline{\gamma}}}{2\underline{\gamma}}\,\mathrm{d}s\\
&\leq\frac{\overline{K}^2}{N^2}\frac{1-e^{-2T\underline{\gamma}}}{2\underline{\gamma}}(N-1)^2\max_{j\neq i}\int_0^T\mathbb{E}[\Delta\alpha_s^{j,n-1}]^2\,\mathrm{d}s\\
&\leq\frac{1-e^{-2T\underline{\gamma}}}{2\underline{\gamma}}(1-\frac{1}{N})^2\overline{K}^2\max_{i\in\mathcal{I}}\left\|\Delta\alpha^{i,n-1}\right\|_2^2,
\end{aligned}$$

where $\underline{\gamma}=a+(1-\frac{1}{N})q+(1-\frac{1}{N})^2\underline{K}$. Since the RHS of the above inequality is independent of i, taking maximum over $\mathcal{I}$ yields

$$\max_{i\in\mathcal{I}}\left\|\Delta\psi^{i,n}\right\|_2^2\leq\frac{1-e^{-2T\underline{\gamma}}}{2\underline{\gamma}}(1-\frac{1}{N})^2\overline{K}^2\max_{i\in\mathcal{I}}\left\|\Delta\alpha^{i,n-1}\right\|_2^2.\tag{3.18}$$

Similarly, the dynamics of $\Delta\widetilde{X}_t^{i,n}$ can be derived from (3.13):

$$\mathrm{d}\Delta\widetilde{X}_t^{i,n}=[\frac{1}{N}\sum_{j\neq i}\Delta\alpha_t^{j,n-1}-\gamma_t\Delta\widetilde{X}_t^{i,n}+(1-\frac{1}{N})^2\Delta\psi_t^{i,n}]\,\mathrm{d}t,\quad\Delta\widetilde{X}_0^{i,n}=0,$$

which admits the solution:

$$\Delta\widetilde{X}_t^{i,n}=\int_0^t\left(\frac{1}{N}\sum_{j\neq i}\Delta\alpha_s^{j,n-1}+(1-\frac{1}{N})^2\Delta\psi_s^{i,n}\right)e^{-\int_s^t\gamma_u\,\mathrm{d}u}\,\mathrm{d}s.$$

We next give an upper bound of increment of the forward process $\Delta\widetilde{X}_\cdot^{i,n}$:

$$\begin{aligned}\left\|\Delta\widetilde{X}^{i,n}\right\|_2^2&\leq\int_0^T\int_0^t\mathbb{E}\left(\frac{1}{N}\sum_{j\neq i}\Delta\alpha_s^{j,n-1}+(1-\frac{1}{N})^2\Delta\psi_s^{i,n}\right)^2e^{-2\int_s^t\gamma_u\,\mathrm{d}u}\,\mathrm{d}s\,\mathrm{d}t\\&\leq 2\int_0^T\int_0^t\left(\mathbb{E}[\frac{1}{N}\sum_{j\neq i}\Delta\alpha_s^{j,n-1}]^2+(1-\frac{1}{N})^4\mathbb{E}[\Delta\psi_s^{i,n}]^2\right)e^{-2(t-s)\underline{\gamma}}\mathrm{d}s\,\mathrm{d}t\\&\leq 2\int_0^T\left(\mathbb{E}[\frac{1}{N}\sum_{j\neq i}\Delta\alpha_s^{j,n-1}]^2+(1-\frac{1}{N})^4\mathbb{E}[\Delta\psi_s^{i,n}]^2\right)\frac{1-e^{-2(T-s)\underline{\gamma}}}{2\underline{\gamma}}\mathrm{d}s\\&\leq\frac{1-e^{-2T\underline{\gamma}}}{\underline{\gamma}}\left((1-\frac{1}{N})^2\max_{j\neq i}\left\|\Delta\alpha^{j,n-1}\right\|_2^2+(1-\frac{1}{N})^4\left\|\Delta\psi^{i,n}\right\|_2^2\right).\end{aligned}$$

Again by taking maximum over $\mathcal{I}$ on both sides, one has:

$$\max_{i\in\mathcal{I}}\left\|\Delta\widetilde{X}^{i,n}\right\|_2^2\leq\frac{1-e^{-2T\underline{\gamma}}}{\underline{\gamma}}\left((1-\frac{1}{N})^2\max_{i\in\mathcal{I}}\left\|\Delta\alpha^{i,n-1}\right\|_2^2+(1-\frac{1}{N})^4\max_{i\in\mathcal{I}}\left\|\Delta\psi^{i,n}\right\|_2^2\right).\tag{3.19}$$

Recall from (3.15) that the increment in the strategy can be decomposed as

$$\Delta\alpha_t^{i,n}=(q+(1-\frac{1}{N})K_t)\Delta\widetilde{X}_t^{i,n}-(1-\frac{1}{N})\Delta\psi_t^{i,n},$$

together with estimates (3.18) and (3.19), we obtain:

$$\begin{aligned}\max_{i\in\mathcal{I}}\left\|\Delta\alpha^{i,n}\right\|_2^2&\leq 2(q+(1-\frac{1}{N})\overline{K})^2\max_{i\in I}\left\|\Delta\widetilde{X}^{i,n}\right\|_2^2+2(1-\frac{1}{N})^2\max_{i\in\mathcal{I}}\left\|\Delta\psi^{i,n}\right\|_2^2\\&\leq\frac{1-e^{-2T\underline{\gamma}}}{\underline{\gamma}}C\max_{i\in\mathcal{I}}\left\|\Delta\alpha^{i,n-1}\right\|_2^2,\end{aligned}$$

where C is a constant given in (3.17). Under condition (3.16), the mapping $\Delta\boldsymbol{\alpha}^{n-1}\hookrightarrow\Delta\boldsymbol{\alpha}^n$ is a contraction. Therefore, this proposed learning process converges in the linear-quadratic games.

Denote the limit of $\{\boldsymbol{\alpha}^n\}$ by $\boldsymbol{\alpha}^\infty=[\alpha^{1,\infty},\ldots,\alpha^{N,\infty}]$ where the learning process starts with an initial belief $\boldsymbol{\alpha}^0$. Let $(\widetilde{X}_t^{i,\alpha},\psi_t^{i,\alpha},\phi_t^{i,\alpha})$ be the solution to the decoupled system (3.13) with $\{\alpha^{j,n},j\in\mathcal{I}\setminus\{i\}\}$ replaced by $\{\alpha^{j,\infty},j\in\mathcal{I}\setminus\{i\}\}$. On one hand, this corresponds to the problem of identifying player i's best strategy, while others using $\boldsymbol{\alpha}^{-i,\infty}$, and her best choice is

$$(q+(1-\frac{1}{N})K_t)\widetilde{X}_t^{i,\alpha}-(1-\frac{1}{N})\psi_t^{i,\alpha}.$$

On the other hand, by stability theorems (e.g. [64, Theorem 3.4.2, Theorem 4.4.3]), this triple $(\widetilde{X}_t^{i,\alpha},\psi_t^{i,\alpha},\phi_t^{i,\alpha})$ is also the L^2 limit of $(\widetilde{X}_t^{i,n},\psi_t^{i,n},\phi_t^{i,n})$. Therefore, letting $n\to\infty$ in Equation (3.15) gives

$$\alpha^{i,\infty}=(q+(1-\frac{1}{N})K_t)\widetilde{X}_t^{i,\alpha}-(1-\frac{1}{N})\psi_t^{i,\alpha}.\tag{3.20}$$

Therefore, the best response for player i is $\alpha^{i,\infty}$, given others play $\boldsymbol{\alpha}^{-i,\infty}$, indicating that the limit $\boldsymbol{\alpha}^\infty$ forms an open-loop Nash equilibrium.

It remains to prove that the limit is independent from the initial belief. Suppose that there exist two limits $\boldsymbol{\alpha}^\infty$ and $\boldsymbol{\beta}^\infty$ arisen from two distinguished initial beliefs $\boldsymbol{\alpha}^0$ and $\boldsymbol{\beta}^0$, and let $(\widetilde{X}_t^{i,\beta},\psi_t^{i,\beta},\phi_t^{i,\beta})$ be the solution to (3.13) associated with $\boldsymbol{\beta}^\infty$. Following similar derivations in the proof of convergence gives:

$$\begin{aligned}
&\max_{i\in\mathcal{I}}\left\|\psi^{i,\alpha}-\psi^{i,\beta}\right\|_2^2 \le \frac{1-e^{-2T\underline{\gamma}}}{2\underline{\gamma}}(1-\frac{1}{N})^2\overline{K}^2\max_{i\in\mathcal{I}}\left\|\alpha^{i,\infty}-\beta^{i,\infty}\right\|_2^2,\\
&\max_{i\in\mathcal{I}}\left\|\widetilde{X}^{i,\alpha}-\widetilde{X}^{i,\beta}\right\|_2^2\\
\le&\frac{1-e^{-2T\underline{\gamma}}}{\underline{\gamma}}\left((1-\frac{1}{N})^2\max_{i\in\mathcal{I}}\left\|\alpha^{i,\infty}-\beta^{i,\infty}\right\|_2^2+(1-\frac{1}{N})^4\max_{i\in\mathcal{I}}\left\|\psi^{i,\alpha}-\psi^{i,\beta}\right\|_2^2\right).
\end{aligned}$$

Combining the above equations together, and using (3.20) for both $\alpha^{i,\infty}$ and $\beta^{i,\infty}$, we deduce:

$$\max_{i\in\mathcal{I}}\left\|\alpha^{i,\infty}-\beta^{i,\infty}\right\|_2^2 \le \frac{1-e^{-2T\underline{\gamma}}}{\underline{\gamma}}C\max_{i\in\mathcal{I}}\left\|\alpha^{i,\infty}-\beta^{i,\infty}\right\|_2^2.$$

Under the same condition (3.16), $\boldsymbol{\alpha}^\infty=\boldsymbol{\beta}^\infty$ in the L^2 sense. Therefore, we have shown that, independent of initial belief, the fictitious play will converge and the limit is unique.

3.3. Identifying the limit. As proved in Theorem 3.1, the limiting strategy $\boldsymbol{\alpha}^\infty$ forms an open-loop Nash equilibrium, and in this section, we verify it coincides with the equilibrium provided in [12] by direct calculations.

Recall from [12], the open-loop Nash equilibrium to the original N-player problem (3.1)–(3.2) is:

$$\alpha_t^{i,*}=[q+(1-\frac{1}{N})\eta_t](\overline{X}_t^*-X_t^{i,*}), \tag{3.21}$$

where $X_t^{i,*}$ is the solution to (3.1) associated with $\alpha_t^{i,*}$, $\overline{X}_t^*$ is the average of $X_t^{i,*}$, and η_t solves a Riccati equation:

$$\dot{\eta}_t=2(a+(1-\frac{1}{2N})q)\eta_t+(1-\frac{1}{N})\eta_t^2-(\epsilon-q^2),\quad \eta_T=c. \tag{3.22}$$

Note that, the expression (3.21) means the open-loop equilibrium happens to be expressed as a function of the states in the equilibrium, but not a closed-loop feedback equilibrium. To be more precise, plugging (3.21) into (3.1) yields

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

Thus, $\alpha_t^{i,*}$ is indeed $\mathcal{F}_t$-measurable. To avoid further confusion in the sequel, we denote by Ξ_t^i the solution to the above SDE, then

$$\alpha_t^{i,*}=[q+(1-\frac{1}{N})\eta_t]\Xi_t^i, \tag{3.23}$$

and Ξ_t^i is the unique strong solution to the SDE:

$$\mathrm{d}\Xi_t^i = -\kappa_t \Xi_t^i \,\mathrm{d}t + \sigma\sqrt{1-\rho^2}\left(\frac{1}{N}\sum_{i=1}^N \mathrm{d}W_t^i - \mathrm{d}W_t^i\right), \quad \Xi_0^i = \overline{x}_0 - x_0^i, \tag{3.24}$$

with

$$\kappa_t = a + q + (1-\frac{1}{N})\eta_t. \tag{3.25}$$

Two properties regarding Ξ_t^i will be used in sequel: firstly, $\sum_{i=1}^N \Xi_t^i = 0$, $\forall t \in [0,T]$. This is straightforward by deriving the SDE for $\overline{\Xi}_t$ via summing (3.24) over $i \in \mathcal{I}$, and using $\overline{\Xi}_0 = 0$. Consequently, we also have $\sum_{i=1}^N \alpha_t^{i,*} = 0$, $\forall t \in [0,T]$. Secondly, one has that $e^{\int_0^t \kappa_u \,\mathrm{d}u}\Xi_t^i$ is a martingale, follows by the SDE (3.24) and the boundedness of η_t on $[0,T]$.

We next verify that the limit $\alpha^{i,\infty}$ coincides with (3.23) by showing the optimal control to the problem (3.5) is $\alpha^{i,*}$ where other players' are following $\alpha^{j,*}$, $j \neq i$, and by the uniqueness of limit under condition (3.16). Denote by $(\widetilde{X}_t^{i,*}, \psi_t^{i,*}, \phi_t^{i,*})$ the solution to the FBSDEs (3.13) with $\alpha^{j,n}$ replaced by $\alpha^{j,*}$, $j \in \mathcal{I}\setminus\{i\}$. Essentially, the problem is to show the player i's optimal response, represented by the solution of FBSDEs, $(q + (1-\frac{1}{N})K_t)\widetilde{X}_t^{i,*} - (1-\frac{1}{N})\psi_t^{i,*}$ matches her Nash strategy $\alpha^{i,*}$. Note that this is not a fixed-point argument as usually seen in mean-field games, since only $\boldsymbol{\alpha}^{-i,*}$ is needed to solve $(\widetilde{X}_t^{i,*}, \psi_t^{i,*}, \phi_t^{i,*})$.

We first solve $\psi^{i,*}$ from the backward process in (3.13). The BSDE is of affine form, and thus possesses a unique solution:

$$\begin{aligned}
\psi_t^{i,*} &= \mathbb{E}\left[\int_t^T -\frac{K_s}{N}\sum_{j\neq i}\alpha_s^{j,*} e^{\int_s^t \gamma_u \,\mathrm{d}u}\,\mathrm{d}s \middle| \mathcal{F}_t\right] = \mathbb{E}\left[\int_t^T \frac{K_s}{N}\alpha_s^{i,*} e^{\int_s^t \gamma_u \,\mathrm{d}u}\,\mathrm{d}s \middle| \mathcal{F}_t\right] \\
&= \mathbb{E}\left[\int_t^T \frac{K_s}{N}[q + (1-\frac{1}{N})\eta_s]\Xi_s^i e^{\int_s^t \gamma_u \,\mathrm{d}u}\,\mathrm{d}s \middle| \mathcal{F}_t\right] \\
&= \int_t^T \frac{K_s}{N}[q + (1-\frac{1}{N})\eta_s]\Xi_t^i e^{-\int_t^s \kappa_u + \gamma_u \,\mathrm{d}u}\,\mathrm{d}s \\
&:= F(t)\Xi_t^i.
\end{aligned}$$

The function $F(t)$ satisfies

$$\dot{F}(t) = F(t)(\kappa_t + \gamma_t) - \frac{K_t}{N}(q + (1-\frac{1}{N})\eta_t), \quad F(T) = 0, \tag{3.26}$$

where γ_t and κ_t are given by (3.14) and (3.25) respectively, and η_t solves (3.22). Note that (3.26) is a first order linear ordinary differential equation (ODE) with smooth coefficients, whose solution in uniqueness is ensured by standard ODE theory. A straightforward calculation shows $K_t - \eta_t$ solves (3.26), thus $\psi_t^{i,*} = (K_t - \eta_t)\Xi_t^i$.

Now to solve the forward equation for $\widetilde{X}_t^{i,*}$, we first calculate

$$\begin{aligned}
\frac{\sum_{j\neq i}\alpha^{j,*}}{N} + (1-\frac{1}{N})^2\psi_t^{i,*} &= -\frac{\alpha^{i,*}}{N} + (1-\frac{1}{N})^2(K_t - \eta_t)\Xi_t^i \\
&= (-\frac{q}{N} + (1-\frac{1}{N})^2 K_t - (1-\frac{1}{N})\eta_t)\Xi_t^i,
\end{aligned}$$

therefore

$$\mathrm{d}\widetilde{X}_t^{i,*} = [(-\frac{q}{N} + (1-\frac{1}{N})^2 K_t - (1-\frac{1}{N})\eta_t)\Xi_t^i - \gamma_t \widetilde{X}_t^{i,*}]\,\mathrm{d}t + \sigma\sqrt{1-\rho^2}\left(\frac{1}{N}\sum_{i=1}^{N}\mathrm{d}W_t^i - \mathrm{d}W_t^i\right).$$

Comparing it to (3.24), one deduces $\widetilde{X}_t^{i,*} = \Xi_t^i$. Therefore, player i's optimal response to her opponents' strategy $\boldsymbol{\alpha}^{-i,*}$ is

$$\begin{aligned}(q + (1-\frac{1}{N})K_t)\widetilde{X}_t^{i,*} - (1-\frac{1}{N})\psi_t^{i,*} &= (q + (1-\frac{1}{N})K_t)\Xi_t^i - (1-\frac{1}{N})(K_t - \eta_t)\Xi_t^i \\ &= (q + (1-\frac{1}{N})\eta_t)\Xi_t^i \equiv \alpha_t^{i,*},\end{aligned}$$

which implies the limit of fictitious play gives an open-loop Nash equilibrium in the linear quadratic case.

4. Numerical experiments

In this section, we present the proof of methodology for deep fictitious play by applying our algorithm to the linear-quadratic game (3.1)-(3.2), which was first introduced in [12] to study the systemic risk. We choose this model as our study for two reasons: firstly, convergence of fictitious play under this setting has been proved in Section 3 under model assumptions. Secondly, closed-form solution exists for this model, which enables us to benchmark the performance of our proposed scheme. Numerical results are shown in three examples of $N = 5, 10, 24$ players.

The Euler scheme (with time step $h = T/N_T$) of the dynamics (3.3)-(3.4) follows from (2.8) with:

$$b^\ell(t, \boldsymbol{x}, \boldsymbol{\alpha}) = a(\overline{\boldsymbol{x}} - x^\ell) + \alpha^\ell, \quad \sigma^\ell(t, \boldsymbol{x}, \boldsymbol{\alpha}) = \sigma^0(t, \boldsymbol{x}, \boldsymbol{\alpha}) \equiv \sigma, \quad \ell \in \mathcal{I}.$$

The model parameters chosen by numerical experiments are

$$T = 1, \quad \sigma = 1, \quad a = 1, \quad q = 0, \quad \rho = 0, \quad \epsilon = 1, \quad c = 1.$$

Remark that in the above choice, if one computes the factor in (3.16), which gives $\frac{1-e^{-2T\gamma}}{\gamma}C = 0.9568, 1.5420,\ 1.9995$ for $N = 5, 10, 24$ respectively, then all the three cases in Proposition 3.1 failed. However, we can still obtain convergent numerical results, which shows the robustness of the proposed algorithms and potential improvement of our theoretical analysis. We choose $M = 2^{16}$ samples for training of the DNNs, and $M' = 10^6$ out-of-samples for final evaluation. A validation split ratio of 25% and callbacks are set to avoid over-fitting. The subnetwork for policy approximation at each time step contains 2 hidden layers and $8+8$ neurons. During each stage, each network is trained for 200 epochs with a mini-batch of size 1024 . A total of 10 stages are played. The true (benchmark) optimal control is computed according (3.21)-(3.22), with η_t given in the closed form.

Example 1 ($N = 5$). We set the initial states of the five players as $x_0 = (1, 5, 7, 3, 8)^{\mathrm{T}}$ and discretize the time interval $[0, 1]$ into $N_T = 50$ steps. In Figure 4.1, we compare the cost functions computed by deep fictitious play to the closed-form solution. One can see that, the relative errors of cost function for all players drop quickly under 5% after a few iterations, and then steadily under 2% after only ten iterations. In Figure 4.2,

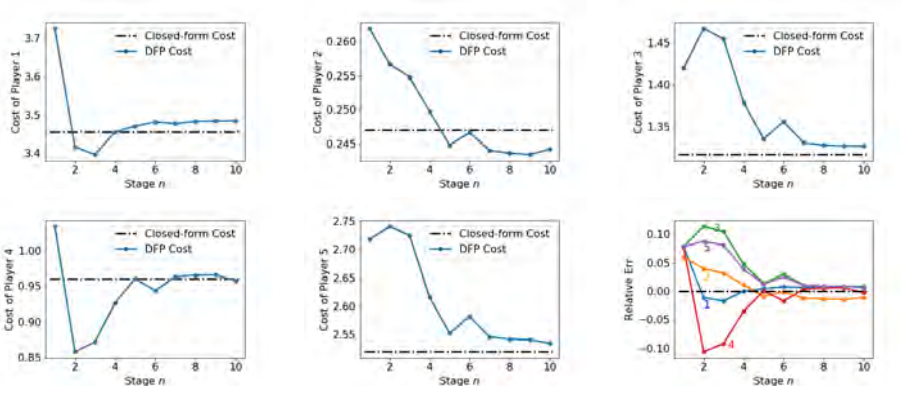

FIG. 4.1. *Comparisons of cost functions for $N=5$ players in the linear quadratic game. The dotted dash lines are the analytical cost functions given by the closed-form solution for each individual player. The solid lines are the cost functions given by deep fictitious play for each player at the first* 10 *iterations. The bottom-right panel shows the relative errors of cost function for the five players, which are pretty small at the 10^{th} iteration.*

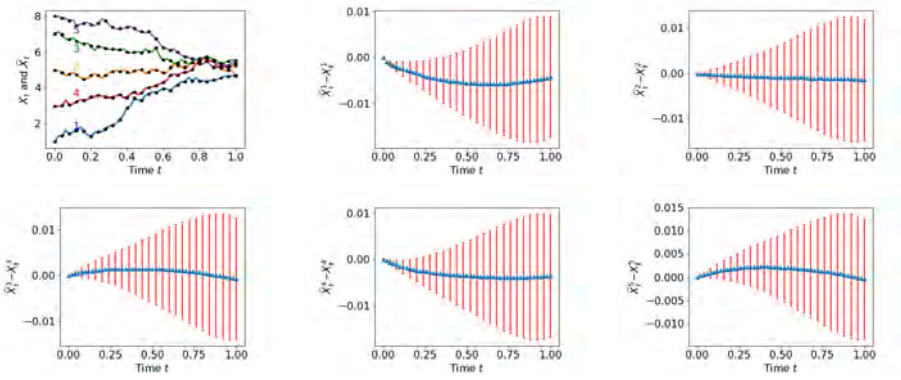

FIG. 4.2. *Comparisons of optimal trajectories for $N=5$ players in the linear quadratic game. Top-left panel: a single sample path of the true optimal trajectories X_t (solid lines)* vs. *the ones computed by deep fictitious play $\widehat{X}_t$ (star lines). The other panels show the mean (blue triangles) and standard deviation (red bars, plotted every other time step) of optimal trajectories errors for five players using a total sample of 10^6 paths. Overall, they show a good approximation of deep fictitious play to the linear quadratic game by $N=5$ players.*

we show in the top-left panel optimal trajectories from total five players computed by deep fictitious play (black star lines) *vs.* by closed-form formulae (colored solid lines) at one representative realization. One can observe that players, although start away from each other, become closer as time evolves. This is consistent with the mechanism of costs functions, as they are in favor of being together. To quantitatively measure the performance of our algorithm, we show the mean and standard deviation of the difference between NN predictions and the true solutions in the remaining panels based on a total of 10^6 sample paths. The means are almost zero, with slightly convex or concave curves depending on player's relative ranking initially. Players starting below average tend to have convex feature.

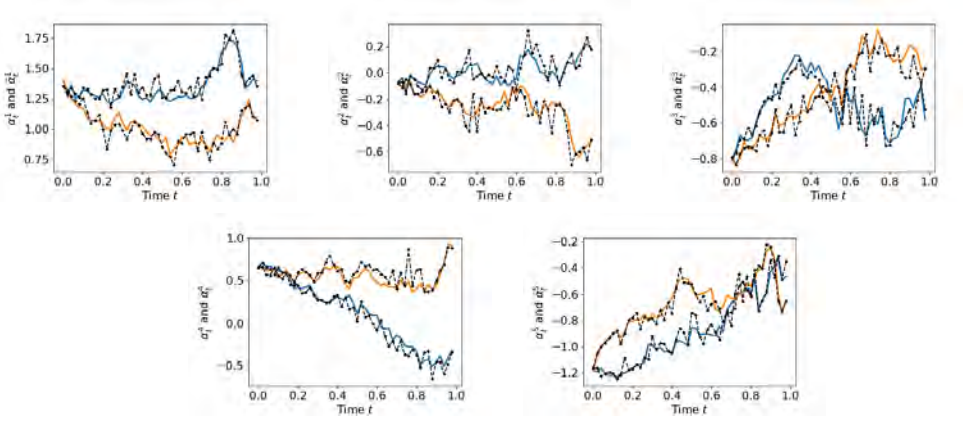

FIG. 4.3. *Comparisons of optimal controls for $N=5$ players in the linear quadratic game. For a sake of clarity, we only show two sample paths of optimal controls for each player. The solid lines are optimal controls given by the closed-form solution, and the dotted dash lines are computed by deep fictitious play.*

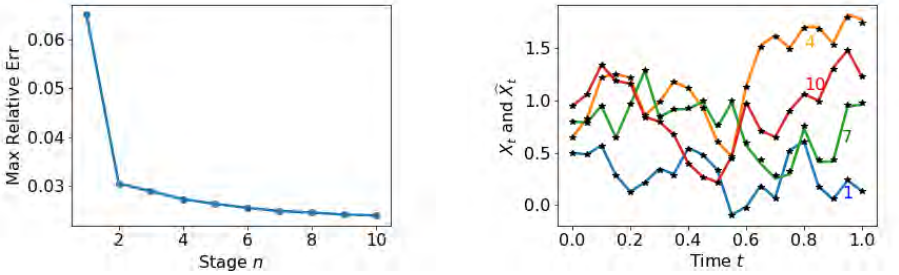

FIG. 4.4. *Comparisons of cost functions and optimal trajectories for $N=10$ players in the linear quadratic game. Left: the maximum relative errors of the cost functions for ten players; Right: for a sake of clarity, we only present the comparison of optimal trajectories for the 1^{st}, 4^{th}, 7^{th} and 10^{th} players, where the solid lines are given by the closed-form solution and the stars are computed by deep fictitious play.*

Standard numerical schemes can do well to approximate cost functions, but not on the derivatives, which are related to the controls, while our deep learning algorithm computes directly the control, which shows a good approximation. Figure 4.3 plots two visualized paths of controls for an illustration purpose.

Example 2 ($N=10$)**.** The initial state for i^{th} player is $x_0^i = 0.5 + 0.05(i-1)$. We use $N_T = 20$ time steps for the discretization of the time interval $[0,1]$. Such choices enable us to investigate the sensitivity of deep learning algorithm on initial positions and time step. In Figure 4.4, we compare the cost functions computed by deep fictitious play to the closed-form solution, where, after only ten iterations, the maximum relative error of cost function for all players have been reduced to less than 3%, and the computed optimal trajectories (one visualized sample path) of selected four players by fictitious play coincide with those of the closed-form solution. The standard deviation of difference between approximated and true optimal trajectories is less than 2% for $t \in [0,1]$ for all players, and we present a selection of six in Figure 4.5.

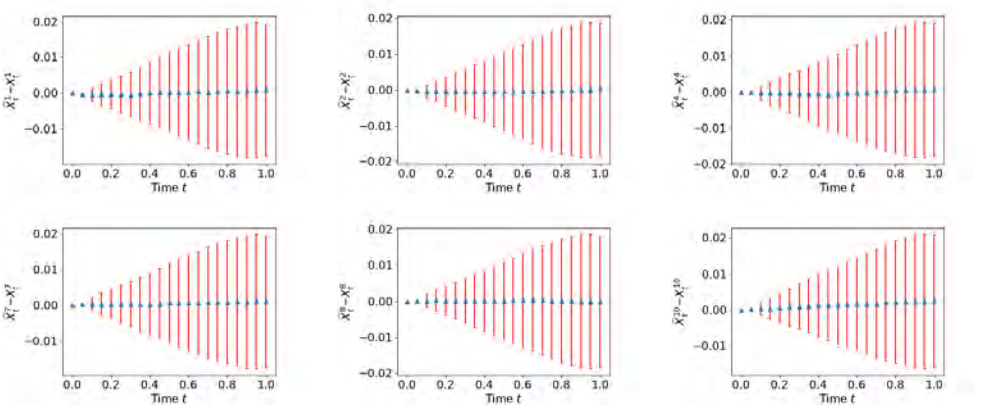

FIG. 4.5. *Comparisons of optimal trajectories for $N=10$ players in the linear quadratic game. For a sake of clarity, we only show the mean (blue triangles) and standard deviation (red bars) of optimal trajectories errors for the 1^{st}, 2^{nd}, 4^{th}, 7^{th}, 8^{th} and 10^{th} player, respectively. The results are based on a total sample of 65536 paths, and show that deep fictitious play provides a uniformly good accuracy of optimal trajectories.*

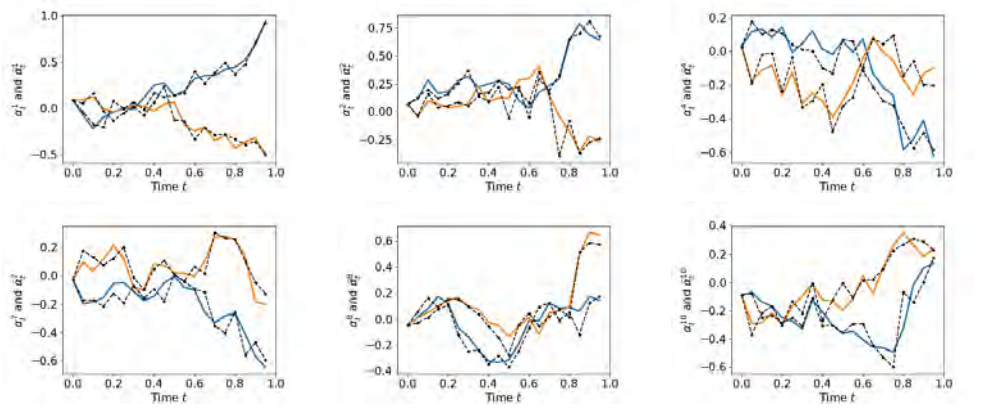

FIG. 4.6. *Comparisons of optimal controls for $N=10$ players in the linear quadratic game. For a sake of clarity, we only show two sample paths of optimal controls for the 1^{st}, 2^{nd}, 4^{th}, 7^{th}, 8^{th} and 10^{th} player, respectively. The solid lines are optimal controls given by the closed-form solution, and the dotted dash lines are computed by deep fictitious play.*

Note that, although the time step h is twice larger than $N=5$, the relative error does not increase significantly. However, we do not observe that the trajectories are getting closer and closer as in the case of $N=5$, since they already start in the neighborhood of each other. We do not observe the curve either, which justifies our assertion that the curvature depends on $\overline{x}_0 - x_0^i$. We also show two visualized sample paths of optimal control in Figure 4.6, which presents a good approximation of the policy.

Example 3 ($N=24$). The initial positions for the i^{th} player is $x_0^i = 0.5i$. We set the time steps $N_T = 20$, after observing the relative errors did not increase too much from $N_T = 50$ to $N_T = 20$. The problem by nature is high-dimensional: the k^{th} "Sequential" subnetwork maps $\mathbb{R}^{Nk}$ to $\mathbb{R}$. To accelerate the computation, we distribute the training

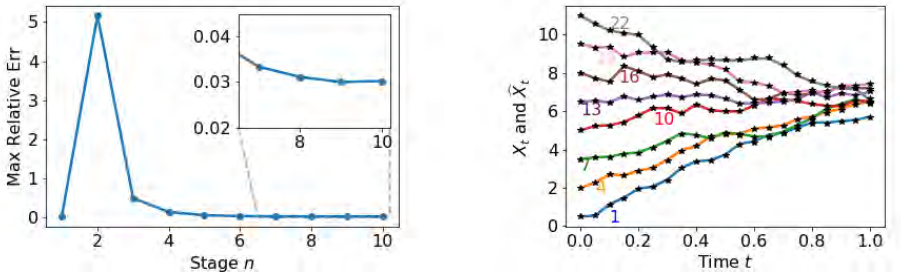

FIG. 4.7. *Comparisons of cost functions and optimal trajectories for $N = 24$ players in the linear quadratic game. Left: the maximum relative errors of the cost functions for ten players; Right: for a sake of clarity, we only present the comparison of optimal trajectories for the 1^{st}, 4^{th}, 7^{th}, 10^{th}, 13^{th}, 16^{th}, 19^{th} and 22^{th} players, where the solid lines are given by the closed-form solution and the stars are computed by deep fictitious play.*

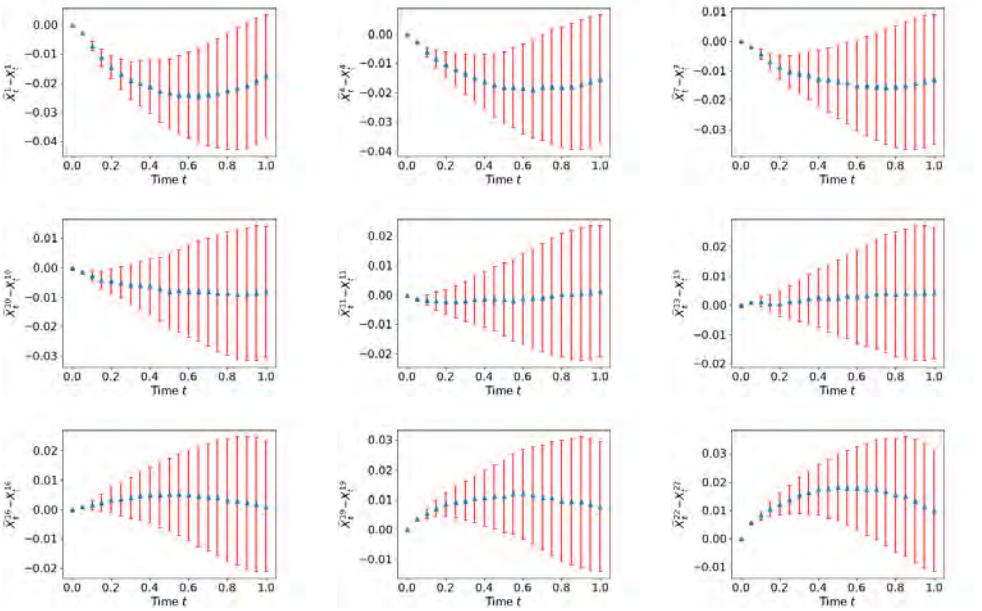

FIG. 4.8. *Comparisons of optimal trajectories for $N = 24$ players in the linear quadratic game. For a sake of clarity, we only show the mean (blue triangles) and standard deviation (red bars) of optimal trajectories errors for the 1^{st}, 4^{th}, 7^{th}, 10^{th}, 11^{th}, 13^{th}, 16^{th}, 19^{th} and 22^{th} player, respectively. The results are based on a total sample of 65536 paths, show that deep fictitious play provides a uniformly good accuracy of optimal trajectories.*

to 8 GPUs. Similar studies to the $N = 10$ case are presented in Figures 4.7-4.9. Some key features that have been observed from previous numerical experiments: the maximum of relative error drops below 3% after ten iterations; the average error of estimated trajectories are convex/concave functions of time t; the standard deviation of estimated error aggregates step by step. In fact, the convexity/concavity with respect to time t is caused by two factors: the propagation of errors, which produces a magnitude increase in error mean; and the existence of terminal cost, which puts more weights on X_T than

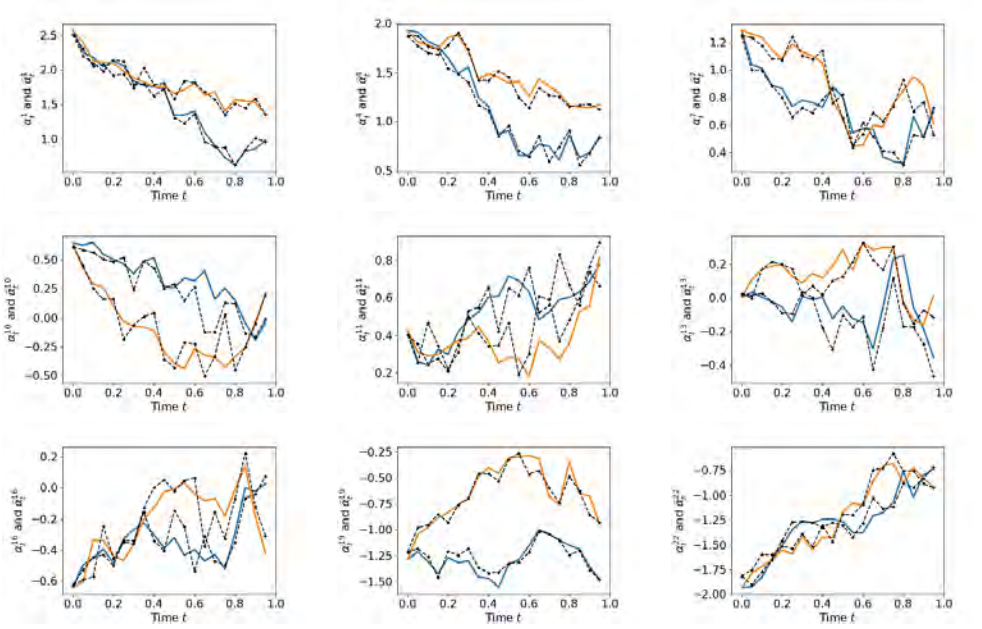

FIG. 4.9. *Comparisons of optimal controls for $N=24$ players in the linear quadratic game. For a sake of clarity, we only show two sample paths of optimal controls for the 1^{st}, 4^{th}, 7^{th}, 10^{th}, 11^{th}, 13^{th}, 16^{th}, 19^{th} and 22^{th} player, respectively. The solid lines are optimal controls given by the closed-form solution, and the dotted dash lines are computed by deep fictitious play.*

TABLE 4.1. *Hyperparameters and runtime for the numerical examples presented in Section 4.*

Problem	N = 5	N = 10	N = 24
N_T	50	20	20
Max Relative Err	1.15%	2.45%	2.95%
# of GPUs used	1	1	8
runtime (hours) †	2.15	14.03	12.10
L^1 error of $\widehat{X}$	1.09e-2	1.49e-2	2.08e-2

† The numerical experiments were conducted using Amazon EC2 services with P2 instances. We remark that the runtime is subject to further reduction with a multi-GPU system or more efficient GPUs.

$X_t, t \in (0,T)$, resulting in a better estimate of X_T and a decreasing effect.

To better illustrate that our algorithm can overcome the curse of dimensionality, we compare the performance across different N. Particularly, we compute

$$\max_{i\in\mathcal{I}} \max_{k\le N_T} \left| X^i_{kh} - \widehat{X}^i_{kh} \right|$$

where X denotes the state process following the open-loop Nash equilibrium, while $\widehat{X}$ is the deep fictitious play counterpart. The L^1 error is 1.09×10^{-2} for $N=5$, 1.49×10^{-2} for $N=10$ and 2.08×10^{-2} for $N=24$. Table 4.1 gives the running time and other hyper-parameters used in the numerical examples.

5. Conclusion, discussion and extension

In this paper, the deep fictitious play theory is proposed to compute the Nash equilibrium of asymmetric N-player non-zero-sum stochastic differential games. We apply the strategy of fictitious play by letting individual player optimize her own payoff while fixing the control of the other players at each stage, and then repeat the game until their responses do not change too much from stage to stage. Finding the best response for each player at each stage is a stochastic optimal control problem, which we approximate by deep neural networks (DNNs). By the nature of open-loop strategies, the problem is recasted into repeated training of N decoupled neural networks (NNs), where inputs of each NN depend on the other NNs' outputs from previous training. Using Keras and parallel GPU simulation, the deep learning algorithm can be applied to any N-player stochastic differential game with different symmetries and heterogeneities. The numerical accuracy and efficiency is illustrated by comparing to the closed-form solution of the linear quadratic case. We also prove the convergence of fictitious play under appropriate assumptions, and show that the convergent limit forms an open-loop Nash equilibrium. We remark that the implementation of this algorithm causes no extra difficulties beyond the linear-quadratic game, but the verification of convergence to the true equilibrium is in general hard due to the lack of benchmark solution. Although one may observe the convergence of the proposed algorithm by tracking the relative change of cost (cf. Step 11 in Algorithm 1), it may actually be trapped in a local (but not true) equilibrium.

In the following, we shall discuss the extensions to other neural network architectures, other strategies of fictitious play and closed-loop Nash equilibrium.

5.1. Other neural network architectures. In the open-loop framework, the searching space for optimal policies contains all $\mathcal{F}_t$-progressively measurable processes, which possesses a path-dependent feature. When using a feedforward architecture, in order to better capture this feature, one needs to partition $[0,T]$ into a sufficiently large number of N_T intervals. Then, a sub-network is used to approximate the optimal policy at each time point (2.9), whose size becomes larger as the time approaches the terminal time T since more history needs to be fed as input. Therefore, the training time increases significantly when one uses large N_T. To improve the performance, architectures based on recurrent neural networks can be considered in solving the stochastic control problem (2.6)–(2.7), for example, using long short-term memory (LSTM), gated recurrent units (GRUs), *etc.* This will be part of our future work [21].

5.2. Belief based on time average of past play. In the formulation (2.2), players' belief is based on their actions during last round, i.e. at stage $n+1$, players myopically respond to their opponents' policies at stage n without considering all decisions before n. This is in fact a bit discrepant from Brown's definition [6, 7], where players responses take into account all past policies. Denote by $\widetilde{\boldsymbol{\alpha}}^{-i,n}$ the weighted average of past play,

$$\widetilde{\boldsymbol{\alpha}}^{-i,n} = \frac{1}{n}\sum_{k=1}^{n} \boldsymbol{\alpha}^{-i,k}, \tag{5.1}$$

then Brown's original idea corresponds to the control problem:

$$\alpha^{i,n+1} := \underset{\beta^i \in \mathbb{A}}{\arg\min}\, J^i(\beta^i; \widetilde{\boldsymbol{\alpha}}^{-i,n}), \quad \forall i \in \mathcal{I}, n \in \mathbb{N}.$$

where J^i is defined as in (2.1).

In general, convergence in the strategy $\boldsymbol{\alpha}^n$ implies convergence in the average of past play $\widetilde{\boldsymbol{\alpha}}^n$, but not vice versa. Therefore, convergence in $\widetilde{\boldsymbol{\alpha}}^n$ does not necessarily lead to a Nash equilibrium. Our numerical tests show that, if the algorithm converges in $\boldsymbol{\alpha}^n$, then using $\widetilde{\boldsymbol{\alpha}}^n$ tends to give a better rate for linear quadratic cases. In practice, within the framework of deep fictitious play, one can generalize (5.1) to any weighted average of past policies: $\sum_{k=0}^n c_k \boldsymbol{\alpha}^{-i,k}$, where $(c_k)_{k=0}^n$ is a n-simplex with $c_n > 0$. We plan to further investigate the comparison between different beliefs for practical problems in future.

5.3. Belief updated alternatively. We shall also mention that, there are actually two versions of fictitious play, the alternating fictitious play (AFP), originally invented in [6], and the simultaneous fictitious play (SFP) mentioned as a minor variant of AFP in [6]. In contrast to (2.2), the players under AFP update their beliefs alternatively. For example, in the case $N=2$, the learning process is:

$$\alpha^{1,n+1} := \operatorname*{arg\,min}_{\beta^1 \in \mathbb{A}} J^1(\beta^1; \alpha^{2,n}), \quad \alpha^{2,n} := \operatorname*{arg\,min}_{\beta^2 \in \mathbb{A}} J^2(\beta^2; \alpha^{1,n}), \quad n \geq 1,$$

and the computation follows $\alpha^{2,0}(\text{initial belief}) \to \alpha^{1,1} \to \alpha^{2,1} \to \alpha^{1,2} \to \alpha^{2,2} \to \ldots$. The dependence of $\alpha^{2,n}$ on $\alpha^{1,n}$ makes one unable to update them simultaneously, which is the main difference from SFP.

Indeed, SFP can be considered as a simpler learning process than AFP, as players are treated symmetrically in time. This usually enhances analytical convenience as well as numerical efficiency (with possible parallel implementation in Step 5-9 of Algorithm 1). Gradually, the original AFP seems to disappear from the literature, and people focus on SFP, even though SFP may generate subtle problems which do not arise under AFP. For a comparison study, we refer to [4], where they also related this subtly to Monderer and Sela's Improvement Principle [51]. We focused on SFP in this paper, where the beliefs can be updated in parallel, and leave the AFP learning process for future studies.

5.4. The algorithm for closed-loop Nash equilibrium. Depending on the space we search for β^i in (2.2), the algorithm can lead to a Nash equilibrium in different setting. Indeed, if we consider $[0,T] \times (\mathbb{R}^d)^N \ni (t, \boldsymbol{x}) \to \beta^i \in A \subset \mathbb{R}^k$ as a function of current states, then the limit yields a feedback strategy for Nash equilibrium. Mathematically,

$$\alpha^{i,n+1}(t,\boldsymbol{x}) := \operatorname*{arg\,min}_{\beta^i(t,\boldsymbol{x}) \in A} J^i(\beta^i(X_t^{i,\beta^i}, \boldsymbol{X}_t^{-i,\boldsymbol{\alpha}^{-i,n}}); \boldsymbol{\alpha}^{-i,n}(X_t^{i,\beta^i}, \boldsymbol{X}_t^{-i,\boldsymbol{\alpha}^{-i,n}})), \tag{5.2}$$

where $\boldsymbol{X}_t^{-i,\boldsymbol{\alpha}^{-i,n}}$ represents players $j \neq i$ state processes following policies $\boldsymbol{\alpha}^{-i,n}$.

This setup can be analyzed by the the partial differential equation (PDE) approach. Assuming enough regularity, the minimal cost can be reformulated as the classical solution to HJB equation where others' strategies are given by deterministic functions obtained from previous round. Consequently, at each stage, the task is to solve N independent HJB equations, which can still be implemented in parallel. Moreover, if the players are statistically identical, one actually only needs to solve one PDE. Denote by $V^{i,n+1}(t,\boldsymbol{x})$ the value function of problem (5.2) at time t with initial states $\boldsymbol{X}_t = \boldsymbol{x}$, by dynamic programming, it satisfies

$$\partial_t V^{i,n+1} + \inf_{\beta} \left\{ b^i(t,\boldsymbol{x},\beta) \partial_{x^i} V^{i,n+1} + f^i(t,\boldsymbol{x},\beta) + \frac{1}{2} \mathrm{Tr}\left[\partial^2_{x^i,x^i} V^{i,n+1} \sigma^i(t,\boldsymbol{x},\beta) \sigma^i(t,\boldsymbol{x},\beta)^\dagger \right] \right.$$

$$+\sum_{\substack{j=1\\ j\neq i}}^{N}\mathrm{Tr}\left[\partial^2_{x^i,x^j}V^{i,n+1}\sigma^i(t,\boldsymbol{x},\beta)\Sigma^{i,j}\sigma^j(t,\boldsymbol{x},\alpha^{j,n})^\dagger\right]\Bigg\}+\sum_{\substack{j=1\\ j\neq i}}^{N}b^j(t,\boldsymbol{x},\alpha^{j,n})\partial_{x^j}V^{i,n+1}$$

$$+\frac{1}{2}\sum_{\substack{j,k=1\\ j\neq i\\ k\neq i}}^{N}\mathrm{Tr}\left[\partial^2_{x^j,x^k}V^{i,n+1}\sigma^j(t,\boldsymbol{x},\alpha^{j,n})\Sigma^{j,k}\sigma^k(t,\boldsymbol{x},\alpha^{k,n})^\dagger\right]=0,$$

$$\alpha^{i,n}\equiv\alpha^{i,n}(t,\boldsymbol{x}):=\operatorname*{arg\,min}_{\beta\in A}\left\{b^i(t,\boldsymbol{x},\beta)\partial_{x^i}V^{i,n}+f^i(t,\boldsymbol{x},\beta)\right\},\quad \Sigma^{j,k}\,\mathrm{d}t:=\mathrm{d}\left\langle W^j,W^k\right\rangle_t.$$

Then, numerically, one can design traditional finite difference/element methods, or use deep learning which has shown excellent performance in overcoming the curse of dimensionality in high-dimensional PDEs [18, 24]. After all, the optimal response function $\alpha^{i,n+1}$ is given in terms of $\partial_{x^i}V^{i,n+1}$,$\partial^2_{x^i,x^j}V^{i,n+1}$. However, a common drawback of working on the value function J^i is that numerical schemes usually well approximate the solution but not the derivative of the solution, which is more sensitive.

An alternative way is to work directly on the control. By a stochastic maximum principle argument, the optimal control is linked to the solution (not the derivative) of FBSDEs, see, *e.g.*, [10, Section 2.2]. Then it is promising to apply the recent deep learning algorithm for the coupled FBSDEs [25]. In this case, at each stage, the task is to solve N independent FBSDEs and parallel implementation is still possible.

Both approaches rely on the property of the reformulated problem: the solution's regularity in the PDE approach and the Hamiltonian's convexity in the FBSDEs approach. A third possibility is to work with the optimization (5.2) directly as we do in the open-loop case. That is, using the deep NN to approximate the control and find the optimal parameters that minimize (5.2). However, due to the feedback reaction, the Algorithm 1 and architectures proposed in Section 2.2 are no longer suitable. It is this "indirect" reaction nature of the open-loop strategy that enables us to design N separate NNs and a scalable algorithm. While working with feedback controls, the realized opponents' strategies $\boldsymbol{\alpha}^{-i,n}(t,\boldsymbol{X}_t)$ depend on β^i. Further explained by Figure 2.2, this means that, $\boldsymbol{\alpha}_1^{-i}$, previously considered as intermediate outputs from NNs of other players at previous training, now depend on β_0^i through X_1^i. Consequently, to take into account the direct reaction of her opponents, one needs to feed β_0^i to player j^{th} NN, $j\neq i$ for intermediate output $\boldsymbol{\alpha}_1^{-i}$. This makes the N-neural networks coupled with each other, and hard to implement in parallel.

Apparently, using deep fictitious play for Markovian Nash equilibrium is not a simple modification of Algorithm 1, and two of the three aforementioned approaches (PDE and direct) are studied in the follow-up works [22, 23].

Acknowledgment. I am grateful to Professor Marcel Nutz for the stimulating and fruitful discussions on fictitious play and convergence of linear quadratic case.

REFERENCES

[1] A. Bachouch, C. Huré, N. Langrené, and H. Pham, *Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications*, Methodol. Comput. Appl. Probab., 2021. 1, 1

[2] Y. Bengio, *Learning deep architectures for AI*, Found. Trends Mach. Learn., 2(1):1–127, 2009. 1, 1

[3] U. Berger, *Fictitious play in 2×n games*, J. Econ. Theory, 120(2):139–154, 2005. 1

[4] U. Berger, *Brown's original fictitious play*, J. Econ. Theory, 135(1):572–578, 2007. 5.3

[5] A. Briani and P. Cardaliaguet, *Stable solutions in potential mean field game systems*, Nonlinear Diff. Eqs. Appl., 25(1):1, 2018. 1
[6] G.W. Brown, *Some notes on computation of games solutions*, Technical report, Rand Corp Santa Monica, CA, 1949. 1, 5.2, 5.3
[7] G.W. Brown, *Iterative solution of games by fictitious play*, in T.C. Koopmans (eds.), Activity Analysis of Production and Allocation, John Wiley & Sons, New York, 374–376, 1951. 1, 5.2
[8] P. Cardaliaguet and S. Hadikhanloo, *Learning in mean field games: the fictitious play*, ESAIM Control Optim. Calc. Var., 23(2):569–591, 2017. 1
[9] R. Carmona and F. Delarue, *Probabilistic analysis of mean-field games*, SIAM J. Control Optim., 51(4):2705–2734, 2013. 1
[10] R. Carmona and F. Delarue, *Probabilistic Theory of Mean Field Games with Applications I*, Springer, 2017. 1, 5.4
[11] R. Carmona and F. Delarue, *Probabilistic Theory of Mean Field Games with Applications II*, Springer, 2018. 1, 2
[12] R. Carmona, J.-P. Fouque, and L.-H. Sun, *Mean field games and systemic risk*, Commun. Math. Sci., 13(4):911–933, 2015. 3, 3, 3.3, 4
[13] F. Chollet et al., Keras, 2015. 2.3
[14] R. Cressman and C. Ansell, *Evolutionary dynamics and Extensive Form Games*, MIT Press, 5, 2003. 1
[15] G. Cybenko, *Approximations by superpositions of a sigmoidal function*, Math. Control Signals Syst., 2:303–314, 1989. 2.1
[16] C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou, *The complexity of computing a Nash equilibrium*, SIAM J. Comput., 39:195–259, 2009. 1
[17] T. Dozat, *Incorporating Nesterov momentum into Adam*, International Conference on Learning Representations 2016 - Workshop Track, 2016. 2.1
[18] W. E, J. Han, and A. Jentzen, *Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations*, Commun. Math. Stat., 5(4):349–380, 2017. 5.4
[19] D.P. Foster and H.P. Young, *On the nonconvergence of fictitious play in coordination games*, Games Econ. Behav., 25(1):79–96, 1998. 1
[20] J. Han and W. E, *Deep learning approximation for stochastic control problems*, Deep Reinforcement Learning Workshop, NIPS, 2016. 1, 1, 2.2.1
[21] J. Han and R. Hu, *Deep learning-based methods for stochastic control problems with delay*, in preparation, 2020. 5.1
[22] J. Han and R. Hu, *Deep fictitious play for finding Markovian Nash equilibrium in multi-agent games*, Proceedings of The First Mathematical and Scientific Machine Learning Conference, 107:221-245, 2020. 5.4
[23] J. Han, R. Hu, and J. Long, *Convergence of deep fictitious play for stochastic differential games*, arXiv preprint, arXiv:2008.05519, 2020. 5.4
[24] J. Han, A. Jentzen, and W. E, *Solving high-dimensional partial differential equations using deep learning*, Proc. Natl. Acad. Sci., 115(34):8505–8510, 2018. 5.4
[25] J. Han and J. Long, *Convergence of the deep BSDE method for coupled FBSDEs*, Probab. Uncertain. Quant. Risk, 5(1):1–33, 2020. 5.4
[26] J. Heinrich and D. Silver, *Deep reinforcement learning from self-play in imperfect-information games*, arXiv preprint, arXiv:1603.01121, 2016. 1
[27] J. Hofbauer and W.H. Sandholm, *On the global convergence of stochastic fictitious play*, Econometrica, 70(6):2265–2294, 2002. 1
[28] S. Hon-Snir, D. Monderer, and A. Sela, *A learning approach to auctions*, J. Econ. Theory, 82(1):65–88, 1998. 1
[29] K. Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural Netw., 4(2):251–257, 1991. 2.1
[30] R. Hu, *Deep learning for ranking response surfaces with applications to optimal stopping problems*, Quant. Finance, 20(9):1567–1581, 2020. 2.1
[31] 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 Trans. Automat. Control, 52(9):1560–1571, 2007. 1
[32] 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*, Commun. Info. Sys., 6(3):221–252, 2006. 1
[33] C. Huré, H. Pham, A. Bachouch, and N. Langrené, *Deep neural networks algorithms for stochastic control problems on finite horizon, part I: convergence analysis*, arXiv:1812.04300, 2018. 1, 1, 2.1

[34] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, Int. Conf. Mach. Learn., 448–456, 2015. 2.3
[35] J.S. Jordan, *Three problems in learning mixed-strategy Nash equilibria*, Games Econ. Behav., 5(3):368–386, 1993. 1
[36] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, International Conference of Learning Representations, 2015. 2.1, 2.3
[37] A.N. Kolmogorov, *On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition*, Dokl. Akad. Nauk SSSR, 114(5):953–956, 1957. 2.1
[38] V. Krishna and T. Sjöström, *On the convergence of fictitious play*, Math. Oper. Res., 23(2):257–511, 1998. 1
[39] A. Krizhevsky, I. Sutskever, and G.E. Hinton, *Imagenet classification with deep convolutional neural networks*, Adv. Neur. Inf. Process. Syst., 1097–1105, 2012. 1
[40] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel, *A unified game-theoretic approach to multiagent reinforcement learning*, Adv. Neur. Inf. Process. Syst., 4190–4203, 2017. 1
[41] J.-M. Lasry and P.-L. Lions, *Jeux à champ moyen. I. Le cas stationnaire*, C.R. Math. Acad. Sci. Paris, 9:619–625, 2006. 1
[42] 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, 2006. 1
[43] J.-M. Lasry and P.-L. Lions, *Mean field games*, Japanese J. Math., 2:229–260, 2007. 1
[44] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521(7553):436–444, 2015. 1, 1
[45] J. Ma, J.-M. Morel, and J. Yong, *Forward-backward Stochastic Differential Equations and their Applications*, Springer Science & Business Media, 1702, 1999. 2
[46] J. Ma, Z. Wu, D. Zhang, and J. Zhang, *On well-posedness of forward-backward SDEs–A unified approach*, Ann. Appl. Probab., 25(4):2168–2214, 2015. 2
[47] D. Mguni, J. Jennings, and E.M. de Cote, *Decentralised learning in systems with many, many strategic agents*, Thirty-Second AAAI Conference on Artificial Intelligence, 2018. 1
[48] P. Milgrom and J. Roberts, *Adaptive and sophisticated learning in normal form games*, Games Econ. Behav., 3(1):82–100, 1991. 1
[49] 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. 1
[50] D. Monderer and A. Sela, *A 2×2 game without the fictitious play property*, Games Econ. Behav., 14(1):144–148, 1996. 1
[51] D. Monderer and A. Sela, *Fictitious play and no-cycling conditions*, Technical report, Technion - Israel Institute of Technology, 1997. 5.3
[52] D. Monderer and L.S. Shapley, *Fictitious play property for games with identical interests*, J. Econ. Theory, 68(1):258–265, 1996. 1
[53] D. Monderer and L.S. Shapley, *Potential games*, Games. Econ. Behav., 14(1):124–143, 1996. 1
[54] M. Nielsen, *Neural networks and deep learning*, http://neuralnetworksanddeeplearning.com/. 2.2.1
[55] M. Nutz, J. San Martin, and X. Tan, *Convergence to the mean field game limit: A case study*, Ann. Appl. Probab., 30(1):259–286, 2020. 1
[56] E. Pardoux and S. Peng, *Adapted solution of a backward stochastic differential equation*, Syst. Control Lett., 14(1):55–61, 1990. 3.1
[57] E. Pardoux and S. Tang, *Forward-backward stochastic differential equations and quasilinear parabolic PDEs*, Probab. Theory Relat. Fields, 114(2):123–150, 1999. 2
[58] S. Peng and Z. Wu, *Fully coupled forward-backward stochastic differential equations and applications to optimal control*, SIAM J. Control Optim., 37(3):825–843, 1999. 2, 3.1
[59] A. Pinkus, *Approximation theory of the MLP model in neural networks*, Acta Numer., 8:143–195, 1999. 1
[60] W.B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, John Wiley & Sons, 703, 2007. 1
[61] S.J. Reddi, S. Kale, and S. Kumar, *On the convergence of Adam and beyond*, 6th International Conference on Learning Representations, 2018. 2.1
[62] J. Robinson, *An iterative method of solving a game*, Ann. Math., 54(2):296–301, 1951. 1
[63] L. Shapley, *Some topics in two-person games*, Adv. Game Theory, 52:1–29, 1964. 1
[64] J. Zhang, *Backward Stochastic Differential Equations: From Linear to Fully Nonlinear Theory*, Springer, 2017. 3.2