University of California

Santa Barbara

Machine Learning Methods for Stochastic Differential Games and those with Delay: Applications and Modeling in Epidemiology and Finance

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy

in

Mathematics

by

Robert Andrew Balkin

Committee in charge:

Professor Hector D. Ceniceros, Co-Chair

Professor Ruimeng Hu, Co-Chair

Professor Carlos Garcia-Cervera

September 2023

The dissertation of Robert Andrew Balkin is approved.

Professor Carlos Garcia-Cervera

Professor Hector D. Ceniceros, Co-Chair

Professor Ruimeng Hu, Co-Chair

September 2023

Machine Learning Methods for Stochastic Differential Games and those with Delay: Applications and Modeling in Epidemiology and Finance

by

Robert Andrew Balkin

Acknowledgements

I am exceedingly grateful to my advisors, Hector Ceniceros and Ruimeng Hu, for their guidance and support during my time at UCSB. The process of mathematical research was especially challenging during the COVID-19 pandemic, and Hector and Ruimeng were steadfast in their support during the hardest of times. I consider myself extremely fortunate to have worked with the two of them. I am thankful to my committee, including Carlos Garcia-Cervera, along with my advisors Hector and Ruimeng. Carlos was instrumental in offering feedback for the dissertation. I wish to extend gratitude towards fellow co-authors and collaborators in my research papers. Yao Xuan, Jiequn Han as well as my aforementioned advisors all played a massive role in the research for stochastic differential games applied to COVID-19. I am also grateful for my friends in graduate school who helped me throughout this experience. I am thankful to friends both in and outside of graduate school who have made these past years special.

I am filled with gratitude for my family who supported me throughout my life and education. Mom and dad, you have believed in me throughout everything and given me the opportunities to succeed. I cannot possibly express my gratitude to you in words. Matt and Lauren, you have shaped me immensely to become the person I am today, and I am extremely fortunate and beyond grateful to have you as my brother and sister. Taylor, I am incredibly lucky to know you, and you have shown unwavering support to me throughout my time in graduate school. I could not have done it without you.

Robert Andrew Balkin

Education

University of California, Santa Barbara
Ph.D. in Mathematics Sep. 2017 - Sep. 2023
Master's in Mathematics GPA: 4.00 Sep. 2017 - Jun. 2019
Colorado School of Mines
B.S. in Computational and Applied Mathematics Aug. 2013 - Dec. 2016

Publications

R. Balkin, H.D. Ceniceros, and R. Hu, "Stochastic Delay Differential Games: Financial Modeling and Machine Learning Algorithms". 2023. (Under review). https://arxiv.org/pdf/2307.06450.pdf.

Y. Xuan, R. Balkin, J. Han, R. Hu, H.D. Ceniceros, "Optimal policies for a pandemic: a stochastic game approach and a deep learning algorithm. " *Mathematical and Scientific Machine Learning*, pages 987 - 1012. PMLR, 2022. https://msml21.github.io/papers/id42.pdf.

Y. Xuan, R. Balkin, J. Han, R. Hu and H.D. Ceniceros, "Pandemic Control, Game Theory and Machine Learning", *Notices of the AMS*, 69(11), pages 1878 - 1887. Dec. 2022. https://www.ams.org/journals/notices/202211/rnoti-p1878.pdf.

Research Experience

Graduate Researcher, UC Santa Barbara Sep. 2017 - Sep. 2023
Advised by Dr. Hector D. Ceniceros and Dr. Ruimeng Hu
Research in stochastic optimal control and games. Worked on new machine learning methodologies for solving stochastic differential games with delay. Developed new financial models for competitive portfolio allocation amongst financial market participants with taxes taken into consideration. Worked on machine learning methodologies for stochastic differential games to model COVID-19 dynamics and policies.

Research Assistant, Colorado School of Mines Aug. 2016 - Dec. 2016
Under Dr. Paul Constantine
Numerical analysis of a new approximation methodology to estimate moments of solutions to a class of elliptic PDEs with random field diffusivity.

Undergraduate Researcher, Florida Institute of Technology Jun. 2016 - Aug. 2016
Under Dr. Ugur Abdulla
Research Experience for Undergraduates (REU) in mathematics studying evolution of free boundaries for nonlinear convection-diffusion-reaction equations.

Work Experience

Teaching Assistant, UC Santa Barbara Sep. 2017 - Sep. 2023
Teaching in various math classes including, PDEs, ODEs, and numerical analysis.

Investment Research Intern, Kamehameha Schools Apr. 2017 - Aug. 2017
Developed models in VBA for portfolio optimization and asset allocation simulation.

Abstract

Machine Learning Methods for Stochastic Differential Games and those with Delay: Applications and Modeling in Epidemiology and Finance

by

Robert Andrew Balkin

Stochastic differential games and those with delay play a crucial role in modeling complex, real-world phenomena. The ability to find Nash equilibria in these games enhances the predictive capabilities of scientists and professionals across various fields and informs optimal decision-making processes. These problems can be computationally demanding to solve, especially in the case of delayed dynamics and interaction among a large number of agents.

This dissertation begins with an overview of stochastic differential games and existing machine learning methodologies designed to find their Nash equilibria. We then extend these existing methodologies to the challenging case of stochastic *delay* differential games with a new algorithm for finding their closed-loop Nash equilibria. To evaluate the effectiveness of our proposed algorithm, we test it on problems with known solutions. In particular, we introduce new financial models based on competing portfolio managers taking into consideration delayed tax-effects. We derive analytical Nash equilibrium solutions for these newly introduced stochastic delay differential games, serving as additional benchmarks to assess the performance of our proposed machine learning approach.

Finally, building on the existing machine learning methodologies for stochastic differential games, we introduce a new modified algorithm that we use to solve the Nash equilibrium problem for a game-theoretic, stochastic SEIR (Susceptible-Exposed-Infectious-Recovered) model applied to the COVID-19 pandemic. Solving this proposed model

demonstrates the effects of differing policies on the spread of disease over different regions and how these policies affect each other, illustrating the practical effectiveness of the proposed numerical approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The work presented in this chapter, which was prepared for this dissertation, includes material previously published [43] and material submitted for publication [3].

This dissertation explores machine learning methodologies for solving the Nash equilibrium problem for stochastic differential games (SDGs) and stochastic delay differential games (SDDGs). We begin by introducing the idea behind both SDGs and SDDGs, discussing the importance and difficulty of these problems, and introduce the SDG mathematically. Chapter 2 reviews existing methodologies for finding the Nash equilibrium of an SDG with deep learning techniques known as deep fictitious play (DFP). The DFP methodology will be the basis of our work which involves extending this methodology to new cases and applying it to solve problems.

As our first main contribution, we introduce a new algorithm for finding closed-loop Nash equilibria of SDDGs by extending existing DFP methods using memory-dependent neural network architectures such as the recurrent neural network (RNN) and in particular, the long short-term memory (LSTM) network. In Chapter 3, we introduce this novel DFP extension for the case of SDDGs, and we demonstrate the algorithm's successful performance by testing it on SDDGs with known closed-form Nash equilibrium solutions used as benchmarks.

Chapter 4 focuses on introducing newly considered SDDGs from the field of finance and deriving their analytical closed-loop Nash equilibrium solutions. These problems first appear in Chapter 3 and represent a portion of the benchmark problems used to validate the numerical algorithm for SDDGs. The analytical solutions derived for these newly considered problems represent new contributions in the field of financial modeling.

Using DFP for SDGs, one can solve problems corresponding to relevant real-world phenomena that would otherwise be intractable. In Chapter 5, we introduce a game-theoretic epidemiological model to analyze and predict pandemic decision-making with the focus on building a model for COVID-19 dynamics. We extend the DFP methodology for SDGs presented in Chapter 2 with a modified DFP algorithm used to find the Nash equilibrium solution of this COVID-19 model.

1.1 Motivation

Differential games, initiated by Isaacs [27] as an offspring of game theory and optimal control, provide modeling and analysis of conflict in the context of a dynamical system. Differential games model the interaction among all participants (or players) who select their controls to optimize their objectives. The controls of each player affect the system dynamics, which are modeled by a system of differential equations. SDGs extend differential games by considering stochastic dynamics.

SDDGs further extend the notion of SDG by considering general delayed features in the stochastic dynamics. SDDGs and their single-player counterparts, stochastic control problems with delay, encompass various models applicable to economics, advertising, and finance. For instance, in determining a firm's optimal advertising policy, Gozzi and Marinelli [17] 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 [35].

Of fundamental importance to the game is the concept of Nash equilibrium, which is a collection of all players' choices, ensuring that no player has the incentive to deviate unilaterally. The critical problem of finding Nash equilibria for SDGs and SDDGs allows one to model behavior throughout many disciplines, including management science, economics, social science, biology, military science, and finance. However, a major bottleneck comes from the notorious intractability of N-player games, and the direct computation of Nash equilibria is extremely time-consuming and memory demanding. The primary reason for these difficulties lies in the inherent high dimensionality of the problem.

However, deep learning methodologies are natural choices for solving problems with high dimensionality. In a series of recent works [26, 20, 22], the DFP theory and algorithms were developed for SDGs with a large number of heterogeneous players. The DFP framework embeds the fictitious play idea, introduced by Brown [9, 8], into designed architectures of deep neural networks to produce accurate and parallelizable algorithms with convergence analysis, and resolve the intractability issue (curse of dimensionality) caused by the complex modeling and underlying high-dimensional space in SDG. These materials are reviewed in Chapter 2.

SDGs already face the curse of dimensionality when the number of players N is large. Adding to this inherent complexity, SDDGs introduce a possibly infinite-dimensional component, as the drift and volatility of the associated stochastic delay differential equation (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 Section 2.6.8 in the book by Fabbri, Gozzi, and Swiech [15]. Numerically solving this HJB system would require an additional high-dimensional approximation to handle the infinite dimensionality arising from the delay.

However, deep learning approaches have been used to solve problems in similar instances. For example, Fouque and Zhang [16] parametrize 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 [21] solve the stochastic control problems with delay using neural networks. While the DFP framework originally applied to SDGs, we propose in Chapter 3 a new machine learning-based numerical methodology involving recurrent neural networks, and in particular, the long short-term memory architecture, that extends DFP to SDDGs.

Leveraging the DFP methodology, we can solve real-world problems that would be otherwise intractable from typical approaches. For instance, in Chapter 5 we extend the DFP methodology to solve a new stochastic, game-theoretic epidemiological model applied to COVID-19 dynamics among various interacting regions with separate central planners. This modeling is one example of a broader theme we believe DFP methodologies can address. Specifically, we believe the DFP methodologies, both existing and those we introduce in this work, can enable scientists, professionals, and other practitioners to approximate the behavior of agents in complex models with fast and efficient algorithms, allowing for insight into prediction and decision-making capabilities in otherwise intractable cases of model complexity.

1.2 Nash Equilibrium Problem for Stochastic Differential Games

We present a brief review of the mathematical set-up for stochastic differential games (SDGs). For a detailed viewpoint of SDGs, we refer to Volume I, Chapter 2 of the book

on mean field games by Carmona and Delarue [10].

Formally, we describe a general SDG by first considering a complete probability space $(\Omega, \mathcal{F}, \mathbb{P})$ on which we have a controlled SDE system:

$$\begin{aligned} &\mathrm{d}\boldsymbol{X}_t^{\boldsymbol{\alpha}} = \boldsymbol{b}(t, \boldsymbol{X}_t^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}_t)\,\mathrm{d}t + \sigma(t, \boldsymbol{X}_t^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}_t)\mathrm{d}\boldsymbol{W}_t,\ t \in [0, T], \\ &\boldsymbol{X}_0^{\boldsymbol{\alpha}} = \boldsymbol{x}_0 \in \mathbb{R}^n, \end{aligned} \tag{1.1}$$

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 $\mathcal{A}^i \subset \mathbb{R}^{m_i}$. The drift $\boldsymbol{b}$ and volatility σ are functions that map into $\mathbb{R}^n$ and $\mathbb{R}^{n\times k}$, respectively, and $\boldsymbol{W}$ is a k-dimensional, standard Brownian motion with filtration given by $(\mathcal{F}_t)_{t\in[0,T]}$.

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.

As important as the dynamics (1.1) are in dictating the game, so are the requirements of the control $\boldsymbol{\alpha}$. On one hand, we must specify what information the control variables are "allowed" to know based on what the underlying model represents. For instance, we enforce that $\boldsymbol{\alpha}_t$ only depends on past information up to and including time t, which represents the fact that future noises are unknowable to the controllers. However, we must specify the precise nature of this past "information" that is used in determining a given player's control. There are two main frameworks: the closed-loop and open-loop controls. Informally, the closed-loop case corresponds to players having information on the history of the state process $\boldsymbol{X}^{\boldsymbol{\alpha}}$, while the open-loop case corresponds to players only knowing the past information of the noise process $\boldsymbol{W}$. Formally, closed-loop controls are progressively measurable processes with respect to the filtration generated by the

state process ($\boldsymbol{X}_t^{\boldsymbol{\alpha}}$), while open-loop controls satisfy this condition with respect to the filtration generated by the noises themselves ($\boldsymbol{W}_t$).

Formally, we understand the set of allowable controls by defining the so-called admissible set for each player's control. For example, we define the admissible set of open-loop controls $\mathbb{A}^i_{\text{op}-\text{loop}}$ for player i to be

$$\mathbb{A}^i_{\text{op}-\text{loop}} = \left\{ \mathcal{F}_t-\text{prog. meas. } \beta^i : [0,T] \times \Omega \to \mathcal{A}^i \subset \mathbb{R}^{m_i} \middle| \int_0^T \mathbb{E}[|\beta_t^i|^2]\, \mathrm{d}t < \infty \right\}, \quad (1.2)$$

where we are using the abbreviation "prog. meas." to refer to the property of being progressively measurable. Here, we recall that $(\mathcal{F}_t)_{t\in[0,T]}$ is the filtration generated by the Brownian motion $\boldsymbol{W}$, and thus open-loop controls represent decisions at time t based on observing the noise process up to and including time t.

Denoting $\mathcal{F}_t^{\boldsymbol{X}}$ to be the filtration generated by $\boldsymbol{X}_t^{\boldsymbol{\alpha}}$, we define the admissible set of closed-loop controls $\mathbb{A}^i_{\text{cl}-\text{loop}}$ for player i to be

$$\mathbb{A}^i_{\text{cl}-\text{loop}} = \left\{ \mathcal{F}_t^{\boldsymbol{X}}-\text{prog. meas. } \beta^i : [0,T] \times \Omega \to \mathcal{A}^i \subset \mathbb{R}^{m_i} \middle| \int_0^T \mathbb{E}[|\beta_t^i|^2]\, \mathrm{d}t < \infty \right\}, \quad (1.3)$$

and thus we see that closed-loop controls represent decisions at time t based on observing the entire past history of the state-process $\boldsymbol{X}^{\boldsymbol{\alpha}}$ up to and including time t.

The definition of the appropriate admissible set of controls is not just to distinguish the informational dependency of the allowed controls. In fact, in both set definitions (1.2) and (1.3), there is a requirement for the controls of each player to be square integrable, i.e. $\int_0^T \mathbb{E}[|\beta_t^i|^2]\, \mathrm{d}t < \infty$, which is important for the well-posedness of the problem. Secondly, an important constraint on the choice of controls is enforced through the fact that player i's control takes values in $\mathcal{A}^i \subset \mathbb{R}^{m_i}$. The generality of $\mathcal{A}^i$ allows one to represent context dependent constraints on the output of controls for various problems. For example, a control representing the rate of fuel-consumption at time t may not be allowed to be negative. In essence, the choice of admissible sets relates to both well-posedness concerns of the underlying problem as well as model interpretation of the game itself by defining

the relevant output space of controls as well as their informational dependencies.

Lastly, we emphasize an important special case of closed-loop controls: the so-called "closed-loop feedback form" or "Markovian" controls. A Markovian control is one which represents a decision at time t made only based on the information of the current time t and the current state $\boldsymbol{X}_t^{\boldsymbol{\alpha}}$. In this case, we will slightly abuse notation and write player i's control at time t by $\alpha_t^i = \alpha^i(t, \boldsymbol{X}_t^{\boldsymbol{\alpha}})$. Formally, the admissible set for player i in the case of Markovian controls will be:

$$\mathbb{A}^i_{\mathrm{MC}} = \left\{ \text{Borel meas. } \beta^i : [0,T] \times \mathbb{R}^n \to \mathcal{A}^i \subset \mathbb{R}^{m_i} \,\middle|\, \int_0^T \mathbb{E}[|\beta^i(t, \boldsymbol{X}_t^{\boldsymbol{\alpha}})|^2] \, \mathrm{d}t < \infty \right\}. \quad (1.4)$$

The choice of admissible set of controls is context dependent, and we will in general write $\mathbb{A}^i$ for the admissible set for player i, whether this relates to potentially set definition (1.2), (1.3), or (1.4). 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 $\mathcal{A} = \otimes_{i=1}^N \mathcal{A}^i$.

Having defined the dynamics of the game as well as outlined the common admissible sets for choices of controls, we now describe what actually makes the problem a *game*. The game dynamics are understood by the fact that costs (or rewards) are incurred by each of the players based on 1) their choices of controls and 2) the realized state dynamics $\boldsymbol{X}^{\boldsymbol{\alpha}}$ associated to the choices of controls. Formally, we define the running and terminal costs for player i as f^i and g^i, respectively. Here, $f^i : [0,T] \times \mathbb{R}^n \times \mathcal{A} \to \mathbb{R}$ and $g^i : \mathbb{R}^n \to \mathbb{R}$ are deterministic measurable functions. From these functions, we define the expected cost J^i for player i to be

$$J^i[\boldsymbol{\alpha}] = \mathbb{E}\left[\int_0^T f^i(t, \boldsymbol{X}_t^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}_t) \mathrm{d}t + g^i(\boldsymbol{X}_T^{\boldsymbol{\alpha}}) \right]. \quad (1.5)$$

The idea is that each player selects their control α^i so as to minimize their expected cost (1.5) given the game dynamics (1.1). Player i's control influences the other players' expected costs which in turn influences their decisions and thus player i's expected cost and choice of control. Simply put, the cost functions are coupled with each player's

choice of controls, and thus the idea that each player minimizes their own expected cost requires a specific notion of solution. In the context of non-cooperative games, this notion of solution is encapsulated by the Nash equilibrium.

The problem we consider is to find the Nash equilibrium in $\mathbb{A}$ for the SDG described in (1.1) and (1.5). 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. In other words, a Nash equilibrium is an N-tuple consisting of each player's strategy (i.e. choice of controls) such that no player has an incentive to unilaterally deviate their strategy. To be precise, we understand the Nash equilibrium through the following definition.

Definition 1.1. We say that $\boldsymbol{\alpha}^* = (\alpha^{*1}, \cdots, \alpha^{*N}) \in \mathbb{A}$ is a Nash equilibrium if

$$J^i[\boldsymbol{\alpha}^*] \leq J^i[\alpha^{*1}, \cdots, \alpha^{*i-1}, \ \beta^i, \alpha^{*i+1}, \cdots, \alpha^{*N}], \quad \forall i \in \{1, \cdots, N\}, \forall \beta^i \in \mathbb{A}^i. \tag{1.6}$$

In conclusion, the full problem of finding the Nash equilibrium for an SDG can be understood through the following key components. First, the admissible set of all players' controls $\mathbb{A}$ is determined by potentially (1.2), (1.3), or (1.4), which defines an appropriate space of controls based on problem interpretation and well-posedness concerns. Most notably, the choice of $\mathbb{A}$ reflects the set of information the players are allowed to access in determining their decisions at a given time. Next, Eqs. (1.1) and (1.5) define a map from the choices of controls for each player $\boldsymbol{\alpha} \in \mathbb{A}$ to the cost experienced by a given player J^i. The condition (1.6) expresses how each player rationally chooses their strategy based on their own cost, given the choices of the other players, leading to equilibrium.

Chapter 2

Machine Learning for Stochastic Differential Games

The work presented in this chapter, which was prepared for this dissertation, includes material previously published [43] and material submitted for publication [3].

2.1 Deep Fictitious Play for Approximating Nash Equilibrium

As mentioned in Section 1.1, the problem of finding Nash equilibria for a SDG is incredibly challenging computationally especially when the number of players N is large. In this chapter, we review the methodology in [26, 20], which establishes a set of numerical techniques known as deep fictitious play (DFP) for solving SDGs with a large number of heterogeneous players.

DFP is a broad technique whereby Nash equilibrium controls are approximated iteratively via deep learning techniques in a manner akin to Brown's fictitious play [9, 8]. 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

$$\inf_{\beta^i \in \mathbb{A}^i} J^i[\alpha^1, \cdots, \alpha^{i-1}, \beta^i, \alpha^{i+1}, \cdots, \alpha^N], \tag{2.1}$$

and then iterate over the solutions to these optimization problems.

Assuming a unique minimum occurs at $\beta^{i,*}$, one uses $\beta^{i,*}$ to inform α^i in future iterations of the optimization (2.1). Doing this for each player $i \in \{1, \cdots, N\}$ constitutes one stage of this modified fictitious play. In the case of Brown's fictitious play, α^j in the optimization problem (2.1) would be given by the empirical average of player j's control taken over the previous rounds of play. However, for DFP methodologies, one will usually take α^j to be the exact control from the previous round of play for memory efficiency reasons (see [20, Remarks 3.1 and 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 m 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

1: Initialize each $\alpha^{i,0} \in \mathbb{A}^i$.
2: **for** m in 1 to N_{stages} **do**
3: **for** i in 1 to N **do**
4: $\alpha^{i,m} = \arg\min_{\beta^i \in \mathbb{A}^i} J^i[\alpha^{1,m-1}, \cdots, \alpha^{i-1,m-1}, \beta^i, \alpha^{i+1,m-1}, \cdots, \alpha^{N,m-1}]$
5: **end for**
6: **end for**

In Algorithm 1, $\alpha^{i,m}$ is the control for player i at stage m, 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 [26] 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 (2.1). In reality, (2.1) 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 (2.1) via deep learning and any of these would be considered deep fictitious play. One approach to solve the optimization problem (2.1) is based on the so-called direct parametrization discussed by Han and E in [19]. 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 [26]. This is discussed in Section 2.2 of this chapter.

The DFP methodology introduced in [20] seeks to solve the Hamilton-Jacobi-Bellman (HJB) equation given by the decoupled optimal control problem defined by the optimization problem (2.1). 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, Han, and Jentzen [13, 23]. This method will be covered in Section 2.3 and adapted to solve the COVID-19 model problem in Chapter 5. Alternatively, the solution of the HJB equation could be approximated with the Deep Galerkin method introduced by Sirignano and Spiliopoulos [41]. This would also be considered deep fictitious play as it solves the optimization problem posed by the fictitious-play like Algorithm 1 through deep learning.

2.2 Deep Fictitious Play via Direct Parametrization Method

Deep fictitious play with the direct parametrization method involves approaching the optimization problems in Algorithm 1 directly with neural network parametrized controls. Specifically, we take the control β^i represented in Algorithm 1 to be a neural network in some appropriate class, and the optimization is done with stochastic gradient descent

based methods using the cost function J^i as the loss. In order to make this computationally tractable, we will have to approximate the expected cost J^i numerically. This is done by first discretizing the SDE (1.1) according to the Euler-Maruyama method:

$$\hat{\boldsymbol{X}}_{k+1} = \hat{\boldsymbol{X}}_k + \boldsymbol{b}(t_k, \hat{\boldsymbol{X}}_k, \hat{\boldsymbol{\alpha}}_k)\Delta t_k + \sigma(t_k, \hat{\boldsymbol{X}}_k, \hat{\boldsymbol{\alpha}}_k)\Delta \boldsymbol{W}_k, \quad k = 0, \dots, N_T - 1, \tag{2.2}$$

where $\Delta \boldsymbol{W}_k = \boldsymbol{W}_{t_{k+1}} - \boldsymbol{W}_{t_k}$ from the original Brownian motion $\boldsymbol{W}$ in Eq. (1.1). 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 neural networks; each player's control is parametrized by their own neural network. We consider the stochastic differential game defined by Eqs. (1.1) and (1.5) in the case of Markovian controls, i.e. $\mathbb{A}^i = \mathbb{A}^i_{\text{MC}}$ from the set definition (1.4). In this case of Markovian controls, we can parametrize the decision at time t_k by:

$$\hat{\alpha}_k^i = \phi_{NN}^i(t_k, \hat{\boldsymbol{X}}_k; \vartheta_i),$$

where ϑ_i are the parameters of player i's corresponding neural network.

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

$$\hat{J}^i[\hat{\boldsymbol{\alpha}}] = \frac{1}{N_{\text{batch}}} \sum_{\ell=1}^{N_{\text{batch}}} \left[\sum_{k=1}^{N_T} f^i(t_k, \hat{\boldsymbol{X}}_k(\omega_\ell), \hat{\boldsymbol{\alpha}}_k)\Delta t + g^i(\hat{\boldsymbol{X}}_{N_T}(\omega_\ell)) \right], \tag{2.3}$$

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 (2.2).

In essence, each player i will select their desired neural network parameters ϑ_i, determining their choice of control. 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 $(\hat{J}^i)_{i=1}^N$. The neural network parameters are then adjusted in a manner akin to fictitious play . This leads us to Algorithm 2 below.

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 NNs at stage 0.
Select N_{stages} of deep fictitious play based on the computational budget.
for m in 0 to N_{stages} **do**
    for i in 1 to N **do**
        Compute N_{batch} trajectories $\hat{\boldsymbol{X}}$ of the numerical SDE (2.2) under the given controls $(\hat{\alpha}^{j,m})_j$, where the control for each player j is given by $\hat{\alpha}^{j,m} = \phi^j_{NN}(\cdot; \vartheta_{j,m})$.
        Compute the numerical cost $\hat{J}^i$ from Eq. (2.3).
        Compute via automatic differentiation $\nabla_{\vartheta_{i,m}} \hat{J}^i$.
        Do a gradient descent step or similar (e.g. Adam) on $\vartheta_{i,m}$ with learning rate l_r, i.e.
        $$\vartheta_{i,m+1} = \vartheta_{i,m} - l_r \nabla_{\vartheta_{i,m}} \hat{J}^i.$$
    end for
end for

As before, we consider N_{stages} of iteration, where $\phi^j_{NN}(\cdot, \vartheta_{j,m})$ parametrizes the control for player j at stage m as selected by the neural network parameters $\vartheta_{j,m}$. As motivated by Algorithm 1, we would then like to compute the optimal $\vartheta_{j,m}$ holding the other neural network parameters, $(\vartheta_{j',m})_{j' \neq j}$, 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 $\vartheta_{i,m}$-gradient of $\hat{J}^i$. This is possible numerically through automatic differentiation [7]. Instead of standard gradient descent, one may 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 advantages to choosing the Adam optimization over traditional stochastic gradient descent is due to it having improved convergence properties in many cases [28]. 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 neural network parameters after each gradient descent step, leading to increased costs.[1] Instead, in Algorithm 2, we move on immediately to the next

[1]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

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.[2]

2.3 Deep Fictitious Play via Hamilton-Jacobi-Bellman System and Associated BSDE

We consider the SDG defined by Eqs. (1.1) and (1.5) in the case of Markovian controls, i.e. $\mathbb{A}^i = \mathbb{A}^i_{\mathrm{MC}}$ from the set definition (1.4). That is, we can understand the control(s) for each player i as a function of the current time and state $\alpha^i_t = \alpha^i_t(t, \boldsymbol{X}_t)$. We also assume that the control does not enter into the volatility, i.e. σ in Eq. (1.1) has the form $\sigma = \sigma(t, \boldsymbol{X}_t)$. We define the value function of player i by

$$V^i(t, \boldsymbol{x}) = \inf_{\alpha^i \in \mathbb{A}^i} \mathbb{E}\left[\int_t^T f^i(s, \boldsymbol{X}_s, \boldsymbol{\alpha}(s, \boldsymbol{X}_s))\,\mathrm{d}s + g^i(\boldsymbol{X}_T) | \boldsymbol{X}_t = \boldsymbol{x}\right].$$

By dynamic programming, $\boldsymbol{V}$ solves the following Hamilton-Jacobi-Bellman (HJB) system

$$\begin{cases} \partial_t V^i + \inf_{\alpha^i \in \mathbb{A}^i} H^i(t, \boldsymbol{x}, \boldsymbol{\alpha}(t, \boldsymbol{x}), \nabla_{\boldsymbol{x}} V^i) + \frac{1}{2}\mathrm{Tr}(\sigma(t, \boldsymbol{x})^{\mathrm{T}} \mathrm{Hess}_{\boldsymbol{x}} V^i \sigma(t, \boldsymbol{x})) = 0, \\ V^i(T, \boldsymbol{x}) = g(\boldsymbol{x}), \quad i \in \{1, \cdots, N\}, \end{cases} \tag{2.4}$$

the beginning stages, since the approximate controls $(\hat{\alpha}^{i,m})_i$ 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 $\hat{\alpha}^{i,m}$ given the choices of the other players.

[2]Of course it is exceedingly unlikely that controls of the form $(\phi^i_{NN}(\cdot, \cdot; \vartheta^{i,*}))_i$ happen to be a Nash equilibrium. However, if it so happens to be the case that $(\phi^i_{NN}(\cdot, \cdot; \vartheta^{i,*}))_i$ is a Nash equilibrium for $\hat{\boldsymbol{J}}$, and assuming sufficient smoothness of $\hat{\boldsymbol{J}}$, then for each i it holds that $\nabla_{\vartheta^{i,*}} \hat{J}^i[(\phi^j_{NN}(\cdot, \cdot; \vartheta^{j,*}))_j] = 0$, and therefore $(\vartheta^{i,*})_{i=1}^N$ is a fixed point of the iteration.

where Tr is the trace of a matrix, and H^i is the Hamiltonian defined by

$$H^i(t, \boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{p}) = \boldsymbol{b}(t, \boldsymbol{x}, \boldsymbol{\alpha}) \cdot \boldsymbol{p} + f^i(t, \boldsymbol{x}, \boldsymbol{\alpha}). \tag{2.5}$$

Finding the optimal policies for N players is therefore equivalent to solving N-coupled n-dimensional nonlinear equations (2.4). The recently proposed DFP algorithm in [20] has shown excellent numerical performance in solving these high-dimensional, coupled HJB equations with convergence analysis in [22]. In this work, each individual problem is solved by the deep BSDE method [13, 23]. The algorithm starts with some initial guess $\boldsymbol{\alpha}^0$. At the $(m+1)^{th}$ stage, given the optimal policies $\boldsymbol{\alpha}^m$ at the previous stage, the algorithm solves the following PDEs

$$\begin{cases} \partial_t V^{i,m+1} + \inf\limits_{\alpha^i \in \mathbb{A}^i} H^i(t, \boldsymbol{x}, (\alpha^i, \boldsymbol{\alpha}^{-i,m})(t, \boldsymbol{x}), \nabla_{\boldsymbol{x}} V^{i,m+1}) \\ \qquad\qquad + \dfrac{1}{2}\mathrm{Tr}(\sigma(t, \boldsymbol{x})^{\mathrm{T}} \mathrm{Hess}_{\boldsymbol{x}} V^{i,m+1} \sigma(t, \boldsymbol{x})) = 0, \\ V^{i,m+1}(T, \boldsymbol{x}) = g^i(\boldsymbol{x}), \quad i \in \{1, \cdots, N\}, \end{cases} \tag{2.6}$$

and obtains the $(m+1)^{th}$ stage's optimal strategy by:

$$(\alpha^{i,m+1})(t, \boldsymbol{x}) = \arg\min_{\alpha^i \in \mathbb{A}^i} H^i(t, \boldsymbol{x}, (\alpha^i, \boldsymbol{\alpha}^{-i,m})(t, \boldsymbol{x}), \nabla_{\boldsymbol{x}} V^{i,m+1}(t, \boldsymbol{x})). \tag{2.7}$$

Here, $\boldsymbol{\alpha}^{-i,m}$ stands for all others' optimal policies besides player i from the m^{th} stage and are considered to be fixed functions when solving the PDE at the current stage. In the sequel, to simplify notations, we omit the stage label m in the superscript when there is no risk of confusion. To solve Eq. (2.6) at each stage, it is first rewritten as

$$\begin{aligned} \partial_t V^i + \frac{1}{2}\mathrm{Tr}(\sigma(t, \boldsymbol{x})^{\mathrm{T}} \mathrm{Hess}_{\boldsymbol{x}} V^i \sigma(t, \boldsymbol{x})) + \mu^i(t, \boldsymbol{x}; \boldsymbol{\alpha}^{-i}) \cdot \nabla_{\boldsymbol{x}} V^i \\ + \zeta^i(t, \boldsymbol{x}, \sigma(t, \boldsymbol{x})^{\mathrm{T}} \nabla_{\boldsymbol{x}} V^i; \boldsymbol{\alpha}^{-i}) = 0, \end{aligned} \tag{2.8}$$

for some functions μ^i and ζ^i such that

$$\inf_{\alpha^i \in \mathbb{A}^i} H^i(t, \boldsymbol{x}, (\alpha^i, \boldsymbol{\alpha}^{-i,m})(t, \boldsymbol{x}), \nabla_{\boldsymbol{x}} V^i) = \mu^i(t, \boldsymbol{x}; \boldsymbol{\alpha}^{-i}) \cdot \nabla_{\boldsymbol{x}} V^i + \zeta^i(t, \boldsymbol{x}, \sigma(t, \boldsymbol{x})^{\mathrm{T}} \nabla_{\boldsymbol{x}} V^i; \boldsymbol{\alpha}^{-i}).$$

The solution to Eq. (2.6) can then found by solving the equivalent BSDE $(\boldsymbol{X}_t^i, Y_t^i, \boldsymbol{Z}_t^i) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^k$:

$$\begin{cases} \boldsymbol{X}_t^i = \boldsymbol{x}_0 + \displaystyle\int_0^t \mu^i(s, \boldsymbol{X}_s^i; \boldsymbol{\alpha}^{-i}(s, \boldsymbol{X}_s^i))\,\mathrm{d}s + \int_0^t \sigma(s, \boldsymbol{X}_s^i)\,\mathrm{d}\boldsymbol{W}_s, & (2.9) \\ Y_t^i = g^i(\boldsymbol{X}_T^i) + \displaystyle\int_t^T \zeta^i(s, \boldsymbol{X}_s^i, \boldsymbol{Z}_s^i; \boldsymbol{\alpha}^{-i}(s, \boldsymbol{X}_s^i))\,\mathrm{d}s - \int_t^T (\boldsymbol{Z}_s^i)^{\mathrm{T}}\,\mathrm{d}\boldsymbol{W}_s, & (2.10) \end{cases}$$

in the sense of [36, 14, 37],

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

The high-dimensional BSDE (2.9)–(2.10) is tackled by the deep BSDE method proposed in [13, 23]. This is approached by simulating both SDEs forward with a guess for the Y_0^i value and the $\boldsymbol{Z}^i$ process. Numerically, this is done by discretizing (2.9)–(2.10). With a neural network estimating the value for Y_0^i as a function of $\boldsymbol{x}_0$ and neural network estimating the discretized $\boldsymbol{Z}^i$ process, one can simulate (2.10) forward in time, where the loss function for these networks is given by $\mathbb{E}\left[|Y_T^i - g^i(\boldsymbol{X}_T^i)|^2\right]$ with respect to the discretized processes.

Specifically, this is done as follows. One takes a partition π of size N_T on the time interval $[0, T]$, $0 = t_0 < t_1 < .... < t_{N_T} = T$. We use the superscript π to denote the discretized processes corresponding to Eqs. (2.9)–(2.10). To ease notation, we replace the subscript t_k by k. One aims to solve the minimization problem over the class of neural networks in hypothesis spaces $\mathcal{N}_0^{i'}, \{\mathcal{N}_k^i\}$:

$$\begin{aligned} &\inf_{\psi_0 \in \mathcal{N}_0^{i'}, \left\{\phi_k \in \mathcal{N}_k^i\right\}_{k=0}^{N_T-1}} \mathbb{E}\left[|Y_T^{i,\pi} - g^i(\boldsymbol{X}_T^{i,\pi})|^2\right] \\ \text{s.t. } &\boldsymbol{X}_0^{i,\pi} = \boldsymbol{x}_0, \quad Y_0^{i,\pi} = \psi_0\left(\boldsymbol{X}_0^{i,\pi}\right), \quad \boldsymbol{Z}_k^{i,\pi} = \phi_k\left(\boldsymbol{X}_k^{i,\pi}\right), \\ &\boldsymbol{X}_{k+1}^{i,\pi} = \boldsymbol{X}_k^{i,\pi} + \mu^i\left(t_k, \boldsymbol{X}_k^{i,\pi}; \boldsymbol{\alpha}_k^{-i,\pi}(\boldsymbol{X}_k^{i,\pi})\right)\Delta t_k + \sigma\left(t_k, \boldsymbol{X}_k^{i,\pi}\right)\Delta \boldsymbol{W}_k, \\ &Y_{k+1}^{i,\pi} = Y_k^{i,\pi} - \zeta^i\left(t_k, \boldsymbol{X}_k^{i,\pi}, \boldsymbol{Z}_k^{i,\pi}; \boldsymbol{\alpha}_k^{i,\pi}(\boldsymbol{X}_k^{i,\pi})\right)\Delta t_k + \left(\boldsymbol{Z}_k^{i,\pi}\right)^{\mathrm{T}} \Delta \boldsymbol{W}_k, \end{aligned} \tag{2.11}$$

where $\Delta t_k = t_{k+1} - t_k$, $\Delta \boldsymbol{W}_k = \boldsymbol{W}_{t_{k+1}} - \boldsymbol{W}_{t_k}$. Approximating the solution to the BSDE (2.9)–(2.10), one has approximated $Y_t^i = V^i(t, \boldsymbol{X}_t)$ and $\boldsymbol{Z}_t^i = \sigma(t, \boldsymbol{X}_t^i)^{\mathrm{T}} \nabla_{\boldsymbol{x}} V^i(t, \boldsymbol{X}_t^i)$. With approximation of these quantities, one can compute player i's control at the next stage $m+1$, $\alpha^{i,m+1}$, according to Eq. (2.7). This allows one to proceed with the idea of Algorithm 1, where the interior optimization problem for player i at stage m is approximated from the optimization problem (2.11) using stochastic gradient descent.

Chapter 3

Machine Learning for Stochastic Delay Differential Games and Financial Modeling

The work described in this chapter, which was prepared for this dissertation, contains material that has already been submitted for publication [3].

We now focus on the problem of finding the closed-loop Nash equilibrium for stochastic delay differential games (SDDGs). As mentioned in Section 1.1, SDDGs are challenging to approximate numerically due to their inherent dimensionality arising from the number of players as well as the delayed dynamics which effectively cause these problems to become infinite dimensional.

To address the challenge of the dimensionality of SDDGs, we propose a deep learning-based method that effectively handles the delay. Inspired by the approach presented in [21], which utilizes recurrent neural networks (RNNs) to solve stochastic control problems with delay, we introduce an algorithm for finding the Nash equilibrium of SDDGs. Specifically, we parametrize players' controls using RNNs and approximate their objective functions by sampling the game dynamics under these RNN-based controls. The param-

eters of the RNNs are then optimized using the concept of deep fictitious play (DFP), as introduced in [26, 20] and discussed in Chapter 2. 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.

We then validate the proposed deep learning algorithm numerically on a set of problems with known closed-form solutions. This includes a new class of problems that we formally introduce and solve in Chapter 4. We summarize our results for these new problems in Sections 3.3.1 and 3.3.2. We also consider the model introduced and solved in [11] 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 this chapter is as follows. In Section 3.1, we introduce the mathematical problem of finding the Nash equilibria for a SDDG. In Section 3.2, we propose a numerical algorithm for approximating SDDGs. In Section 3.3, we formulate and provide analytical solutions for the closed-loop Nash equilibrium of the SDDGs we later solve numerically using our proposed algorithm. Specifically, we devote Sections 3.3.1 and 3.3.2 to present the newly considered SDDGs and their solutions. Since this chapter is dedicated to presenting and displaying the results of our numerical methodology for SDDGs, we postpone the derivation, interpretation, and proofs of Nash equilibrium solutions of these newly considered SDDGs to Chapter 4. In Section 3.4, we demonstrate the numerical results of our algorithm compared to the known solutions of the considered problems.

3.1 Nash Equilibrium Problem for Stochastic Delay Differential Games

We now consider the problem of finding closed-loop Nash equilibria for SDDGs. We define the problem mathematically following the similar setup in [21] for stochastic control problems with delay. Because of the additional formality required by SDDGs, we state the problem in detail for this case even though it is similar to the SDG presented in Section 1.2.

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

$$\begin{aligned} \mathrm{d}\boldsymbol{X}_t^{\boldsymbol{\alpha}} &= \boldsymbol{b}(t, \boldsymbol{X}_{[t-\tau,t]}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}_{[t-\tau,t]})\,\mathrm{d}t + \sigma(t, \boldsymbol{X}_{[t-\tau,t]}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}_{[t-\tau,t]})\mathrm{d}\boldsymbol{W}_t,\ t \in [0,T], \\ \boldsymbol{X}_t^{\boldsymbol{\alpha}} &= \boldsymbol{\zeta}(t),\ t \in [-\tau, 0], \\ \boldsymbol{\alpha}_t &= \boldsymbol{\phi}(t),\ t \in [-\tau, 0). \end{aligned} \tag{3.1}$$

Similarly to the SDG dynamics (1.1), we have the state process, $\boldsymbol{X}^{\boldsymbol{\alpha}}$, takes values in $\mathbb{R}^n$, and $\boldsymbol{\alpha} = (\alpha^1, \cdots, \alpha^N)$ is the collection of all players' controls, where α_t^i 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. (3.1) 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, $\boldsymbol{b}$, 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.

Formally, we define $\boldsymbol{X}_{[t-\tau,t]}^{\boldsymbol{\alpha}}$ to be a map from $[-\tau, 0]$ to the space of square integrable random variables $L^2(\Omega)$ given by $\boldsymbol{X}_{[t-\tau,t]}^{\boldsymbol{\alpha}}(s) := \boldsymbol{X}_{s+\tau+t}^{\boldsymbol{\alpha}}$. In particular, we seek solutions

$\boldsymbol{X}^{\boldsymbol{\alpha}}$ to Eq. (3.1) 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 $\boldsymbol{\alpha}_{[t-\tau,t]}(s) := \boldsymbol{\alpha}_{s+\tau+t}$, where the stochastic processes $(\boldsymbol{\alpha}_t)_{t\in[0,T]}$ belongs to an admissible set $\mathbb{A}$ defined later by the set definition (3.2).

For a fixed choice of controls $\boldsymbol{\alpha}$, one can consider the existence and uniqueness of the SDDE (3.1). For this SDDE, requiring $\boldsymbol{b}$ and σ to be Lipschitz in the second argument to ensures the existence and uniqueness of a strong solution. To be precise, this Lipschitz condition is

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

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 [34].

We require that each control, α^i, is in the class of closed-loop controls which will be similar to admissible set of controls defined for the SDG case in set definition (1.3), but with some differences due to the delay that will require us to restate the definition for the case of SDDGs. Intuitively, the closed-loop controls are those that take the form $\alpha^i_t = \phi^i(t, \boldsymbol{X}^{\boldsymbol{\alpha}}_{[-\tau,t]})$, and therefore represent a decision at time t based on observation of the state process up to and including this time. The set of closed-loop controls is a natural choice for SDDGs as the dynamics themselves are memory dependent. Intuitively, the increment at time t depends on past information, and thus a reasonably informed decision at time t should incorporate the available past information as well.

Formally, closed-loop controls are progressively measurable processes with respect

to the filtration generated by the state process $\boldsymbol{X}_t^{\boldsymbol{\alpha}}$. Denoting $\mathcal{F}_t^{\boldsymbol{X}}$ to be the filtration generated by $\boldsymbol{X}_{[-\tau,t]}^{\boldsymbol{\alpha}}$, we define the admissible set of closed-loop controls $\mathbb{A}^i$ for player i to be

$$\mathbb{A}^i = \left\{ \mathcal{F}_t^{\boldsymbol{X}} - \text{prog. meas. } \beta^i : [-\tau, T] \times \Omega \to \mathcal{A}^i \subset \mathbb{R}^{m_i} \middle| \int_{-\tau}^{T} \mathbb{E}[|\beta_t^i|^2] \, \mathrm{d}t < \infty \right\}, \tag{3.2}$$

where we are using the abbreviation "prog. meas." to refer to the property of being progressively measurable. Again, 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 $\mathcal{A} = \otimes_{i=1}^N \mathcal{A}^i$.

The running and terminal costs for player i are f^i and g^i, respectively. Here, $f^i : [0,T] \times L^2(\Omega, C([-\tau,0];\mathbb{R}^n)) \times L^2(\Omega, C([-\tau,0];\mathcal{A})) \to \mathbb{R}$ and $g^i : L^2(\Omega, C([-\tau,0];\mathbb{R}^n)) \to \mathbb{R}$ are deterministic measurable functionals. From these functionals, we define the expected cost J^i for player i to be

$$J^i[\boldsymbol{\alpha}] = \mathbb{E}\left[\int_0^T f^i(t, \boldsymbol{X}_{[t-\tau,t]}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}_{[t-\tau,t]}) \mathrm{d}t + g^i(\boldsymbol{X}_{[T-\tau,T]}^{\boldsymbol{\alpha}}) \right]. \tag{3.3}$$

The problem we consider is to find the Nash equilibrium, Definition (1.1), in the admissible set $\mathbb{A}$ defined from Eq. (3.2) for the SDDG described by the dynamics (3.1) and expected cost for each player (3.3). 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.

3.2 The Deep Learning Algorithm for Games with Delay

In Chapter 2, we presented two methods to approximate Markovian Nash equilibria for SDGs with machine learning algorithms that fall under the category of deep fictitious play (DFP).

We are now interested in the closed-loop Nash equilibrium problem for SDDGs. In the case of a SDDG, the associated HJB equations discussed in Chapter 2 would be infinite-

dimensional due to the delay [15]. Therefore, modifying a DFP approach which involves approximating the HJB equations (for example, approaches similar to that introduced in Section 2.3) would greatly add to the dimensionality. Because of this, we propose a DFP algorithm for SDDGs based on extending the direct parametrization method outlined in Section 2.2.

We recall that the key steps to DFP with the direct parametrization method presented in Section 2.2 are 1) approximating the dynamics and expected cost functionals with discrete dynamics and empirical cost estimates 2) parametrizing the controls as functions in a class of suitable neural networks 3) iteratively updating the parameters of the neural network control functions in a method akin to fictitious play.

We propose to parametrize each player's controls with RNNs (and in particular LSTMs) so as to capture past information and respect the closed-loop structure of the problem. The approximation of dynamics and expected cost functionals are similar to the method outlined in Section 2.2, but with some modifications due to the dynamics incorporating delayed information.

3.2.1 Discretized Stochastic Delay Differential Game

We now consider the discrete analogue of the stochastic delay differential game defined by Eqs. (3.1)–(3.3). We represent the SDDE (3.1) numerically by its associated Euler Maruyama scheme, and the expected cost (3.3) is estimated with an empirical cost computed by the Monte Carlo method. This is similar to Section 2.2, but modified for delayed dynamics.

The discrete analogue of the SDDE (3.1) 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 [32] 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 $N_\tau = \frac{\tau}{\Delta t}$ and $N_T = \frac{T}{\Delta t}$ are integers without loss of generality[1]. Then, we define the discrete approximation $(\hat{\boldsymbol{X}}_k)_k$ through the Euler-Maruyama scheme

$$\begin{aligned}
\hat{\boldsymbol{X}}_{k+1} = \; & \hat{\boldsymbol{X}}_k + \hat{\boldsymbol{b}}(t_k, \hat{\boldsymbol{X}}_{k-N_\tau}, \cdots, \hat{\boldsymbol{X}}_k, \hat{\boldsymbol{\alpha}}_{k-N_\tau}, \cdots, \hat{\boldsymbol{\alpha}}_k)\Delta t \\
& + \; \hat{\sigma}(t_k, \hat{\boldsymbol{X}}_{k-N_\tau}, \cdots, \hat{\boldsymbol{X}}_k, \hat{\boldsymbol{\alpha}}_{k-N_\tau}, \cdots, \hat{\boldsymbol{\alpha}}_k)\Delta \boldsymbol{W}_k, \quad k = 0, \ldots, N_T - 1, \\
\hat{\boldsymbol{X}}_k = \; & \boldsymbol{\zeta}(t_k), \quad k = -N_\tau, \ldots, 0, \\
\hat{\boldsymbol{\alpha}}_k = \; & \boldsymbol{\phi}(t_k), \quad k = -N_\tau, \ldots, -1,
\end{aligned} \tag{3.4}$$

where $\Delta \boldsymbol{W}_k = \boldsymbol{W}_{t_{k+1}} - \boldsymbol{W}_{t_k}$ from the original Brownian motion $\boldsymbol{W}$ in Eq. (3.1). 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 parametrized by their own RNN.

We have also approximated the functionals $\boldsymbol{b}$ and σ occurring in the SDDE (3.1) with discrete counterparts $\hat{\boldsymbol{b}}$ and $\hat{\sigma}$. While the functionals $\boldsymbol{b}, \sigma$ 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 3.3.1 and 3.3.2 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 3.3.3 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

$$\hat{J}^i[\hat{\boldsymbol{\alpha}}] = \frac{1}{N_{\text{batch}}} \sum_{\ell=1}^{N_{\text{batch}}} \left[\sum_{k=1}^{N_T} f^i(t_k, \hat{\boldsymbol{X}}_k(\omega_\ell), \hat{\boldsymbol{\alpha}}_k)\Delta t + g^i(\hat{\boldsymbol{X}}_{N_T}(\omega_\ell)) \right], \tag{3.5}$$

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

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 (3.4).

3.2.2 Recurrent Neural Network Parametrized Controls

In the mathematical problem we introduced for SDDGs in Section 3.1, each player's control lives in the admissible set $\mathbb{A}^i$ defined by Eq. (3.2), containing suitable $\mathcal{F}_t^X$-progressively measurable strategies, i.e, closed-loop controls. To capture this closed-loop aspect, 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, $(\hat{\boldsymbol{X}}_{k'})_{k' \leq k}$. 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, Hinton, and Williams [39], 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 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

$$\hat{\alpha}_k^i = \phi_{RNN}\left((t_{-N_\tau}, \hat{\boldsymbol{X}}_{-N_\tau}), \cdots, (t_k, \hat{\boldsymbol{X}}_k); \vartheta_i\right). \tag{3.6}$$

The map ϕ_{RNN} in Eq. (3.6) 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

$$\begin{aligned} y_k^i, h_k^i &= \phi_{RNNcell}(t_k, \hat{\boldsymbol{X}}_k, h_{k-1}^i; \vartheta_i), \qquad && k = -N_\tau + 1, \cdots, N_T, \\ \hat{\alpha}_k^i &= y_k^i, && k = 0, \cdots, N_T - 1, \\ h_{-N_\tau}^i &= h_{init}. \end{aligned} \tag{3.7}$$

The time-k map of this recurrence relation defines a map $\Big((t_{-N_\tau}, \hat{\boldsymbol{X}}_{-N_\tau}), \cdots, (t_k, \hat{\boldsymbol{X}}_k)\Big) \mapsto y_k^i = \hat{\alpha}_k^i$, which is precisely the map we call ϕ_{RNN} in Eq. (3.6). 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. (3.6)–(3.7) provides us with a reasonable space to approximate the closed-loop controls in the set definition (3.2). 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 approximating the Nash equilibrium problem through DFP.

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 Eq. (3.4), (3.7) 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 3.2.4 with a deep fictitious play algorithm for approximating the Nash equilibrium.

3.2.3 The Long Short-Term Memory Recurrent Network

RNNs span a vast variety of architectures and the controls defined by Eqs. (3.6)–(3.7) are quite broad. We now focus on the precise architecture we use in our numerical experiments featured in Section 3.4. Motivated by the implementation in [21], we choose to use the specific RNN architecture given by the so-called long short-term memory (LSTM) network.

The LSTM was first introduced by Hochreiter and Schmidhuber [25] 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. (3.7). 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. (3.7). 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

$$\begin{aligned} i_k &= \sigma(W_i x_k + U_i h_{k-1} + b_i), \\ f_k &= \sigma(W_f x_k + U_f h_{k-1} + b_f), \\ o_k &= \sigma(W_o x_k + U_o h_{k-1} + b_o), \\ c_k &= f_k \odot c_{k-1} + i_k \odot \tanh(W_c x_k + U_c h_{k-1} + b_c), \\ h_k &= o_k \odot \tanh(c_k), \\ y_k &= W_y h_k + b_y. \end{aligned} \tag{3.8}$$

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. (3.8) specifically for player j.

With the cell-map, $\phi_{LSTMcell}$, specified by Eq. (3.8), the choice of controls in Eq. (3.4) is determined by taking player j's control at time t_k to be

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

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

$$\begin{aligned} x_k &= (t_k, \hat{\boldsymbol{X}}_k), && k = -N_\tau, \cdots, N_T, \\ y_k^j, c_k^j, h_k^j &= \phi_{LSTMcell}(x_k, h_{k-1}^j, c_{k-1}^j; \vartheta_j), && k = -N_\tau + 1, \cdots, N_T - 1, \\ \hat{\alpha}_k^j &= y_k^j, && k = 0, \cdots, N_T - 1. \end{aligned} \tag{3.9}$$

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 [21]. Note that $x_k = (t_k, \hat{\boldsymbol{X}}_k)$ is a vector in $\mathbb{R}^{1+n}$, as $\hat{\boldsymbol{X}}_k$ 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.

3.2.4 Implementation Details and Full Algorithm

For computational efficiency, we may not be inputting a single sample of $(t_k, \hat{\boldsymbol{X}}_k)$ into the LSTM as indicated by Eq. (3.9), but rather a so-called "batch" which contains N_{batch} paths of $\hat{\boldsymbol{X}}$ determined by respective N_{batch} samples of Brownian paths.

Precisely, we can augment the operations in the Euler-Maruyama method (3.4) so

that it iterates over a batch $(\hat{\boldsymbol{X}}_k(\omega_\ell))_{\ell=1}^{N_{\text{batch}}}$ 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 $(\Delta\boldsymbol{W}_k(\omega_\ell))_{\ell=1}^{N_{\text{batch}}}$. The drift $\hat{\boldsymbol{b}}$ 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. (3.8) naturally acts pointwise along the batch dimension. To have the LSTM defined by Eqs. (3.8)–(3.9) extended to act pointwise on the batch, we will take the input vector x_k to be

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

We notice that this will imply that h_k^j, c_k^j in Eq. (3.9) 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. (3.8) 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. (3.5) in a tensorized form. From a computational perspective, we avoid looping over each sampled trajectory to compute the numerical cost function (3.5). 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 [38, 1].

The final algorithm that uses DFP for the closed-loop Nash equilibrium problem for SDDGs is shown in Algorithm 3 below.

Algorithm 3 A Deep Fictitious Play Algorithm via Direct Parametrization for SDDG

Initialize each $\vartheta_{1,0}, \cdots, \vartheta_{N,0}$ which are the respective parameters of the N different LSTMs at stage 0.
Select N_{stages} of deep fictitious play based on the computational budget.
for m in 0 to N_{stages} **do**
    for i in 1 to N **do**
        Compute N_{batch} trajectories $\hat{\boldsymbol{X}}$ of the numerical SDDE (3.4) under the given controls $(\hat{\alpha}^{j,m})_j$, where the controls $\hat{\alpha}^{j,m} = \phi_{LSTM}(\cdot; \vartheta_{j,m})$ are defined by Eqs. (3.8)–(3.9) for each player j.
        Compute the numerical cost $\hat{J}^i$ from Eq. (3.5).
        Compute via automatic differentiation $\nabla_{\vartheta_{i,m}} \hat{J}^i$.
        Do a gradient descent step or similar (e.g. Adam) on $\vartheta_{i,m}$ with learning rate l_r, i.e. $\vartheta_{i,m+1} = \vartheta_{i,m} - l_r \nabla_{\vartheta_{i,m}} \hat{J}^i$.
    end for
end for

3.3 Three Games with Delay

In this section, we present three SDDGs– two of which are newly consider problems which we will fully motivate and solve later in Chapter 4 and one of which is solved in [11]. In Section 3.3.1 and 3.3.2, we present and summarize the results for these two new model problems inspired by [35, 4, 29, 30].

For clarity, we shall present the mathematical formulations and highlight analytical results below and defer modeling motivations and intuitions for these new problems to Section 4.1 and the proofs of their solutions to Section 4.2. In Section 3.3.3, we briefly review a stochastic delay differential game arising from inter-bank lending, as discussed in [11], and summarize the results therein. All three problems will serve as benchmarks for the numerical methodology we introduced in Section 3.2.

3.3.1 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 [35] and [30], and the full intuition and derivations are elaborated in Section 4.1.

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 $i \in \{1, \cdots, N\}$ are given by

$$\begin{aligned} \mathrm{d}X_t^i &= \left[(\mu_1 - r)\pi_t^i X_t^i + rX_t^i - \mu_2 Y_t^i\right] \mathrm{d}t + \sigma \pi_t^i X_t^i \, \mathrm{d}W_t, \ t \in (0, T], \\ Y_t^i &= \int_{-\infty}^t \lambda e^{-\lambda(t-s)} X_s^i \, \mathrm{d}s, \ t \in (0, T], \\ X_t^i &= \zeta^i(t), \ t \in (-\infty, 0]. \end{aligned} \tag{3.10}$$

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 which will be explained in Section 4.1.1. 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. (3.10) 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

$$J^i[\boldsymbol{\pi}] = \mathbb{E}\left[U_i\left(Z^i_{disc,T} - \theta_i \overline{Z}_{disc,T}\right)\right], \tag{3.11}$$

where $0 < \theta_i < 1$, and

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

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

$$\begin{aligned} Z^i_t &= X^i_t + aY^i_t, \qquad \text{where } a = \frac{-(r+\lambda) + \sqrt{(r+\lambda)^2 - 4\lambda\mu_2}}{2\lambda}, \\ Z^i_{disc,t} &= e^{-(r+\lambda a)t} Z^i_t. \end{aligned} \tag{3.12}$$

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

$$J^i[\boldsymbol{\pi}] = \mathbb{E}\left[U_i\left(Z^i_{disc,T}\overline{Z}^{-\theta_i}_{disc,T}\right)\right], \tag{3.13}$$

where $0 < \theta_i < 1$ and

$$\overline{Z}_{disc,T} = \left(\prod_{i=1}^{N} Z^i_{disc,T}\right)^{1/N}, \qquad U_i(z) = \begin{cases} \frac{1}{1-\frac{1}{\delta_i}} z^{1-\frac{1}{\delta_i}}, & \delta_i \neq 1, \\ \log(z), & \delta_i = 1, \end{cases} \tag{3.14}$$

with $\delta_i > 0$ and Z^i_{disc} defined by Eq. (3.12).

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 Eq. (3.11) and Eq. (3.13) have meaningful interpretations, as discussed in Sections 4.1.2 and 4.1.4, respectively.

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

$$\mathbb{A}^i = \left\{ \mathcal{F}^{\boldsymbol{X}}_t\text{−prog. meas. } \pi^i : [0,T] \times \Omega \to \mathbb{R} \;\middle|\; \exists K > 0 : |\pi^i_t X^i_t| \leq K|Z^i_t| \right\}, \tag{3.15}$$

where $\mathcal{F}^{\boldsymbol{X}}_t$ is the filtration generated by $(X^1_{(-\infty,t]}, \cdots, X^N_{(-\infty,t]})$. 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. (3.13)–(3.14) is well defined. This is shown in Section 4.2.2.

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 (3.10) with reward for each player $i \in \{1, \cdots, N\}$ given by $J^i = J^i[\boldsymbol{\pi}]$ as defined through Eq. (3.11)–(3.12). Then, there is a closed-loop Nash equilibrium $\boldsymbol{\pi}^*$ given by the controls*

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

where $\bar{\delta} = \frac{1}{N} \sum_{i=1}^{N} \delta_i$, $\bar{\theta} = \frac{1}{N} \sum_{i=1}^{N} \theta_i$, and $X_t^{i,*}$ satisfies the dynamics in (3.10) 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 Section 4.2.1. □

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 (3.10), the reward $J^i = J^i[\boldsymbol{\pi}]$ for each player $i \in \{1, \cdots, N\}$ defined through Eq. (3.12)–(3.14), and the admissible space $\otimes_{i=1}^{N} \mathbb{A}^i$ given by Eq (3.15). 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 $\bar{\delta} = \frac{1}{N} \sum_{i=1}^{N} \delta_i$ and $\overline{\theta(\delta - 1)} = \frac{1}{N} \sum_{i=1}^{N} \theta_i(\delta_i - 1)$.

Proof. See Section 4.2.2. □

3.3.2 Consumption and Portfolio Allocation Game with Delayed Tax Effects

In addition to the delay effects in investment strategies as analyzed in Section 3.3.1, here we also consider players' consumption strategies which contribute to their relative utility in the reward. Such a problem without delay was studied in [29].

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. (3.2), we require that $c_t^i \geq 0$ for all $t \in (0, T]$. In this case, the wealth dynamics for player $i \in \{1, \cdots, N\}$ are given by

$$\begin{aligned} \mathrm{d}X_t^i &= \left[(\mu_1 - r)\pi_t^i X_t^i + rX_t^i - \mu_2 Y_t^i - c_t^i X_t^i\right] \mathrm{d}t + \sigma \pi_t^i X_t^i \, \mathrm{d}W_t, \ t \in (0, T], \\ X_t^i &= \zeta^i(t), \ t \in (-\infty, 0], \end{aligned} \tag{3.16}$$

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 3.3.1. Again, the length of delay is $\tau = \infty$ as Y_t^i in Eq. (3.16) depends on the entire path $X_{(-\infty,t]}^i$.

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

$$J^i[\boldsymbol{\pi}, \boldsymbol{c}] = \mathbb{E}\left[\int_0^T U^i(C_{disc,t}^i \overline{C_{disc,t}}^{-\theta_i}) \, \mathrm{d}t + \epsilon_i U_i\left(Z_{disc,T}^i \overline{Z_{disc,T}}^{-\theta_i}\right)\right], \tag{3.17}$$

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

$$\begin{aligned} Z_t^i &= X_t^i + aY_t^i, & a &= \frac{-(r+\lambda) + \sqrt{(r+\lambda)^2 - 4\lambda\mu_2}}{2\lambda}, \\ C_{disc,t}^i &= e^{-(r+\lambda a)t} c_t^i X_t^i, & \overline{C_{disc,t}} &= \left(\prod_{i=1}^N C_{disc,t}^i\right)^{1/N}, \\ Z_{disc,t}^i &= e^{-(r+\lambda a)t} Z_t^i, & \overline{Z_{disc,t}} &= \left(\prod_{i=1}^N Z_{disc,t}^i\right)^{1/N}, \end{aligned}$$

and the utility function is the constant relative risk aversion (CRRA) utility given by

$$U_i(z) = \begin{cases} \frac{1}{1-\frac{1}{\delta_i}} z^{1-\frac{1}{\delta_i}}, & \delta_i \neq 1, \\ \log(z), & \delta_i = 1, \end{cases} \tag{3.18}$$

with $\delta_i > 0$.

In this case, we can again interpret $r+\lambda a$ as the "tax-adjusted risk-free rate" and Z_t^i as "tax-adjusted wealth", and the interpretation of the expected utility (3.17) is discussed in Section 4.1.5. This example mainly contrasts with that in Section 3.3.1 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. (3.17)–(3.18) is well defined, we require that the controls for player i, (π^i, c^i), live in the admissible set $\mathbb{A}^i$:

$$\begin{aligned}\mathbb{A}^i = \Big\{ &\mathcal{F}_t^{\boldsymbol{X}}\text{−prog. meas. } (\pi^i, c^i) : [0,T]\times\Omega \to \mathbb{R}\times\mathbb{R}^+ \Big| \\ &\exists K>0 : |\pi_t^i X_t^i|, |c_t^i X_t^i| \le K|Z_t^i| \Big\},\end{aligned} \tag{3.19}$$

where $\mathcal{F}_t^{\boldsymbol{X}}$ is the filtration generated by $(X^1_{(-\infty,t]}, \cdots, X^N_{(-\infty,t]})$. With $(\pi^i, c^i) \in \mathbb{A}^i$ and taking the initial path ζ^i chosen such that $Z_t^i = X_t^i + aY_t^i > 0$ for all $t \le 0$, we can show for all i, $C^i_{disc,t}, Z^i_{disc,t} > 0$ a.s. (see details in Section 4.2.3).

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 (3.16), the reward $J^i = J^i[\boldsymbol{\pi}, \boldsymbol{c}]$ for each player $i \in \{1, \cdots, N\}$ defined through Eqs. (3.17)–(3.18), and the admissible space $\otimes_{i=1}^N \mathbb{A}^i$ defined by Eq (3.19). Then, there is a closed-loop Nash equilibrium $(\boldsymbol{\pi}^*, \boldsymbol{c}^*)$ given by the controls*

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

$$c_t^{i,*} = \begin{cases} \left(\beta_i^{-1} + (\gamma_i^{-1} - \beta_i^{-1})e^{-\beta_i(T-t)}\right)^{-1}\frac{X_t^i+aY_t^i}{X_t^i}, & \delta_i \neq 1, \\ \left(T - t - \gamma_i^{-1}\right)^{-1}\frac{X_t^i+aY_t^i}{X_t^i}, & \delta_i = 1, \end{cases}$$

where $\bar{\delta} = \frac{1}{N}\sum_{i=1}^N \delta_i$, $\overline{\theta(\delta-1)} = \frac{1}{N}\sum_{i=1}^N \theta_i(\delta_i - 1)$ *and the parameters* β_i *and* γ_i *are given*

by

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

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

Proof. See Section 4.2.3. □

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, Fouque, Mousavi, and Sun [11]. 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:

$$\begin{aligned} \mathrm{d}X_t^i &= (\alpha_t^i - \alpha_{t-\tau}^i)\mathrm{d}t + \sigma \mathrm{d}W_t^i, \qquad && t \in [0,T], \\ \alpha_t^i &= 0 \in \mathbb{R}, && t \in [-\tau, 0], \\ X_0^i &= \xi^i \in \mathbb{R}. \end{aligned} \tag{3.20}$$

The cost function for bank i is given by

$$J^i[\boldsymbol{\alpha}] = \mathbb{E}\left[\int_0^T \left(\frac{1}{2}(\alpha_t^i)^2 - q\alpha_t^i(\bar{X}_t - X_t^i) + \frac{\epsilon}{2}(\bar{X}_t - X_t^i)^2\right)\mathrm{d}t + \frac{c}{2}(\bar{X}_T - X_T^i)^2\right]. \tag{3.21}$$

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. (3.21). 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 $q\alpha_t^i(\bar{X}_t - X_t^i)$, $\frac{\epsilon}{2}(\bar{X}_t - X_t^i)^2$, and $\frac{c}{2}(\bar{X}_T - X_T^i)^2$ in Eq. (3.21). 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, and 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, all else equal, to avoid lending or borrowing as represented by the quadratic penalty $\frac{1}{2}(\alpha_t^i)^2$ in Eq. (3.21).

The closed-loop Nash equilibrium for the stochastic delay differential game (3.20)–(3.21) is derived and proven in [11]. The result is restated in the proposition below for convenience.

Proposition 3.4 ([11, Proposition 6.1])**.** *Consider the stochastic differential game with delay defined by the dynamics (3.20) and with the reward for each player $i \in \{1, \cdots, N\}$ given by $J^i = J^i[\boldsymbol{\alpha}]$ as defined through Eq. (3.21). Then, there exists a closed-loop Nash equilibrium $\boldsymbol{\alpha}^*$ given by the control for each player $i \in \{1, \cdots, N\}$ by*

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

where $\overline{\alpha^*} = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N}\alpha^{i,*}$, and where $E_0, \cdots, E_2$ are given by the following PDE system in

the region $(t,s,r) \in [0,T] \times [-\tau,0] \times [-\tau,0]$*:*

$$E_0'(t) + \frac{\epsilon}{2} = 2\left(1 - N^{-2}\right)\left(E_1(t,0) + E_0(t)\right)^2 + 2q\left(E_1(t,0) + E_0(t)\right) + \frac{q^2}{2},$$

$$\partial_t E_1(t,s) - \partial_s E_1(t,s) =$$

$$2\left(1 - N^{-2}\right)\left(E_1(t,0) + E_0(t) + \frac{q}{2\left(1 - N^{-2}\right)}\right)\left(E_2(t,s,0) + E_1(t,s)\right),$$

$$\partial_t E_2(t,s,r) - \partial_s E_2(t,s,r) - \partial_r E_2(t,s,r) =$$

$$2\left(1 - N^{-2}\right)\left(E_2(t,s,0) + E_1(t,s)\right)\left(E_2(t,r,0) + E_1(t,r)\right),$$

with boundary conditions given by

$$E_0(T) = \frac{c}{2}, \quad E_1(T,s) = 0, \quad E_2(T,s,r) = 0, \quad E_2(t,s,r) = E_2(t,r,s),$$

$$E_1(t,-\tau) = -E_0(t), \quad E_2(t,s,-\tau) = -E_1(t,s).$$

This Nash equilibrium is derived in [11] 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 [15].

3.4 Numerical Results

In Section 3.3, we have presented three SDDGs 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 3 in Section 3.2.

We introduce below in Section 3.4.1 the precise details of our numerical experiments that serve as a reference point for the construction of the plots shown in Sections 3.4.2–3.4.4, the parameter values used for each of these problems, and an important implementation detail for the problems with infinite delay. In Sections 3.4.2–3.4.4 we present and

interpret the numerical results for each of the considered problems.

3.4.1 Numerical Results Methodology

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 (3.5), 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 3, we compute the 2-norm relative error:

$$\text{Relative 2-Norm Error} = \frac{||\hat{\boldsymbol{J}}[\boldsymbol{\phi}_{LSTM}] - \hat{\boldsymbol{J}}[\boldsymbol{\alpha}^*]||_2}{||\hat{\boldsymbol{J}}[\boldsymbol{\alpha}^*]||_2}, \tag{3.22}$$

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. (3.8)–(3.9) and the true Nash equilibrium controls $(\alpha^{*1}, \cdots, \alpha^{*N})$ of the mathematical problem defined by Eqs. (3.1),(3.3). 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 3.4.2–3.4.4.

Comparison of State and Control Trajectories

After training, we have the collection $(\phi_{LSTM}(\cdot;\vartheta_i))_{i=1}^N$ of LSTM control functions for each player. 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 (3.4) 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 3.4.2–3.4.4. 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.

Model Parameters

The model parameters chosen for the numerical experimentation of each problem are shown below. We express a dependency on i for parameters specific to player i, e.g. $\delta_i = \frac{3}{10} + \frac{4}{9}(i-1)$ in Table 3.1 is player i's risk tolerance parameter. Here, we are still using the indexation $i \in \{1, \cdots, N = 10\}$.

Table 3.1: 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	$X^i_{(-\infty,0]} = x_0^i$
10	10.0	0.08	0.2	0.04	2.0	0.01	$\frac{3}{10} + \frac{4}{9}(i-1)$	$\frac{3}{10} + \frac{4}{9}(i-1)$	$2 + \frac{1}{10}(i-1)$

Table 3.2: 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	$X^i_{(-\infty,0]} = x_0^i$
10	1.0	0.08	0.2	0.04	1.0	0.2	$\frac{3}{10} + \frac{4}{9}(i-1)$	$\frac{3}{10} + \frac{4}{9}(i-1)$	$1 + \frac{1}{20}(i-1)$

Note that in Tables 3.1–3.3, we write $X^i_{(-\infty,0]} = x_0^i$, indicating the initial path for each player is taken to be constant. Table 3.3 is split in two parts for page size considerations.

Table 3.3: Parameters for consumption and portfolio allocation game with delayed tax effects.

N	T	μ_1	σ	r	λ	μ_2	ϵ_i
10	2.0	0.08	0.2	0.04	1.0	0.01	50.0

δ_i	θ_i	$X^i_{(-\infty,0]} = x^i_0$
$\frac{3}{10} + \frac{4}{9}(i-1)$	$\frac{3}{10} + \frac{4}{9}(i-1)$	$1 + \frac{1}{20}(i-1)$

Table 3.4: 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	$1 + 0.1 \cdot 1.15^{i-1}$

Approaching the Infinite Delay Cases

We remark on an approximation used in the infinite delay cases appearing in the problems in Sections 3.3.1 and 3.3.2 whose numerical results we display in 3.4.2 and 3.4.3. 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. (3.9) to be initialized at some finite $-N_\tau$. For the numerical results shown in both Sections 3.4.2 and 3.4.3, we have used $\tau = 1.0$ as the truncation for the infinite delay.

3.4.2 Results for Competition between Portfolio Managers with Delayed Tax Effects

CARA Case

We consider now the problem of competition between portfolio managers with delay tax effects, Eq. (3.10), in the CARA case where the rewards are given by Eqs. (3.11)–(3.12). In our numerical experiment, we select the parameter values as shown in Table 3.1.

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 Figure 3.1 as explained in Section 3.4.1.

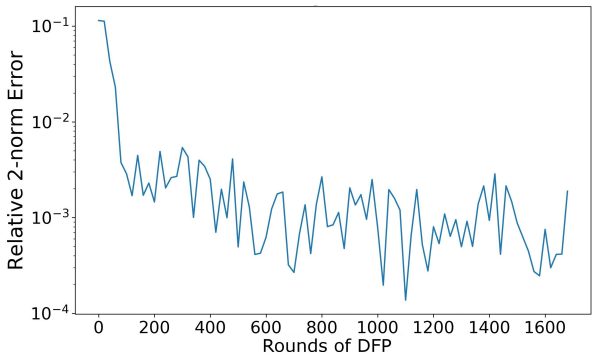

Figure 3.1: The relative 2-norm error (3.22) over the course of training between $(\hat{J}^1, \cdots, \hat{J}^N)$ under the LSTM controls and the true Nash equilibrium controls for the CARA case. We take $N_{\text{batch}} = 2^{15}$ in the computation of $(\hat{J}^1, \cdots, \hat{J}^N)$ according to Eq. (3.5). The length of training is measured in terms of rounds of DFP.

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 3.4.1, we compare the trajectories under the LSTM controls to their true Nash equilibrium counterparts, and the resulting plots are shown in Figure 3.2.

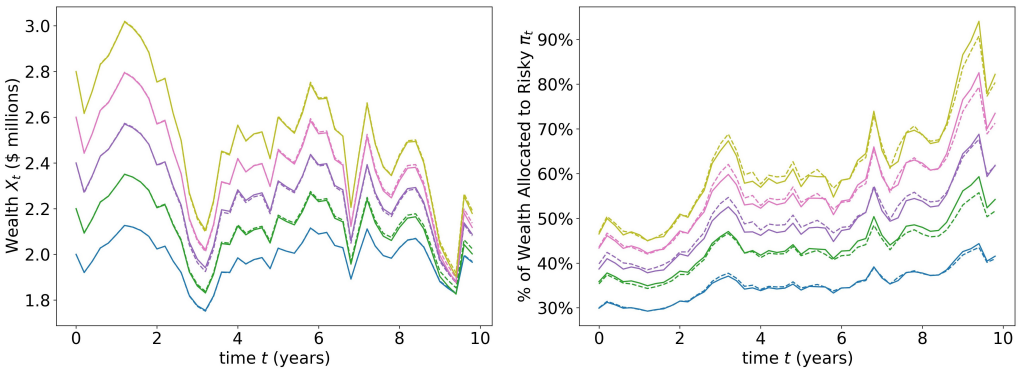

Figure 3.2: Left: A sample path of the wealth processes for players $1, 3, 5, 7$, and 9 (each player corresponds to a unique color) under the true Nash equilibrium controls (dashed) and the trained LSTM control (solid). Right: True Nash equilibrium controls (dashed) vs LSTM controls (solid). Controls represent the fraction of total wealth allocated to the risky asset at time t for a given player.

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.

CRRA Case

We now consider the problem of competition between portfolio managers with delayed tax effects, Eq. (3.10), in the CRRA case where the rewards are given by Eqs. (3.13)–(3.14). In our numerical experiments, we select the parameters for this problem as shown in Table 3.2. 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 3.4.1 for details) is examined by Figure 3.3 below.

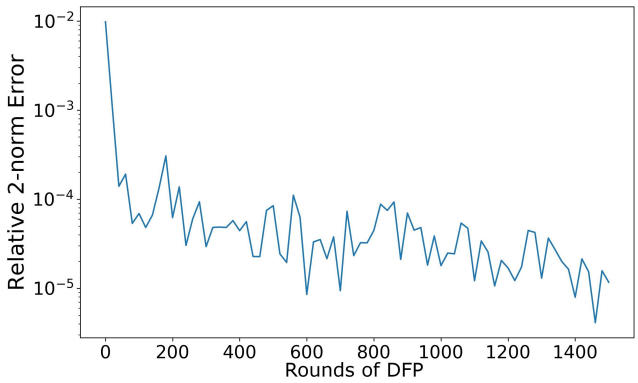

Figure 3.3: The relative 2-norm error (3.22) over the course of training between $(\hat{J}^1, \cdots, \hat{J}^N)$ under the LSTM controls and the true Nash equilibrium controls for the CRRA case. We take $N_{\text{batch}} = 2^{15}$ in the computation of $(\hat{J}^1, \cdots, \hat{J}^N)$ according to Eq. (3.5). The length of training is measured in terms of rounds of DFP.

The relative 2-norm error decreasing throughout training and plateauing 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. Figure 3.4 below allows us to compare the paths induced by these trained controls to their true Nash equilibrium counterparts.

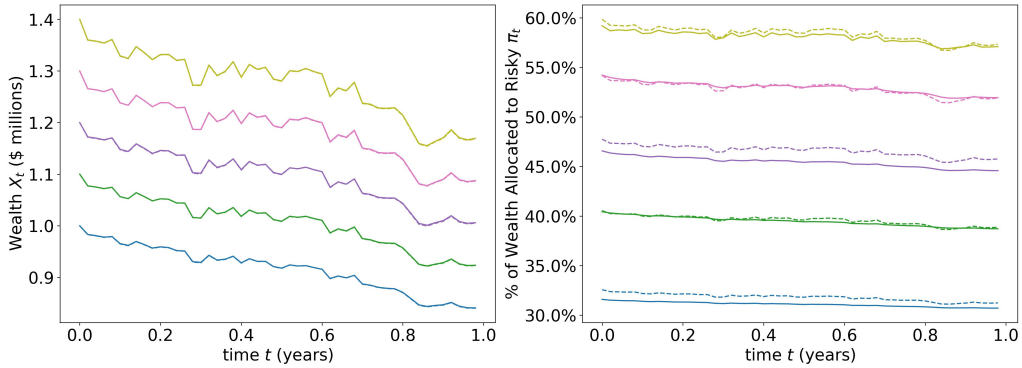

Figure 3.4: Left: A sample path of the wealth processes for players $1, 3, 5, 7$, and 9 (each player corresponds to a unique color) under the true Nash equilibrium controls (dashed) and the trained LSTM control (solid). Right: True Nash equilibrium controls (dashed) vs LSTM controls (solid). Controls represent the fraction of total wealth allocated to the risky asset at time t for a given player.

As indicated by Figure 3.4, 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.

3.4.3 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. (3.16)–(3.18). For our numerical experiments, we select the value of the parameters as shown in Table 3.3. 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 3.4.1, the successive approximation of the Nash equilibrium rewards over training is illustrated by Figure 3.5.

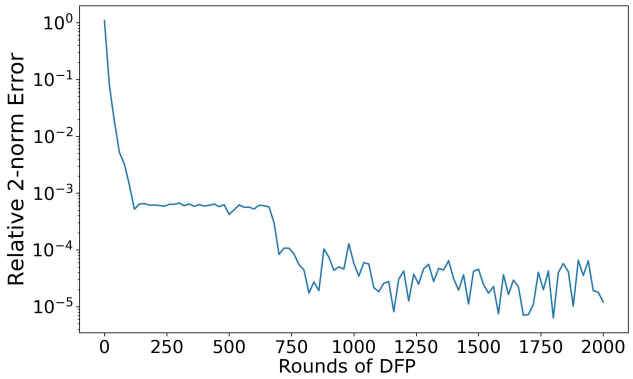

Figure 3.5: The relative 2-norm error (3.22) over the course of training between $(\hat{J}^1, \cdots, \hat{J}^N)$ under the LSTM controls and the true Nash equilibrium controls. We take $N_{\text{batch}} = 2^{15}$ in the computation of $(\hat{J}^1, \cdots, \hat{J}^N)$ according to Eq. (3.5). The length of training is measured in terms of rounds of DFP.

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 Figure 3.6. 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 Figure 3.6 their corresponding unnormalized versions which are the wealth allocated to the stock and the annualized run rate of wealth consumed respectively.

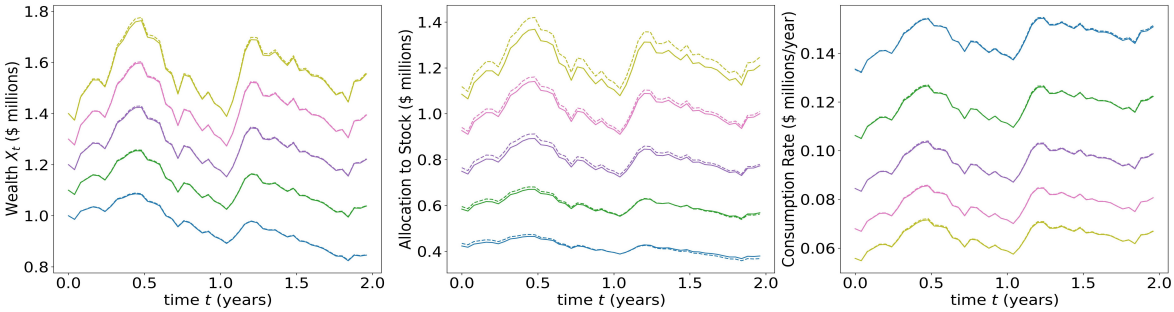

Figure 3.6: Left: A sample path of the wealth processes of players $1, 3, 5, 7$, and 9 under the true Nash equilibrium controls (dashed) and the trained LSTM control (solid) of the corresponding players. Center: Nash equilibrium wealth allocation to stock (dashed) vs LSTM allocation (solid). Right: Nash equilibrium total consumption rate (dashed) vs LSTM consumption rate (solid).

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.

3.4.4 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. (3.20)–(3.21). The parameter choices are summarized in Table 3.4.

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 Figure 3.7.

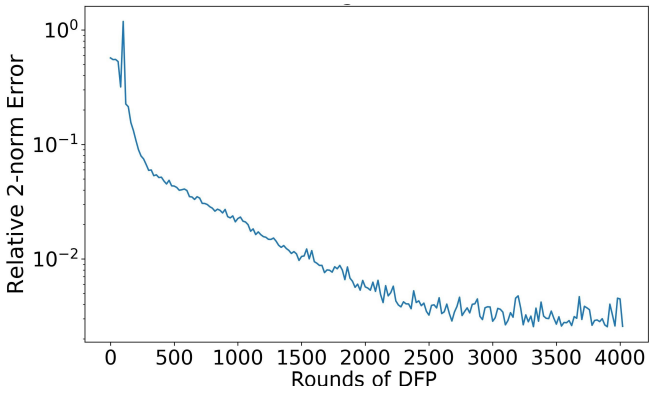

Figure 3.7: The relative 2-norm error (3.22) over the course of training between $(\hat{J}^1, \cdots, \hat{J}^N)$ under the LSTM controls and the true Nash equilibrium controls for the inter-bank lending model. We take $N_{\text{batch}} = 2^{12}$ in the computation of $(\hat{J}^1, \cdots, \hat{J}^N)$ according to Eq. (3.5). The length of training is measured in terms of rounds of DFP.

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).[2] This additional source of error from the PDE approximation is likely the cause of the increased smoothness of the curve in Figure 3.7 and the slightly higher levels of relative error compared to previous cases. 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, Figure 3.8, is the foremost illustration showcasing the success of the numerical algorithm for this particular example.

[2]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.

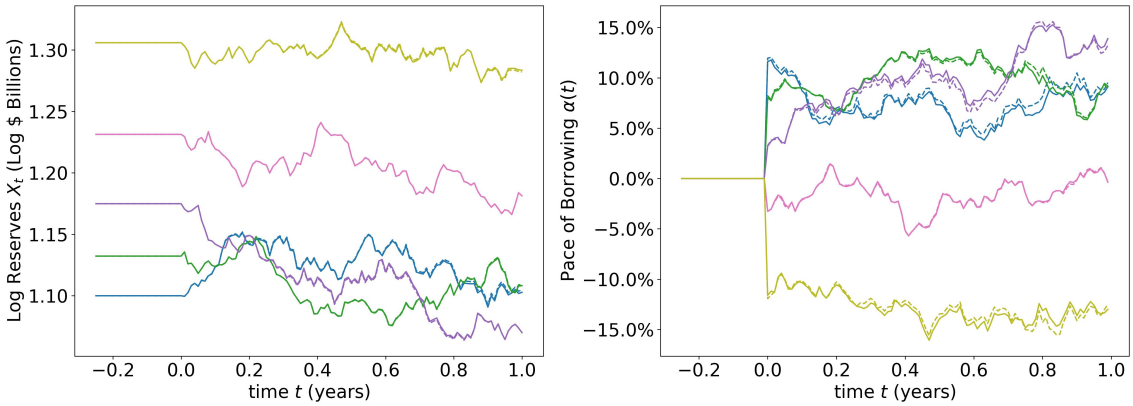

Figure 3.8: Left: A sample path of the log-monetary reserves of banks $1, 3, 5, 7$, and 9 under the true Nash equilibrium controls (dashed) and the trained LSTM control (solid). Right: The corresponding paths (with respect to the left picture) of Nash equilibrium controls (dashed) and LSTM controls (solid) 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.

Chapter 4

Financial Modeling of Portfolio Games with Delayed Tax Effects

The work described in this chapter, which was prepared for this dissertation, contains material that has already been submitted for publication [3].

In this chapter, we elaborate on the newly considered problems featured in Sections 3.3.1 and 3.3.2, explaining their formulation and the intuition behind the models in Section 4.1, while the proofs of solution for each of the considered problems is given in Section 4.2.

4.1 Interpretation and Derivations of the Portfolio Games with Delayed Tax Effects

4.1.1 Portfolio Optimization with Delayed Taxes Effects

We start with restating the original Merton problem [33]. 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 $\frac{\mathrm{d}S_t}{S_t} = \mu\,\mathrm{d}t + \sigma\,\mathrm{d}W_t$. Denoting by X_t the investor's wealth at time t, one then has

$$\mathrm{d}X_t = [(\mu - r)\pi_t X_t + rX_t]\,\mathrm{d}t + \sigma\pi_t X_t\,\mathrm{d}W_t,\ t \in (0, T]. \tag{4.1}$$

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

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

subject to the dynamics (4.1). 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. (4.1). 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

$$\mathrm{d}X_t = ((\mu_1 - r)\pi_t X_t + rX_t - \mu_2 Y_t)\,\mathrm{d}t + \sigma\pi_t X_t\,\mathrm{d}W_t,\ t \in (0, T]. \tag{4.2}$$

This model with $\mu_2 < 0$ was introduced and solved in [35] 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's 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 $\lambda > 0$, 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. (4.2). 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. (4.2) 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 Section 4.1.2. 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 Section 4.1.2 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

$$J[\pi] = \mathbb{E}[U(Z_T)], \tag{4.3}$$

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

Lastly, we mention that the stochastic control problem with such delay structure (4.2) and $\mu_2 < 0$ was considered in [35], including notably the form of the utility (4.3) 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.

4.1.2 Tax-Adjusted Wealth and the Tax-Adjusted Risk-Free Rate

We have discussed the motivation and interpretation of Eqs. (4.2)–(4.3), 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 $\pi_t \equiv 0$:

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

Recall that $Y_t = \int_{-\infty}^{t} \lambda e^{-\lambda(t-s)} X_s \,\mathrm{d}s$ and $\mathrm{d}Y_t = \lambda(X_t - Y_t)\,\mathrm{d}t$, then 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 $c_+ = \frac{a_-}{a_- - a_+}(X_0 + a_+ Y_0)$ and $c_- = \frac{-a_+}{a_- - a_+}(X_0 + a_- Y_0)$. 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 $|r| << \lambda$. 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.

4.1.3 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 [4, 29, 30]. We will briefly summarize them for convenience as they inspire the form of the new problems we had considered in Sections 3.3.1–3.3.2. The motivation [4] 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. [4, 29, 30] 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 $(1-\theta_i)X_T^i + \theta_i(X_T^i - \overline{X}_T) = X_T^i - \theta_i \overline{X}_T$ 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, [30] consider the constant absolute risk aversion (CARA) case, and the reward for player i is given by

$$J^i = E[U_i(X_T^i - \theta_i \overline{X}_T)], \quad U_i(z) = -\exp\left(-\frac{1}{\delta_i} z\right), \tag{4.4}$$

where $\delta_i > 0$ is the risk tolerance of manager i. They consider constant relative risk aversion (CRRA) utilities

$$U_i(z) = \begin{cases} \frac{1}{1-\frac{1}{\delta_i}} z^{1-\frac{1}{\delta_i}}, & \delta_i > 0, \delta_i \neq 1, \\ \log(z), & \delta_i = 1, \end{cases} \tag{4.5}$$

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

The CRRA case is also considered in [30]. In this case, for tractability, the competing managers compare relative performance to absolute performance modeled by the ratio $\frac{X_T^i}{(\overline{X}_T)^{\theta_i}}$, where $\overline{X} = (\prod_{i=1}^N X^i)^{1/N}$ is the geometric average. Then we can say manager i seeks to maximize her expected utility given by

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

4.1.4 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. (4.2), and given by

$$\mathrm{d}X_t^i = \left((\mu_1 - r)\pi_t^i X_t^i + rX_t^i - \mu_2 Y_t^i\right)\,\mathrm{d}t + \sigma\pi_t^i X_t^i\,\mathrm{d}W_t,\ t \in (0,T], \tag{4.6}$$

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 (4.4) and the CRRA case (4.5), but replace the terminal wealth with the discounted, tax-adjusted terminal wealth. To recall the discussion in Section 4.1.1 and 4.1.2, 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

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

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. (4.5), one has

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

where in this case $\overline{Z}_{disc,T} = \left(\prod_{i=1}^N Z_{disc,T}^i\right)^{1/N}$.

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 Section 4.1.3 in determining her utility.

4.1.5 Consumption and Portfolio Allocation Game with Delayed Tax Effects

We further extend the modeling (4.6) 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

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

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 [29], which has the utility of consumption given by $U^i((c_t^i X_t^i)(\overline{cX}_t)^{-\theta_i})$ with $\overline{cX}_t = (\Pi_{i=1}^N (c_t^i X_t^i))^{1/N}$. 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 $i \in \{1, \cdots, N\}$ is given by

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

where

$$C^i_{disc,t} = e^{-(r+\lambda a)t} c^i_t X^i_t, \quad \overline{C_{disc,t}} = \Big(\prod_{i=1}^N C^i_{disc,t}\Big)^{1/N},$$

$$Z^i_{disc,t} = e^{-(r+\lambda a)t} Z^i_t, \qquad \overline{Z_{disc,t}} = \Big(\prod_{i=1}^N Z^i_{disc,t}\Big)^{1/N},$$

and

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

Here $Z^i_{disc,t}$ is the tax-adjusted wealth for player i discounted according to the tax-adjusted risk-free rate. $C^i_{disc,t}$ 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.

4.2 Proofs of Nash Equilibrium Solutions

4.2.1 Proof of Proposition 3.1

For convenience, we restate the dynamics (3.10), which is

$$\mathrm{d}X^i_t = \big((\mu_1 - r)\pi^i_t X^i_t + r X^i_t - \mu_2 Y^i_t\big)\,\mathrm{d}t + \sigma \pi^i_t X^i_t\,\mathrm{d}W_t, \ t \in (0, T],$$

$$X^i_t = \zeta^i(t), \ t \in (-\infty, 0],$$

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 [35], we note that differentiating Y_t^i gives the following differential relation

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

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

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

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

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

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

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

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

$$\tilde{\pi}_t^i = \pi_t^i X_t^i e^{-(r+\lambda a)t}. \tag{4.7}$$

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

$$\mathrm{d}Z_{disc,t}^i = (\mu_1 - r)\tilde{\pi}_t^i\,\mathrm{d}t + \sigma\tilde{\pi}_t^i\,\mathrm{d}W_t, \quad t \in (0, T]. \tag{4.8}$$

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

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

where

$$\overline{Z}_{disc,T} = \frac{1}{N}\sum_{i=1}^N Z_{disc,T}^i, \quad U_i(z) = -\exp\left(-\frac{1}{\delta_i}z\right). \tag{4.9}$$

Therefore, the characterization of controls for the Nash equilibrium can be considered through Eqs. (4.8)–(4.9). This is precisely the same problem described in [30, Section 2, Corollary 4], which gives a closed-loop Nash equilibrium given by the controls

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

for $i \in \{1, \cdots, N\}$, where $\bar{\delta} = \frac{1}{N}\sum_{i=1}^N \delta_i$ and $\bar{\theta} = \frac{1}{N}\sum_{i=1}^N \theta_i$. Substituting $\tilde{\pi}_t^{i,*}$ into Eq. (4.7), we get that there is a closed-loop Nash equilibrium control given by the choices of each player i by

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

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.

4.2.2 Proof of Proposition 3.2

Since the dynamical system is exactly the same as in Section 4.2.1, we know that $Z_t^i = X_t^i + aY_t^i$ satisfies

$$\mathrm{d}Z_t^i = \left((\mu_1 - r)\pi_t^i X_t^i + (r+\lambda a)Z_t^i\right)\mathrm{d}t + \sigma \pi_t^i X_t^i \,\mathrm{d}W_t. \tag{4.10}$$

We are assuming that the initial path $X_t^i = \zeta^i(t)$ for $t \le 0$ is chosen so that $Z_0^i = X_0^i + aY_0^i > 0$.

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

$$\tilde{\pi}_t^i = \begin{cases} \pi_t^i \frac{X_t^i}{Z_t^i}, & Z_t^i \neq 0, \\ 0, & Z_t^i = 0. \end{cases} \tag{4.11}$$

Since $|\pi_t^i X_t^i| \le K|Z_t^i|$, we have $|\tilde{\pi}_t^i| \le K$ and claim that the dynamics (4.10) can be

written in terms of $\tilde{\pi}_t^i$ by

$$\mathrm{d}Z_t^i = \left((\mu_1 - r)\tilde{\pi}_t^i Z_t^i + (r + \lambda a)Z_t^i\right)\,\mathrm{d}t + \sigma\tilde{\pi}_t^i Z_t^i\,\mathrm{d}W_t. \tag{4.12}$$

Since $|\tilde{\pi}_t^i| \leq K$, we have that $\mathbb{E}\left[\int_0^t (\tilde{\pi}_s^i)^2\,\mathrm{d}s\right] < \infty$ for each $t \in (0, T]$ and therefore, Z_t^i has the unique solution

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

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. (4.11) can be written simply as

$$\tilde{\pi}_t^i = \pi_t^i \frac{X_t^i}{Z_t^i}, \tag{4.13}$$

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

Now, multiplying Eq. (4.12) by the integrating factor $e^{-(r+\lambda a)t}$, we can write the equation for $Z^i_{disc,t} = e^{-(r+\lambda a)t} Z_t^i$ as

$$\mathrm{d}Z^i_{disc,t} = (\mu_1 - r)\tilde{\pi}_t^i Z^i_{disc,t}\,\mathrm{d}t + \sigma\tilde{\pi}_t^i Z^i_{disc,t}\,\mathrm{d}W_t, \quad t \in (0, T]. \tag{4.14}$$

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

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

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

$$\overline{Z}_{disc,T} = \left(\prod_{i=1}^{N} Z^i_{disc,T} \right)^{1/N}, \qquad U_i(z) = \begin{cases} \frac{1}{1-\frac{1}{\delta_i}} z^{1-\frac{1}{\delta_i}}, & \delta_i \neq 1, \\ \log(z), & \delta_i = 1, \end{cases} \tag{4.15}$$

with $\delta_i > 0$. The Nash equilibrium problem defined by Eqs. (4.14)–(4.15) for generic, progressively measurable controls $(\tilde{\pi}^i)_{i=1}^N$ such that $\mathbb{E}\left[\int_0^t (\tilde{\pi}^i_s)^2 \, \mathrm{d}s\right] < \infty$ falls into the formulation of [30, Section 3, Corollary 15]. Using the results therein, one has a closed-loop Nash equilibrium given by the controls

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

for $i \in \{1, \cdots, N\}$. Solving for $\pi^{i,*}_t$, from Eq. (4.13) we get

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

which holds as we have shown that $X^i_t > 0$. Lastly, we see that the Nash equilibrium controls do satisfy the admissibility condition as $|\pi^{i,*}_t X^i_t| = \left| \frac{\mu_1 - r}{\sigma^2} \left(\delta_i - \frac{\theta_i(\delta_i - 1)\bar{\delta}}{1 + \theta(\delta - 1)} \right) \right| |Z^i_t|$. This completes the proof of Proposition 3.2.

4.2.3 Proof of Proposition 3.3

We now form the solution for the problem defined by Eq. (3.16)–(3.18), proceeding similarly to the proof in Section 4.2.2. Restating the dynamics given in Proposition 3.3, we have for each $i \in \{1, \cdots, N\}$,

$$\begin{aligned} \mathrm{d}X^i_t &= \left((\mu_1 - r)\pi^i_t X^i_t + r X^i_t - \mu_2 Y^i_t - c^i_t X^i_t \right) \mathrm{d}t + \sigma \pi^i_t X^i_t \, \mathrm{d}W_t, \ t \in (0, T], \\ X^i_t &= \zeta^i(t), \ t \in (-\infty, 0], \end{aligned}$$

where the delay variable, Y^i_t, is given by $Y^i_t = \int_{-\infty}^t \lambda e^{-\lambda(t-s)} X^i_s \, \mathrm{d}s$. As usual, define $Z^i_t = X^i_t + aY^i_t$ and $Z^i_{disc,t} = e^{-(r+\lambda a)t} Z^i_t$, where $a = \frac{-(r+\lambda) + \sqrt{(r+\lambda)^2 - 4\lambda\mu_2}}{2\lambda}$. Following the

similar computation in the proof in Section 4.2.1, one can show

$$\mathrm{d}Z_t^i = \big((r+\lambda a)Z_t^i + (\mu_1 - r)\pi_t^i X_t^i - c_t^i X_t^i\big)\,\mathrm{d}t + \sigma\pi_t^i X_t^i\,\mathrm{d}W_t. \tag{4.16}$$

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

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

Then, Eq. (4.16) can be written as

$$\mathrm{d}Z_t^i = \big((r+\lambda a)Z_t^i + (\mu_1 - r)\tilde{\pi}_t^i Z_t^i - \tilde{c}_t^i Z_t^i\big)\,\mathrm{d}t + \sigma\tilde{\pi}_t^i Z_t^i\,\mathrm{d}W_t. \tag{4.17}$$

Since $|\tilde{\pi}_t^i|, |\tilde{c}_t^i| \leq K$, we have that $\mathbb{E}\left[\int_0^t (\tilde{\pi}_s^i)^2 + (\tilde{c}_s^i)^2\,\mathrm{d}s\right] < \infty$ for each $t \in (0,T]$, and Z_t^i has the unique solution

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

Therefore, $Z_t^i > 0$ and $X_t^i > 0$, following the same argument as in Section 4.2.2, and the transformed controls become simply

$$(\tilde{\pi}_t^i, \tilde{c}_t^i) = (\pi_t^i \frac{X_t^i}{Z_t^i}, c_t^i \frac{X_t^i}{Z_t^i}), \tag{4.18}$$

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

Next, multiplying Eq. (4.17) by the integrating factor $e^{-(r+\lambda a)t}$, we get

$$\mathrm{d}Z_{disc,t}^i = \big((\mu_1 - r)\tilde{\pi}_t^i Z_{disc,t}^i - \tilde{c}_t^i Z_{disc,t}^i\big)\,\mathrm{d}t + \sigma\tilde{\pi}_t^i Z_{disc,t}^i\,\mathrm{d}W_t, \quad t \in (0,T]. \tag{4.19}$$

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

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

where

$$C^i_{disc,t} = e^{-(r+\lambda a)t} c^i_t X^i_t, \quad \overline{C_{disc,t}} = \Big(\prod_{i=1}^N C^i_{disc,t}\Big)^{1/N},$$
$$Z^i_{disc,t} = e^{-(r+\lambda a)t} Z^i_t, \qquad \overline{Z_{disc,t}} = \Big(\prod_{i=1}^N Z^i_{disc,t}\Big)^{1/N},$$

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

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

with $\delta_i > 0$. In terms of the transformed controls defined by Eq. (4.18), we can write the reward function for player i as

$$J^i = E\Big[\int_0^T U^i(\tilde{c}^i_t Z^i_{disc,t}(\overline{\tilde{c}_t Z_{disc,t}})^{-\theta_i})\,\mathrm{d}t + \epsilon_i U_i\Big(Z^i_{disc,T}\overline{Z_{disc,T}}^{-\theta_i}\Big)\Big], \tag{4.20}$$

where $\overline{\tilde{c}_t Z_{disc,t}} = \big(\prod_{i=1}^N \tilde{c}^i_t Z^i_{disc,t}\big)^{1/N}$.

The Nash equilibrium problem defined by Eqs. (4.19)–(4.20) is given in [29, Corollary 2.3] for the generic, progressively measurable controls $(\tilde{\pi}^i, \tilde{c}^i)_{i=1}^N$ that satisfy the admissibility conditions $\mathbb{E}\left[\int_0^t (\tilde{\pi}^i_s)^2 + (\tilde{c}^i_s)^2\,\mathrm{d}s\right] < \infty$ and $\tilde{c}^i_t \geq 0$ for each $t \in [0,T]$. Specifically, [29, Corollary 2.3] gives that there exists a closed-loop Nash equilibrium given by the controls

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

for $i \in \{1, \cdots, N\}$, 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

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

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

Substituting back for $(\pi^{i,*}, c^{i,*})$ through the transformations defined by Eq. (4.18), we see that $(\pi^{i,*}, c^{i,*})$ is indeed in $\mathbb{A}^i$ and the result from Proposition 3.3 holds.

Chapter 5

Stochastic Differential Games for Multi-Region Epidemiological Models

The work described in this chapter, which was prepared for this dissertation, contains material that has already been published [43, 42].

In a classic, compartmental epidemiological model each individual is assigned a label, *e.g.*, **S**usceptible, **E**xposed, **I**nfectious, **R**emoved, **V**accinated. The labels' order shows the flow patterns between the compartments (SIR, SEIR, SIRV models). Other approaches include network models, which explicitly include the interaction of individuals, in addition to the modeling of each individual's dynamics, and agent-based models that are useful in informing decision making when accurately calibrated. Moreover, the consideration of pharmaceutical and/or non-pharmaceutical intervention policies naturally couples game theory to epidemiological models by controlling when and how the game is played in such models. For example, early studies such as Bauch and Earn [5] as well as Bauch, Galvani, and Earn [6], demonstrate that one can use non-repeated games to incorporate game theory into modeling at the individual level, where individuals maximize their gain by weighing the costs and benefits of different strategies. We refer to the review paper of Chang, Piraveenan, Pattison, and Prokopenko [12] and the references

therein for more details.

Building from the DFP theory and algorithms for computing Nash equilibria in SDG, we propose here to strengthen the classical SEIR model by taking into account the social and health policies issued by multiple region planners. We call this new model a stochastic multi-region SEIR model because it couples the stochastic differential game theory with the SEIR model, making it more realistic and powerful. The computational challenge introduced by the high-dimensionality of the multi-region solution space is addressed by the enhanced deep fictitious play algorithm shown in Section 5.2.1. To showcase the performance of the proposed model and algorithm, we apply them to a case study of the COVID-19 pandemic in three states: New York (NY), New Jersey (NJ), and Pennsylvania (PA), presenting the optimal lockdown policy corresponding to the Nash equilibrium of the multi-region SEIR model for this case study. This case study is analyzed for the "would-be" optimal lockdown policies during the pre-vaccination period of the COVID-19 pandemic.

5.1 A Multi-Region SEIR Model

We consider an epidemic occurring across N geographical regions, and each planner controls the loss of her region by implementing some policies. We aim to study how the region planners' policies affect each other and determine the equilibrium policies.

Let us start with a modified version of the very-known epidemic SEIR model shown in Liu, Hethcote, and Levin [31], where each region's population is assigned to compartments with four labels: **S**usceptible, **E**xposed, **I**nfectious, and **R**emoved. Individuals with different labels denote S: those who are not yet infected; E: who have been infected but are not yet infectious themselves; I: who have been infected and are capable of spreading the disease to those in the susceptible category, and R: who have been infected and then removed from the disease due to recovery or death. The region planners

can issue certain policies to mitigate the pandemic, for instance, policies that can help reduce the transmission rates and death rates. Mathematically, denote by $S_t^i, E_t^i, I_t^i, R_t^i$ the *proportion* of population in the four compartments of region i at time t. We consider the following stochastic multi-region SEIR model:

$$\mathrm{d}S_t^i = -\sum_{j=1}^{N} \beta^{ij} S_t^i I_t^j (1-\theta\ell_t^i)(1-\theta\ell_t^j)\,\mathrm{d}t - v(h_t^i) S_t^i\,\mathrm{d}t - \sigma_{s_i} S_t^i\,\mathrm{d}W_t^{s_i}, \tag{5.1}$$

$$\mathrm{d}E_t^i = \sum_{j=1}^{N} \beta^{ij} S_t^i I_t^j (1-\theta\ell_t^i)(1-\theta\ell_t^j)\,\mathrm{d}t - \gamma E_t^i\,\mathrm{d}t + \sigma_{s_i} S_t^i\,\mathrm{d}W_t^{s_i} - \sigma_{e_i} E_t^i\,\mathrm{d}W_t^{e_i}, \tag{5.2}$$

$$\mathrm{d}I_t^i = (\gamma E_t^i - \lambda(h_t^i) I_t^i)\,\mathrm{d}t + \sigma_{e_i} E_t^i\,\mathrm{d}W_t^{e_i}, \tag{5.3}$$

$$\mathrm{d}R_t^i = \lambda(h_t^i) I_t^i\,\mathrm{d}t + v(h_t^i) S_t^i\,\mathrm{d}t, \quad i \in \mathcal{N} := \{1, 2, \ldots, N\}, \tag{5.4}$$

where $\boldsymbol{\ell}_t \equiv (\ell_t^1, \ldots, \ell_t^N)$ and $\boldsymbol{h}_t \equiv (h_t^1, \ldots, h_t^N)$ are policies chosen by the region planners at time t. Each planner i seeks to minimize its region's cost within a period $[0, T]$:

$$J^i(\boldsymbol{\ell}, \boldsymbol{h}) := \mathbb{E}\bigg[\int_0^T e^{-rt} P^i\big[(S_t^i + E_t^i + I_t^i)\ell_t^i w + a(\kappa I_t^i \chi + p I_t^i c)\big] + e^{-rt}\eta(h_t^i)^2\,\mathrm{d}t\bigg]. \tag{5.5}$$

We now give detailed description of this model (5.1)–(5.5):

S: β^{ij} denotes the average number of contacts per person per time. The transition rate between S^i and E^i due to contacting infectious people in the region j is proportional to the fraction of those contacts between an infectious and a susceptible individual, which result in the susceptible one becoming infected, *i.e.*, $\beta S_t^i I_t^j$. Although some regions may not be geographically connected, the transmission between the two is still possible due to air travels but is less intensive than the transmission within the region, *i.e.*, $\beta^{ij} > 0$ and $\beta^{ii} \gg \beta^{ij}$ for all $j \neq i$.

$\ell_t^i \in [0, 1]$ denotes the decision of the planner i on the fraction of population being locked down at time t. We assume that those in lockdown cannot be infected. However, the policy may only be partially effective as essential activities (food production and distribution, health, and basic services) have to continue. Here we

use $\theta \in [0,1]$ to measure this effectiveness, and the transition rate under the policy $\boldsymbol{\ell}$ thus become $\beta^{ij} S_t^i I_t^j (1-\theta\ell_t^i)(1-\theta\ell_t^j)$. The case $\theta = 1$ means the policy is fully effective.

$h_t^i \in [0,1]$ denotes the effort the planner i decides to put into the health system, which we refer as *health policy*. It will influence the vaccination availability $v(\cdot)$ and the recovery rate $\lambda(\cdot)$ of this model.

$v(h_t^i)$ denotes the vaccination availability of region i at time t. Once vaccinated, the susceptible individuals $v(h_t^i)S_t^i$ become immune to the disease, and join the removed category R_t^i. We model it as an increasing function of h_t^i, and if the vaccine has not been developed yet, we can define $v(x) = 0$ for $x \leq \overline{h}$.

E: In Eq. (5.2), γ describes the latent period when the person has been infected but not infectious yet. It is the inverse of the average latent time, and we assume γ to be identical across all regions. The transition between E^i and I^i is proportional to the fraction of exposed, *i.e.*, γE_t^i.

I: $\lambda(\cdot)$ represents the recovery rate. For the infected individuals, a fraction $\lambda(h^i)I^i$ (including both death and recovery from the infection) joins the removed category R^i per time unit. The rate is determined by the average duration of infection. We model the duration (and hence the recovery rate) related to the health policy h_t^i decided by the region planner i. The more effort put into the region (i.e., expanding hospital capacity, creating more drive-thru testing sites), the more clinical resources the region will have and the more resources will be accessible by patients, which could accelerate the recovery and slow down death. The death rate, denoted by $\kappa(\cdot)$, is crucial for computing the cost of the region i; see the next item.

Cost: Each region planner faces four types of cost. One is the economic activity loss due to the lockdown policy, where w is the productivity rate per individual, and P^i is the population of the region i. The second one is due to the death of infected

individuals. Here κ is the death rate which we assume for simplicity to be constant, and χ denotes the cost of each death. The hyperparameter a describes how planners weigh deaths and infections comparing to other costs. The third one is the inpatient cost, where p is the hospitalization rate, and c is the cost per inpatient day. The last term $\eta(h_t^i)^2$ is the grants putting into the health system. We choose a quadratic form to account for diminishing marginal utility (view it from $\eta(h_t^i)^2$ to h_t^i). All costs are discounted by an exponential function e^{-rt}, where r is the risk-free interest rate, to take into account the time preference. Note that region i's cost depends on all regions' policies $(\boldsymbol{\ell}, \boldsymbol{h})$, as $\{I^j, j \neq i\}$ appearing in the dynamics of S^i. Thus we write the expected cost of player i as $J^i(\boldsymbol{\ell}, \boldsymbol{h})$ emphasizing its dependence on the other players' controls.

The choices of epidemiological parameters will be discussed in Section 5.3.2. Next, we summarize the key assumptions in the above model:

1. The dynamics of an epidemic are much faster than the vital (birth and death) dynamics. So vital dynamics are omitted in the above model.

2. The planning is of a short horizon and must be adjusted frequently as the epidemic develops. For simplicity, we assume there is no migration between regions over the time $[0, T]$.

3. Individuals once recovered from the disease are immune and free of lockdown policy.

4. The dynamics obeys the conservation law: $S_t^i + E_t^i + I_t^i + R_t^i = P^i$. This means that the process R^i is redundant.

5. The dynamics of S, E and I are subjected to random noise, to account for the noise introduced during data recording, false-positive/negative test results, exceptional cases when recovered individuals become susceptible again, minor individual differences in the latent period, etc.

6. Individuals who are not under lockdown have the same productivity, no matter their categories. We assume this for simplicity and remark that this can be improved by assigning different productivity to individuals with or without symptoms.

The above modeling and objectives can be viewed as a stochastic differential game between N players[1]. Here we view the whole problem as a non-cooperative game, as many regions make decisions individually and indeed even compete for scarce resources (frontline workers, personal protective equipment, etc.) during the outbreak. Each player i controls her states (S^i, E^i, I^i, R^i) through her strategy (ℓ^i, h^i) in order to minimize the associated cost J^i. The optimizers then are interpreted as the optimal lockdown policy and optimal effort put into the health system.

Since we are considering the problem as a non-cooperative game, we use the notion of Nash equilibrium to model optimality and the outcome of the model problem. The Nash equilibrium for this problem can be understood through the Definition 1.1, where $\alpha^i = (\ell^i, h^i)$ and with the admissible set of controls given by Markovian controls, i.e. $\mathbb{A} = \otimes_{i=1}^N \mathbb{A}^i_{\text{MC}}$, where $\mathbb{A}^i_{\text{MC}}$ is given by the set definition (1.4). To be more precise, we require the controls for each player i to be in the set of Borel measurable functions: $(\ell^i, h^i) : [0, T] \times \mathbb{R}^{3N} \to [0, 1]^2$. In other words, the policies (ℓ^i_t, h^i_t) at time t are functions of the time t and the current values of all players' state processes $(\boldsymbol{S}_t, \boldsymbol{E}_t, \boldsymbol{I}_t)$. We omit the dependence on $\boldsymbol{R}_t$ as it is redundant as a consequence of the conservation law.

5.2 Machine Learning Algorithm and Implementation

The idea is to solve the SDG (5.1)–(5.5) via the DFP methodology in Chapter 2, Section 2.3. First, this involves deriving the Hamilton-Jacobi-Bellman (HJB) equations characterizing the Markovian Nash equilibrium. To simplify the notation, we first rewrite the

[1]Henceforth, in this Chapter, we shall use *planner* and *player* interchangeably.

dynamics of $(\boldsymbol{S}_t, \boldsymbol{E}_t, \boldsymbol{I}_t)$ defined in (5.1)–(5.3) into a vector form

$$\boldsymbol{X}_t \equiv [\boldsymbol{S}_t, \boldsymbol{E}_t, \boldsymbol{I}_t]^{\mathrm{T}} \equiv [S_t^1, \cdots, S_t^N, E_t^1, \cdots, E_t^N, I_t^1, \cdots, I_t^N]^{\mathrm{T}} \in \mathbb{R}^{3N}.$$

Again, we shall drop the redundant process $\boldsymbol{R}_t$ defined in (5.4). We also combine the two controls for player i into a single vectorized control so that α^i. That is $\alpha^i = (\ell^i, h^i)$ for each $i \in \mathcal{N}$, and $\boldsymbol{\alpha} = (\alpha^1, \cdots, \alpha^N)$. The dynamics of $\boldsymbol{X}_t$ is in the general form:

$$\mathrm{d}\boldsymbol{X}_t = \boldsymbol{b}(t, \boldsymbol{X}_t, \boldsymbol{\alpha}(t, X_t)\,\mathrm{d}t + \sigma(\boldsymbol{X}_t)\,\mathrm{d}\boldsymbol{W}_t,$$

where $\boldsymbol{b}, \sigma$ are deterministic functions in $\mathbb{R}^{3N}$ and $\mathbb{R}^{3N\times 2N}$, and $\{\boldsymbol{W}_t\}_{0\le t\le T}$ is a $2N$-dimensional standard Brownian motion. Each player i aims to minimize the expected running cost

$$\mathbb{E}\left[\int_0^T f^i(t, \boldsymbol{X}_t, \alpha^i(t, \boldsymbol{X}_t))\,\mathrm{d}t\right], \tag{5.6}$$

as there is no terminal cost in this case. We defer the specific definitions of $\boldsymbol{b}, \sigma, \boldsymbol{W}$, and f^i to Section 5.3.1 to facilitate the exposition. The value function for player i is given by

$$V^i(t, \boldsymbol{x}) = \inf_{\alpha^i \in \mathbb{A}^i} \mathbb{E}\left[\int_t^T f^i(s, \boldsymbol{X}_s, \alpha^i(s, \boldsymbol{X}_s))\,\mathrm{d}s | \boldsymbol{X}_t = \boldsymbol{x}\right].$$

In this form, we see the problem for finding the Nash Equilibrium could be solved by the method outlined in Chapter 2, Section 2.3 with the DFP methodology presented in [20]. However, we will see that for our particular problem which involves a particularly difficult structure, we can adjust this methodology to greatly reduce its memory costs.

Referring to the equations in Chapter 2, Section 2.3, the algorithm in [20] solves the BSDE (2.9)–(2.10) by parametrizing $Y_t^i = V^i(t, \boldsymbol{x})$ using neural networks (NNs) and then obtains the approximate optimal policy by plugging the NN outputs into Eq. (2.7). For memory efficiency, the algorithm only stores the NNs' parameters at the current and the one-step previous stages. This strategy works well for games like the linear-quadratic game, but it would be ineffective if $\boldsymbol{\alpha}^{-i}$ explicitly appears in the minimizer in Eq. (2.7). In this case, when evaluating others' strategy $\boldsymbol{\alpha}^{-i}$ at stage m, it does not only need NNs

at stage m but also at stages $m-1$, $m-2$, ..., 0. This means one needs to store NNs' parameters for all the previous stages from $1, \ldots, m$, and evaluate the associated output. To overcome this problem, one considers a modification of the original algorithm called *Enhanced Deep Fictitious Play*.

5.2.1 Enhanced Deep Fictitious Play

In order to reduce the computational complexity of evaluating $\boldsymbol{\alpha}^{-i}$ in the situation when $\boldsymbol{\alpha}^{-i}$ explicitly appears in the minimizer in (2.7), we propose the *Enhanced Deep Fictitious Play* which parametrizes both $V^i(t, \boldsymbol{x})$ and control $\alpha^i(t, \boldsymbol{x})$ by NNs. This method represents an adaptation to the DFP method introduced in Section 2.3.

In each stage of the *Enhanced Deep Fictitious Play*, for each player i, the loss that the algorithm aims to minimize consists of two parts: (1) the loss related to solving Eqs. (2.9)–(2.10) and (2) the error of approximating the optimal strategy α^i within some hypothesis spaces. The resulted approximation $\tilde{\alpha}^i$ will be used in the next stage of fictitious play:

$$
\begin{aligned}
&\inf_{Y_0^i, \tilde{\alpha}^i, \{\boldsymbol{Z}_t^i\}_{0\le t\le T}} \mathbb{E}\Big(\big|Y_T^i\big|^2 + \nu \int_0^T |\alpha^i(s, \boldsymbol{X}_s^i) - \tilde{\alpha}^i(s, \boldsymbol{X}_s^i)|^2 \,\mathrm{d}s\Big) \\
\text{s.t. } &\boldsymbol{X}_t^i = \boldsymbol{x}_0 + \int_0^t \mu^i(s, \boldsymbol{X}_s^i; \tilde{\boldsymbol{\alpha}}^{-i}(s, \boldsymbol{X}_s^i))\,\mathrm{d}s + \int_0^t \sigma(\boldsymbol{X}_s^i)\,\mathrm{d}\boldsymbol{W}_s, \\
&Y_t^i = Y_0^i - \int_0^t \zeta^i(s, \boldsymbol{X}_s^i, \boldsymbol{Z}_s^i; \tilde{\boldsymbol{\alpha}}^{-i}(s, \boldsymbol{X}_s^i))\,\mathrm{d}s + \int_0^t (\boldsymbol{Z}_s^i)^{\mathrm{T}}\,\mathrm{d}\boldsymbol{W}_s, \\
&\alpha^i(s, \boldsymbol{X}_s^i) = \arg\min_{\beta^i} H^i(s, \boldsymbol{X}_s^i, (\beta^i, \tilde{\boldsymbol{\alpha}}^{-i})(s, \boldsymbol{X}_s^i), \boldsymbol{Z}_s^i),\,[2]
\end{aligned}
\tag{5.7}
$$

where $\tilde{\boldsymbol{\alpha}}^{-i}$ denotes the collection of approximated optimal controls from the previous stage except player i, and ν is a hyperparameter denoting the weight between two terms in the loss function. The hypothesis space for which we search $\tilde{\alpha}^i$ is characterized by another NN, in addition to the one to approximate Y_0 and $\{\boldsymbol{Z}_t^i\}_{0\le t\le T}$. Although representing $\tilde{\alpha}^i$ with a neural network introduces approximation errors, it allows us to efficiently access

[2]Here we have assumed that the Hamiltonian H^i depends on $\nabla_{\boldsymbol{x}} V$ through $\sigma^{\mathrm{T}} \nabla_{\boldsymbol{x}} V$.

the proxy of the optimal strategy $\boldsymbol{\alpha}^{-i}$ in the last stage by calling corresponding networks, instead of storing and calling all the previous strategies $\boldsymbol{\alpha}^{-i,m-1}, \ldots, \boldsymbol{\alpha}^{-i,1}$ due to the recursive dependence.

Numerically, we solve a discretized version of (5.7). Given a partition π of size N_T on the time interval $[0, T]$, $0 = t_0 < t_1 < .... < t_{N_T} = T$, the algorithm reads (to ease the notation, we replace the subscript t_k by k):

$$\inf_{\psi_0 \in \mathcal{N}_0^{i'}, \left\{\phi_k \in \mathcal{N}_k^i, \xi_k \in \mathcal{N}_k^{i''}\right\}_{k=0}^{N_T-1}} \mathbb{E}\{|Y_T^{i,\pi}|^2 + \nu \sum_k |\alpha_k^{i,\pi} - \tilde{\alpha}_k^{i,\pi}(\boldsymbol{X}_k^{i,\pi})|^2 \Delta t_k\} \tag{5.8}$$

$$\text{s.t. } \boldsymbol{X}_0^{i,\pi} = \boldsymbol{X}_0, \quad Y_0^{i,\pi} = \psi_0\left(\boldsymbol{X}_0^{i,\pi}\right), \quad \boldsymbol{Z}_k^{i,\pi} = \phi_k\left(\boldsymbol{X}_k^{i,\pi}\right), \quad \tilde{\alpha}_k^{i,\pi}(\boldsymbol{X}_k^{i,\pi}) = \xi_k(\boldsymbol{X}_k^{i,\pi}),$$

$$\alpha_k^{i,\pi} = \arg\min_{\beta^i} H^i(t_k, \boldsymbol{X}_k^{i,\pi}, (\beta^i, \tilde{\boldsymbol{\alpha}}_k^{-i,\pi})(\boldsymbol{X}_k^{i,\pi}), \boldsymbol{Z}_k^{i,\pi}), \quad k = 0, \ldots, N_T - 1$$

$$\boldsymbol{X}_{k+1}^{i,\pi} = \boldsymbol{X}_k^{i,\pi} + \mu^i\left(t_k, \boldsymbol{X}_k^{i,\pi}; \tilde{\boldsymbol{\alpha}}_k^{-i,\pi}(\boldsymbol{X}_k^{i,\pi})\right)\Delta t_k + \sigma\left(t_k, \boldsymbol{X}_k^{i,\pi}\right)\Delta \boldsymbol{W}_k, \tag{5.9}$$

$$Y_{k+1}^{i,\pi} = Y_k^{i,\pi} - \zeta^i\left(t_k, \boldsymbol{X}_k^{i,\pi}, \boldsymbol{Z}_k^{i,\pi}; \tilde{\boldsymbol{\alpha}}_k^{i,\pi}(\boldsymbol{X}_k^{i,\pi})\right)\Delta t_k + \left(\boldsymbol{Z}_k^{i,\pi}\right)^{\mathrm{T}} \Delta \boldsymbol{W}_k, \tag{5.10}$$

where $\Delta t_k = t_{k+1} - t_k$, $\Delta \boldsymbol{W}_k = \boldsymbol{W}_{t_{k+1}} - \boldsymbol{W}_{t_k}$, and $\mathcal{N}_0^{i'}$, $\{\mathcal{N}_k^i\}_{k=0}^{N_T-1}$, $\{\mathcal{N}_k^{i''}\}_{k=0}^{N_T-1}$ are the hypothesis spaces of NNs for player i. With this algorithm in mind, we iterate according to the method proposed by Algorithm 1 in Chapter 2 where the interior optimization problem in Algorithm 1 at stage m for player i is determined by the solution to Eqs. (5.8)–(5.10) above. This new algorithm is still theoretical as we cannot find the precise neural network minimizes within the class of neural networks, and hence we will have to approximate this with a sequence of stochastic gradient descent (SGD) steps. Moreover, the expectation in (5.8) is further approximated by Monte Carlo samples of (5.9)-(5.10). This is made clear by the full algorithm for *Enhanced Deep Fictitious Play* in Algorithm 4 presented later in this section.

The parameters in the hypothesis spaces are determined by stochastic gradient descent algorithms such that the approximated expectation is minimized, which in turn gives the optimal deterministic functions $(\psi_0^*, \phi_k^*, \xi_k^*)$. We expect that $(\psi_0^*, \phi_k^*, \xi_k^*)$ will approximate $(V^i, \nabla_{\boldsymbol{x}} V^i, \alpha^i)$ well when this proxy of (5.8) is small. Particularly, $\{\xi_k^*\}_{k=0}^{N_T-1}$ serves as

an efficient tool to evaluate the optimal policy at the current stage for finding Nash equilibrium. We note that when $\nu = 0$ and $\tilde{\alpha}$ are replaced by α in the above algorithm, the *Enhanced Deep Fictitious Play* degenerates to the *Deep Fictitious Play* proposed in [20].

Implementation of Enhanced DFP

Here we provide some detail to implement the methodology in Section 5.2.1. First, we specify the hypothesis spaces for neural networks $\mathcal{N}_0^{i'}$,$\{\mathcal{N}_k^i\}_{k=0}^{N_T-1}$, $\{\mathcal{N}_k^{i''}\}_{k=0}^{N_T-1}$, corresponding to $V^i, \nabla_{\boldsymbol{x}} V^i, \alpha^i$ (the superscript m is dropped again for simplicity). $V^i(t,x)$ is parametrized directly by a neural network $\mathrm{NN}(t,\boldsymbol{x})$. Corresponding map $\sigma(\boldsymbol{X})^{\mathrm{T}}\nabla_{\boldsymbol{x}} V^i(t,\boldsymbol{X})$ that defines $\boldsymbol{Z}_t^i$ in the optimization problem (5.7) could be parametrized by $\sigma(\boldsymbol{x})\nabla_{\boldsymbol{x}}\mathrm{NN}(t,\boldsymbol{x})$. Naturally, $\sigma(\boldsymbol{x})\nabla_{\boldsymbol{x}}\mathrm{NN}(t_k,\boldsymbol{x})$ is a hypothesis function in $\mathcal{N}_k^i$. Under this parametrization rule, the hypothesis functions in $\mathcal{N}_0^{i'}$ and $\{\mathcal{N}_k^i\}_{k=0}^{N_T-1}$ share the same set of parameters. The policy function $\alpha^i(t,\boldsymbol{x})$ is parametrized by another neural network $\widetilde{\mathrm{NN}}(t,\boldsymbol{x})$ and then $\widetilde{\mathrm{NN}}(t_k,\boldsymbol{x})$ plays the role of a hypothesis function in $\mathcal{N}_k^{i''}$. In other words, $\{\mathcal{N}_k^{i''}\}_{k=0}^{N_T-1}$ share the same set of neural networks. In a stochastic game of N players, there are $2N$ neural networks in total, with N neural networks corresponding to $V^i(t,\boldsymbol{x})$ and N neural networks corresponding to $\alpha^i(t,\boldsymbol{x})$. At stage m, the N V-networks are trained to approximate the solution of PDE (2.8) and the N α-networks are trained to approximate the current optimal policy computed by (2.7) using the optimal strategies in the last stage. The updated neural networks at stage m would be used at stage $m+1$ to simulate paths $\{X_k^{i,\pi}\}_{k=0}^{N_T-1}$ and optimal strategies by (2.7). In this work, fully connected neural networks with three hidden layers are used.

Second, at each stage, the $2N$ neural networks could be decoupled to N pairs of V-network and α-network based on players. Then, the N pairs of neural networks could be trained in parallel, which dramatically reduces computational time. As [20] and [40] pointed out, it is not necessary to solve the individual control problem accurately in each

stage; the parameters at each stage are updated starting from the optimal parameters in the last stage without re-initialization. This requires only a moderate number of epochs for the stochastic gradient descent at each stage.

This leads us to the full implementation of *Enhanced Deep Fictitious Play* shown in Algorithm 4 below.

Algorithm 4 Enhanced Deep Fictitious Play for Finding Markovian Nash Equilibrium

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

Initialize N deep neural networks to represent $V^{i,0}$ and N deep neural networks to represent $\alpha^{i,0}, i \in \mathcal{N}$
for $m \leftarrow 1$ to N_{stages} **do**
    for $i \in \mathcal{N}$ (in parallel) **do**
        Generate N_{batch} sample paths $\{\boldsymbol{X}_k^{i,\pi}\}_{k=0}^{N_T}$ according to (5.9) and the realized approximate optimal policies $\tilde{\boldsymbol{\alpha}}^{-i,m-1}(t_k, \boldsymbol{X}_k^{i,\pi})$
        for $e \leftarrow 1$ to $N_{\text{SGD\_per\_stage}}$ **do**
            Update the parameters of the i^{th} V-neural network and α-neural network one step with N_{batch} paths using the SGD algorithm (or its variant), based on the loss function (5.8)
        end for
        Obtain the approximate optimal policy $\tilde{\alpha}^{i,m}$ represented by the latest policy neural network
    end for
    Collect the approximate optimal policies at stage m: $\tilde{\boldsymbol{\alpha}}^m \leftarrow (\tilde{\alpha}^{1,m}, \ldots, \tilde{\alpha}^{N,m})$
end for

The exact choice of NN architectures will be detailed in Section 5.4. To determine the total stages of fictitious play N_{stages}, we monitor the relative changes of α^i and V^i, and stop the process when the relative change from stage to stage is below a threshold. Regarding the total number of SGD per stage, as shown in [20, Figure 1], the original DFP is insensitive to the choice of $N_{\text{SGD\_per\_stage}}$. We find the enhanced version sharing the same behavior when applied to our COVID-19 case study of our game-theoretic, epidemiological model defined by Eqs. (5.1)–(5.6). We give more details in Section 5.3.3, and further experiments regarding different choices of N_{stages} and $N_{\text{SGD\_per\_stage}}$ in Section 5.4.1.

For problems without analytical solutions, one natural concern is the reliability of numerical solutions. Theoretically, the quantity (5.8) serves as the indicator of the numerical accuracy. In the original DFP where the second term in (5.8) does not exist, Theorem 3 in [22] ensures the convergence to the true Nash equilibrium under technical assumptions when the quantity (5.8) is small enough for each fictitious play stage and with sufficiently large N_{stages} and small Δt_k. In practice, the quantity (5.8) is approximated by its Monte Carlo counterpart, which we define as the loss function of our algorithms. Therefore, having small training losses during all stages will ensure convergence. Extending Theorem 3 in [22] to the current setting is left for further work.

5.3 Application on COVID-19

Our case study is based on COVID-19 dynamics in the pre-vaccination period. We focus mainly on the lockdown/travel ban policy between different regions to evaluate the theoretical optimal lockdown policies that could have been implemented before the widespread deployment of vaccines. Studying this pre-vaccination period of COVID-19 dynamics, we take $v(\cdot) = 0$. Also, we focus mainly on the lockdown/travel ban policy between different regions. Therefore, to simplify the presentation, we omit the health policy h, and take $\lambda(\cdot) = \lambda$, and $\eta = 0$ in the model problem given by Eqs. (5.1)–(5.5).

5.3.1 The Decoupled HJB equations for the COVID-19 Model

Since we are not considering health policy and vaccinations, but only the effects of travel ban and lockdown, the dynamics of (S_t^n, E_t^n, I_t^n) defined in (5.1)–(5.3) in the vector form can be written as

$$\mathrm{d}\boldsymbol{X}_t = \boldsymbol{b}(t, \boldsymbol{X}_t, \boldsymbol{\ell}(t, \boldsymbol{X}_t))\,\mathrm{d}t + \sigma(\boldsymbol{X}_t)\,\mathrm{d}\boldsymbol{W}_t,$$

where again we restate $\boldsymbol{X}_t \equiv [\boldsymbol{S}_t, \boldsymbol{E}_t, \boldsymbol{I}_t]^{\mathrm{T}} \equiv [S_t^1, \cdots, S_t^N, E_t^1, \cdots, E_t^N, I_t^1, \cdots, I_t^N]^{\mathrm{T}} \in \mathbb{R}^{3N}$, and the Markovian controls $\boldsymbol{\ell}$ are given by $\boldsymbol{\ell}(t, \boldsymbol{x}) = [\ell^1, \ldots, \ell^N]^{\mathrm{T}}(t, \boldsymbol{x})$. In the

sequel, for a vector $\boldsymbol{x} \equiv (\boldsymbol{s}, \boldsymbol{e}, \boldsymbol{i}) \in \mathbb{R}^{3N}$, we shall index them in two ways,

$$(s_1, \cdots, s_N, e_1, \cdots, e_N, i_1, \cdots, i_N) \text{ or } (x_1, \ldots, x_{3N}).$$

and use them interchangeably.

The former one emphasizes the dependence on each category, while the later notation is more condensed. Similarly, for partial derivatives, we will have two set of notations

$$(\partial_{s_1}, \cdots, \partial_{s_N}, \partial_{e_1}, \cdots, \partial_{e_N}, \partial_{i_1}, \cdots, \partial_{i_N}) \text{ or } (\partial_{x_1}, \cdots, \partial_{x_{3N}}).$$

Notation. Within this section, (Section 5.3.1 only), we will often be indexing over players using the variable n, temporarily ignoring the previous use of this variable as the dimension of the state space originally introduced as related to the generic SDG in Eq. (1.1). We have previously used i to index over the players, but this conflicts with the variable $\boldsymbol{i}$, an important independent variable in this section. To summarize, we now use n to represent the n-th player out of N total, replacing what would have typically been the index i in previous sections.

In this case, player n has cost functional given by

$$\mathbb{E}\left[\int_0^T f^n(t, \boldsymbol{X}_t, \ell^n(t, \boldsymbol{X}_t))\,\mathrm{d}t\right],$$

and we recall the value function of player n is given by

$$V^n(t, \boldsymbol{x}) = \inf_{\ell^n \in \mathbb{A}^n} \mathbb{E}\left[\int_t^T f^n(s, \boldsymbol{X}_s, \ell^n(s, \boldsymbol{X}_s))\,\mathrm{d}s | \boldsymbol{X}_t = \boldsymbol{x}\right].$$

We now specify $\boldsymbol{b}, \sigma, \boldsymbol{W}$, and f^n for the COVID-19 model problem from the structure of the dynamics (5.1)–(5.3) with the modifications that h is omitted as a control, $v(\cdot) = 0$, $\lambda(\cdot) = \lambda$, and $\eta = 0$.

First, we see that $\boldsymbol{b}(t, \boldsymbol{x}, \boldsymbol{\ell}) = [b_1, \ldots, b_{3N}]^{\mathrm{T}}(t, \boldsymbol{x}, \boldsymbol{\ell})$ is a deterministic vector-valued

function given by:

$$b_j(t,\boldsymbol{x},\boldsymbol{\ell}) = \begin{cases} -\sum_{k=1}^{N} \beta^{jk} s_j i_k (1-\theta\ell^j(t,\boldsymbol{x}))(1-\theta\ell^k(t,\boldsymbol{x})), & j \in \mathcal{N}, \\ \sum_{k=1}^{N} \beta^{j'k} s_{j'} i_k (1-\theta\ell^{j'}(t,\boldsymbol{x}))(1-\theta\ell^k(t,\boldsymbol{x})) - \gamma e_{j'}, & j \in \mathcal{N}+N, \\ \gamma e_{j'} - \lambda i_{j'}, & j \in \mathcal{N}+2N, \text{ and } j' = j \bmod N. \end{cases}$$

Next, $\sigma(\boldsymbol{x}) = (\sigma_{j,k}(\boldsymbol{x}))$ is a matrix-valued deterministic function in $\mathbb{R}^{3N\times 2N}$ with non-zero entries given below:

$$\sigma_{j,j}(\boldsymbol{x}) = -\sigma_{s_j} s_j, \qquad \sigma_{j+N,j}(\boldsymbol{x}) = \sigma_{s_j} s_j,$$
$$\sigma_{j+N,j+N}(\boldsymbol{x}) = -\sigma_{e_j} e_j, \qquad \sigma_{j+2N,j+N}(\boldsymbol{x}) = \sigma_{e_j} e_j, \quad j \in \mathcal{N}.$$

and $\{\boldsymbol{W}_t\}_{0\le t\le T}$ is a $2N$-dimensional standard Brownian motion:

$$\boldsymbol{W}_t = [W_t^{s_1}, \cdots, W_t^{s_N}, W_t^{e_1}, \cdots, W_t^{e_N}]^{\mathrm{T}}.$$

Modifying (5.5), with the absence of health policies and vaccinations, we have that each region's running cost f^n is

$$f^n(t,\boldsymbol{x},\boldsymbol{\ell}) = e^{-rt} P^n[(s_n + e_n + i_n)\ell^n(t,\boldsymbol{x})w + a(\kappa i_n \chi + p i_n c)].$$

Recall that we aim to solve (2.6) using the BSDE approach (nonlinear Feynman Kac relation). To this end, we will rewrite it in the form of (2.8) and identify $\boldsymbol{\mu}$ and $\boldsymbol{\zeta}$. The first step is to identify the minimizer in the Hamiltonian (2.5).

In this case, the Hamiltonian for the HJB corresponding to player n is:

$$H^n(t,\boldsymbol{x},(\ell^n,\boldsymbol{\ell}^{-n,m}),\nabla_{\boldsymbol{x}} V^{n,m+1})$$
$$= \boldsymbol{b}(t,\boldsymbol{x},(\ell^n,\boldsymbol{\ell}^{-n,m}))\cdot\nabla_{\boldsymbol{x}} V^{n,m+1} + f^n(t,\boldsymbol{x},\ell^n)$$

$$= \sum_{j=1}^{3N} b_j(t, \boldsymbol{x}, (\ell^n, \boldsymbol{\ell}^{-n,m})) \frac{\partial V^{n,m+1}}{\partial x_j} + e^{-rt} P^n [(s_n + e_n + i_n)\ell^n(t, \boldsymbol{x})w + a(\kappa i_n \chi + p i_n c)],$$

and recall that $\boldsymbol{\ell}^{-n,m} = (\ell^{1,m}, \ldots, \ell^{n-1,m}, \ell^{n+1,m}, \ldots, \ell^{N,m})$ represents the m^{th} stage strategies of all players other than n, which are given functions in this derivation. The first order condition requires for ℓ^n:

$$0 = \sum_{\substack{j=1 \\ j \neq i}}^{N} (1 - \theta \ell^{j,m}) \left[\beta^{jn} s_j i_n \left(\frac{\partial V^{n,m+1}}{\partial e_j} - \frac{\partial V^{n,m+1}}{\partial s_j} \right) + \beta^{nj} s_n i_j \left(\frac{\partial V^{n,m+1}}{\partial e_n} - \frac{\partial V^{n,m+1}}{\partial s_n} \right) \right]$$

$$+ 2(1 - \theta \ell^n) \beta^{nn} s_n i_n \left(\frac{\partial V^{n,m+1}}{\partial e_n} - \frac{\partial V^{n,m+1}}{\partial s_n} \right) - \frac{1}{\theta} e^{-rt} P^n (s_n + e_n + i_n) w.$$

The critical point given by the above equation indeed gives a minimizer of the Hamiltonian, as long as it is in $[0, 1]$. Because we can show $\left(\frac{\partial V^{n,m+1}}{\partial e_n} - \frac{\partial V^{n,m+1}}{\partial s_n} \right) > 0$ by comparing $V^{n,m+1}(t, \boldsymbol{x} + \epsilon_{n+N})$ and $V^{n,m+1}(t, \boldsymbol{x} + \epsilon_n)$ using their definitions, where ϵ_j is a $3N$-vector with only one nonzero entry $\epsilon \ll 1$ at j^{th} position. Intuitively, with all others players' initial condition the same, if player n starts with a higher exposed proportion $e_n + \epsilon$, it will produce more cost, comparing with the same increase proportion still being susceptible $s_n + \epsilon$. To summarize, we deduce the optimal policy for player n at stage $m + 1$ is given by:

$$\ell^{n,m+1}(t, \boldsymbol{x}) = \left\{ 2\beta^{nn} s_n i_n \left(\frac{\partial V^{n,m+1}}{\partial e_n} - \frac{\partial V^{n,m+1}}{\partial s_n} \right) - \frac{1}{\theta} e^{-rt} P^n (s_n + e_n + i_n) w \right. \tag{5.11}$$

$$\left. + \sum_{\substack{j=1 \\ j \neq n}} (1 - \theta \ell^{j,m}) \left[\beta^{jn} s_j i_n \left(\frac{\partial V^{n,m+1}}{\partial e_j} - \frac{\partial V^{n,m+1}}{\partial s_j} \right) + \beta^{nj} s_n i_j \left(\frac{\partial V^{n,m+1}}{\partial e_n} - \frac{\partial V^{n,m+1}}{\partial s_n} \right) \right] \right\}$$

$$\times \left\{ 2\theta \beta^{nn} s_n i_n \left(\frac{\partial V^{n,m+1}}{\partial e_n} - \frac{\partial V^{n,m+1}}{\partial s_n} \right) \right\}^{-1} \wedge 1 \vee 0,$$

where we use the conventional notations $a \wedge b = \min\{a, b\}$ and $a \vee b = \max\{a, b\}$. Plugging (5.11) into equation (2.6) and by straightforward computation, one obtains for the $(m+1)^{th}$ stage, $\mu^{n,m+1}(t, \boldsymbol{x}; \boldsymbol{\ell}^{-n,m}) = [\mu_1^{n,m+1}, \ldots, \mu_{3N}^{n,m+1}](t, \boldsymbol{x}; \boldsymbol{\ell}^{-n,m})^{\mathrm{T}}$ is

$$\mu_j^{n,m+1} = -\beta^{jn} s_j i_n (1 - \theta \ell^{j,m}(t, \boldsymbol{x})) - \sum_{\substack{k=1 \\ k \neq n}}^{N} \beta^{jk} s_j i_k (1 - \theta \ell^{j,m}(t, \boldsymbol{x}))(1 - \theta \ell^{k,m}(t, \boldsymbol{x})),$$

$$j \in \mathcal{N} \setminus n$$

$$\mu_n^{n,m+1} = -\beta^{nn} s_n i_n - \sum_{\substack{k=1 \\ k \neq n}} \beta^{nk} s_n i_k (1 - \theta \ell^{k,m}(t, \boldsymbol{x}))$$

$$\mu_{N+j}^{n,m+1} = -\mu_j^{n,m+1} - \gamma e_j, \quad \mu_{2N+j}^{n,m+1} = \gamma e_j - \lambda i_j, \quad j \in \mathcal{N}.$$

To write $\zeta^{n,m+1}$ as a function of $(t, \boldsymbol{x}, z)$, we first compute

$$\sigma(\boldsymbol{x})^{\mathrm{T}} \nabla_{\boldsymbol{x}} V^n(t, \boldsymbol{x}) = \Big[\sigma_{s_1} s_1 \big(\frac{\partial V^n}{\partial e_1} - \frac{\partial V^n}{\partial s_1}\big), \cdots$$
$$\cdots, \sigma_{s_N} s_N \big(\frac{\partial V^n}{\partial e_N} - \frac{\partial V^n}{\partial s_N}\big), \sigma_{e_1} e_1 \big(\frac{\partial V^n}{\partial i_1} - \frac{\partial V^n}{\partial e_1}\big), \cdots, \sigma_{e_N} e_N \big(\frac{\partial V^n}{\partial i_N} - \frac{\partial V^n}{\partial e_N}\big)\Big]^{\mathrm{T}}.$$

and then $\zeta^{n,m+1}$ is given by:

$$\zeta^{n,m+1}(t, \boldsymbol{x}, z; \boldsymbol{\ell}^{-n,m}) =$$
$$\frac{\theta^2}{\sigma_{s_n}} \beta^{nn} z_n i_n [\ell^{n,m+1}(t, \boldsymbol{x})]^2 + \Big\{ e^{-rt} P^n (s_n + e_n + i_n) w - 2 \frac{\theta}{\sigma_{s_n}} \beta^{nn} z_n i_n -$$
$$\sum_{\substack{j=1 \\ j \neq n}}^{N} \theta (1 - \theta \ell^{j,m}(t, \boldsymbol{x})) \big(\frac{\beta^{nj}}{\sigma_{s_n}} z_n i_j + \frac{\beta^{jn}}{\sigma_{s_j}} z_j i_n\big) \Big\} \ell^{n,m+1}(t, \boldsymbol{x}) + e^{-rt} P^n a (\kappa i_n \chi + p i_n c).$$

Therefore, we have defined $\boldsymbol{b}, \sigma, \boldsymbol{\mu}$, and $\boldsymbol{\zeta}$ for the COVID-19 model problem in question. With these clarified and fully specified for our considered COVID-19 model, one can implement the *enhanced* DFP algorithm introduced in Section 5.2.1 to find its Nash equilibrium given specific choices of parameter values for the model. We now discuss what parameters are chosen to simulate COVID-19 dynamics in the period between 2020

and 2021.

5.3.2 Parameter Derivations and Choices

The Inter-Region Transmission Rate Derivation

We first clarify the precise meaning of β^{ij} beyond it representing transmission between the regions. We derive β^{ij} in terms of an underlying transmission rate, β, and the travel between region.

In single-region SEIR models, the transmission rate, β, is the basic reproductive number divided by the length of time an individual is infectious. In our model, we assume that there is a region-independent constant β that underlies the rate of infections for each population. The transmission rates β^{ij} between regions are related to the underlying transmission rate β, and the amount of travel between regions i and j.

To quantify the size of travel between regions, we assume there is a constant fraction of people from region i that travel to region j, f^{ij}, at any given moment in time. We note that realistically one may expect f^{ij} to depend on time and also on the epidemic status of regions i and j. However, for simplicity, we will not consider these scenarios in our numerical experiments. We assume that $f^{ii} \gg f^{ij}$, $f^{ii} \gg f^{ji}$, $\forall j \neq i$, meaning that most of the population i resides in region i at any given time, and also that most of the people in region i at a given time are from i and not travelers from another region. We will see later that this implies $\beta^{ii} \gg \beta^{ij}$ $\forall j \neq i$.

The transmission rate, β, is the average number of people infected by an infectious person per day (assuming that the susceptible population is at 100% of the population). Thus, if a proportion S of the population is susceptible, then βS represents the average number of people infected per infectious person per day. If there are a total of P people in a population with a fraction I being infectious, then $\beta S(PI)$ is the number of newly infected per day.

Thus, in the context of a single-region SEIR model, the number of newly infected

(or the influx to the exposed population) that occurs within $(t, t + \mathrm{d}t)$ is given by $\beta S(t)(I(t)P)\,\mathrm{d}t$. Dividing by P, the influx to the scaled exposed population in $(t, t + \mathrm{d}t)$ is $\beta S(t)I(t)\,\mathrm{d}t$.

Now let us consider the multi-region case and temporarily ignore the effect of lockdown. The term from (5.1)–(5.3) that gives the influx to E^i due to infection from I^j is $\beta^{ij} S^i(t)I^j(t)\,\mathrm{d}t$. To determine β^{ij}, we build this exact influx from core assumptions.

First, we quantify the number of people from region i that are infected by those from region j. Specifically, the influx in the interval $(t, t + \mathrm{d}t)$ to the unscaled exposed population i due to transmission from population j is given by

$$\sum_{\ell} (\beta f^{i\ell} S^i(t))(f^{j\ell} I^j(t) P^j)\,\mathrm{d}t, \tag{5.12}$$

where f^{ij} is the (assumed constant) fraction of people from i currently in region j at any moment in time.

Equation (5.12) is obtained by summing the number of infections in population i due to population j across each region. The summand represents these infections occurring in region ℓ. This can be seen as the term $f^{i\ell} S^i(t)$ is the proportion of population i that are susceptible and within region ℓ. Of this population, there will be $\beta f^{i\ell} S^i(t)$ infections per infectious individual per day. Since the number of infectious from j that are in ℓ is $f^{j\ell} I^j(t) P^j$, we have that $(\beta f^{i\ell} S^i(t))(f^{j\ell} I^j(t) P^j)\,\mathrm{d}t$ is the number of new infections in population i due to population j occurring in the region ℓ within the time interval $(t, t + \mathrm{d}t)$.

Let us assume for now that $i \neq j$. Since we assume that $1 > f^{ii} \gg f^{ij}$, the terms in (5.12) besides the cases where $\ell = i$ or $\ell = j$ are negligible. Removing the negligible terms and dividing by the population P^i, we see that the influx to the scaled exposed population E^i due to transmission from population j over the interval $(t, t + \mathrm{d}t)$ is

$$\frac{P^j}{P^i} \beta (f^{ii} f^{ji} + f^{ij} f^{jj}) S^i(t) I^j(t)\,\mathrm{d}t,$$

which is exactly the influx represented by the model of $\beta^{ij} S^i(t) I^j(t)\,\mathrm{d}t$. This gives us

$$\beta^{ij} = \begin{cases} \beta(f^{ij} f^{jj} + f^{ji} f^{ii}) \dfrac{P^j}{P^i}, & \text{if } j \neq i \\ \beta(f^{ii})^2, & \text{if } j = i. \end{cases} \tag{5.13}$$

We note that the last equation in 5.13, which defines β^{ii} is found similarly by taking $i = j$ in (5.12) and ignoring all terms other than $\ell = i(= j)$.

Therefore, to specify β^{ij}, we need to provide β and f^{ij} for the regions of concern. We will conduct a thorough explanation in the choices of parameters, recalling that the model is for COVID-19 dynamics for the 2020-2021 time period in order to analyze optimal lockdown policies.

Choices of Model Parameters

The motivation for these parameter choices are based on 2020 estimates which are discussed fully in [43, Section 4.1]. These choices are used to describe dynamics of COVID-19 for the 180 day period starting on March 15, 2020.

We choose a basic reproductive number $R_0 = 2.2$, and assume that the length of each individual being infectious is 13 days. More precisely, we assume infectious individuals either recover or die in 13 days. Under these assumptions, we obtain $\beta = \frac{2.2}{13} \approx 0.17$. The infection fatality rate, or IFR, is the fraction of those infected who died from the infection. We choose the IFR to be 0.65%. We choose the latent period to be 5 days. The assumptions of an IFR of 0.65% and an infectious period of 13 days determines that the recovery rate (including both recovery and death due to infection) is $\lambda = \frac{1}{13} \approx 0.0769$, and the death rate is $\kappa = \frac{(0.65\%)}{13} = 0.0005$. We assume that the parameters for noise-level $\sigma_{s_i}, \sigma_{e_i}$, $i \in \mathcal{N}$ are all 0.0002, and the extent to which one adheres to the social distancing policy, θ, is either $\theta = 0.9$ or $\theta = 0.99$.

Our model will concern transmission between New York (NY), New Jersey (NJ), and Pennsylvania (PA), treating them as a closed-system to the outside world. We refer to

the state of New York as region 1, New Jersey as region 2, and Pennsylvania as region 3. Their respective populations in 2020 were $P^1 = 19.54$ million, $P^2 = 8.91$ million, and $P^3 = 12.81$ million. Regarding β^{ij}, $\forall i, j = 1, 2, 3$, we assume that: (a) 90% of any state's population is residing in their state at a given time; (b) the remaining population (travelers) visit the other regions in an equal proportion; and (c) there is no travel outside of the considered regions, *i.e.*, the NY-NJ-PA is a closed system. The reasoning for (c) is that, under our model assuming that infection only occurs in the regions considered, (c) is equivalent to allowing people traveling outside the considered regions, but the travelers cannot be affected. For simplicity, we assume this is the case. Under these assumptions, we will have $f^{ii} = 90\%$ for $i = 1, 2, 3$ and $f^{ij} = 5\%$ for $i \neq j$, and obtain the values of β^{ij} through (5.13).

With the parameters for the SDE model (5.1)-(5.3) discussed, we now address those specific to defining the cost in Eq. (5.5). We choose for simplicity that $r = 0$ as the US Effective Federal Funds rate was around 0.1% for the time period in question, causing the discounting for a 180 day simulation to be negligible. The parameter w represents the dollar output per individual per day. To estimate w, we use the 2020 estimate of GDP per capita per day, yielding the estimate $w = 172.6$ dollars per person per day. Following [2] and [18], we use the value of a statistical life, χ, to be 20 times GDP per capita. This results in $\chi = 1.95 \cdot 10^6$ dollars per person. According to the CDC summary of U.S. COVID-19 activity, the hospitalization rate was 228.7 per 100,000 population by 11/14/2020. Thus, we set $p = 228.7 \times 10^{-5}$. The cost per inpatient day is $c = 73300/13$ dollars, estimated according to [24]. The attention hyperparameter a takes various values in the case study, and will be specified in Section 5.3.3.

5.3.3 NY-NJ-PA COVID-19 Case Study

In this section, we apply our model (5.1)–(5.5) to analyze COVID-19 related policy in three adjacent states: New York, New Jersey, and Pennsylvania. This case study is

done over 180 days starting from 03/15/2020, using the *Enhanced Deep Fictitious Play* Algorithm 4 and the parameters discussed in Section 5.3.2, where the exact formulas of μ^i and ζ^i in the equation (2.8) are stated in Section 5.3.1.

Figure 5.1 presents the equilibrium policy issued by the governors of NY, NJ, and PA when the policy effectiveness is $\theta = 0.99$, *i.e.*, 99% of the population follow the lockdown order. The hyperparameter is $a = 100$, *i.e.*, each governor values people's death 100 times the lockdown cost. In this scenario, the governors take action at an early stage and soon reach the strictest policy. Once the disease is under control, they may relax the policy later. The percentage of Susceptible, Exposed, Infectious, and Removed stays almost constant in the end. As a comparison, Figure 5.2 illustrates how the pandemic gets out of control if governors show inaction or issue mild lockdown policies.

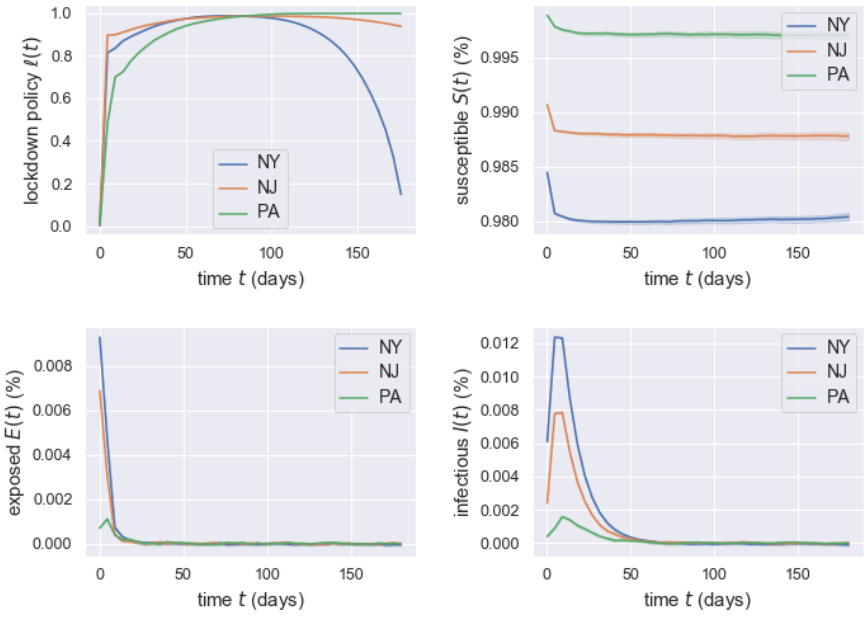

Figure 5.1: Plots of optimal policies (top-left), Susceptible (top-right), Exposed (bottom-left) and Infectious (bottom-right) for three states: New York (blue), New Jersey (orange) and Pennsylvania (green). The shaded areas depict the mean and 95% confidence interval over 256 sample paths. Choices of parameters are in Section 5.3.2, $a = 100$ and $\theta = 0.99$.

Experiment 1: dependence on a. We further analyze how the planners' view on the death of human beings changes their policies. In reality, economic loss is not the only factor the planners concern about. It is also important to mitigate the infections and

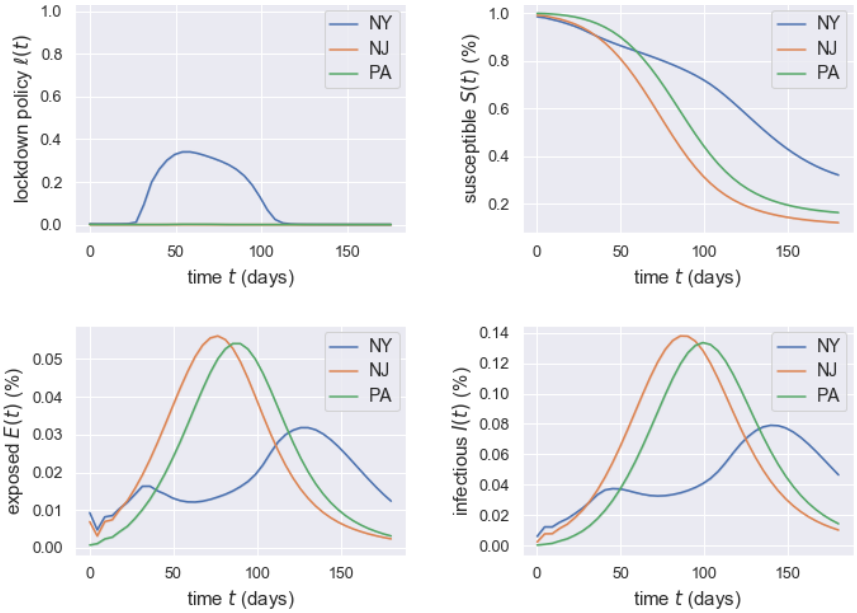

Figure 5.2: An illustration that governors' inaction or mild control leads to disease spreading.

deaths within the budget and available resources. Different views and values from the planners will lead to different policies. In this experiment, we consider different attitudes towards the infection, especially death caused by COVID-19. This is reflected by the attention hyperparameter a. Large a implies that planners care more about human beings and are willing to spend more effort or endure more economic loss on lockdown to avoid further infection and death. In comparison, smaller a implies that planners care less about infection and death and instead pay more attention to minimizing the total cost.

The numerical results in Figures 5.3 and 5.4 are consistent with intuition. With a large a (top-left panels of Figures 5.3 and 5.4), meaning the planners give more consideration to infection and death, they tend to issue a restrict lockdown policy, which helps slow down the disease spreads and reduce the percentage of infected people. As a becomes smaller (top-right panels of Figures 5.3 and 5.4), planners weigh more the economic loss and spend fewer efforts on lockdown.

When the attention a is small enough, some states even give up controlling the disease

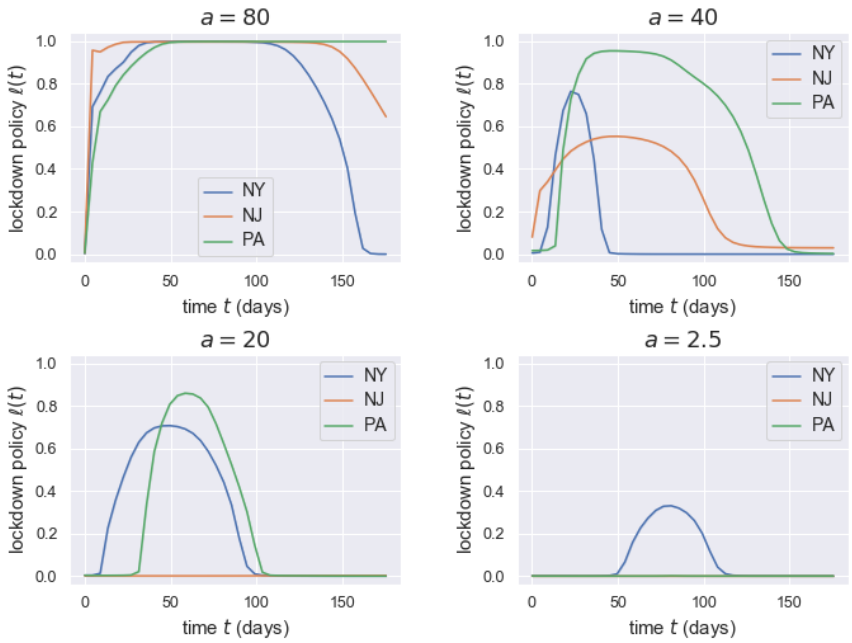

Figure 5.3: Plots of optimal policies with different choice of a for three states: New York (blue), New Jersey (orange) and Pennsylvania (green), when the lockdown efficiency is $\theta = 0.9$.

spread due to economic concern (bottom panels of Figures 5.3 and 5.4). As a result, the pandemic would get out of control by the end of the simulation period. This mild lockdown policy leads to a natural spread of disease (also shown in Figure 5.2).

Experiment 2: dependence on θ. We next analyze how the residents' willingness to comply with the lockdown policy changes the optimal policies and the development of a pandemic. The larger the θ is, the more likely the residents will follow the lockdown policy, and the larger the difference the control makes on the pandemic situation. Conversely, small θ weakens the effect of the lockdown policy. In the extreme case of $\theta = 0$, no matter how strict the lockdown policy is, the pandemic will become a natural spread because the control term in Eq. (5.1) disappears. In short, this willingness to policy compliance should be an essential factor in decision-making.

To this end, we compare the optimal policy when $\theta = 0.9$ and $\theta = 0.99$ in Figure 5.5. Panels (a-d) show the difference of optimal policies $\ell(t)$ and the Susceptible $S(t)$ in the tri-state game under different θ when $a = 50$. In both situations, the pandemic

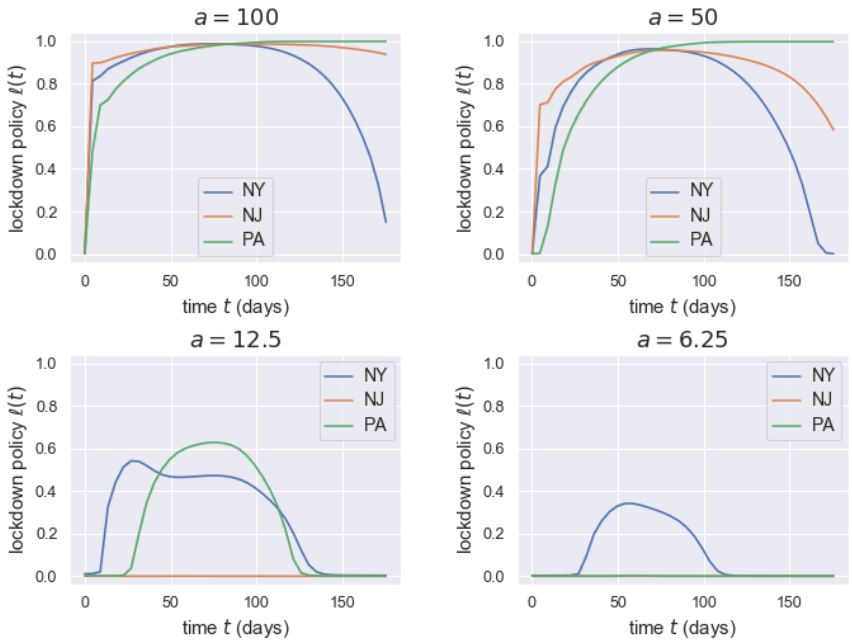

Figure 5.4: Plots of optimal policies with different choice of a for three states: New York (blue), New Jersey (orange) and Pennsylvania (green), when the lockdown efficiency is $\theta = 0.99$.

is well-controlled, with the percentage of susceptible people staying stable in the end. Moreover, in the case of $\theta = 0.99$, people are more willing to comply with the policies. Consequently, the planners are allowed to use a less strict lockdown policy as shown in Figure 5.5(b) compared to 5.5(a), which saves the lockdown cost. Figure 5.5 (e-h) shows an interesting case in the comparison of $\theta = 0.9$ and $\theta = 0.99$. In this scenario, with the same attention parameter ($a = 25$), $\theta = 0.9$ leads to a mild lockdown policy, see Figure 5.5(e), while $\theta = 0.99$ provides a possibility to stop the spread of virus, see Figure 5.5(f). We believe that the decision when $\theta = 0.9$ is a compromise as the lockdown is not efficient enough to reduce largely the infection and death loss by paying lockdown cost, and also due to the limited simulation period, *i.e.*, the policies could have been different if we had the simulation until the disease dies out. We also believe that the early give-up by NJ drives NY and PA to lift lockdown policies at a later stage, because even NY and PA issue strict policies, they are still facing severe infections from NJ due to its high infected percentages and the existence of travel between states whatever the policy is.

So their interventions are not worth the candle. Figure 5.5(f)(h) further elucidate the importance of residents' support in slowing down the pandemic.

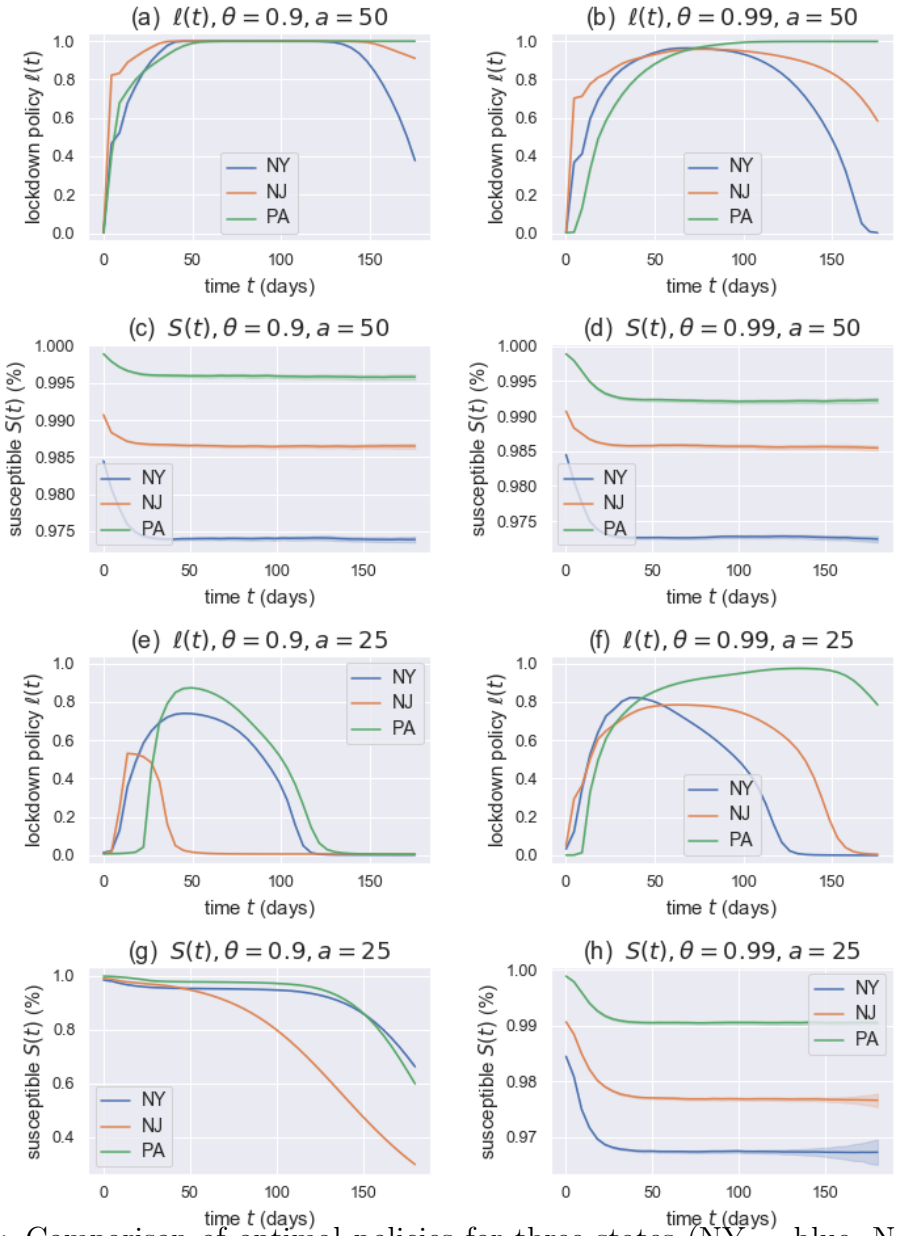

Figure 5.5: Comparison of optimal policies for three states (NY = blue, NJ = orange, PA = green) and their susceptibles between different policy effectiveness θ and hyperparameter a.

To summarize, the numerical experiments illustrate that both the balance of economy and infection/death from the view of plan-makers and the willingness of residents to follow the lockdown policy play an important role in decision-making. In reality, all three states

issued stay-at-home orders in March 2020, and attempted to reopen in June 2020. By comparing real world policies and our simulations of $\boldsymbol{\ell}(x)$, we may infer α and θ for NY, NJ, and PA in our model, *i.e.*, $\theta = 0.99$ and $a = 25$.

5.4 Implementation Details

In the NY-NJ-PA case study, we choose feedforward architectures for both V-networks and α-networks. Both have three hidden layers with a width of 40 neurons. The activation function in each hidden layer is $\tanh(x)$. We do not apply activation function to the output layer of V-networks, and choose sigmoid function $\rho_s(x) = \frac{1}{1+e^{-x}}$ for the α-networks. Other hyperparameters are summarized in the table below.

hyperparameter	lr	N_{stages}	$N_{\text{SGD\_per\_stage}}$	N_{batch}	N_T	ν
value	5e-4	250	100	256	40	$1e^{-3}/180$

Table 5.1: Hyperparameters in the case study: lr denotes the learning rate in stochastic gradient descent method, N_{stages} is the total stages of fictitious play, $N_{\text{SGD\_per\_stage}}$ is the number of stochastic gradient descent done in each minimization problem (5.8), N_{batch} is batch size in each stochastic gradient descent, N_T is the discretization steps on $[0, T]$, and ν is the weight of the control part in the loss function (5.8).

5.4.1 Discussion on the Choice of N_{stages} and $N_{\text{SGD\_per\_stage}}$

We provide further experiments here on various choices of N_{stages} and $N_{\text{SGD\_per\_stage}}$. In Figure 5.6, we plot both validation loss and log loss against N_{stages} for three states, which are produced by evaluating the NNs using unseen data after each fictitious play stage. In each panel, loss curves associated with different number of SGDs per stage are presented in different colors (blue = 50, orange = 75, green = 100, red = 125, purple = 150).

The numerical results show that the validation losses for all states decrease as the number of DFP stages N_{stages} increases. Moreover, it shows that different $N_{\text{SGD\_per\_stage}}$ generates loss curves with similar patterns. Smaller $N_{\text{SGD\_per\_stage}}$ is more stable on the

validation loss of PA. This result is consistent with [20] and [40], which convey that it is unnecessary to solve the problem extremely accurately in each stage and that a moderate number of $N_{\mathrm{SGD\_per\_stage}}$ is sufficient. As a result, we choose $N_{\mathrm{SGD\_per\_stage}} = 100$ in our case study.

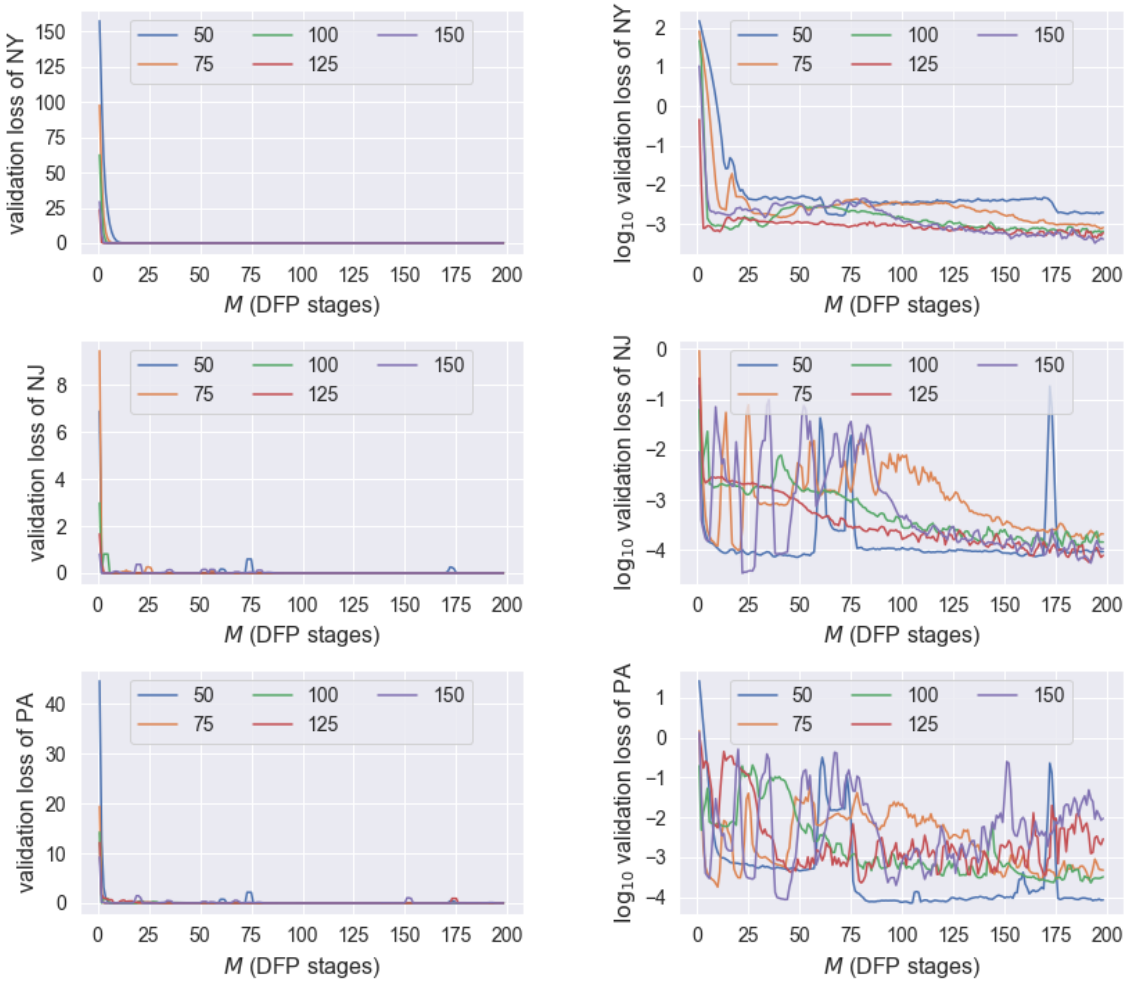

Figure 5.6: Loss curves of each state. Left: validation losses versus rounds N_{stages} of the enhanced deep fictitious play; Right: $\log_{10}$ validation loss versus rounds N_{stages} of the enhanced deep fictitious play. The loss curves with respect to $N_{\mathrm{SGD\_per\_stage}} = 50, 75, 100, 125, 150$ are depicted in blue, orange, green, purple and red. A smoothed moving average with window size 3 is applied in the final plots.

5.4.2 Stability over Different Experiments

Here we present experiments to investigate the Nash equilibrium of the model with different combinations of parameters. For each combination of parameters, we use the same hyper-parameters and repeat the experiments several times. We run the algorithm for a certain computational budget, and then filter out the results with a fluctuating loss near the stopping and check the converged equilibrium. In the first combination of parameters, we take $\theta = 0.99$ and $a = 100$, corresponding to the case that a governor weighs the deaths much more than the economic loss and tries to avoid it, and the residents have a strong willingness to follow the governor's policies. Intuitively, the pandemic is possible to get well-controlled. Our numerical experiments confirm this intuition: all converging trails lead to the same Nash equilibrium. A representative plot of $X(t) = (S(t), E(t), I(t))$ is shown in Figure 5.7.

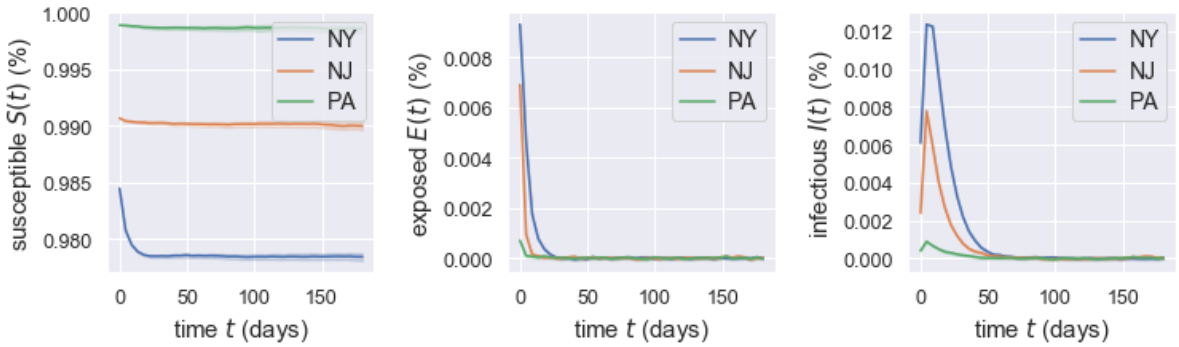

Figure 5.7: With the parameter combination $\theta = 0.99$, $a = 100$, the algorithm identifies one Nash equilibrium for the NY-NJ-PA case study.

In the second batch of experiments, we take $\theta = 0.9$ and $a = 50$, corresponding to the case that a governor pays less attention to the number of deaths, and the residents are less willing to follow the policies compared to the first batch of experiments. The change leads to the possibility of multiple Nash equilibrium and the pandemic being out of control. In this case, with different NNs' initialization, the algorithm identifies two Nash equilibria: 75% of the experiments converge to the Nash equilibrium that the pandemic gets controlled and 25% of the experiments converge to the other Nash equilibrium where

the pandemic gets out of control.

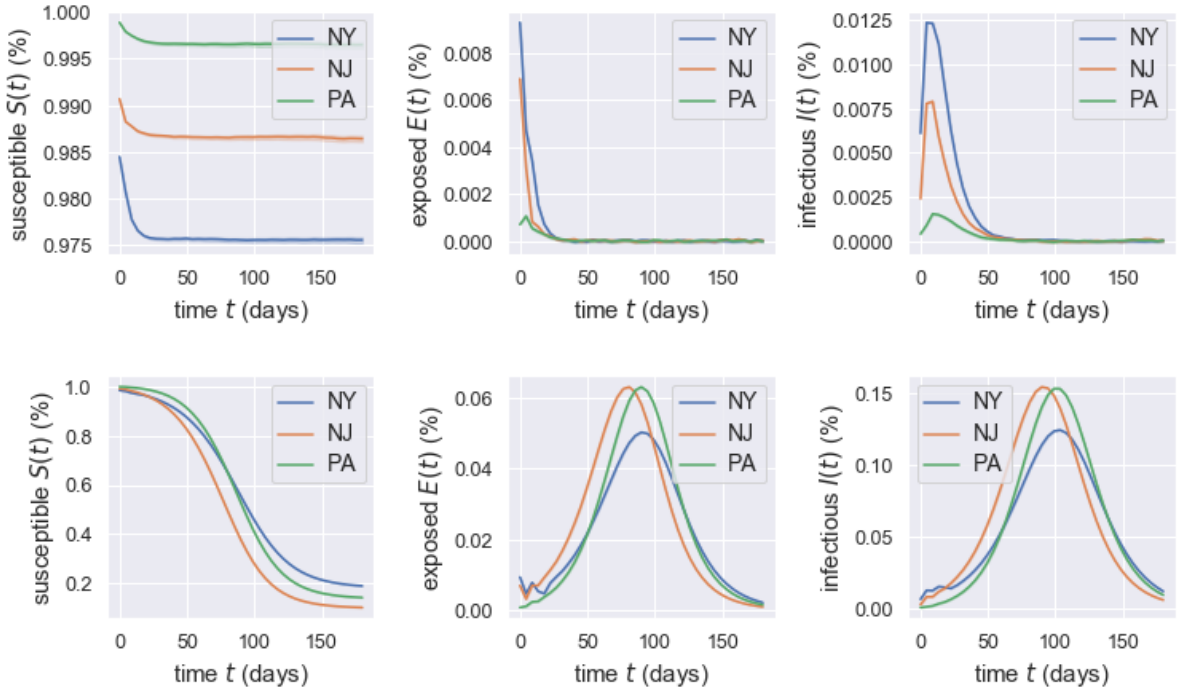

Figure 5.8: With the parameter combination $\theta = 0.9$, $a = 50$, the algorithm identifies two possible Nash equilibria: an under control one (topic panels, with 75% of the experiments) and on out-of-control one (bottom panels, with 25% of the experiments).

In conclusion, it is possible to have multiple Nash equilibria depending on the parameter chosen in our stochastic multi-region SEIR model. There is usually a single Nash equilibrium for parameters chosen at extreme values, while for the parameters selected in the middle range, there could exist multiple Nash equilibria. When multiple equilibria exist, we conjecture that the possibility to reach a particular one depends on where we start the fictitious play (the initialization of the NNs' parameters).

Bibliography

[1] M. Abadi et al. "TensorFlow: A System for Large-Scale Machine Learning." In: *OSDI*. Vol. 16. 2016, pp. 265–283.

[2] F. E. Alvarez, D. Argente, and F. Lippi. *A Simple Planning Problem for COVID-19 Lockdown*. Working Paper 26981. National Bureau of Economic Research, 2020.

[3] R. Balkin, H.D. Ceniceros, and R. Hu. *Stochastic Delay Differential Games: Financial Modeling and Machine Learning Algorithms*. 2023. arXiv: 2307.06450.

[4] S. Basak and D. Makarov. "Competition among Portfolio Managers and Asset Specialization". In: w0194 (Apr. 2013). URL: https://ideas.repec.org/p/abo/neswpt/w0194.html.

[5] C. T. Bauch and D. J. D. Earn. "Vaccination and the theory of games". In: *Proceedings of the National Academy of Sciences* 101.36 (2004), pp. 13391–13394.

[6] C. T. Bauch, A. P. Galvani, and D. J. D. Earn. "Group interest versus self-interest in smallpox vaccination policy". In: *Proceedings of the National Academy of Sciences* 100.18 (2003), pp. 10564–10567.

[7] A. G. Baydin, B. A. Pearlmutter, and A. A. Radul. "Automatic differentiation in machine learning: a survey". In: *CoRR* abs/1502.05767 (2015). arXiv: 1502.05767.

[8] G. W. Brown. "Iterative solution of games by fictitious play". In: *Activity Analysis of Production and Allocation* 13.1 (1951), pp. 374–376.

[9] G. W. Brown. *Some notes on computation of games solutions*. Tech. rep. Rand Corp Santa Monica CA, 1949.

[10] R. Carmona and F. Delarue. *Probabilistic Theory of Mean Field Games with Applications I*. Probability Theory and Stochastic Modelling. Springer, 2018.

[11] R. Carmona, J. Fouque, S. Mousavi, and L. Sun. "Systemic Risk and Stochastic Games with Delay". In: *Journal of Optimization Theory and Applications* 179 (2018), pp. 366–399.

[12] S. L. Chang, M. Piraveenan, P. Pattison, and M. Prokopenko. "Game theoretic modelling of infectious disease dynamics and intervention methods: a review". In: *Journal of Biological Dynamics* 14.1 (2020), pp. 1–33.

[13] W. E, J. Han, and A. Jentzen. "Deep Learning-Based Numerical Methods for High-Dimensional Parabolic Partial Differential Equations and Backward Stochastic Differential Equations". In: *Communications in Mathematics and Statistics* 5.4 (Nov. 2017), pp. 349–380.

[14] N. El Karoui, S. Peng, and M. C. Quenez. "Backward stochastic differential equations in finance". In: *Mathematical Finance* 7.1 (1997), pp. 1–71.

[15] G. Fabbri, F. Gozzi, and A. Swiech. *Stochastic optimal control in infinite dimension*. Probability and Stochastic Modelling. Springer, 2017.

[16] J. Fouque and Z. Zhang. "Deep Learning Methods for Mean Field Control Problems with Delay". In: *Frontiers in Applied Mathematics and Statistics* 6 (2020).

[17] F. Gozzi and C. Marinelli. "Stochastic Optimal Control of Delay Equations Arising in Advertising Models". In: 245 (Jan. 2005).

[18] R. E. Hall, C. I Jones, and P. J. Klenow. *Trading Off Consumption and COVID-19 Deaths*. Working Paper 27340. National Bureau of Economic Research, 2020.

[19] J. Han and W. E. *Deep Learning Approximation for Stochastic Control Problems*. 2016. arXiv: 1611.07422 [cs.LG].

[20] J. Han and R. Hu. "Deep Fictitious Play for Finding Markovian Nash Equilibrium in Multi-Agent Games". In: *Proceedings of The First Mathematical and Scientific Machine Learning Conference (MSML)*. Vol. 107. 2020, pp. 221–245.

[21] J. Han and R. Hu. "Recurrent Neural Networks for Stochastic Control Problems with Delay". In: *Mathematics of Control, Signals, and Systems* 33 (2021), pp. 775–795.

[22] J. Han, R. Hu, and J. Long. "Convergence of Deep Fictitious Play for Stochastic Differential Games". In: *Frontiers of Mathematical Finance* 1.2 (2022), pp. 287–319.

[23] J. Han, A. Jentzen, and W. E. "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences* 115.34 (Aug. 2018), pp. 8505–8510.

[24] Fair Health. *Costs for a Hospital Stay for COVID-19*. 2020. URL: https://www.fairhealth.org/article/costs-for-a-hospital-stay-for-covid-19.

[25] S. Hochreiter and J. Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80.

[26] R. Hu. "Deep Fictitious Play for Stochastic Differential Games". In: *Communications in Mathematical Sciences* 19(2) (2021), pp. 325–353.

[27] R. Isaacs. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. London: John Wiley and Sons, 1965.

[28] D.P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." In: *ICLR (Poster)*. 2015.

[29] D. Lacker and A. Soret. "Many-player games of optimal consumption and investment under relative performance criteria". In: *Mathematics and Financial Economics* 14 (2019), pp. 263–281.

[30] D. Lacker and T. Zariphopoulou. "Mean field and n-agent games for optimal investment under relative performance criteria". In: *Mathematical Finance* 29.4 (2019), pp. 1003–1038.

[31] W.-M. Liu, H. W. Hethcote, and S. A. Levin. "Dynamical behavior of epidemiological models with nonlinear incidence rates". In: *Journal of Mathematical Biology* 25.4 (1987), pp. 359–380.

[32] X. Mao. "Approximate solutions for a class of delay stochastic differential equations". In: *Stochastics and Stochastic Reports* 35.2 (1991), pp. 111–123.

[33] R. Merton. "Lifetime Portfolio Selection under Uncertainty: The Continuous-Time Case". In: *The Review of Economics and Statistics* 51.3 (1969), pp. 247–57.

[34] S. A. Mohammed. "Stochastic Differential Systems With Memory: Theory, Examples and Applications". In: *Stochastic Analysis and Related Topics VI*. Boston, MA: Birkhäuser Boston, 1998, pp. 1–77. ISBN: 978-1-4612-2022-0.

[35] T. Pang and A. Hussain. "A stochastic portfolio optimization model with complete memory". In: *Stochastic Analysis and Applications* 35.4 (2017), pp. 742–766.

[36] E. Pardoux and S. Peng. "Backward stochastic differential equations and quasilinear parabolic partial differential equations". In: *Stochastic Partial Differential Equations and Their Applications*. Springer, 1992, pp. 200–217.

[37] E. Pardoux and S. Tang. "Forward-backward stochastic differential equations and quasilinear parabolic PDEs". In: *Probability Theory and Related Fields* 114.2 (1999), pp. 123–150.

[38] A. Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703.

[39] D. Rumelhart, G. Hinton, and R. Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[40] D. Seale and J. Burnett. "Solving large games with simulated fictitious play". In: *International Game Theory Review* 8.03 (2006), pp. 437–467.

[41] J. Sirignano and K. Spiliopoulos. "DGM: A deep learning algorithm for solving partial differential equations". In: *Journal of Computational Physics* 375 (Dec. 2018), pp. 1339–1364.

[42] Y. Xuan, R. Balkin, J. Han, R. Hu, and H. D. Ceniceros. "Pandemic Control, Game Theory and Machine Learning". In: *Notices of the American Mathematical Society* 69.11 (2022), pp. 1878–1887.

[43] Y. Xuan, R. Balkin, J. Han, R. Hu, and H.D. Ceniceros. "Optimal Policies for a Pandemic: A Stochastic Game Approach and a Deep Learning Algorithm". In: *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*. Vol. 145. Proceedings of Machine Learning Research. PMLR, 16–19 Aug 2022, pp. 987–1012.