Journal of Machine Learning **ISSN**: 2790-2048(e), 2790-203X(p)

Stochastic Delay Differential Games: Financial Modeling and Machine Learning Algorithms

Robert Balkin * [1], Hector D. Ceniceros † [1], and Ruimeng Hu ‡ [1,2]

[1]Department of Mathematics, University of California, Santa Barbara, CA 93106-3080, USA.
[2]Department of Statistics and Applied Probability, University of California, Santa Barbara, CA 93106-3080, USA.

Abstract. In this paper, we propose a numerical methodology for finding the closed-loop Nash equilibrium of stochastic delay differential games through deep learning. These games are prevalent in finance and economics where multi-agent interaction and delayed effects are often desired features in a model, but are introduced at the expense of increased dimensionality of the problem. This increased dimensionality is especially significant as that arising from the number of players is coupled with the potential infinite dimensionality caused by the delay. Our approach involves parameterizing the controls of each player using distinct recurrent neural networks. These recurrent neural network-based controls are then trained using a modified version of Brown's fictitious play, incorporating deep learning techniques. To evaluate the effectiveness of our methodology, we test it on finance-related problems with known solutions. Furthermore, we also develop new problems and derive their analytical Nash equilibrium solutions, which serve as additional benchmarks for assessing the performance of our proposed deep learning approach.

Keywords:
Stochastic delay differential games,
Deep fictitious play,
Recurrent neural networks,
Portfolio games.

Article Info.:
Volume: 3
Number: 1
Pages: 23 - 63
Date: March/2024
doi.org/10.4208/jml.230713

Article History:
Received: 13/07/2023
Accepted: 15/02/2024

Communicated by:
Jiequn Han

1 Introduction

Stochastic delay differential games combine game theory and stochastic control problems with delay. These control problems encompass various models applicable to economics, advertising, and finance. For instance, in determining a firm's optimal advertising policy, Gozzi and Marinelli [10] consider a model which incorporates the delayed impact of advertising expenditures on the firm's goodwill. Similarly, in finance, optimal investment and consumption decisions could also take into account delayed market features as is done by Pang and Hussain [23]. Furthermore, these delayed stochastic control problems can often be extended to incorporate interaction with competitors, who can influence both the underlying system dynamics and the objectives of individual actors. In the context of such scenarios, the stochastic control problem with delay can be further extended to a stochastic delay differential game. This framework captures the interaction among all

*Corresponding author. rbalkin@ucsb.edu
†ceniceros@ucsb.edu
‡rhu@ucsb.edu

participants (or players) who select their controls to optimize their objectives. The controls of each player affect the system dynamics, which are modeled as a system of stochastic delay differential equations (SDDEs). The outcome of the game is represented by the concept of Nash equilibrium, which is a collection of all players' choices, ensuring that no player has the incentive to deviate unilaterally.

Despite introducing mathematical and computational challenges, incorporating delay is crucial for developing more complex and realistic models that capture real-world phenomena. For instance, in the analysis of systemic risk, Carmona *et al.* [6] model bank lending and borrowing as a stochastic differential game without delay, assuming a specific form of bank repayments at time t. To enhance the model's realism, the same authors in collaboration with Mousavi [5] consider banks that must repay their borrowings at time t by time $t + \tau$, introducing a delayed factor into the governing dynamics. While this model effectively captures the nature of delayed repayments, it also increases the mathematical and computational complexity of the underlying problem. However, given the widespread occurrence and realistic nature of delayed problems, it is essential to address the computational challenges they present.

The primary reason for these difficulties lies in the inherent dimensionality of the problem. Stochastic differential games already face the curse of dimensionality when the number of players, denoted as N, is large. Adding to this inherent complexity, stochastic delay differential games introduce a possibly infinite-dimensional component, as the drift and volatility of the associated SDDE depend on the entire path. To formalize this, we note that one can employ the approach of dynamic programming to characterize the value functions associated with the closed-loop Nash equilibrium through a system of Hamilton-Jacobi-Bellman (HJB) equations, enabling the determination of Nash equilibrium controls. However, the resulting HJB equations in the delayed case involve derivatives with respect to variables in an infinite-dimensional Hilbert space as detailed in the book by Fabbri *et al.* [8, Section 2.6.8]. Numerically solving this HJB system would require an additional high-dimensional approximation to handle the infinite dimensionality arising from the delay. However, deep learning methodologies are natural choices for solving problems with high dimensionality and have been used in similar instances. For example, Fouque and Zhang [9] parameterize the optimal control with neural networks to solve a mean field control problem arising from an inter-bank lending model with delayed repayments, and Han and Hu [14] solve the stochastic control problems with delay using neural networks.

To address the challenge of high dimensionality in these problems, we propose a deep learning-based method that effectively handles the delay. Inspired by the approach presented in [14], which utilizes recurrent neural networks (RNNs) to solve stochastic control problems with delay, we introduce an algorithm for finding the Nash equilibrium of stochastic delay differential games. Specifically, we parameterize players' controls using RNNs and approximate their objective functions by sampling the game dynamics under these RNN-based controls. The parameters of the RNNs are then optimized using the concept of deep fictitious play, as introduced in [12,16]. The utilization of neural network-based control functions enables us to reformulate the problem in a finite-dimensional setting. Now, the optimization for a given player revolves around selecting the neural network parameters. Moreover, employing RNNs, in particular, allows us to effectively

capture the influence of delay in players' controls, as RNNs have the capability to learn the appropriate memory dependencies present in the true Nash equilibrium controls.

Separately, we introduce a new class of problems motivated by portfolio optimization with delayed dynamics and competition among portfolio managers [2, 18, 19, 23]. These problems arise from a model where portfolio managers trade in a financial market, considering tax consequences and being incentivized to perform well on both an absolute and relative basis. Specifically, there is a delay between when taxes are realized and when they become due. We summarize our results for these new problems in Sections 3.1 and 3.2, with full motivations provided in Appendix A and proofs in Appendix B.

We then validate the proposed deep learning algorithm numerically on a set of problems with known closed-form solutions. This includes the new class of problems that we introduce and solve in this paper as well as the model introduced in [5] to study the systemic risk in bank lending. By considering both new and existing problems, we assess the accuracy of our proposed method using their closed-form solutions as benchmarks. Our numerical experiments confirm the success of our algorithm in approximating the true Nash equilibrium for all the problems considered.

The outline of the paper is as follows. In Section 2, we introduce the mathematical description of stochastic delay differential games and define the notion of closed-loop Nash equilibrium. In Section 3, we formulate and provide analytical solutions for the closed-loop Nash equilibrium of the stochastic delay differential games we later solve numerically using our proposed algorithm. Specifically, we devote Sections 3.1 and 3.2 to present the new problems and their solutions. The model construction of these problems can be found in Appendix A, while the proofs of their solutions are in Appendix B. In Section 4, we propose a numerical algorithm for approximating stochastic delay differential games, and in Section 5, we demonstrate the numerical results of our algorithm compared to the known solutions of the considered problems. Finally, we provide some concluding remarks in Section 6.

2 The mathematical problem

We define the problem mathematically following the similar setup in [14] for stochastic control problems with delay. The general problem we consider begins with an SDDE system, where the delay can be potentially present in both the state variables as well as the controls. Formally, on a complete probability space $(\Omega, \mathcal{F}, \mathbb{P})$, we have the N-player stochastic delay differential game driven by the dynamics

$$\begin{aligned} &\mathrm{d}\boldsymbol{X}_t^{\boldsymbol{\alpha}} = \mu\left(t, \boldsymbol{X}_{[t-\tau,t]}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}_{[t-\tau,t]}\right)\mathrm{d}t + \sigma\left(t, \boldsymbol{X}_{[t-\tau,t]}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}_{[t-\tau,t]}\right)\mathrm{d}\boldsymbol{W}_t, \quad t \in [0,T], \\ &\boldsymbol{X}_t^{\boldsymbol{\alpha}} = \boldsymbol{\zeta}(t), \qquad\qquad t \in [-\tau,0], \\ &\boldsymbol{\alpha}_t = \boldsymbol{\phi}(t), \qquad\qquad t \in [-\tau,0), \end{aligned} \tag{2.1}$$

where $\boldsymbol{X}^{\boldsymbol{\alpha}}$ is the state process which takes values in $\mathbb{R}^n$, and $\boldsymbol{\alpha} = (\alpha^1, \cdots, \alpha^N)$ is the collection of all players' controls. Here, α_t^i is the strategy or decision of player i at time t and takes values in the control space $\mathcal{A}^i \subset \mathbb{R}^{m_i}$. We use the notation $\boldsymbol{X}_{[t-\tau,t]}^{\boldsymbol{\alpha}}$ to represent

the paths of the stochastic process $\boldsymbol{X}^{\boldsymbol{\alpha}}$ along the interval $[t-\tau, t]$, and similar notation for $\boldsymbol{\alpha}_{[t-\tau,t]}$. We call $\tau > 0$ (deterministic) the length of the delay or simply the delay as Eq. (2.1) shows that the increment of the state process at time t depends on the entire history of the state and control processes as far back as τ units in the past. The drift μ and volatility σ are functionals that map into $\mathbb{R}^n$ and $\mathbb{R}^{n\times k}$ respectively, and $\boldsymbol{W}$ is a k-dimensional, standard Brownian motion.

We remark that there is a special case in which one can write the state process as $\boldsymbol{X} = (X^1, \cdots, X^N)$, where X^i is affected only through the control α^i. In this instance, X^i is the private state of player i. In our case, we generically assume $\boldsymbol{X}_t^{\boldsymbol{\alpha}}(\omega) \in \mathbb{R}^n$ and could represent a combination of both private states as well as public states – those that are shared and influenced collectively.

Formally, we define $\boldsymbol{X}^{\boldsymbol{\alpha}}_{[t-\tau,t]}$ to be a map from $[-\tau, 0]$ to the space of square integrable random variables $L^2(\Omega)$ given by $\boldsymbol{X}^{\boldsymbol{\alpha}}_{[t-\tau,t]}(s) := \boldsymbol{X}^{\boldsymbol{\alpha}}_{s+t}$. In particular, we seek solutions $\boldsymbol{X}^{\boldsymbol{\alpha}}$ to Eq. (2.1) such that for each $t \in [0, T]$, we have that $\boldsymbol{X}^{\boldsymbol{\alpha}}_{[t-\tau,t]} \in L^2(\Omega; C([-\tau, 0]; \mathbb{R}^n))$. Here, the space $L^2(\Omega; C([-\tau, 0]; \mathbb{R}^n))$ is defined as the normed space of $C([-\tau, 0]; \mathbb{R}^n)$ valued random variables with the norm given by

$$\|\boldsymbol{Z}\|_{L^2(\Omega;C([-\tau,0];\mathbb{R}^n))} = \left(\mathbb{E}\left[\sup_{s\in[-\tau,0]} |\boldsymbol{Z}_s(\omega)|^2\right]\right)^{\frac{1}{2}}.$$

The stochastic path $\boldsymbol{\alpha}_{[t-\tau,t]}$ is defined analogously by $\boldsymbol{\alpha}_{[t-\tau,t]}(s) := \boldsymbol{\alpha}_{s+t}$, where the stochastic processes $(\boldsymbol{\alpha}_t)_{t\in[0,T]}$ belongs to an admissible set $\mathbb{A}$ defined later by the set definition (2.2).

For a fixed choice of controls $\boldsymbol{\alpha}$, one can consider the existence and uniqueness of the SDDE (2.1). For this SDDE, one can require μ and σ to be Lipschitz in the second argument to ensure the existence and uniqueness of a strong solution. To be precise, this Lipschitz condition is

$$\begin{aligned}
\|\mu(t, \boldsymbol{x_1}, \boldsymbol{\alpha}_{[t-\tau,t]}) - \mu(t, \boldsymbol{x_2}, \boldsymbol{\alpha}_{[t-\tau,t]})\|_{L^2(\Omega)} &\leq L\|\boldsymbol{x_1} - \boldsymbol{x_2}\|_{L^2(\Omega;C([-\tau,0];\mathbb{R}^n))},\\
\|\sigma(t, \boldsymbol{x_1}, \boldsymbol{\alpha}_{[t-\tau,t]}) - \sigma(t, \boldsymbol{x_2}, \boldsymbol{\alpha}_{[t-\tau,t]})\|_{L^2(\Omega)} &\leq L\|\boldsymbol{x_1} - \boldsymbol{x_2}\|_{L^2(\Omega;C([-\tau,0];\mathbb{R}^n))}
\end{aligned}$$

for some $L > 0$ and for all $t \in [0, T]$, $\boldsymbol{x_1}, \boldsymbol{x_2} \in L^2(\Omega; C([-\tau, 0]; \mathbb{R}^n))$. For more details of the existence and uniqueness theory for SDDEs, we refer to the work by Mohammed [22].

Now that we have defined the SDDE and considered its existence and uniqueness for a given $\boldsymbol{\alpha}$, we proceed by specifying the requirements for $\boldsymbol{\alpha}$. We require that each control, α^i, is in the class of closed-loop controls within an admissible set. The closed-loop controls are those that take the form $\alpha_t^i = \phi^i(t, \boldsymbol{X}^{\boldsymbol{\alpha}}_{[-\tau,t]})$, and therefore represent a decision at time t based on observation of the state process up to and including this time. Formally, we define the admissible set of closed-loop controls $\mathbb{A}^i$ for player i to be

$$\begin{aligned}
\mathbb{A}^i = \Bigg\{\beta^i \,\Big|\, \beta_t^i = \phi^i\Big(t, \boldsymbol{X}^{\boldsymbol{\alpha}}_{[-\tau,t]}\Big), \phi^i \text{ measurable, } \phi^i : [-\tau, T] \times L^2\big(\Omega; C([-\tau, T]; \mathbb{R}^n)\big)\\
\to \mathcal{A}^i \subset \mathbb{R}^{m_i}, \int_{-\tau}^{T} \mathbb{E}\Big[\big|\beta_t^i\big|^2\Big]\, \mathrm{d}t < \infty\Bigg\},
\end{aligned} \tag{2.2}$$

and we denote the product space of admissible controls by $\mathbb{A} = \otimes_{i=1}^{N} \mathbb{A}^i$ along with the control space for all players by $\mathcal{A} = \otimes_{i=1}^{N} \mathcal{A}^i$.

Next, we define the running and terminal costs for player i as f^i and g^i, respectively. Here,

$$\begin{aligned} f^i &: [0,T] \times L^2(\Omega, C([-\tau,0];\mathbb{R}^n)) \times L^2(\Omega, C([-\tau,0];\mathcal{A})) \;\; \to \;\; \mathbb{R}, \\ g^i &: L^2(\Omega, C([-\tau,0];\mathbb{R}^n)) \;\; \to \;\; \mathbb{R} \end{aligned}$$

are deterministic measurable functionals. From these functionals, we define the expected cost J^i for player i to be

$$J^i[\boldsymbol{\alpha}] = \mathbb{E}\left[\int_0^T f^i\left(t, \boldsymbol{X}^{\boldsymbol{\alpha}}_{[t-\tau,t]}, \boldsymbol{\alpha}_{[t-\tau,t]}\right) \mathrm{d}t + g^i\left(\boldsymbol{X}^{\boldsymbol{\alpha}}_{[T-\tau,T]}\right)\right]. \tag{2.3}$$

We remark that one can also consider games where J^i in Eq. (2.3) defines the reward of player i rather than cost. In this case, one can cast the problem as one of costs by taking the cost of player i to be $-J^i$. Because of this, we will be referring to problems where J^i represents the cost of player i unless otherwise stated.

The problem we consider is to find the Nash equilibrium in $\mathbb{A}$ for the stochastic delay differential game described in (2.1) and (2.3). The Nash equilibrium is a set of controls whereupon each player has optimally chosen their own control given the choices of controls for the other players. To be precise, we understand the Nash equilibrium through the following definition.

Definition 2.1. *We say that $\boldsymbol{\alpha}^* = (\alpha^{*1}, \cdots, \alpha^{*N}) \in \mathbb{A}$ is a Nash equilibrium if*

$$J^i[\boldsymbol{\alpha}^*] \leq J^i\left[\alpha^{*1}, \cdots, \alpha^{*i-1}, \beta^i, \alpha^{*i+1}, \alpha^{*N}\right], \quad \forall\, i \in \{1, \ldots, N\}, \quad \forall\, \beta^i \in \mathbb{A}^i. \tag{2.4}$$

We remark that since $\mathbb{A}$ by definition contains the admissible, closed-loop controls, we call a Nash equilibrium $\boldsymbol{\alpha}^* \in \mathbb{A}$ a closed-loop Nash equilibrium.

To summarize, the full problem of finding the closed-loop Nash equilibrium for a stochastic delay differential game is defined through the key components (2.1)-(2.4). In essence, Eq. (2.2) defines the appropriate space of controls, while Eqs. (2.1), (2.3) define a map from the choices of controls for each player $\alpha^i \in \mathbb{A}^i$ to the cost experienced by a given player J^i. The condition (2.4) expresses how each player rationally chooses her strategy based on her own cost, given the choices of the other players, leading to equilibrium.

3 Three games with delay

In this section, we present three stochastic differential games with delay. In Sections 3.1 and 3.2, we derive two novel problems that are inspired by [2, 18, 19, 23]. For clarity, we shall present the mathematical formulations and highlight analytical results below and defer modeling motivations and intuitions for these new problems to Appendix A and the proofs of their solutions to Appendix B. In Section 3.3, we briefly review a stochastic

delay differential game arising from inter-bank lending, as discussed in [5], and summarize the results therein. All three problems will serve as benchmarks for the numerical methodology we propose in Section 4.

3.1 Competition between portfolio managers with delayed tax effects

We consider a portfolio game between N managers where everyone's award depends on both their absolute and relative performance, subject to delayed tax effects. Such a problem is inspired by the model problems introduced in [23] and [19], and the full intuition and derivations are elaborated in Appendix A.

Let $X_t^i \in \mathbb{R}$ be the wealth at time t of an investor i. Her wealth process is influenced by $\pi_t^i \in \mathbb{R}$, the fraction of wealth she chooses at time t to allocate into a risky asset, while the remaining is left in a money market account accruing at a risk-free rate $r \in \mathbb{R}$. At time t, the investor pays taxes at a rate of μ_2 on her exponentially averaged past wealth Y_t^i. Precisely, the dynamics for the wealth of each player $i \in \{1, \dots, N\}$ are given by

$$\begin{aligned} \mathrm{d}X_t^i &= \left[(\mu_1 - r)\pi_t^i X_t^i + rX_t^i - \mu_2 Y_t^i\right]\mathrm{d}t + \sigma\pi_t^i X_t^i \,\mathrm{d}W_t, && t \in (0, T], \\ Y_t^i &= \int_{-\infty}^{t} \lambda e^{-\lambda(t-s)} X_s^i \,\mathrm{d}s, && t \in (0, T], \\ X_t^i &= \zeta^i(t), && t \in (-\infty, 0]. \end{aligned} \tag{3.1}$$

Here, W is a 1-D Brownian motion, and the initial wealth ζ^i is positive and bounded for $t \in (-\infty, 0]$. The parameter $\mu_1 \in \mathbb{R}$ is the mean return of the stock with $\mu_1 > r$, and $\sigma > 0$ is its volatility from the Black-Scholes model. The parameter $\lambda > 0$ is the arrival rate of tax billings as explained in Appendix A.1. We note that in this case, the length of the delay is $\tau = \infty$ as seen through the dependency on Y_t^i in Eq. (3.1) which itself depends on the entire path $X^i_{(-\infty,t]}$.

We consider two cases for the reward for player i. The first case is based on the constant absolute risk aversion (CARA) utility and is given by

$$J^i[\boldsymbol{\pi}] = \mathbb{E}\left[U_i\left(Z^i_{disc,T} - \theta_i \overline{Z}_{disc,T}\right)\right], \tag{3.2}$$

where $0 < \theta_i < 1$, and

$$\overline{Z}_{disc,T} = \frac{1}{N}\sum_{i=1}^{N} Z^i_{disc,T}, \quad U_i(z) = -\exp\left(-\frac{1}{\delta_i} z\right) \tag{3.3}$$

with $\delta_i > 0$ and Z^i_{disc} defined by

$$\begin{aligned} Z_t^i &= X_t^i + aY_t^i, \quad a = \frac{1}{2\lambda}\left(-(r+\lambda) + \sqrt{(r+\lambda)^2 - 4\lambda\mu_2}\right), \\ Z^i_{disc,t} &= e^{-(r+\lambda a)t} Z_t^i. \end{aligned} \tag{3.4}$$

The second case is based on the constant relative risk aversion (CRRA) utility and given by

$$J^i[\boldsymbol{\pi}] = \mathbb{E}\left[U_i\left(Z^i_{disc,T}\overline{Z}^{-\theta_i}_{disc,T}\right)\right], \tag{3.5}$$

where $0 < \theta_i < 1$ and

$$\overline{Z}_{disc,T} = \left(\prod_{i=1}^{N} Z^i_{disc,T}\right)^{\frac{1}{N}}, \quad U_i(z) = \begin{cases} \dfrac{1}{1-1/\delta_i} z^{1-\frac{1}{\delta_i}}, & \delta_i \neq 1, \\ \log(z), & \delta_i = 1 \end{cases} \tag{3.6}$$

with $\delta_i > 0$ and Z^i_{disc} defined by Eq. (3.4).

From the definition above, we notice that $(r+\lambda)^2 - 4\lambda\mu_2 > 0$ must be required, which essentially means that the tax effect cannot be too large. We further require $r + \lambda > 0$, resulting in $a < 0$. We also remark that $r + \lambda a$ can be ascribed the meaning of a "tax-adjusted risk-free rate" and Z^i_t can be ascribed the "tax-adjusted wealth". The utilities in Eqs. (3.2) and (3.5) have meaningful interpretations, as discussed in Appendices A.2 and A.4, respectively.

Lastly, for the CRRA case, the admissible set for π^i is extended with additional requirements, i.e. we will take $\pi^i \in \mathbb{A}^i$,

$$\mathbb{A}^i = \left\{\pi^i \,\middle|\, \pi^i_t = \phi^i\left(t, \boldsymbol{X}_{[-\tau,T]}\right) \in \mathbb{R},\ \phi^i \text{ measurable},\ \exists\, K > 0 : \left|\pi^i_t X^i_t\right| \leq K\left|Z^i_t\right|\right\}. \tag{3.7}$$

With $\pi^i \in \mathbb{A}^i$ and taking $\zeta^i(t)$ chosen such that $Z^i_t = X^i_t + aY^i_t > 0$ for all $t \leq 0$, we can show for all i, $Z^i_{disc,t} > 0$ a.s., and therefore the utility given by Eqs. (3.5)-(3.6) is well defined. This is shown in Appendix B.2.

The solution to the CARA case is summarized in the following proposition.

Proposition 3.1. *Consider the stochastic delay differential game defined by the dynamics* (3.1) *with reward for each player $i \in \{1, \ldots, N\}$ given by $J^i = J^i[\boldsymbol{\pi}]$ as defined through Eqs.* (3.2)-(3.4). *Then, there is a closed-loop Nash equilibrium $\boldsymbol{\pi}^*$ given by the controls*

$$\pi^{i,*}_t X^{i,*}_t = \frac{\mu_1 - r}{\sigma^2}\left(\delta_i + \frac{\theta_i\bar{\delta}}{1-\bar{\theta}}\right)\frac{1}{e^{-(r+\lambda a)t}},$$

where

$$\bar{\delta} = \frac{1}{N}\sum_{i=1}^{N}\delta_i, \quad \bar{\theta} = \frac{1}{N}\sum_{i=1}^{N}\theta_i,$$

and $X^{i,*}_t$ satisfies the dynamics in (3.1) *associated with $\pi^{i,*}$. More precisely, the Nash equilibrium strategy at time t is to invest a deterministic dollar amount into the risky asset, independent of the current wealth level.*

Proof. See Appendix B.1. □

This proposition characterizes the resulting Nash equilibrium for the CARA case. For the CRRA case, we have the following result.

Proposition 3.2. *Consider the stochastic differential game with delay defined by the dynamics* (3.1)*, the reward* $J^i = J^i[\boldsymbol{\pi}]$ *for each player* $i \in \{1, \dots, N\}$ *defined through Eqs.* (3.4)-(3.6)*, and the admissible space* $\otimes_{i=1}^N \mathbb{A}^i$ *given by Eq.* (3.7)*. Then, there is a closed-loop Nash equilibrium* $\boldsymbol{\pi}^*$ *given by the controls*

$$\pi_t^{i,*} = \frac{\mu_1 - r}{\sigma^2}\left(\delta_i - \frac{\theta_i(\delta_i - 1)\bar{\delta}}{1 + \overline{\theta(\delta - 1)}}\right)\frac{X_t^i + aY_t^i}{X_t^i},$$

where

$$\bar{\delta} = \frac{1}{N}\sum_{i=1}^N \delta_i, \quad \overline{\theta(\delta - 1)} = \frac{1}{N}\sum_{i=1}^N \theta_i(\delta_i - 1).$$

Proof. See Appendix B.2. □

3.2 Consumption and portfolio allocation game with delayed tax effects

In addition to the delay effects in investment strategies as analyzed in Section 3.1, here we also consider players' consumption strategies which contribute to their relative utility in the reward. Such a problem without delay was studied in [18].

As before, π_t^i represents the fraction of player i's wealth allocated to the risky asset at time t. The second control process c_t^i is person i's rate of consumption at time t as a fraction of her wealth. In addition to the usual admissibility conditions given by Eq. (2.2), we require that $c_t^i \geq 0$ for all $t \in (0, T]$. In this case, the wealth dynamics for player $i \in \{1, \dots, N\}$ are given by

$$\begin{aligned} \mathrm{d}X_t^i &= \left[(\mu_1 - r)\pi_t^i X_t^i + rX_t^i - \mu_2 Y_t^i - c_t^i X_t^i\right]\mathrm{d}t + \sigma\pi_t^i X_t^i\,\mathrm{d}W_t, && t \in (0, T], \\ X_t^i &= \zeta^i(t), && t \in (-\infty, 0], \end{aligned} \tag{3.8}$$

where

$$Y_t^i = \int_{-\infty}^t \lambda e^{-\lambda(t-s)} X_s^i\,\mathrm{d}s$$

is the exponentially decayed moving average of past wealth. The parameters are taken as $\mu_1, r, \lambda, \sigma \in \mathbb{R} : \lambda, \sigma > 0$, $\mu_1 > r$, and $\mu_2 > 0$. The results will stay the same if one has $\mu_2 \in \mathbb{R}$, but only $\mu_2 > 0$ corresponds to taxes. We further require that $(r + \lambda)^2 - 4\lambda\mu_2 > 0$ and $r + \lambda > 0$, as we did in Section 3.1. Again, the length of delay is $\tau = \infty$ as Y_t^i in Eq. (3.8) depends on the entire path $X^i_{(-\infty,t]}$.

With the dynamics fully described, we now define the reward function for player i to be given by

$$J^i[\boldsymbol{\pi}, \boldsymbol{c}] = \mathbb{E}\left[\int_0^T U^i\left(C^i_{disc,t}\overline{C_{disc,t}}^{-\theta_i}\right)\mathrm{d}t + \epsilon_i U_i\left(Z^i_{disc,T}\overline{Z_{disc,T}}^{-\theta_i}\right)\right], \tag{3.9}$$

where $0 < \theta_i < 1$, $\epsilon_i > 0$, and

$$Z_t^i = X_t^i + aY_t^i, \quad a = \frac{1}{2\lambda}\left(-(r + \lambda) + \sqrt{(r + \lambda)^2 - 4\lambda\mu_2}\right), \tag{3.10a}$$

$$C^i_{disc,t} = e^{-(r+\lambda a)t} c^i_t X^i_t, \quad \overline{C_{disc,t}} = \left(\prod_{i=1}^N C^i_{disc,t}\right)^{\frac{1}{N}}, \tag{3.10b}$$

$$Z^i_{disc,t} = e^{-(r+\lambda a)t} Z^i_t, \qquad \overline{Z_{disc,t}} = \left(\prod_{i=1}^N Z^i_{disc,t}\right)^{\frac{1}{N}}, \tag{3.10c}$$

and the utility function is the CRRA utility given by

$$U_i(z) = \begin{cases} \dfrac{1}{1-1/\delta_i} z^{1-\frac{1}{\delta_i}}, & \delta_i \neq 1, \\ \log(z), & \delta_i = 1 \end{cases} \tag{3.11}$$

with $\delta_i > 0$.

In this case, we can again interpret $r + \lambda a$ as the tax-adjusted risk-free rate and Z^i_t as tax-adjusted wealth, and the interpretation of the expected utility (3.9) is discussed in Appendix A.5. This example mainly contrasts with that in Section 3.1 in that each player now has two controls. This changes the mathematical structure and poses additional numerical challenges.

Lastly, to ensure the utility in Eqs. (3.9)-(3.11) is well defined, we require that the controls for player i, (π^i, c^i), live in the admissible set $\mathbb{A}^i$

$$\begin{aligned} \mathbb{A}^i = \Big\{ (\pi^i, c^i) \,\big|\, (\pi^i_t, c^i_t) = \phi^i(t, \boldsymbol{X}_{(-\infty,t]}) \in \mathbb{R} \times \mathbb{R}^+, \, \phi^i \text{ measurable}, \\ \exists K > 0 : \big|\pi^i_t X^i_t\big|, \big|c^i_t X^i_t\big| \leq K \big|Z^i_t\big| \Big\}. \end{aligned} \tag{3.12}$$

With $(\pi^i, c^i) \in \mathbb{A}^i$ and taking the initial path ζ^i chosen such that $Z^i_t = X^i_t + aY^i_t > 0$ for all $t \leq 0$, we can show for all i, $C^i_{disc,t}, Z^i_{disc,t} > 0$ a.s., see details in Appendix B.3.

The characterization of the closed-loop Nash equilibrium is given by the following proposition.

Proposition 3.3. *Consider the stochastic differential game with delay defined by the dynamics* (3.8)*, the reward* $J^i = J^i[\boldsymbol{\pi}, \boldsymbol{c}]$ *for each player* $i \in \{1, \ldots, N\}$ *defined through Eqs.* (3.9)-(3.11)*, and the admissible space* $\otimes_{i=1}^N \mathbb{A}^i$ *defined by Eq.* (3.12)*. Then, there is a closed-loop Nash equilibrium* $(\boldsymbol{\pi}^*, \boldsymbol{c}^*)$ *given by the controls*

$$\begin{aligned} \pi^{i,*}_t &= \frac{\mu_1 - r}{\sigma^2} \left(\delta_i - \frac{\theta_i(\delta_i - 1)\bar{\delta}}{1 + \overline{\theta(\delta - 1)}}\right) \frac{X^i_t + aY^i_t}{X^i_t}, \\ c^{i,*}_t &= \begin{cases} \left(\beta_i^{-1} + (\gamma_i^{-1} - \beta_i^{-1}) e^{-\beta_i(T-t)}\right)^{-1} \dfrac{X^i_t + aY^i_t}{X^i_t}, & \delta_i \neq 1, \\ \left(T - t - \gamma_i^{-1}\right)^{-1} \dfrac{X^i_t + aY^i_t}{X^i_t}, & \delta_i = 1, \end{cases} \end{aligned} \tag{3.13}$$

where

$$\bar{\delta} = \frac{1}{N} \sum_{i=1}^N \delta_i, \quad \overline{\theta(\delta - 1)} = \frac{1}{N} \sum_{i=1}^N \theta_i(\delta_i - 1),$$

and the parameters β_i and γ_i are given by

$$\begin{aligned}\beta_i &= \frac{1}{2}(1-\delta_i)\left(\frac{\mu_1 - r}{\sigma}\right)^2 \left(1 - \frac{\theta_i\bar{\delta}}{1+\overline{\theta(\delta-1)}}\right)\left(\delta_i - \frac{\theta_i\bar{\delta}}{1+\overline{\theta(\delta-1)}}(\delta_i - 1)\right), \\ \gamma_i &= \epsilon_i^{-\delta_i}\left(\left(\prod_{k=1}^{N}\epsilon_k^{\delta_k}\right)^{\frac{1}{N}}\right)^{\frac{\theta_i(\delta_i-1)}{1+\overline{\theta(\delta-1)}}}.\end{aligned} \tag{3.14}$$

Proof. See Appendix B.3. □

3.3 Inter-bank lending model for systemic risk

The last problem we present comes from the study of systemic risk within inter-bank lending. The particular model we follow is a stochastic delay differential game introduced and studied by Carmona *et al.* [5]. In this model, each bank lends/borrows monetary reserves to/from a central bank with their controls being their pace of lending/borrowing. The model includes loan repayments after a fixed time $\tau > 0$, which leads to the delayed component in the SDDE. The log-monetary reserves of bank i, X_t^i, change in a differential manner with respect to this lending/borrowing dynamics along with some noise. Mathematically, X_t^i satisfies the SDDE

$$\begin{aligned}&\mathrm{d}X_t^i = \left(\alpha_t^i - \alpha_{t-\tau}^i\right)\mathrm{d}t + \sigma\mathrm{d}W_t^i, && t \in [0,T], \\ &\alpha_t^i = 0 \in \mathbb{R}, && t \in [-\tau, 0], \\ &X_0^i = \xi^i \in \mathbb{R}.\end{aligned} \tag{3.15}$$

The cost function for bank i is given by

$$J^i[\boldsymbol{\alpha}] = \mathbb{E}\left[\int_0^T \left(\frac{1}{2}\left(\alpha_t^i\right)^2 - q\alpha_t^i(\bar{X}_t - X_t^i) + \frac{\epsilon}{2}\left(\bar{X}_t - X_t^i\right)^2\right)\mathrm{d}t + \frac{c}{2}\left(\bar{X}_T - X_T^i\right)^2\right]. \tag{3.16}$$

The control $\alpha_t^i \in \mathbb{R}$ is their corresponding pace of borrowing ($\alpha_t^i > 0$) or lending ($\alpha_t^i < 0$) at time t. Although the level of volatility, $\sigma > 0$, is the same for each bank, we have that $\{W^i\}_{i=1}^N$ are independent 1-D Brownian motions meaning that each bank experiences their own idiosyncratic noise, and

$$\bar{X}_t = \frac{1}{N}\sum_{i=1}^{N} X_t^i.$$

A bank's choice of action is dictated by its incentive mechanisms illustrated through Eq. (3.16). The main incentive of bank i can be simply stated as a desire to borrow when they deem their reserves to be too low and lend when deemed too high. In particular, bank i will arithmetically compare their level of log-monetary reserves to the mean log-monetary reserves of all banks, $\bar{X}$, with a preference for bank i to have $X_t^i \approx \bar{X}_t$. This is exemplified by the terms $q\alpha_t^i(\bar{X}_t - X_t^i), \epsilon(\bar{X}_t - X_t^i)^2/2$, and $c(\bar{X}_T - X_T^i)^2/2$ in Eq. (3.16). The parameters $q \geq 0, \epsilon > 0$ and $c \geq 0$ respectively represent the degrees to which a bank desires:

1) to borrow when there are too little monetary reserves,

2) to maintain near average capitalization of log-monetary reserves at all times,

3) to have near average capitalization at the final time.

These incentives to have near average levels of log-monetary reserves are balanced by the bank's inclination to avoid lending or borrowing all else equal as represented by the quadratic penalty $(\alpha_t^i)^2/2$ in Eq. (3.16).

The closed-loop Nash equilibrium for the stochastic delay differential game (3.15)-(3.16) is derived and proven in [5]. The result is restated in the proposition below for convenience.

Proposition 3.4 ([5, Proposition 6.1])**.** *Consider the stochastic differential game with delay defined by the dynamics* (3.15) *and with the reward for each player $i \in \{1,\dots,N\}$ given by $J^i = J^i[\boldsymbol{\alpha}]$ as defined through Eq.* (3.16)*. Then, there exists a closed-loop Nash equilibrium $\boldsymbol{\alpha}^*$ given by the control for each player $i \in \{1,\dots,N\}$ by*

$$\alpha_t^{i,*} = 2\big(1-N^{-1}\big)\Bigg[\left(E_1(t,0)+E_0(t)+\frac{q}{2(1-N^{-1})}\right)\big(\bar{X}_t - X_t^i\big) + \int_{t-\tau}^{t}\big(E_2(t,s-t,0)+E_1(t,s-t)\big)\big(\overline{\alpha^*}_s - \alpha_s^{i,*}\big)\,\mathrm{d}s\Bigg],$$

where

$$\overline{\alpha^*} = \frac{1}{N}\sum_{i=1}^{N}\alpha^{i,*},$$

and where $E_0,\cdots,E_2$ are given by the following PDE system in the region $(t,s,r) \in [0,T]\times[-\tau,0]\times[-\tau,0]$:

$$\begin{aligned}
&E_0'(t)+\frac{\epsilon}{2} = 2\big(1-N^{-2}\big)\big(E_1(t,0)+E_0(t)\big)^2 + 2q\big(E_1(t,0)+E_0(t)\big)+\frac{q^2}{2},\\
&\partial_t E_1(t,s)-\partial_s E_1(t,s)\\
&= 2\big(1-N^{-2}\big)\left(E_1(t,0)+E_0(t)+\frac{q}{2\,(1-N^{-2})}\right)\big(E_2(t,s,0)+E_1(t,s)\big),\\
&\partial_t E_2(t,s,r)-\partial_s E_2(t,s,r)-\partial_r E_2(t,s,r)\\
&= 2\big(1-N^{-2}\big)\big(E_2(t,s,0)+E_1(t,s)\big)\big(E_2(t,r,0)+E_1(t,r)\big)
\end{aligned}$$

with boundary conditions given by

$$\begin{aligned}
&E_0(T)=\frac{c}{2},\quad E_1(T,s)=0,\quad E_2(T,s,r)=0,\quad E_2(t,s,r)=E_2(t,r,s),\\
&E_1(t,-\tau)=-E_0(t),\quad E_2(t,s,-\tau)=-E_1(t,s).
\end{aligned}$$

This Nash equilibrium is derived in [5] by first formulating the delayed problem as a stochastic differential game in an infinite-dimensional Hilbert space, and then characterizing

the Nash equilibrium through Hamilton-Jacobi-Bellman equations over a Hilbert space of functions. A thorough discussion of this technique as well as the theory of infinite-dimensional stochastic control problems can be found in [8].

4 The deep learning algorithm

In Section 2, we presented the mathematical formulation for the Nash equilibrium problem of a stochastic delay differential game as the collection of conditions (2.1)-(2.4). We remarked that Eqs. (2.1) and (2.3) allow one to define a map from choices of controls to the corresponding expected costs of each player. With respect to this map, one can define the notion of solution as that of Nash equilibrium (2.4), where one searches for such equilibrium for "allowed" controls in an admissible set given by Eq. (2.2). To approach this problem numerically, we will need to approximate the map from controls to cost functions by discretizing the dynamics of the game and expected cost. Additionally, to make the space of controls numerically tractable, we will have to represent the controls in a finite-dimensional space that still respects the closed-loop structure demanded by the problem. To this end, we describe in Section 4.1 the discretized set-up that allows us to define a map from a finite-dimensional space of RNN-based controls to the numerically approximated expected costs of each player. Such a discretization scheme is straightforward, but we decide to include it for the sake of clarity and completeness. Building on Section 4.1, which focuses on approximating the cost of a single player given a choice of RNN-based controls, Section 4.2 is dedicated to the algorithm for games with delay. The algorithm we propose belongs to a broader methodology called deep fictitious play, introduced in [12,16]. In Section 4.2.1, we provide a general discussion of the main ideas behind deep fictitious play and present the precise deep fictitious play algorithm we propose for solving stochastic delay differential games. Lastly, in Section 4.2.2, we highlight the specific choice of RNNs used in our implementation, namely the long short-term memory (LSTM) architecture.

4.1 The discrete problem

We now consider the discrete analogue of the stochastic delay differential game defined by Eqs. (2.1)-(2.3). We represent the SDDE (2.1) numerically by its associated Euler-Maruyama scheme, and the expected cost (2.3) is estimated with an empirical cost computed by the Monte Carlo method. Instead of searching for controls in the space given by Eq. (2.2), we parameterize each player's control through a given RNN which helps to address the delayed aspect of the problem.

The discrete analogue of the SDDE (2.1) is obtained using the Euler-Maruyama method, taking into account the delay. This is a natural choice for approximating SDDEs, and its convergence properties have been well-established in certain cases, for instance, Mao [20] addresses the case of a single pointwise delay. The discretization of the delay needs to be done on a case-by-case basis, and we will describe it later.

For step size $\Delta t > 0$, we consider the partition of $[-\tau, T]$ given by $\{t_k = k\Delta t : -N_\tau \leq k \leq N_T, k \in \mathbb{Z}\}$, where $N_\tau = \tau/\Delta t$ and $N_T = T/\Delta t$ are integers without loss of gen-

erality[1]. Then, we define the discrete approximation $(\hat{\boldsymbol{X}}_k)_k$ through the Euler-Maruyama scheme

$$\begin{aligned}
\hat{\boldsymbol{X}}_{k+1} &= \hat{\boldsymbol{X}}_k + \hat{\mu}\big(t_k, \hat{\boldsymbol{X}}_{k-N_\tau}, \cdots, \hat{\boldsymbol{X}}_k, \hat{\boldsymbol{\alpha}}_{k-N_\tau}, \cdots, \hat{\boldsymbol{\alpha}}_k\big)\Delta t \\
&\qquad + \hat{\sigma}(t_k, \hat{\boldsymbol{X}}_{k-N_\tau}, \cdots, \hat{\boldsymbol{X}}_k, \hat{\boldsymbol{\alpha}}_{k-N_\tau}, \cdots, \hat{\boldsymbol{\alpha}}_k)\Delta \boldsymbol{W}_k, \quad k = 0, \ldots, N_T - 1, \\
\hat{\boldsymbol{X}}_k &= \boldsymbol{\zeta}(t_k), \qquad k = -N_\tau, \ldots, 0, \\
\hat{\boldsymbol{\alpha}}_k &= \boldsymbol{\phi}(t_k), \qquad k = -N_\tau, \ldots, -1,
\end{aligned} \tag{4.1}$$

where $\Delta \boldsymbol{W}_k = \boldsymbol{W}_{t_{k+1}} - \boldsymbol{W}_{t_k}$ from the original Brownian motion $\boldsymbol{W}$ in Eq. (2.1). The value for $\hat{\boldsymbol{\alpha}}_k = (\hat{\alpha}_k^1, \cdots, \hat{\alpha}_k^N)$ will be determined as the output of N separate RNNs, each player's control is parameterized by their own RNN.

We have also approximated the functionals μ and σ occurring in the SDDE (2.1) with discrete counterparts $\hat{\mu}$ and $\hat{\sigma}$. While the functionals μ, σ are generic, there will be natural choices for their discrete counterparts in some of the common cases that we consider. For example, the problems occurring in Sections 3.1 and 3.2 contain a delay variable given by an integral over the past history of the state process. This can be approximated by a numerical quadrature along the partition or through discretization of a separate ODE that produces this integral. The problem introduced in Section 3.3 contains a delay variable in the form of pointwise evaluation of the control at a time $t - \tau$. This case is easily dealt with because the delay evaluation occurs on the partition as we have that τ and T are both divisible by Δt.

Having defined the discrete approximation to the SDDE, the numerical approximation of the expected cost for player i is given by

$$\hat{J}^i[\hat{\boldsymbol{\alpha}}] = \frac{1}{N_{\text{batch}}} \sum_{\ell=1}^{N_{\text{batch}}} \left[\sum_{k=1}^{N_T} f^i\big(t_k, \hat{\boldsymbol{X}}_k(\omega_\ell), \hat{\boldsymbol{\alpha}}_k\big)\Delta t + g^i\big(\hat{\boldsymbol{X}}_{N_T}(\omega_\ell)\big) \right], \tag{4.2}$$

which is computed by taking N_{batch} samples of the discrete Brownian paths given by $\{(\Delta \boldsymbol{W}_k(\omega_\ell))_{k=0}^{N_T-1} : \omega_\ell \in \Omega\}_{\ell=1}^{N_{\text{batch}}}$, producing the realized trajectories $(\hat{\boldsymbol{X}}_k(\omega_\ell))$ through the discrete dynamics (4.1).

We know that in the continuous setting, each player's control lives in the admissible set $\mathbb{A}^i$ (2.2) of closed-loop controls, meaning decisions at time t are based on the past history of $\boldsymbol{X}$ up to and including time t. To capture this closed-loop aspect in the discrete case, we will require that $\hat{\boldsymbol{\alpha}}_k = (\hat{\alpha}_k^1, \cdots, \hat{\alpha}_k^N)$ be given as outputs of functions of the past history of the state space, $(\hat{\boldsymbol{X}}_{k'})_{k' \le k}$. In particular, each player's strategy will be given through the outputs of an RNN of a fixed architecture.

The concept of RNN was first introduced by Rumelhart *et al.* [25], a work that demonstrates the implementation of the backpropagation algorithm of a neural network that includes hidden units. In our case, the RNN structure naturally allows the control for player i to encapsulate the past history of the state process. We denote the RNN characterizing the actions of player i by $\phi_{RNN}(\cdot; \vartheta_i)$, where ϑ_i represents the parameters of the

[1]If the divisibility is not met, one may perturb Δt, τ and/or T in order to ensure divisibility of $N_\tau = \tau/\Delta t, N_T = T/\Delta t$. Generality is still respected as the perturbations of each can be taken to be arbitrarily small.

RNN for player i. Specifically, player i's control at time t_k for the discretized problem is given as a function of the input sequence $(t_{-N_\tau}, \hat{\boldsymbol{X}}_{-N_\tau}), \cdots, (t_k, \hat{\boldsymbol{X}}_k)$, or

$$\hat{\alpha}_k^i = \phi_{RNN}\left((t_{-N_\tau}, \hat{\boldsymbol{X}}_{-N_\tau}), \cdots, (t_k, \hat{\boldsymbol{X}}_k); \vartheta_i\right). \tag{4.3}$$

The map ϕ_{RNN} in Eq. (4.3) inputs a sequence of arbitrary length by defining it through a recurrence relation. Specifically, the recurrence is on a map we call the RNN cell, or $\phi_{RNNcell}$. The recurrence occurs through a secondary output of the RNN cell called the hidden state, which we label as h^i for player i. In our case, the recurrence starts with an initial value for $h^i_{-N_\tau}$ by some specific choice h_{init}. Then, we define the recurrence relation

$$\begin{aligned} &y_k^i, h_k^i = \phi_{RNNcell}(t_k, \hat{\boldsymbol{X}}_k, h_{k-1}^i; \vartheta_i), && k = -N_\tau + 1, \ldots, N_T, \\ &\hat{\alpha}_k^i = y_k^i, && k = 0, \ldots, N_T - 1, \\ &h_{-N_\tau}^i = h_{init}. \end{aligned} \tag{4.4}$$

The time-k map of this recurrence relation defines a map from $((t_{-N_\tau}, \hat{\boldsymbol{X}}_{-N_\tau}), \cdots, (t_k, \hat{\boldsymbol{X}}_k)) \mapsto y_k^i = \hat{\alpha}_k^i$, which is precisely the map we call ϕ_{RNN} in Eq. (4.3). We remark that the hidden states h_k^i will correspond to each player i as they are dependent on the parameters ϑ_i.

The key observation is that taking the control to be given by the RNN defined by Eqs. (4.3)-(4.4) provides us with a reasonable space to approximate the closed-loop controls in Eq. (2.2). For one, we see that $\hat{\alpha}_k^i$, the discrete control output at time t_k, now depends on the past trajectory of $\hat{\boldsymbol{X}}$ up to and including time t_k, which encapsulates the closed-loop property. This also addresses the impact of the delay as the control at time t_k has memory of past events. At the same time, the dimension of the search space of the control for each player i is reduced to the finite dimension $\dim(\vartheta_i) < \infty$, which allows for tractability of the Nash equilibrium problem as we will see in Section 4.2.1.

In essence, each player i will select their desired neural network parameters ϑ_i, determining their choice of control. With each player's control chosen, the recurrence relations given by both Eqs. (4.1), (4.4) are then iterated together, which produces simulated dynamics for $\hat{\boldsymbol{X}}$. With these simulated dynamics, one can compute the empirical costs for each player $(\hat{J}^i)_{i=1}^N$. This defines a map from "controls" given by the choices of parameters $(\vartheta_i)_{i=1}^N$ to the empirical costs $(\hat{J}^i)_{i=1}^N$, which will allow us to proceed in Section 4.2.1 with a deep fictitious play algorithm for approximating the Nash equilibrium.

4.2 The numerical algorithm

4.2.1 Deep fictitious play for approximating Nash equilibrium

Deep fictitious play is a broad technique introduced in [12,16] whereby Nash equilibrium controls are approximated iteratively via deep learning techniques in a manner akin to Brown's fictitious play [4]. The primary reason for using deep learning is the high dimensionality in the problem being solved in each iterative step of fictitious play when one has a stochastic differential game.

In general, we consider a game defined by a map from choices of controls $\boldsymbol{\alpha} \in \mathbb{A}$ into costs $\boldsymbol{J}[\boldsymbol{\alpha}]$. The idea of fictitious play is to fix all but one player's control, which leads to the decoupled optimization problems

$$\inf_{\beta^i \in \mathbb{A}^i} J^i\left[\alpha^1, \cdots, \alpha^{i-1}, \beta^i, \alpha^{i+1}, \cdots, \alpha^N\right], \tag{4.5}$$

and then we iterate over the solutions to these optimization problems.

Assuming a unique minimum occurs at $\beta^{i,*}$, we use $\beta^{i,*}$ to inform α^i in future iterations of the optimization (4.5). Doing this for each player $i \in \{1, \ldots, N\}$ constitutes one round of this modified fictitious play. In the case of Brown's fictitious play, α^j in the optimization problem (4.5) would be given by the empirical average of player j's control taken over the previous rounds of play. However, for deep fictitious play methodologies, we will usually take α^j to be the exact control from the previous round of play for memory efficiency reasons (see [12, Remarks 3.1, 3.2] for more information).

In this approach, we choose N_{stages} to be the number of stages of fictitious play. Player i at stage s selects her best response given that all other players are using their strategies from the previous round. This leads to the theoretical Algorithm 1 below.

Algorithm 1 Modified Fictitious Play

Initialize each $\alpha^{i,0} \in \mathbb{A}^i$.
for s in 1 to N_{stages} **do**
  for i in 1 to N **do**
    $\alpha^{i,s} = \arg\min_{\beta^i \in \mathbb{A}^i} J^i\left[\alpha^{1,s-1}, \cdots, \alpha^{i-1,s-1}, \beta^i, \alpha^{i+1,s-1}, \cdots, \alpha^{N,s-1}\right]$.
  end for
end for

In Algorithm 1, $\alpha^{i,s}$ is the control for player i at stage s, and we are assuming that the minimizer exists and is unique. If it is not unique, we could choose a particular minimizer. The idea of Algorithm 1 is that the Nash equilibria are characterized precisely by the fixed points of this iteration. However, the convergence of the method outlined by Algorithm 1 is done on a case-by-case basis. For example, in [16] it is shown that Algorithm 1 converges for a Linear-Quadratic stochastic differential game.

Next, Algorithm 1 is purely theoretical as it assumes the solution to the optimization problem (4.5). In reality, (4.5) may be difficult to directly solve due to high dimensionality and may be best approached by deep learning techniques. There are several methods we may take to solve (4.5) via deep learning and any of these would be considered deep fictitious play. Namely, the approach in [12] considers the Hamilton-Jacobi-Bellman (HJB) equation given by the decoupled optimal control problem defined by the optimization problem (4.5). The HJB solution can be framed in terms of a system of backward SDEs (BSDEs) which can be solved via the Deep BSDE method introduced by E *et al.* [7, 13]. Alternatively, the solution of the HJB equation could be approximated with the Deep Galerkin method introduced by Sirignano and Spiliopoulos [26].

In the case of stochastic delay differential games, the associated HJB equation would be infinite-dimensional [8]. Because of this, we use a third approach to solve the optimization

problem (4.5) that is based on the so-called direct parametrization as discussed by Han and E [11]. This approach was originally introduced for stochastic control problems but can be extended into the game setting via the iteration in Algorithm 1 as demonstrated in [16]. In this method, we approach the optimization problems in Algorithm 1 with neural networks. This is done by taking the control β^i to be a neural network and the optimization is done with gradient descent using the cost function J^i as the loss.

In our case, there are a few steps that must be computationally approximated. First, we must approximate the true expected cost J^i with its empirical, discrete counterpart $\hat{J}^i$, Eq. (4.2). This is computed under controls given by the RNNs, $(\phi_{RNN}(\cdot, \vartheta_1), \cdots, \phi_{RNN}(\cdot, \vartheta_N))$, which give the choices of controls for each player defined through Eqs. (4.3)-(4.4) as discussed in Section 4.1. This leads us to Algorithm 2 which is the basis of our proposed numerical method.

Algorithm 2 A Deep Fictitious Play Algorithm via Direct Parametrization

Initialize each $\vartheta_{1,0}, \cdots, \vartheta_{N,0}$ which are the respective parameters of the N different RNNs at stage 0.
Select N_{stages} of deep fictitious play based on the computational budget.
for s in 0 to N_{stages} **do**
    for i in 1 to N **do**
        Compute N_{batch} trajectories $\hat{\boldsymbol{X}}$ of the numerical SDDE (4.1) under the given controls $(\hat{\alpha}^{j,s})_j$, where the controls $\hat{\alpha}^{j,s} = \phi_{RNN}(\cdot; \vartheta_{j,s+1})$ for $j < i$, and $\hat{\alpha}^{j,s} = \phi_{RNN}(\cdot; \vartheta_{j,s})$ for $j \geq i$ are defined by Eqs. (4.3)-(4.4) for each player j.
        Compute the numerical cost $\hat{J}^i$ from Eq. (4.2).
        Compute via automatic differentiation $\nabla_{\vartheta_{i,s}} \hat{J}^i$.
        Do a gradient descent step or similar (e.g. Adam) on $\vartheta_{i,s}$ using learning rate l_r, i.e. $\vartheta_{i,s+1} = \vartheta_{i,s} - l_r \nabla_{\vartheta_{i,s}} \hat{J}^i$.
    end for
end for

As before, we consider N_{stages} of iteration, where $\phi_{RNN}(\cdot, \vartheta_{j,s})$ parameterizes the control for player j at stage s as selected by the neural network parameters $\vartheta_{j,s}$. As motivated by Algorithm 1, we would then like to compute the optimal $\vartheta_{j,s}$ holding the other RNN parameters, $(\vartheta_{j',s})_{j' \neq j}$, fixed. In practice, we do a gradient descent step or a sequence of gradient descent steps to approximate this behavior. Of course, to do this, we will have to compute the $\vartheta_{i,s}$-gradient of $\hat{J}^i$. This is possible numerically through automatic differentiation [3].

As opposed to Algorithm 1, which involves waiting for an entire stage to use the updated controls, we have found the method to be more effective when the updated controls are immediately applied in the training at stage s. To achieve this, for player i, we use controls $\hat{\alpha}^{j,s} = \phi_{RNN}(\cdot; \vartheta_{j,s+1})$ for $j < i$, and $\hat{\alpha}^{j,s} = \phi_{RNN}(\cdot; \vartheta_{j,s})$ for $j \geq i$ to simulate the trajectories $\hat{\boldsymbol{X}}$, allowing for the immediate use of the updated parameters to train the controls of player i within the same stage. It is worth mentioning that this corresponds to

alternating fictitious play in [4], while Algorithm 1 implements the simultaneous fictitious play.

Lastly, instead of using standard gradient descent, we choose to use the Adam optimization which adaptively chooses the learning rate based on the mean and variance of the gradients involved in the computation of the loss. The choice of using the Adam optimization over traditional gradient descent is due to it having improved convergence properties in many cases [17]. This gives us the deep fictitious play algorithm shown above in Algorithm 2.

To better approximate the argument minimizer in Algorithm 1, several gradient steps might be needed. However, this would require a new computation of $\hat{J}^i$ and its gradient with respect to the updated RNN parameters after each gradient descent step, leading to increased costs.[2] Instead, in Algorithm 2, we move on immediately to the next player's optimization after a single gradient descent step of the current player. For this reason, Algorithm 2 is not meant to be a perfect numerical analogue of Algorithm 1, but is instead merely based on it. Importantly, Algorithm 2 still reflects the property that Nash equilibria are fixed points of the iteration from an intuitive point of view.[3]

4.2.2 The long short-term memory recurrent network

Algorithm 2 showcases the numerical method we use to approximate the Nash equilibrium controls of stochastic delay differential games using RNN-based controls. It only remains to specify a precise form of the RNNs as we have presented these networks quite broadly through Eqs. (4.3)-(4.4). Motivated by the implementation in [14], we choose to use the specific RNN architecture given by the so-called long short-term memory (LSTM).

The LSTM was first introduced by Hochreiter and Schmidhuber [15] and was built to effectively handle the vanishing gradient problem. The LSTM will have two variables that both play the role of the hidden state of an RNN demonstrated in Eq. (4.4). Confusingly, one is called the hidden state h and the other is the cell state c, although we will see they are both defined recurrently and therefore act as hidden states with respect to the generic RNN architecture we have defined by Eq. (4.4). The map ϕ_{LSTM} maps an input vector $(x_0, \cdots, x_k)$ of arbitrary length to an output, hidden, and cell state through a recursive dependence on its previous outputs of the previous input $(x_0, \cdots, x_{k-1})$. The recurrence is given through a function called the LSTM cell, which we will denote $\phi_{LSTMcell}$. In particular $\phi_{LSTMcell}$ directly maps the inputs (x_k, c_{k-1}, h_{k-1}) to the outputs c_k, h_k, y_k according to

$$i_k = \sigma(W_i x_k + U_i h_{k-1} + b_i), \tag{4.6a}$$

[2] One possibility would be to incorporate having additional gradient descent steps for a single player before moving onto the next when one is at later stages of play. The idea is that at the beginning stages, since the approximate controls $(\hat{\alpha}^{i,s})_i$ are not yet close to the Nash equilibrium, it is not as important to fully optimize a single player given the choices of others since the other players are not yet close enough to the Nash equilibrium. However, at later stages, as the Nash equilibrium is approached, it may be beneficial to more fully optimize $\hat{\alpha}^{i,s}$ given the choices of the other players.

[3] Of course it is exceedingly unlikely that controls of the form $(\phi_{RNN}(\cdot, \vartheta^{i,*}))_i$ happen to be a Nash equilibrium due to it being a finite-dimensional object in an infinite-dimensional space. However, if it so happens to be the case that $(\phi_{RNN}(\cdot, \vartheta^{i,*}))_i$ is a Nash equilibrium for $\hat{J}$, and assuming sufficient smoothness of $\hat{J}$, then for each i it holds that $\nabla_{\vartheta^{i,*}} \hat{J}^i[(\phi_{RNN}(\cdot, \vartheta^{j,*}))_j] = 0$, and therefore $(\vartheta^{i,*})_{i=1}^N$ is a fixed point of the iteration.

$$f_k = \sigma(W_f x_k + U_f h_{k-1} + b_f), \tag{4.6b}$$

$$o_k = \sigma(W_o x_k + U_o h_{k-1} + b_o), \tag{4.6c}$$

$$c_k = f_k \odot c_{k-1} + i_k \odot \tanh(W_c x_k + U_c h_{k-1} + b_c), \tag{4.6d}$$

$$h_k = o_k \odot \tanh(c_k), \tag{4.6e}$$

$$y_k = W_y h_k + b_y. \tag{4.6f}$$

Here, $\odot$ denotes the Hadamard product. The individual mappings within the LSTM cell to i_k, f_k, and o_k are known as the input, forget, and output gate respectively. Denoting the input dimension, $\dim(x_k)$, to be N_{input}, and denoting the hidden dimension N_{hidden} for the size of each $i_k, f_k, \cdots, h_k$, we see that each matrix $W_i, W_f, \cdots, W_c$ is of size $N_{hidden} \times N_{input}$ and the bias vectors $b_i, b_f, \cdots, b_c$ are of size N_{hidden}. The final output is $y_k \in \mathbb{R}^{N_{output}}$, so W_y is in $\mathbb{R}^{N_{output} \times N_{hidden}}$ and b_y is in $\mathbb{R}^{N_{output}}$. For player j, the LSTM cell map, $\phi_{LSTMcell}(\cdot; \vartheta_j)$, is determined by the choice of parameters $\vartheta_j = (W_i^j, \cdots, W_y^j, U_i^j, \cdots, U_c^j, b_i^j, \cdots, b_y^j)$, representing the weight matrices and bias vectors in Eq. (4.6) specifically for player j.

With the cell-map, $\phi_{LSTMcell}$, specified by Eq. (4.6), the choice of controls in Eq. (4.1) is determined by taking player j's control at time t_k to be

$$\hat{\alpha}_k^j = \phi_{LSTM}\left((-\tau, \hat{X}_{-N_\tau}), \cdots, (t_k, \hat{X}_k); \vartheta_j\right),$$

where this mapping $\phi_{LSTM}(\cdot; \vartheta_j)$ is given by the forward iteration of the recurrence relation

$$\begin{aligned} x_k &= (t_k, \hat{X}_k), && k = -N_\tau, \ldots, N_T, \\ y_k^j, c_k^j, h_k^j &= \phi_{LSTMcell}(x_k, h_{k-1}^j, c_{k-1}^j; \vartheta_j), && k = -N_\tau + 1, \ldots, N_T - 1, \\ \hat{\alpha}_k^j &= y_k^j, && k = 0, \ldots, N_T - 1. \end{aligned} \tag{4.7}$$

In the cases where the dimension of the state process is equal to the number of players (i.e. $n = N$), we will choose $h_{-N_\tau}^j = c_{-N_\tau}^j = (\hat{X}_0^j, 0, \cdots, 0) \in \mathbb{R}^{N_{hidden}}$ to start the forward iteration, following the implementation in [14]. Note that $x_k = (t_k, \hat{X}_k)$ is a vector in $\mathbb{R}^{1+n}$, as $\hat{X}_k$ is a vector in $\mathbb{R}^n$ for each k. This means that while the input dimension is fixed $N_{input} = 1 + n$, one is free to choose the size of the hidden dimension, N_{hidden}, depending on the user's desired size of the network. In our implementation, we choose $N_{hidden} = 2^6$.

4.2.3 Implementation details

For computational efficiency, we may not be inputting a single sample of $(t_k, \hat{X}_k)$ into the LSTM as indicated by Eq. (4.7), but rather a so-called "batch" which contains N_{batch} paths of $\hat{X}$ determined by respective N_{batch} samples of Brownian paths.

Precisely, we can augment the operations in the Euler-Maruyama method (4.1) so that it iterates over a batch $(\hat{X}_k(\omega_\ell))_{\ell=1}^{N_{\text{batch}}}$ which is represented as a matrix in $\mathbb{R}^{n \times N_{\text{batch}}}$. This is done by generating N_{batch} samples of the Brownian increments $(\Delta W_k(\omega_\ell))_{\ell=1}^{N_{\text{batch}}}$. The drift $\hat{\mu}$ and volatility $\hat{\sigma}$ are extended to act pointwise across the batch dimension.

The control given by the neural network must also be able to provide the respective outputs for each sample of $\hat{\boldsymbol{X}}$ along the batch dimension by acting pointwise across the batch dimension. Note that a generic linear layer $x \mapsto Wx + b$ extends to the mapping $(x_1, \cdots, x_{N_{\text{batch}}}) \mapsto W(x_1, \cdots, x_{N_{\text{batch}}}) + (b, \cdots, b)$, with the property that $x_i \mapsto Wx_i + b$. Because of this, we see that the map $\phi_{LSTMcell}$ in Eq. (4.6) naturally acts pointwise along the batch dimension. To have the LSTM defined by Eqs. (4.6)-(4.7) extended to act pointwise on the batch, we will take the input vector x_k to be

$$x_k = \left(t_k, \hat{\boldsymbol{X}}_k(\omega_\ell)\right)_{\ell=1}^{N_{\text{batch}}} \in \mathbb{R}^{(1+n)\times N_{\text{batch}}}.$$

We notice that this will imply that h_k^j, c_k^j in Eq. (4.7) must also be tensorized along this batch dimension and we will have $c_k, h_k \in \mathbb{R}^{N_{hidden}\times N_{\text{batch}}}$. For the operations in Eq. (4.6) to act on a batch, we will have to resize the bias vectors $b_i, \cdots, b_c \in \mathbb{R}^{N_{hidden}}$ to be of size $\mathbb{R}^{N_{hidden}\times N_{\text{batch}}}$ by repeating their original values across the batch dimension. The matrices $W_i, \cdots, W_c$ will remain the same size of $\mathbb{R}^{N_{hidden}\times(1+n)}$.

The end result is the ability to work with the map from the choice of controls $(\phi^1_{LSTM}(\cdot, \vartheta_1), \cdots, \phi^1_{LSTM}(\cdot, \vartheta_N))$ to the cost function $\hat{J}^i$ in Eq. (4.2) in a tensorized form. From a computational perspective, we avoid looping over each sampled trajectory to compute the numerical cost function (4.2). This tensorization is especially useful when working with automatic differentiation supported libraries such as PyTorch or TensorFlow, as these tensorized operations can be easily and automatically parallelized [1, 24].

5 Numerical results

In Section 3, we have presented three stochastic delay differential games with analytical formulas of their closed-loop Nash equilibrium. In each case, we now numerically approximate the closed-loop Nash equilibrium for $N = 10$ players via our proposed numerical method, Algorithm 2 in Section 4.2.

We introduce below in Section 5.1 the precise details of our numerical experiments that serve as a reference point for the construction of the plots shown in Sections 5.2-5.4, the parameter values used for each of these problems, and an important implementation detail for the problems with infinite delay. In Sections 5.2-5.4 we present and interpret the numerical results for each of the considered problems.

5.1 Numerical results methodology

5.1.1 Costs/rewards over training

For typical machine learning problems, one usually has a training curve – a plot of the loss function over the course of training, which serves as an initial gauge of the effectiveness of training. While the loss function for each player can be seen through the player's empirical cost (4.2), the controls are meant to approximate a Nash equilibrium, the trajectory of each player's loss function over the course of training is not an appropriate measure of the effectiveness of training in this case. The impact of training can be better seen through

the relative error of these costs under the LSTM controls to that corresponding to the true Nash equilibrium controls.

Therefore, for every 20 rounds of deep fictitious play (DFP) within Algorithm 2, we compute the 2-norm relative error

$$\text{Relative 2-Norm Error} = \frac{\|\hat{\boldsymbol{J}}[\boldsymbol{\phi}_{LSTM}] - \hat{\boldsymbol{J}}[\boldsymbol{\alpha}^*]\|_2}{\|\hat{\boldsymbol{J}}[\boldsymbol{\alpha}^*]\|_2}, \tag{5.1}$$

where $\hat{\boldsymbol{J}}[\boldsymbol{\phi}_{LSTM}]$ and $\hat{\boldsymbol{J}}[\boldsymbol{\alpha}^*]$ respectively are the vectors containing the empirical cost for each player under the LSTM controls $(\phi^1_{LSTM}, \cdots, \phi^N_{LSTM})$ defined in Eqs. (4.6)-(4.7) and the true Nash equilibrium controls $(\alpha^{*1}, \cdots, \alpha^{*N})$ of the mathematical problem defined by Eqs. (2.1),(2.3). We then plot this relative 2-norm error as it evolves over the course of training for each of the problems we consider throughout Sections 5.2-5.4.

5.1.2 Comparison of state and control trajectories

After training, we have the collection of LSTM control functions for each player, $(\phi_{LSTM}(\cdot;\vartheta_i))_{i=1}^N$. We demonstrate the ability of these surrogate functions to approximate the true Nash equilibrium controls by comparing the trajectories of both control and state processes under both the LSTM and true Nash equilibrium controls for a given sample of Brownian motion.

This is done by selecting a single realization of the discrete Brownian motion's path $(\Delta \boldsymbol{W}_k(\omega))_{k=0}^{N_T-1}$, and with this given noise simulate the discretized dynamics (4.1) once under the Nash equilibrium controls and again under the LSTM controls. We then compare the dynamics given the two different choices of controls. The plots of these dynamics are shown for each problem occurring throughout Sections 5.2-5.4. Each player is distinguished with a given color, while solid and dashed lines correspond to the dynamics under LSTM controls and Nash equilibrium controls respectively. Lastly, while we have performed the training for 10 players to demonstrate the methodologies' ability to handle larger games, we will only plot 5 out of 10 players' trajectories for the sake of visual clarity.

5.1.3 Model parameters

The model parameters chosen for the numerical experimentation of each problem are shown through Tables 5.1-5.4 below. We express a dependency on i for parameters specific to player i, e.g. $\delta_i = 3/10 + 4(i-1)/9$ in Table 5.1 is player i's risk tolerance parameter. Here, we are still using the indexation $i \in \{1, \ldots, N = 10\}$. Note that in Tables 5.1-5.3, we write $X^i_{(-\infty,0]} = x^i_0$, indicating the initial path for each player is taken to be constant.

Table 5.1: Parameters for CARA case of competition between portfolio managers with delayed tax effects.

N	T	μ_1	σ	r	λ	μ_2	δ_i	θ_i	$X^i_{(-\infty,0]} = x^i_0$
10	10.0	0.08	0.2	0.04	2.0	0.01	$3/10 + 4(i-1)/9$	$3/10 + 4(i-1)/9$	$2 + (i-1)/10$

Table 5.2: Parameters for CRRA case of competition between portfolio managers with delayed tax effects.

N	T	μ_1	σ	r	λ	μ_2	δ_i	θ_i	$X^i_{(-\infty,0]} = x^i_0$
10	1.0	0.08	0.2	0.04	1.0	0.2	$3/10 + 4(i-1)/9$	$3/10 + 4(i-1)/9$	$1 + (i-1)/20$

Table 5.3: Parameters for consumption and portfolio allocation game with delayed tax effects.

N	T	μ_1	σ	r	λ	μ_2	ϵ_i	δ_i	θ_i	$X^i_{(-\infty,0]} = x^i_0$
10	2.0	0.08	0.2	0.04	1.0	0.01	50.0	$3/10 + 4(i-1)/9$	$3/10 + 4(i-1)/9$	$1 + (i-1)/20$

Table 5.4: Parameters for inter-bank lending model.

N	T	σ	q	ϵ	c	τ	$X^i_0 = \xi^i$
10	1.0	.05	1.0	2.0	0.25	0.25	$1 + 0.1 \cdot 1.15^{i-1}$

5.1.4 Approaching the infinite delay cases

We remark on an approximation used in the infinite delay cases appearing in the problems in Sections 3.1 and 3.2 whose numerical results we display in Sections 5.2 and 5.3. In both cases, the delay is contained through the variable

$$Y^i_t = \int_{-\infty}^{t} \lambda e^{-\lambda(t-s)} X^i_s \, \mathrm{d}s.$$

While one option is to truncate the delay resulting in a truncated integral for which a standard numerical integration approach can be applied, this is not necessary. The reason is that in each of these problems, one can show Y^i_t satisfies the ODE relation $\mathrm{d}Y^i_t = \lambda(X^i_t - Y^i_t)\,\mathrm{d}t$, allowing us to approximate $\boldsymbol{Y}_t$ through the forward Euler discretization. Moreover, when the initial path $X^i_{(-\infty,0]}$ is constant, we can easily see that $Y^i_0 = X^i_0$, which greatly simplifies the discrete SDDE iteration by altogether avoiding integration, and therefore the truncation of $\tau = \infty$ is no longer important for $\boldsymbol{Y}_t$. However, we still must impose some finite truncation to the delay τ in order for the forward iteration of the LSTM as described in Eq. (4.7) to be initialized at some finite $-N_\tau$. For the numerical results shown in both Sections 5.2 and 5.3, we have used $\tau = 1.0$ as the truncation for the infinite delay.

5.2 Results for competition between portfolio managers with delayed tax effects

5.2.1 CARA case

We consider now the problem of competition between portfolio managers with delay tax effects, Eq. (3.1), in the CARA case where the rewards are given by Eqs. (3.2)-(3.4). In our numerical experiment, we select the parameter values as shown in Table 5.1.

For this problem, we lower the learning rate throughout training by taking it to be 10^{-2} for the first 500 rounds of DFP, 10^{-3} for the next 500, and 10^{-4} for the remaining 700 rounds. The impact of this training on the approximation of the true Nash equilibrium rewards is shown in Fig. 5.1 as explained in Section 5.1.1. The decreasing relative 2-norm error that plateaus at a level near 10^{-3} indicates a successful training of the controls. On one hand, this demonstrates that the rewards simulated under the LSTM controls are close to the true Nash equilibrium rewards. However, we are also interested to see how the trajectories themselves compare under both LSTM and true Nash equilibrium controls. Following the methodology in Section 5.1.2, we compare the trajectories under the LSTM controls to their true Nash equilibrium counterparts, and the resulting plots are shown in Fig. 5.2. We see that the state process trajectories are nearly identical under both true and LSTM controls respectively. At the same time, the LSTM controls themselves are not only approximating the absolute level of control, but adapting to noises within the state process as illustrated by the LSTM controls matching the shape of the Nash equilibrium controls.

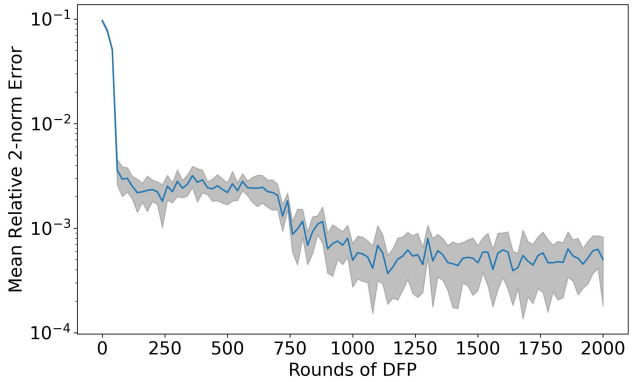

Figure 5.1: The relative 2-norm error (5.1) over the course of training between $(\hat{f}^1, \cdots, \hat{f}^N)$ under the LSTM controls and the true Nash equilibrium controls for the CARA case. The length of training is measured in terms of rounds of DFP. We take $N_{\text{batch}} = 2^{15}$ in the computation of $(\hat{f}^1, \cdots, \hat{f}^N)$ according to Eq. (4.2). For a specific round of DFP, we calculate 10 trials of the relative 2-norm error, as defined in (5.1), and plot its mean in blue and represent the range of one standard deviation by the gray shaded region.

5.2.2 CRRA case

We now consider the problem of competition between portfolio managers with delayed tax effects, Eq. (3.1), in the CRRA case where the rewards are given by Eqs. (3.5)-(3.6). In our numerical experiments, we select the parameters for this problem as shown in Table 5.2. For this problem, we lower the learning rate throughout training by taking it to be 10^{-2} for the first 500 rounds of DFP, 10^{-3} for the next 500, and 10^{-4} for the remaining 500 rounds. The approximation of the empirical rewards under the LSTM controls to that under the true Nash equilibrium controls (see Section 5.1.1 for details) is examined by Fig. 5.3 below. The relative 2-norm error decreasing throughout training and plateauing

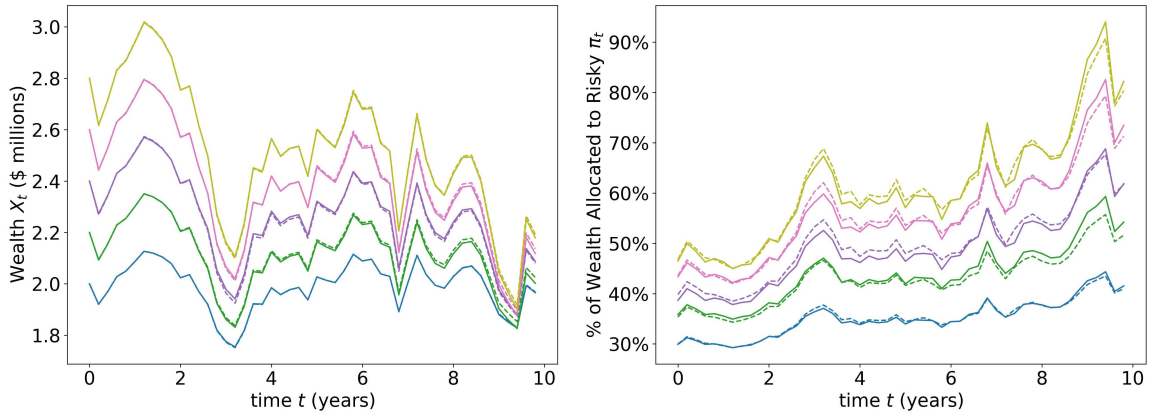

Figure 5.2: Left: A sample path of the wealth processes for players $1, 3, 5, 7$, and 9 (each player corresponds to a unique color) under the true Nash equilibrium controls (dashed) and the trained LSTM control (solid). Right: True Nash equilibrium controls (dashed) vs LSTM controls (solid). Controls represent the fraction of total wealth allocated to the risky asset at time t for a given player.

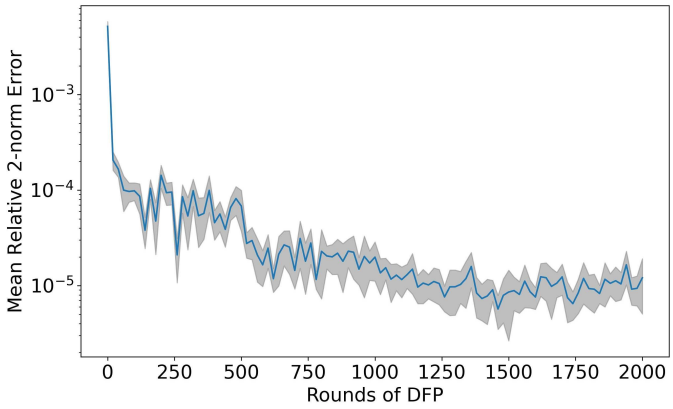

Figure 5.3: The relative 2-norm error (5.1) over the course of training between $(\hat{f}^1, \cdots, \hat{f}^N)$ under the LSTM controls and the true Nash equilibrium controls for the CARA case. The length of training is measured in terms of rounds of DFP. We take $N_{\text{batch}} = 2^{15}$ in the computation of $(\hat{f}^1, \cdots, \hat{f}^N)$ according to Eq. (4.2). For a specific round of DFP, we calculate 10 trials of the relative 2-norm error, as defined in (5.1), and plot its mean in blue and represent the range of one standard deviation by the gray shaded region.

at levels near 10^{-5} indicates that the training is successful and the rewards experienced under the LSTM controls are approximating the rewards experienced under the true Nash equilibrium controls. Fig. 5.4 below allows us to compare the paths induced by these trained controls to their true Nash equilibrium counterparts. As indicated by Fig. 5.4, we see that the numerical methodology results in LSTM controls that accurately depict the true Nash equilibrium dynamics of the problem in question. The corresponding wealth processes under the true and LSTM controls are nearly identical, while the paths of the LSTM controls themselves are coinciding well with their true counterparts.

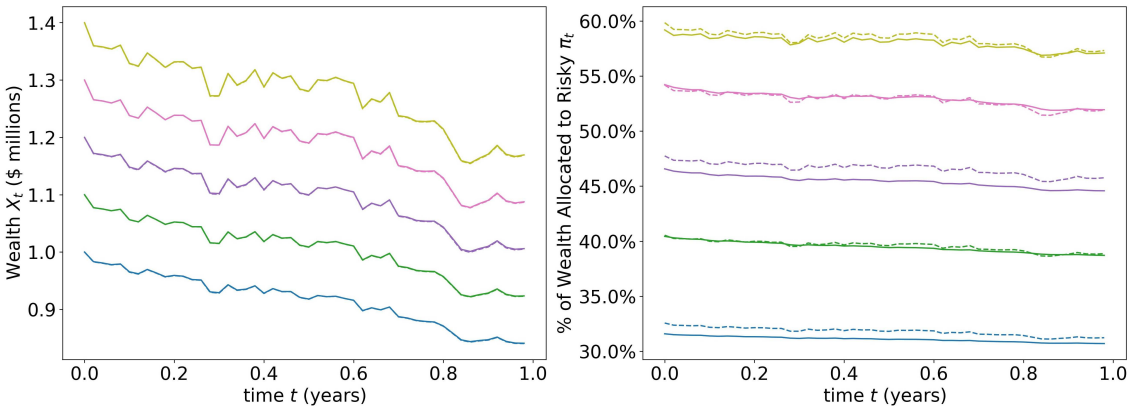

Figure 5.4: Left: A sample path of the wealth processes for players $1, 3, 5, 7$, and 9 (each player corresponds to a unique color) under the true Nash equilibrium controls (dashed) and the trained LSTM control (solid). Right: True Nash equilibrium controls (dashed) vs LSTM controls (solid). Controls represent the fraction of total wealth allocated to the risky asset at time t for a given player.

5.3 Results for consumption and portfolio allocation game with delayed tax effects

We now consider the consumption and portfolio allocation game with delayed tax effects given by Eqs. (3.8)-(3.11). For our numerical experiments, we select the value of the parameters as shown in Table 5.3. In this example, we use 10^{-2} as the learning rate for the first 500 rounds of DFP, 10^{-3} for the subsequent 500 rounds, and 10^{-4} for the final 1000 rounds. Following Section 5.1.1, the successive approximation of the Nash equilibrium rewards over training is illustrated by Fig. 5.5. This example showcases the importance of the training schedule. We notice that there is an initial plateau in the approximation of the true Nash equilibrium empirical rewards around the level of 10^{-3} relative 2-norm error. The learning rate first changes after 500 rounds of DFP and the plateau breaks shortly thereafter. The learning rate is again decreased at round 1000 of DFP, yet another substantial decrease in relative error following this change is not seen. The relative 2-norm error reaches a final plateau near 10^{-5} indicating a successful training.

The success of training is also reflected in the results from comparing the realized trajectories under both true and LSTM controls in Fig. 5.6. In this case, player i has two controls representing the stock allocation at time t, π_t^i, as well as the consumption rate at time t, c_t^i. We plot in Fig. 5.6 their corresponding unnormalized versions which are the wealth allocated to the stock and the annualized run rate of wealth consumed respectively. We see that the trajectories produced by the trained LSTM controls coincide well with those produced under the true Nash equilibrium controls. This is especially apparent in the consumption control, which like the wealth process, contains trajectories that almost entirely overlap with the true Nash equilibrium dynamics. This example highlights the ability of the proposed algorithm to succeed in the important case where each player has multiple controls.

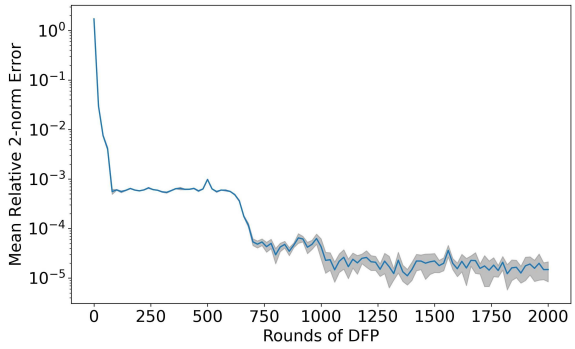

Figure 5.5: The relative 2-norm error (5.1) over the course of training between $(\hat{f}^1, \cdots, \hat{f}^N)$ under the LSTM controls and the true Nash equilibrium controls for the CARA case. The length of training is measured in terms of rounds of DFP. We take $N_{\text{batch}} = 2^{15}$ in the computation of $(\hat{f}^1, \cdots, \hat{f}^N)$ according to Eq. (4.2). For a specific round of DFP, we calculate 10 trials of the relative 2-norm error, as defined in (5.1), and plot its mean in blue and represent the range of one standard deviation by the gray shaded region.

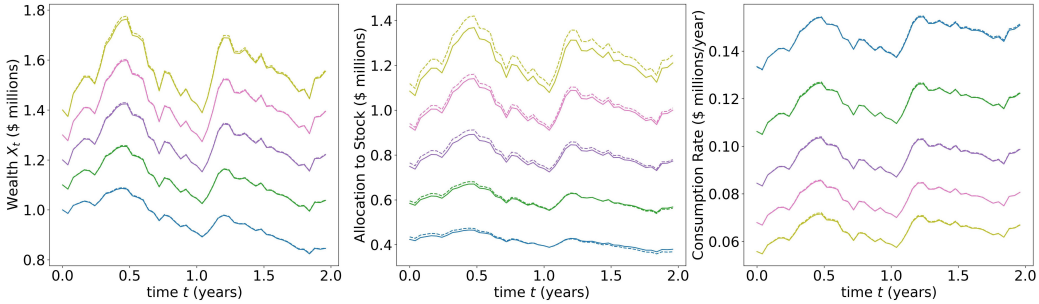

Figure 5.6: Left: A sample path of the wealth processes of players $1, 3, 5, 7$, and 9 under the true Nash equilibrium controls (dashed) and the trained LSTM control (solid) of the corresponding players. Center: Nash equilibrium wealth allocation to stock (dashed) vs LSTM allocation (solid). Right: Nash equilibrium total consumption rate (dashed) vs LSTM consumption rate (solid).

5.4 Results for the inter-bank lending model

Lastly, we present the numerical results for the inter-bank lending model for systemic risk given by Eqs. (3.15)-(3.16). The parameter choices are summarized in Table 5.4. The learning rate iterated throughout the training is chosen to be 10^{-2} in the first 500 rounds of DFP, 10^{-3} in the next 500 rounds, 10^{-4} in the subsequent 500 rounds, and 10^{-5} in the last 2500 rounds. The relative 2-norm error between empirical rewards over training, in this case, is shown in Fig. 5.7.

The 2-norm relative error steadily decreases illustrating a continual improvement throughout training. It is important to note that in this case, what we call the "true" Nash equilibrium controls are actually themselves approximated as the true controls are given in terms of a PDE system (see Proposition 3.4).[4] This additional source of error from the

[4]The PDE system is solved numerically with a discretization of 800 equally spaced slices for $t \in [0, T]$ and 50 slices for both $s \in [-\tau, 0]$ and $r \in [-\tau, 0]$. The PDE is a nonlinear transport equation and the forward Euler scheme is used to solve the PDE.

PDE approximation is likely the cause of the increased smoothness of the curve in Fig. 5.7 and the slightly higher levels of relative error compared to previous cases. We would like to highlight that the gray shaded area, representing plus or minus one standard deviation from the mean, is not visible in this case. This is due to the computed values of the relative 2-norm error being significantly higher than the standard deviation of this quantity at a given stage. Such an observation suggests a notably stable performance for this particular example. Despite this additional source of error, the relative 2-norm error between these two reaches levels close to 10^{-3}, and the plateau of this error indicates successful training. However, the comparison of trajectories, Fig. 5.8, is the foremost illustration showcasing the success of the numerical algorithm for this particular example.

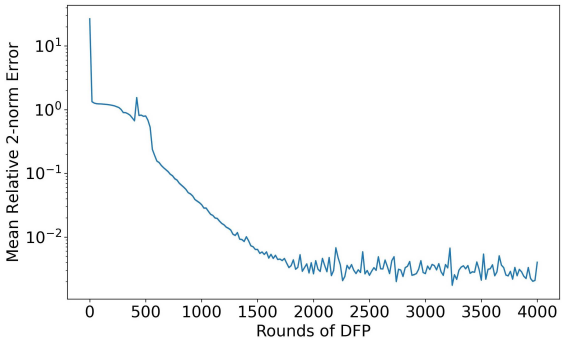

Figure 5.7: The relative 2-norm error (5.1) over the course of training between $(\hat{f}^1, \cdots, \hat{f}^N)$ under the LSTM controls and the true Nash equilibrium controls for the CARA case. The length of training is measured in terms of rounds of DFP. We take $N_{\text{batch}} = 2^{12}$ in the computation of $(\hat{f}^1, \cdots, \hat{f}^N)$ according to Eq. (4.2). For a specific round of DFP, we calculate 10 trials of the relative 2-norm error, as defined in (5.1), and plot its mean in blue and represent the range of one standard deviation by the gray shaded region.

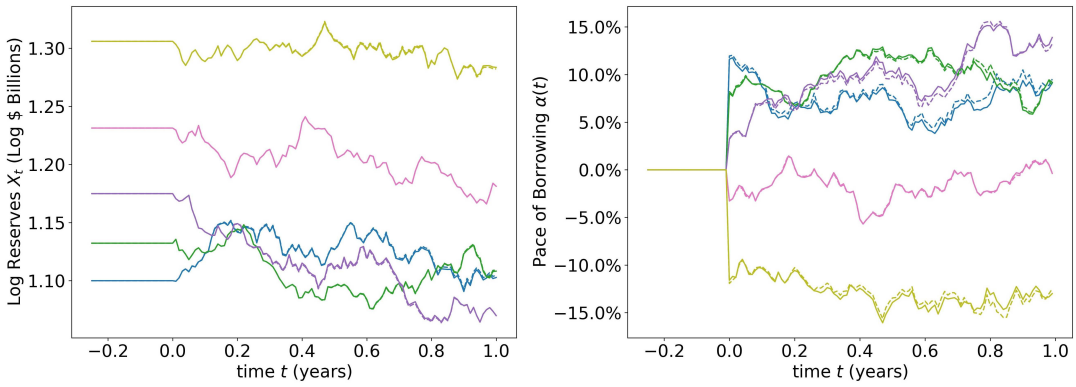

Figure 5.8: Left: A sample path of the log-monetary reserves of banks $1, 3, 5, 7$, and 9 under the true Nash equilibrium controls (dashed) and the trained LSTM control (solid). Right: The corresponding paths (with respect to the left picture) of Nash equilibrium controls (dashed) and LSTM controls (solid) for each bank. The controls represent the pace of borrowing/lending for each bank measured as the annualized run rate of borrowing as a percentage of current monetary reserves.

This is an interesting and unique case in that this problem's dynamics involved delay with respect to the controls themselves rather than the state process. Despite this complexity, we see that the LSTM controls are successful in approximating the true Nash equilibrium control trajectories by conforming to their shape and absolute levels exceptionally well, causing their induced state processes to be nearly identical to that of the true Nash equilibrium.

6 Conclusion

Finding the closed-loop Nash equilibrium for stochastic delay differential games poses challenges due to the inherent high dimensionality of the problem. To address this numerically, we propose a deep learning algorithm that involves parameterizing the controls with LSTM recurrent networks and simulating the trajectories of the discretized dynamical system under these LSTM controls to approximate the expected cost for each player. By approximating the expected costs given a choice of the LSTM-based controls, the LSTMs are trained in an iterative process inspired by the concept of fictitious play.

Separately, we have introduced and analyzed a new class of stochastic delay differential games that arise in finance and have obtained their analytical solutions. The derivation, interpretation, and proofs for these problems are presented in Appendices A and B.

From the numerical experiments we conducted, we have observed compelling evidence that our proposed algorithm successfully approximates the true Nash equilibrium controls. The trajectories of the trained controls closely match those of the true controls after training. Furthermore, we have observed the successful approximation of the costs (or rewards) for each player under the LSTM controls towards the true Nash equilibrium costs (or rewards) during the course of training. This is demonstrated by a decrease in the relative 2-norm error between these vectors throughout the training process, with relative errors reaching values lower than 10^{-5} in some cases.

Appendix A. Interpretation and derivations of the games in Sections 3.1-3.2

A.1. Portfolio optimization with delayed taxes effects

We start with restating the original Merton problem [21]. Consider an investor who chooses between a stock and a bond. At time t, the fraction of wealth π_t is invested in the stock, and $1-\pi_t$ is invested in the bond. In general, we have $\pi_t \in \mathbb{R}$, where $\pi_t > 1$ corresponds to a leveraged stock position, while $\pi_t < 0$, corresponds to the investor being short the stock. The bond is assumed to accrue at a continuously compounded rate of interest r and the stock evolves according to the Black-Scholes model $\mathrm{d}S_t/S_t = \mu\,\mathrm{d}t + \sigma\,\mathrm{d}W_t$. Denoting by X_t the investor's wealth at time t, one then has

$$\mathrm{d}X_t = [(\mu - r)\pi_t X_t + rX_t]\,\mathrm{d}t + \sigma\pi_t X_t\,\mathrm{d}W_t, \quad t \in (0, T]. \tag{A.1}$$

The investor aims to choose a strategy π to optimize her expected utility of terminal wealth

$$J[\pi] = \mathbb{E}[U(X_T)], \tag{A.2}$$

subject to the dynamics (A.1). Intuitively, the expected utility quantifies the investor's desire for the random outcome X_T.

We now consider an additional outflow of wealth due to taxes in Eq. (A.1). We assume that at the end of the period $[t, t + \mathrm{d}t]$, the investor pays the amount $\mu_2 Y_t \, \mathrm{d}t$ in taxes, where

$$Y_t = \int_{-\infty}^{t} \lambda e^{-\lambda(t-s)} X_s \, \mathrm{d}s$$

is the investor's exponentially averaged past wealth. This leads to the modified dynamics of the wealth process X given by

$$\mathrm{d}X_t = \big((\mu_1 - r)\pi_t X_t + r X_t - \mu_2 Y_t\big) \, \mathrm{d}t + \sigma \pi_t X_t \, \mathrm{d}W_t, \quad t \in (0, T]. \tag{A.3}$$

This model with $\mu_2 < 0$ was introduced and solved in [23] arising from a type of momentum effect. For our consideration, we take $\mu_2 > 0$ and the term $\mu_2 Y_t \, \mathrm{d}t$ represents the fact that the investor is paying the fixed percentage (or tax rate) $\mu_2 > 0$ of their historical wealth. This outflow could represent management fees, trading fees, and/or taxes. For simplicity, we shall refer to it as taxes in the sequel.

Such modeling enables us to capture some realistic features: 1) taxes increase with and proportional to wealth, 2) there is a delay between when a tax is realized and when it is paid, 3) this delayed period for taxes varies for a given tax and is itself random. To see this, let us assume that the taxes paid over $[t, t + \mathrm{d}t]$ occur due to numerous tax bills that were realized τ units in the past. If we further assume that the time to pay these taxes τ follows an exponential distribution with a rate λ, then one could approximate the taxes paid in $[t, t + \mathrm{d}t]$ by its mean under the scenario of a high frequency of tax occurrences with small tax amounts at each occurrence. This gives rise to the flux of wealth of

$$-\mu_2 \int_{-\infty}^{t} \lambda e^{-\lambda(t-s)} X_s \, \mathrm{d}s \, \mathrm{d}t = -\mu_2 Y_t \, \mathrm{d}t$$

as it appears in Eq. (A.3). In essence, the tax at time t of $\mu_2 Y_t \, \mathrm{d}t$ naturally represents the delayed accrual of taxes due at time t based on past wealth as a result of a delay in billings.

Since Eq. (A.3) includes an outflow due to taxes, one observes that cash will no longer grow at the risk-free rate r. Therefore, it is natural to ask if there is a tax-adjusted risk-free rate that better represents the growth of cash in this model. This is addressed in Appendix A.2. Moreover, while X_t represents the wealth of an investor at time t, the investor is carrying around a hidden tax liability given through their past history of wealth. We will also argue in Appendix A.2 that the variable $Z_T = X_T + aY_T$ represents the tax-adjusted wealth of the investor at time t when a is given by

$$a = \frac{-(r + \lambda) + \sqrt{(r + \lambda)^2 - 4\lambda\mu_2}}{2\lambda}.$$

With this in mind, the problem we consider is that of an investor who seeks to maximize the quantity

$$J[\pi] = \mathbb{E}[U(Z_T)], \tag{A.4}$$

which is the expected utility of tax-adjusted terminal wealth.

Lastly, we mention that the stochastic control problem with such delay structure (A.3) and $\mu_2 < 0$ was considered in [23], including notably the form of the utility (A.4) depending on $Z_T = X_T + aY_T$, where

$$a = \frac{-(r+\lambda) + \sqrt{(r+\lambda)^2 - 4\lambda\mu_2}}{2\lambda}.$$

Therein, $\mu_2 < 0$ is interpreted as a momentum-like effect arising from the market structure. Our work considers $\mu_2 > 0$, leading to a quite different interpretation as discussed above.

A.2. Tax-adjusted wealth and the tax-adjusted risk-free rate

We have discussed the motivation and interpretation of Eqs. (A.3)-(A.4), which comes from an investor who is paying taxes at time t at a rate $\mu_2 > 0$ on her exponential average of past wealth. In this section, we give further interpretations to the quantities Z_t and $r + \lambda a$, as the tax-adjusted wealth and the tax-adjusted risk-free rate respectively. We define the tax-adjusted risk-free rate to be the long-term exponential growth rate of wealth for an all-bond account. Then tax-adjusted wealth is the process that grows precisely at the tax-adjust risk-free rate when considering the all-bond investor. In other words, X_t is no longer the best measurement of an investor's "true" wealth as it does not incorporate the hidden tax liabilities which arise due to the past history of X_t, yet contributes to taxes beyond time t, and the usual risk-free rate r no longer represents the rate of accrual of a pure bond account due to tax drag.

To see this, we consider the dynamics of an all-bond investor $\pi_t \equiv 0$

$$\mathrm{d}X_t = rX - \mu_2 Y_t \,\mathrm{d}t. \tag{A.5}$$

Recall that

$$Y_t = \int_{-\infty}^{t} \lambda e^{-\lambda(t-s)} X_s \,\mathrm{d}s, \quad \mathrm{d}Y_t = \lambda(X_t - Y_t)\,\mathrm{d}t,$$

one has for arbitrary $a \in \mathbb{R}$

$$\mathrm{d}(X_t + aY_t) = (r + \lambda a)X_t \,\mathrm{d}t + (-\mu_2 - \lambda a)Y_t \,\mathrm{d}t.$$

Therefore if a satisfies $-\mu_2 - \lambda a = a(r + \lambda a)$, then we will have

$$\mathrm{d}(X_t + aY_t) = (r + \lambda a)(X_t + aY_t)\,\mathrm{d}t,$$

which occurs when a takes values of

$$a_\pm = \frac{-(r+\lambda) \pm \sqrt{(r+\lambda)^2 - 4\lambda\mu_2}}{2\lambda}.$$

Note that $a_\pm$ are distinct real numbers since $(r+\lambda)^2 - 4\lambda\mu_2 > 0$ is assumed.

Denoting $Z_t^\pm = X_t + a_\pm Y_t$, we have two linearly independent representations for X, which allows us to eliminate Y resulting in an expression of X as a linear combination of Z^+ and Z^-. Using that $Z_t^\pm = Z_0^\pm e^{(r+\lambda a_\pm)t}$, we obtain the expression

$$X_t = c_+ e^{(r+\lambda a_+)t} + c_- e^{(r+\lambda a_-)t},$$

where

$$c_+ = \frac{a_-}{a_- - a_+}(X_0 + a_+ Y_0), \quad c_- = \frac{-a_+}{a_- - a_+}(X_0 + a_- Y_0).$$

Since $\lambda > 0$ and $a_+ > a_-$, we have that $r + \lambda a_+ > r + \lambda a_-$. Because of this, the wealth of the all-bond investor X_t has the property

$$\lim_{t\to\infty} e^{-kt} X_t = \begin{cases} \infty, & k < r + \lambda a_+, \\ c_+, & k = r + \lambda a_+, \\ 0, & k > r + \lambda a_+, \end{cases}$$

where $c_+ > 0$ holds assuming the initial quantity $X_0 + a_+ Y_0 > 0$ as well as $r + \lambda > 0$. This is guaranteed under either of the realistic assumptions that $r \geq 0$ or $|r| \ll \lambda$. Therefore, the rate $r + \lambda a_+$ is precisely the long-run exponential growth rate of an all-bond account, meaning cash grows at the rate $r + \lambda a_+$ in the long run. And for this choice of a, one has

$$X_t + aY_t = (X_0 + aY_0)e^{(r+\lambda a)t},$$

consequently, $X_t + aY_t$ can be interpreted as tax-adjusted wealth. Note that, in fact, $c(X_t + aY_t)$ for any $c \in \mathbb{R}$ could represent the tax-adjusted wealth by our requirement. However, $X_t + aY_t$ is the correct choice out of these by imposing a natural second requirement: the tax-adjusted wealth should be consistent with the wealth if the investor has no tax liability. Thus $c = 1$. The notion of $X_t + aY_t$ as the tax-adjusted wealth also makes sense intuitively as in our case of taxes ($\mu_2 > 0$) results in $a < 0$ under the realistic assumption that $r + \lambda > 0$. Hence $X_t + aY_t$ represents an adjustment to total assets X_t taking into account the tax liability given through Y_t.

A.3. Competition between portfolio managers

We temporarily ignore the delay arising from taxes and review the game extension of Merton's original problem which has been considered in [2, 18, 19]. We will briefly summarize them for convenience as they inspire the form of the new problems we consider in Sections 3.1-3.2. The motivation [2] comes from competing portfolio managers who are selected by their clients (or awarded bonuses) not only based on the fund's absolute performance, but also based on the fund's performance compared to similar funds. To address this possibility, the portfolio manager may have a utility function that takes into account both their absolute performance, as well as their performance relative to their peer group. [2, 18, 19] model this by considering the interaction between managers occurring through the reward function.

Let X_t^i be manager i's wealth process. Her utility can depend on both her terminal wealth X_T^i as well as her terminal wealth relative to that of her peers $X_T^i - \overline{X}_T$, where

$$\overline{X}_T = \frac{1}{N}\sum_{i=1}^{N} X_T^i$$

is the arithmetic mean. A weighted average

$$(1-\theta_i)X_T^i + \theta_i\big(X_T^i - \overline{X}_T\big) = X_T^i - \theta_i\overline{X}_T$$

can serve as the input for the utility function, where $\theta_i \in (0,1)$ measures the extent that manager i weighs relative versus absolute performance. Specifically, [19] consider CARA case, and the reward for player i is given by

$$J^i = E\big[U_i\big(X_T^i - \theta_i\overline{X}_T\big)\big], \quad U_i(z) = -\exp\left(-\frac{1}{\delta_i}z\right), \tag{A.6}$$

where $\delta_i > 0$ is the risk tolerance of manager i. They consider CRRA utilities

$$U_i(z) = \begin{cases} \dfrac{1}{1-1/\delta_i}z^{1-\frac{1}{\delta_i}}, & \delta_i > 0, \quad \delta_i \neq 1, \\ \log(z), & \delta_i = 1, \end{cases} \tag{A.7}$$

where $\delta_i > 0$ is the risk tolerance of manager i.

The CRRA case is also considered in [19]. In this case, for tractability, the competing managers compare relative performance to absolute performance modeled by the ratio $X_T^i/(\overline{X}_T)^{\theta_i}$, where

$$\overline{X} = \left(\prod_{i=1}^{N} X^i\right)^{\frac{1}{N}}$$

is the geometric average. Then we can say manager i seeks to maximize her expected utility given by

$$J^i = E\big[U_i\big(X_T^i(\overline{X}_T)^{-\theta_i}\big)\big]. \tag{A.8}$$

A.4. Competition between portfolio managers with delayed tax effects

We have now discussed the extension of the Merton problem to one with delay as well as one as a game. Now, we seek to combine both aspects together. The wealth dynamics of manager i, denoted by X_t^i, can easily be generalized via Eq. (A.3), and given by

$$\mathrm{d}X_t^i = \big((\mu_1 - r)\pi_t^i X_t^i + rX_t^i - \mu_2 Y_t^i\big)\,\mathrm{d}t + \sigma\pi_t^i X_t^i\,\mathrm{d}W_t, \quad t \in (0,T], \tag{A.9}$$

where

$$Y_t^i = \int_{-\infty}^{t} \lambda e^{-\lambda(t-s)} X_s^i\,\mathrm{d}s.$$

As for the reward function, we again consider both the CARA case (A.6) and the CRRA case (A.7), but replace the terminal wealth with the discounted, tax-adjusted terminal wealth. To recall the discussion in Appendices A.1 and A.2, we have identified the tax-adjusted risk-free rate to be $r + \lambda a$ and the tax-adjusted wealth for player i at time t to be

$$Z_t^i = X_t^i + aY_t^i,$$

where

$$a = \frac{-(r+\lambda) + \sqrt{(r+\lambda)^2 - 4\lambda\mu_2}}{2\lambda}.$$

The discounted, tax-adjusted wealth at time t is then given by

$$Z_{disc,t}^i = e^{-(r+\lambda a)t} Z_t^i,$$

which discounts the tax-adjusted wealth back to time 0 under the tax-adjusted risk-free rate. Therefore, in the CARA utility case, it is natural to consider the reward for player i by

$$J^i[\boldsymbol{\pi}] = \mathbb{E}\big[U_i\big(Z_{disc,T}^i - \theta_i \overline{Z}_{disc,T}\big)\big],$$

where

$$\overline{Z}_{disc,T} = \frac{1}{N}\sum_{i=1}^N Z_{disc,T}^i.$$

In contrast, for the CRRA utility case with U_i given by Eq. (A.7), one has

$$J^i[\boldsymbol{\pi}] = \mathbb{E}\Big[U_i\Big(Z_{disc,T}^i \overline{Z}_{disc,T}^{-\theta_i}\Big)\Big],$$

where in this case

$$\overline{Z}_{disc,T} = \left(\prod_{i=1}^N Z_{disc,T}^i\right)^{\frac{1}{N}}.$$

In both cases, we see that the portfolio manager is simply comparing her terminal tax-adjusted wealth to that of her peers in the manner discussed in Appendix A.3 in determining her utility.

A.5. Consumption and portfolio allocation game with delayed tax effects

We further extend the modeling (A.9) by considering consumption. In addition to the investment strategy π_t^i, the consumption rate c_t^i will also be chosen by investor i. The consumption rate c_t^i measures investor i's annual run rate of consumption at time t as a fraction of her wealth. Precisely, over $[t, t + \mathrm{d}t]$ investor i consumes the dollar amount given by $c_t^i X_t^i \,\mathrm{d}t$. By including this outflow due to consumption, we have the wealth dynamics for player i given by

$$\mathrm{d}X_t^i = \big((\mu_1 - r)\pi_t^i X_t^i + rX_t^i - \mu_2 Y_t^i - c_t^i X_t^i\big)\,\mathrm{d}t + \sigma\pi_t^i X_t^i \,\mathrm{d}W_t, \quad t \in (0, T],$$

where as before,

$$Y_t^i = \int_{-\infty}^{t} \lambda e^{-\lambda(t-s)} X_s^i \, \mathrm{d}s.$$

The quantity $Z_T^i = X_T^i + aY_T^i$ with

$$a = \frac{-(r+\lambda) + \sqrt{(r+\lambda)^2 - 4\lambda\mu_2}}{2\lambda}$$

again represents the tax-adjusted wealth of player i.

The expected utility will be a sum of the utility from discounted consumption and the utility from discounted terminal wealth. The exact form is motivated from [18], which has the utility of consumption given by $U^i((c_t^i X_t^i)(\overline{cX}_t)^{-\theta_i})$ with

$$\overline{cX}_t = \left(\prod_{i=1}^{N} (c_t^i X_t^i) \right)^{\frac{1}{N}}.$$

With the additional delayed tax effects in our modeling, we consider an analogous utility of discounted consumption and an analogous utility of discounted, tax-adjusted terminal wealth, where the discounting is taken with respect to the tax-adjusted risk-free rate. The resulting reward function for each player $i \in \{1, \ldots, N\}$ is given by

$$J^i[\boldsymbol{\pi}, \boldsymbol{c}] = E\left[\int_0^T U^i \left(C_{disc,t}^i \overline{C_{disc,t}}^{-\theta_i} \right) \mathrm{d}t + \epsilon_i U_i\left(Z_{disc,T}^i \overline{Z_{disc,T}}^{-\theta_i} \right) \right],$$

where

$$C_{disc,t}^i = e^{-(r+\lambda a)t} c_t^i X_t^i, \quad \overline{C_{disc,t}} = \left(\prod_{i=1}^{N} C_{disc,t}^i \right)^{\frac{1}{N}},$$

$$Z_{disc,t}^i = e^{-(r+\lambda a)t} Z_t^i, \qquad \overline{Z_{disc,t}} = \left(\prod_{i=1}^{N} Z_{disc,t}^i \right)^{\frac{1}{N}},$$

and

$$U_i(z) = \begin{cases} \dfrac{1}{1-1/\delta_i} z^{1-\frac{1}{\delta_i}}, & \delta_i \neq 1, \\ \log(z), & \delta_i = 1. \end{cases}$$

Here $Z_{disc,t}^i$ is the tax-adjusted wealth for player i discounted according to the tax-adjusted risk-free rate. $C_{disc,t}^i$ is the discounted total consumption over $[t, t + \mathrm{d}t]$ for player i.

In essence, investor i determines her total utility by summing up her utilities of consumption over intervals of length $\mathrm{d}t$. For example, over the time interval $[t, t + \mathrm{d}t]$, the utility of player i is taken by comparing her own discounted consumption compared to that of her peers and weighted by the amount $\mathrm{d}t$. In the final stage, the utility is taken by comparing her discounted terminal wealth to that of her peers and is weighted ϵ_i. Thus ϵ_i represents player i's preference for wealth as compared to consumption.

Appendix B. Proofs of propositions

B.1. Proof of Proposition 3.1

For convenience, we restate the dynamics (3.1), which is

$$\begin{aligned} \mathrm{d}X_t^i &= \big((\mu_1 - r)\pi_t^i X_t^i + rX_t^i - \mu_2 Y_t^i\big)\,\mathrm{d}t + \sigma\pi_t^i X_t^i\,\mathrm{d}W_t, && t \in (0,T], \\ X_t^i &= \zeta^i(t), && t \in (-\infty, 0], \end{aligned}$$

where the delay variable, Y_t^i, is defined as

$$Y_t^i = \int_{-\infty}^{t} \lambda e^{-\lambda(t-s)} X_s^i\,\mathrm{d}s.$$

Motivated by the analysis in [23], we note that differentiating Y_t^i gives the following differential relation:

$$\mathrm{d}Y_t^i = \lambda\big(X_t^i - Y_t^i\big)\,\mathrm{d}t.$$

Multiplying $\mathrm{d}Y_t^i$ by a and adding with $\mathrm{d}X_t^i$, we get

$$\mathrm{d}(X_t^i + aY_t^i) = \big((\mu_1 - r)\pi_t^i X_t^i + (r + \lambda a)X_t^i + (-\mu_2 - \lambda a)Y_t^i\big)\,\mathrm{d}t + \sigma\pi_t^i X_t^i\,\mathrm{d}W_t.$$

Now, using the parameter value

$$a = \frac{-(r+\lambda) + \sqrt{(r+\lambda)^2 - 4\lambda\mu_2}}{2\lambda},$$

we have that $a(r + \lambda a) = -\mu_2 - \lambda a$ as one can see a is a root of this quadratic equation. Thus denoting

$$Z_t^i = X_t^i + aY_t^i,$$

we get

$$\mathrm{d}Z_t^i = \big((\mu_1 - r)\pi_t^i X_t^i + (r + \lambda a)Z_t^i\big)\,\mathrm{d}t + \sigma\pi_t^i X_t^i\,\mathrm{d}W_t.$$

Multiplying by the integrating factor $e^{-(r+\lambda a)t}$ and denoting

$$Z_{disc,t}^i = e^{-(r+\lambda a)t} Z_t^i,$$

we get

$$\mathrm{d}Z_{disc,t}^i = (\mu_1 - r)\pi_t^i e^{-(r+\lambda a)t} X_t^i\,\mathrm{d}t + \sigma\pi_t^i e^{-(r+\lambda a)t} X_t^i\,\mathrm{d}W_t. \tag{B.1}$$

Now, for each i, define the transformed control $\tilde{\pi}^i$ by

$$\tilde{\pi}_t^i = \pi_t^i X_t^i e^{-(r+\lambda a)t}. \tag{B.2}$$

Substituting the transformed controls, we have the following controlled SDE for each $i \in \{1, \ldots, N\}$:

$$\mathrm{d}Z_{disc,t}^i = (\mu_1 - r)\tilde{\pi}_t^i\,\mathrm{d}t + \sigma\tilde{\pi}_t^i\,\mathrm{d}W_t, \quad t \in (0,T]. \tag{B.3}$$

Restating the reward function defined in Proposition 3.1, one has that the reward for each player $i \in \{1, \dots, N\}$ is given by

$$J^i[\tilde{\boldsymbol{\pi}}] = \mathbb{E}\big[U_i\big(Z^i_{disc,T} - \theta_i \overline{Z}_{disc,T}\big)\big], \tag{B.4}$$

where

$$\overline{Z}_{disc,T} = \frac{1}{N}\sum_{i=1}^{N} Z^i_{disc,T}, \quad U_i(z) = -\exp\left(-\frac{1}{\delta_i}z\right). \tag{B.5}$$

Therefore, the characterization of controls for the Nash equilibrium can be considered through Eqs. (B.3)-(B.5). This is precisely the same problem described in [19, Section 2, Corollary 4], which gives a closed-loop Nash equilibrium given by the controls

$$\tilde{\pi}^{i,*}_t = \frac{\mu_1 - r}{\sigma^2}\left(\delta_i + \frac{\theta_i \bar{\delta}}{1-\bar{\theta}}\right)$$

for $i \in \{1, \dots, N\}$, where

$$\bar{\delta} = \frac{1}{N}\sum_{i=1}^{N}\delta_i, \quad \bar{\theta} = \frac{1}{N}\sum_{i=1}^{N}\theta_i.$$

Substituting $\tilde{\pi}^{i,*}_t$ into Eq. (B.2), we get that there is a closed-loop Nash equilibrium control given by the choices of each player i by

$$\pi^{i,*}_t X^{i,*}_t = \frac{\mu_1 - r}{\sigma^2}\left(\delta_i + \frac{\theta_i \bar{\delta}}{1-\bar{\theta}}\right)\frac{1}{e^{-(r+\lambda a)t}}.$$

That is, at time t, the equilibrium strategy is to invest a deterministic dollar amount into the risky asset, independent of the current wealth level. This proves Proposition 3.1.

B.2. Proof of Proposition 3.2

Proof. Since the dynamical system is exactly the same as in Appendix B.1, we know that

$$Z^i_t = X^i_t + aY^i_t$$

satisfies

$$\mathrm{d}Z^i_t = \big((\mu_1 - r)\pi^i_t X^i_t + (r + \lambda a)Z^i_t\big)\,\mathrm{d}t + \sigma \pi^i_t X^i_t\,\mathrm{d}W_t. \tag{B.6}$$

We are assuming that the initial path $X^i_t = \zeta^i(t)$ for $t \le 0$ is chosen so that

$$Z^i_0 = X^i_0 + aY^i_0 > 0.$$

Next, since π^i is admissible, it has the property $|\pi^i_t X^i_t| \le K|Z^i_t|$ for some $K > 0$. In particular, this means that $\pi^i_t X^i_t = 0$ whenever $Z^i_t = 0$. We will soon show that $Z^i_t > 0$. Now, for each i, we define the transformed control $\tilde{\pi}^i$ by

$$\tilde{\pi}^i_t = \begin{cases} \pi^i_t \dfrac{X^i_t}{Z^i_t}, & Z^i_t \neq 0, \\ 0, & Z^i_t = 0. \end{cases} \tag{B.7}$$

Since $|\pi_t^i X_t^i| \leq K|Z_t^i|$, we have $|\tilde{\pi}_t^i| \leq K$ and claim that the dynamics (B.6) can be written in terms of $\tilde{\pi}_t^i$ by

$$\mathrm{d}Z_t^i = \left((\mu_1 - r)\tilde{\pi}_t^i Z_t^i + (r + \lambda a)Z_t^i\right)\mathrm{d}t + \sigma\tilde{\pi}_t^i Z_t^i \,\mathrm{d}W_t. \tag{B.8}$$

Since $|\tilde{\pi}_t^i| \leq K$, we have that

$$\mathbb{E}\left[\int_0^t \left(\tilde{\pi}_s^i\right)^2 \mathrm{d}s\right] < \infty$$

for each $t \in (0, T]$ and therefore, Z_t^i has the unique solution

$$Z_t^i = Z_0^i \exp\left[(r + \lambda a)t + \int_0^t \left[(\mu_1 - r)\tilde{\pi}_s^i - \frac{1}{2}\sigma^2\left(\tilde{\pi}_s^i\right)^2\right]\mathrm{d}s + \int_0^t \tilde{\pi}_s^i \,\mathrm{d}W_s\right]$$

for all $t \in [0, T]$. This indicates $Z_t^i > 0$ as we have assumed the initial path $X^i_{(-\infty,0]}$ is such that

$$Z_0^i = X_0^i + aY_0^i > 0.$$

Also, one can show $X_t^i > 0$ for all $t \leq T$. First $X_t^i = \zeta^i(t) > 0$ for $t \leq 0$. Now, by contradiction take $t' > 0$ to be the first hitting time $X_{t'}^i = 0$. Since $X_t^i > 0$ for $t < t'$, we see that

$$Y_{t'}^i = \int_{-\infty}^{t'} \lambda e^{-\lambda(t-s)} X_s^i \,\mathrm{d}s > 0.$$

Therefore

$$0 < Z_{t'}^i = X_{t'}^i + aY_{t'}^i = aY_{t'}^i,$$

which is a contradiction as $a < 0$ and $Y_{t'}^i > 0$. Since $Z_t^i > 0$, the transformed control from Eq. (B.7) can be written simply as

$$\tilde{\pi}_t^i = \pi_t^i \frac{X_t^i}{Z_t^i}, \tag{B.9}$$

and since $X_t^i > 0$ as well, this transformation is invertible which we will use later.

Now, multiplying Eq. (B.8) by the integrating factor $e^{-(r+\lambda a)t}$, we can write the equation for

$$Z_{disc,t}^i = e^{-(r+\lambda a)t} Z_t^i$$

as

$$\mathrm{d}Z_{disc,t}^i = (\mu_1 - r)\tilde{\pi}_t^i Z_{disc,t}^i \,\mathrm{d}t + \sigma\tilde{\pi}_t^i Z_{disc,t}^i \,\mathrm{d}W_t, \quad t \in (0, T]. \tag{B.10}$$

We restate the reward function in Proposition 3.1 for convenience. We have that the reward for player i is given by

$$J^i = \mathbb{E}\left[U_i\left(Z_{disc,T}^i \overline{Z}_{disc,T}^{-\theta_i}\right)\right], \tag{B.11}$$

where $0 < \theta_i < 1$ for each i, and where

$$\overline{Z}_{disc,T} = \left(\prod_{i=1}^N Z_{disc,T}^i\right)^{\frac{1}{N}}, \quad U_i(z) = \begin{cases} \dfrac{1}{1 - 1/\delta_i} z^{1-\frac{1}{\delta_i}}, & \delta_i \neq 1 \\ \log(z), & \delta_i = 1 \end{cases} \tag{B.12}$$

with $\delta_i > 0$. The Nash equilibrium problem defined by Eqs. (B.10)-(B.12) for generic, progressively measurable controls $(\tilde{\pi}^i)_{i=1}^N$ such that

$$\mathbb{E}\left[\int_0^t \left(\tilde{\pi}_s^i\right)^2 \mathrm{d}s\right] < \infty$$

falls into the formulation of [19, Section 3, Corollary 15]. Using the results therein, one has a closed-loop Nash equilibrium given by the controls

$$\tilde{\pi}_t^{i,*} = \frac{\mu_1 - r}{\sigma^2}\left(\delta_i - \frac{\theta_i(\delta_i - 1)\bar{\delta}}{1 + \overline{\theta(\delta - 1)}}\right)$$

for $i \in \{1, \ldots, N\}$. Solving for $\pi_t^{i,*}$, from Eq. (B.9) we get

$$\pi_t^{i,*} = \frac{\mu_1 - r}{\sigma^2}\left(\delta_i - \frac{\theta_i(\delta_i - 1)\bar{\delta}}{1 + \overline{\theta(\delta - 1)}}\right)\frac{X_t^i + aY_t^i}{X_t^i},$$

which holds as we have shown that $X_t^i > 0$. Lastly, we see that the Nash equilibrium controls do satisfy the admissibility condition as

$$\left|\pi_t^{i,*} X_t^i\right| = \left|\frac{\mu_1 - r}{\sigma^2}\left(\delta_i - \frac{\theta_i(\delta_i - 1)\bar{\delta}}{1 + \overline{\theta(\delta - 1)}}\right)\right| \left|Z_t^i\right|.$$

This completes the proof of Proposition 3.2. □

B.3. Proof of Proposition 3.3

We now form the solution for the problem defined by Eq. (3.8)-(3.11), proceeding similarly to the proof in Appendix B.2. Restating the dynamics given in Proposition 3.3, we have for each $i \in \{1, \ldots, N\}$,

$$\begin{aligned} &\mathrm{d}X_t^i = \left((\mu_1 - r)\pi_t^i X_t^i + rX_t^i - \mu_2 Y_t^i - c_t^i X_t^i\right)\mathrm{d}t + \sigma\pi_t^i X_t^i \,\mathrm{d}W_t, && t \in (0, T], \\ &X_t^i = \zeta^i(t), && t \in (-\infty, 0], \end{aligned}$$

where the delay variable, Y_t^i, is given by

$$Y_t^i = \int_{-\infty}^t \lambda e^{-\lambda(t-s)} X_s^i \,\mathrm{d}s.$$

As usual, define

$$Z_t^i = X_t^i + aY_t^i, \quad Z_{disc,t}^i = e^{-(r+\lambda a)t} Z_t^i,$$

where

$$a = \frac{-(r + \lambda) + \sqrt{(r + \lambda)^2 - 4\lambda\mu_2}}{2\lambda}.$$

Following the similar computation in the proof in Appendix B.1, one can show

$$\mathrm{d}Z_t^i = \left((r+\lambda a)Z_t^i + (\mu_1 - r)\pi_t^i X_t^i - c_t^i X_t^i\right)\mathrm{d}t + \sigma\pi_t^i X_t^i\,\mathrm{d}W_t. \tag{B.13}$$

Now, for each i, define the transformed controls $(\tilde{\pi}^i, \tilde{c}^i)$ by

$$(\tilde{\pi}_t^i, \tilde{c}_t^i) = \begin{cases} \left(\pi_t^i \dfrac{X_t^i}{Z_t^i}, c_t^i \dfrac{X_t^i}{Z_t^i}\right), & Z_t^i \neq 0, \\ (0,0), & Z_t^i = 0. \end{cases}$$

Then Eq. (B.13) can be written as

$$\mathrm{d}Z_t^i = \left((r+\lambda a)Z_t^i + (\mu_1 - r)\tilde{\pi}_t^i Z_t^i - \tilde{c}_t^i Z_t^i\right)\mathrm{d}t + \sigma\tilde{\pi}_t^i Z_t^i\,\mathrm{d}W_t. \tag{B.14}$$

Since $|\tilde{\pi}_t^i|, |\tilde{c}_t^i| \leq K$, we have that

$$\mathbb{E}\left[\int_0^t \left(\tilde{\pi}_s^i\right)^2 + \left(\tilde{c}_s^i\right)^2 \mathrm{d}s\right] < \infty$$

for each $t \in (0, T]$, and Z_t^i has the unique solution

$$Z_t^i = Z_0^i \exp\left[(r+\lambda a)t + \int_0^t \left[(\mu_1 - r)\tilde{\pi}_s^i - \frac{1}{2}\sigma^2\left(\tilde{\pi}_s^i\right)^2 - \tilde{c}_s^i\right]\mathrm{d}s + \int_0^t \tilde{\pi}_s^i\,\mathrm{d}W_s\right].$$

Therefore, $Z_t^i > 0$ and $X_t^i > 0$, following the same argument as in Appendix B.2, and the transformed controls become simply

$$\left(\tilde{\pi}_t^i, \tilde{c}_t^i\right) = \left(\pi_t^i \frac{X_t^i}{Z_t^i}, c_t^i \frac{X_t^i}{Z_t^i}\right), \tag{B.15}$$

which can be inverted for a given choice of controls $(\tilde{\pi}_t^i, \tilde{c}_t^i)$.

Next, multiplying Eq. (B.14) by the integrating factor $e^{-(r+\lambda a)t}$, we get

$$\mathrm{d}Z_{disc,t}^i = \left((\mu_1 - r)\tilde{\pi}_t^i Z_{disc,t}^i - \tilde{c}_t^i Z_{disc,t}^i\right)\mathrm{d}t + \sigma\tilde{\pi}_t^i Z_{disc,t}^i\,\mathrm{d}W_t, \quad t \in (0, T]. \tag{B.16}$$

For convenience, we restate the reward defined in Proposition 3.3, which is

$$J^i = E\left[\int_0^T U^i\left(C_{disc,t}^i \overline{C_{disc,t}}^{-\theta_i}\right)\mathrm{d}t + \epsilon_i U_i\left(Z_{disc,T}^i \overline{Z_{disc,T}}^{-\theta_i}\right)\right],$$

where

$$C_{disc,t}^i = e^{-(r+\lambda a)t} c_t^i X_t^i, \quad \overline{C_{disc,t}} = \left(\prod_{i=1}^N C_{disc,t}^i\right)^{\frac{1}{N}},$$

$$Z_{disc,t}^i = e^{-(r+\lambda a)t} Z_t^i, \qquad \overline{Z_{disc,t}} = \left(\prod_{i=1}^N Z_{disc,t}^i\right)^{\frac{1}{N}},$$

and where $0 < \theta_i < 1$, $\epsilon_i > 0$, and

$$U_i(z) = \begin{cases} \dfrac{1}{1-1/\delta_i} z^{1-\frac{1}{\delta_i}}, & \delta_i \neq 1, \\ \log(z), & \delta_i = 1 \end{cases} \tag{B.17}$$

with $\delta_i > 0$. In terms of the transformed controls defined by Eq. (B.15), we can write the reward function for player i as

$$J^i = E\left[\int_0^T U^i\left(\tilde{c}_t^i Z_{disc,t}^i \overline{(\tilde{c}_t Z_{disc,t})}^{-\theta_i}\right) \mathrm{d}t + \epsilon_i U_i\left(Z_{disc,T}^i \overline{Z_{disc,T}}^{-\theta_i}\right)\right], \tag{B.18}$$

where

$$\overline{\tilde{c}_t Z_{disc,t}} = \left(\prod_{i=1}^N \tilde{c}_t^i Z_{disc,t}^i\right)^{\frac{1}{N}}.$$

The Nash equilibrium problem defined by Eqs. (B.16)-(B.18) is given in [18, Corollary 2.3] for the generic, progressively measurable controls $(\tilde{\pi}^i, \tilde{c}^i)_{i=1}^N$ that satisfy the admissibility conditions

$$\mathbb{E}\left[\int_0^t \left(\tilde{\pi}_s^i\right)^2 + \left(\tilde{c}_s^i\right)^2 \mathrm{d}s\right] < \infty,$$

and $\tilde{c}_t^i \geq 0$ for each $t \in [0, T]$. Specifically, [18, Corollary 2.3] gives that there exists a closed-loop Nash equilibrium given by the controls

$$\tilde{\pi}_t^{i,*} = \frac{\mu_1 - r}{\sigma^2}\left(\delta_i - \frac{\theta_i(\delta_i - 1)\bar{\delta}}{1 + \overline{\theta(\delta - 1)}}\right),$$

$$\tilde{c}_t^{i,*} = \begin{cases} \left(\beta_i^{-1} + \left(\gamma_i^{-1} - \beta_i^{-1}\right)e^{-\beta_i(T-t)}\right)^{-1}, & \delta_i \neq 1, \\ \left(T - t + \gamma_i^{-1}\right)^{-1}, & \delta_i = 1 \end{cases}$$

for $i \in \{1, \ldots, N\}$, where the notations $\bar{\delta}, \overline{\theta(\delta - 1)}$ correspond to the arithmetic average, and the parameters β_i and γ_i are given by

$$\beta_i = \frac{1}{2}(1 - \delta_i)\left(\frac{\mu_1 - r}{\sigma}\right)^2 \left(1 - \frac{\theta_i \bar{\delta}}{1 + \overline{\theta(\delta - 1)}}\right)\left(\delta_i - \frac{\theta_i \bar{\delta}}{1 + \overline{\theta(\delta - 1)}}(\delta_i - 1)\right),$$

$$\gamma_i = \epsilon_i^{-\delta_i}\left(\left(\prod_{k=1}^N \epsilon_k^{\delta_k}\right)^{\frac{1}{N}}\right)^{\frac{\theta_i(\delta_i - 1)}{1 + \overline{\theta(\delta - 1)}}}.$$

Substituting back for $(\pi^{i,*}, c^{i,*})$ through the transformations defined by Eq. (B.15), we see that $(\pi^{i,*}, c^{i,*})$ is indeed in $\mathbb{A}^i$ and the result from Proposition 3.3 holds.

7 Acknowledgements

R. Hu was partially supported by the NSF (Grant No. DMS-1953035), and by the Early Career Faculty Acceleration funding and the Regents' Junior Faculty Fellowship at the University of California, Santa Barbara. H. D. Ceniceros acknowledges partial support from the NSF (Grant No. DMS-1818821).

References

[1] M. Abadi et al., Tensorflow: A system for large-scale machine learning, in: *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, USENIX Association, Vol. 16, 265–283, 2016.
[2] S. Basak and D. Makarov, *Competition among Portfolio Managers and Asset Specialization*, Working Papers w0194, New Economic School, 2013.
[3] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, Automatic differentiation in machine learning: A survey, *arXiv:1502.05767v2*, 2015.
[4] G. Brown, Iterative solution of games by fictitious play, *Act. Anal. Prod. Allocation*, 13(1):374–376, 1951.
[5] R. Carmona, J. Fouque, S. Mousavi, and L. Sun, Systemic risk and stochastic games with delay, *J. Optim. Theory Appl.*, 179:366–399, 2018.
[6] R. Carmona, J. Fouque, and L. Sun, Mean field games and systemic risk, *Commun. Math. Sci.*, 13(4):911–933, 2015.
[7] 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.
[8] G. Fabbri, F. Gozzi, and A. Swiech, *Stochastic Optimal Control in Infinite Dimension*, in: *Probability Theory and Stochastic Modelling*, Springer, 2017.
[9] J. Fouque and Z. Zhang, Deep learning methods for mean field control problems with delay, *arXiv:1905.00358*, 2019.
[10] F. Gozzi and C. Marinelli, Stochastic optimal control of delay equations arising in advertising models, *arXiv:math/0412403*, 2004.
[11] J. Han and W. E, Deep learning approximation for stochastic control problems, *arXiv:1611.07422*, 2016.
[12] 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*, PMLR, 107:221–245, 2020.
[13] J. Han, A. Jentzen, and W. E, Solving high-dimensional partial differential equations using deep learning, *Proc. Nat. Acad. Sci. India Sect. A*, 115(34):8505–8510, 2018.
[14] R. Han and J. Hu, Recurrent neural networks for stochastic control problems with delay, *Math. Control Signals Systems*, 33:775–795, 2021.
[15] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.*, 9:1735–1780, 1997.
[16] R. Hu, Deep fictitious play for stochastic differential games, *Commun. Math. Sci.*, 19(2):325–353, 2021.
[17] D. Kingma and J. Ba, Adam: A method for stochastic optimization, *arXiv:1412.6980*, 2015.
[18] D. Lacker and A. Soret, Many-player games of optimal consumption and investment under relative performance criteria, *Math. Financ. Econ.*, 14:263–281, 2019.
[19] D. Lacker and T. Zariphopoulou, Mean field and n-agent games for optimal investment under relative performance criteria, *Math. Finance*, 29(4):1003–1038, 2019.
[20] X. Mao, Approximate solutions for a class of delay stochastic differential equations, *Stochastics and Stochastic Reports*, 35(2):111–123, 1991.
[21] R. Merton, Lifetime portfolio selection under uncertainty: The continuous-time case, *Rev. Econ. Stat.*, 51(3):247–57, 1969.
[22] S. A. Mohammed, Stochastic differential systems with memory: Theory, examples and applications, in: *Stochastic Analysis and Related Topics VI*, Vol. 42, 1–77, 1998.

[23] T. Pang and A. Hussain, A stochastic portfolio optimization model with complete memory, *Stoch. Anal. Appl.*, 35(4):742–766, 2017.

[24] A. Paszke et al., Pytorch: An imperative style, high-performance deep learning library, *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Curran Associates Inc., 8026–8037, 2019.

[25] D. Rumelhart, G. Hinton, and R. Williams, Learning representations by back-propagating errors, *Nature*, 323(6088):533–536, 1986.

[26] J. Sirignano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, *J. Comput. Phys.*, 375:1339–1364, 2018.