Inverse linear-quadratic nonzero-sum differential games

Martirosyan, E.; Cao, M.

Inverse Linear-quadratic Nonzero-sum Differential Games

Emin Martirosyan, Ming Cao

***Abstract*—This paper addresses the inverse problem for Linear-Quadratic (LQ) nonzero-sum N-player differential games, where the goal is to learn parameters of an unknown cost function for the game, called observed, given the demonstrated trajectories that are known to be generated by stationary linear feedback Nash equilibrium laws. Towards this end, using the demonstrated data, a synthesized game needs to be constructed, which is required to be equivalent to the observed game in the sense that the trajectories generated by the equilibrium feedback laws of the N players in the synthesized game are the same as those demonstrated trajectories. We show a model-based algorithm that can accomplish this task using the given trajectories. We then extend this model-based algorithm to a model-free setting to solve the same problem in the case when the system's matrices are unknown. The algorithms combine both inverse optimal control and reinforcement learning methods making extensive use of gradient descent optimization for the latter. The analysis of the algorithm focuses on the proof of its convergence and stability. To further illustrate possible solution characterization, we show how to generate an infinite number of equivalent games, not requiring to run repeatedly the complete algorithm. Simulation results validate the effectiveness of the proposed algorithms.**

***Index Terms*—Inverse Differential Game, Inverse Optimal Control, Integral Reinforcement Learning, Continuous-time linear systems**

I. Introduction

DYNAMIC Game Theory is a branch of game theory that focuses on games where the strategies of the players can change over time [1]. Four features arise: the possible presence of multiple players (the number of players $N \geq 2$), players' optimizing behavior, enduring consequences of decisions, and robustness against the changing environment [2]. This dynamic aspect has gained significant attention in recent years, as many real-world problems modeled by games involve situations where the parameters of the game are constantly evolving [3]. For example, dynamic games can be used to model the competition of firms in a market, the evolution of political powers, and the interactions between populations in an ecosystem [4]. The typical dynamic games include differential games [4], repeated games, and evolutionary games [5]. The study of these games has far-reaching implications in a range of fields such as economics [6], political science [7], engineering [8], [9], [10], and biology [11]. Most of the literature has focused on determining the outcome of a game given the players' objective functions. Recently, interest grows in the inverse problem, where, given a player's game-playing behavior, one wants to reverse engineer the objective of this player.

Inverse problems are particularly relevant in guiding a game-playing system to some desired behavior. Inverse Reinforcement Learning (IRL), first introduced in [12], solves the inverse problem in a Markov Decision Process (MDP), using, e.g., maximum entropy methods [13], [14]. Inverse Optimal Control (IOC), a closely related field with a long history, has focused on developing mathematical models and algorithms for inferring the objectives and constraints of a system in view of observed behavior. One of the earliest works in this area is the classic paper by Anderson in 1966 [15], which introduced a linear-quadratic framework for inverse optimal control of linear systems. Further development of this framework leads to more results on IOC, e.g., [16], [17]. IRL and IOC are concerned with similar problems, but differ in structure - the IOC aims to reconstruct an objective function given the state/action samples assuming dealing with a stable control system, while the IRL recovers an objective function using expert demonstration assuming that the expert behavior is optimal [18].

Non-cooperative differential games were first introduced in [19] for zero-sum games. In this work, we consider a particular type of differential game - the LQ nonzero-sum game. This type of game is closely related to Linear Quadratic Regulator (LQR) problem – the dynamics of the system are described by ordinary differential equations and the cost function is quadratic. Thus, those methods used for solving IOC problems can be exploited for solving inverse differential games [20]. There are various works dedicated to the inverse problem for non-cooperative linear-quadratic differential games. Some of them use purely IRL approaches [21], [22], while others are based on IOC [23], [24].

Our work considers LQ N-player differential games with heterogeneous players whose the control input matrices and cost function parameters are different. The solution for the considered type of game, namely its Nash equilibrium, is found via solving the so-called Algebraic Riccati Equation (AREs) [25], [3]. We exploit the result of [26] to accomplish this task. Further, instead of seeking the cost functions that, together with the dynamics, generated the demonstrated behavior, we look for an equivalent cost function that, together with the given dynamics, synthesizes a game that shares the same feedback laws with the original game. This can be done via model-based and model-free algorithms presented in this paper. The model-free algorithm, as an extension of the model-based version, is developed relying on he ideas of [27], [28] and [29] and using integral RL [30]. The extended algorithm possesses the same analytical properties as the model-based one. After characterizing the solution, we show that using the

heterogeneity of the players, the output of the algorithms can be further adjusted allowing to generate an infinite number of equivalent games by exploiting such an algorithm again (thus, low computational costs).

The paper is structured as follows. Section II provides preliminary results on LQ nonzero-sum N-player differential games and formulates the problem addressed in the paper. In section III, we describe each step of the model-based algorithm. Section IV is dedicated to the analysis of the algorithm; we show its convergence and stability and explain how to adjust the output of the algorithm via solution characterization. In section V we provide the model-free extension of the algorithm and show the equivalence of analytical results. Sections VI and VII provide simulation results and conclusion, respectively.

Notations: For a matrix $P \in \mathbb{R}^{m\times n}$, P^k, $P^{(k)}$ denote P to the power of k, and the matrix P at the k-th iteration, respectively. In addition, $P > 0$, $P \geq 0$, $P \leq 0$, and $P < 0$, denote positive (semi-)definiteness, and negative (semi-)definiteness of the matrix P, respectively. The notations $\{P_i\}_{i=1}^N$ and $\{P_{ij}\}_{i,j=1}^N$ denotes the sets of matrices $P_1,\dots,P_N$ and $P_{11},\dots,P_{1N},P_{21},\dots,P_{NN}$, respectively. The notation $\mathrm{tr}P$ denotes the trace of the matrix P. I_k is the $k\times k$ identity matrix.

II. Problem Formulation

This section introduces linear-quadratic (LQ) nonzero-sum differential games. We define stationary linear feedback Nash equilibrium (further referred to as NE). We clarify what an optimal behavior for the game is and introduce the inverse differential games.

A. LQ Nonzero-sum Differential Game

Consider a differential game with N players, labeled by $1,...,N$, under the continuous time dynamics

$$\begin{aligned} &\dot{x}(t) = Ax(t) + \sum_{i=1}^{N} B_i u_i(t), \quad i = 1,\dots,N, \\ &x(0) = x_0 \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state and $u_i \in \mathbb{R}^{m_i}$ is the control input of players i; the plant matrix A, control input matrices B_i have appropriate dimensions.

We consider that the players select their control to be

$$u_i(t) = F_i x(t), \quad i = 1,\dots,N \tag{2}$$

where F_i is an $m_i \times n$ time-invariant feedback matrix of player i. Further, to ease notations, we use $x(t) = x$, $u_i(t) = u_i$ for $i = 1,\dots,N$.

We use $u_{-i} = (u_1,\dots,u_{i-1},\dots,u_{i+1},\dots,u_N)$ to denote an action profile of all the players except for player i. Within the game, player i aims to find a controller u_i that minimizes its cost function $J_i(x_0,u_i,u_{-i})$, which takes the quadratic form

$$J_i(x_0,u_i,u_{-i}) = \int_0^\infty \Big(x^\top Q_i x + \sum_{j=1}^{N} u_j^\top R_{ij} u_j \Big)\, dt, \tag{3}$$

where $Q_i \in \mathbb{R}^{n\times n}$, $R_{ij} \in \mathbb{R}^{m_j\times m_j}$ are symmetric and $R_{ii} > 0$ for $i,j = 1,\dots,N$.

A Nash equilibrium (u_i^*, u_{-i}^*) of the game is characterized by

$$J_i(x_0,u_i^*,u_{-i}^*) \leq J_i(x_0,u_i,u_{-i}^*), \quad i = 1,\dots,N. \tag{4}$$

According to [2, Theorem 8.5], for each player i the cost function under the NE control inputs satisfies

$$J_i(x_0,u_i^*,u_{-i}^*) = x_0^\top K_i x_0 \tag{5}$$

where K_i is a symmetric matrix, sometimes referred to as the value matrix, satisfying the following Algebraic Riccati Equations (AREs)

$$\begin{aligned} &A^\top K_i + K_i A + Q_i + \sum_{j=1}^{N} F_j^\top R_{ij} F_j - \\ &\Big(\sum_{j=1}^{N} F_j^\top B_j^\top\Big) K_i - K_i \Big(\sum_{j=1}^{N} B_j F_j\Big) = 0, \end{aligned} \tag{6}$$

where F_i, for each $i = 1,\dots,N$, is given by

$$F_i = R_{ii}^{-1} B_i^\top K_i, \tag{7}$$

and the control trajectories are

$$u_i^* = -F_i x = -R_{ii}^{-1} B_i^\top K_i x, \quad i = 1,\dots,N. \tag{8}$$

We restrict the set of admissible controller matrices $(F_1,\dots,F_N)$ to the following set

$$\mathscr{F} = \{(F_1,\dots,F_N) | A + \sum_{j=1}^{N} B_j F_j \quad \text{is stable}\}, \tag{9}$$

since $(u_1^*,\dots,u_N^*)$ need to stabilize trajectories to qualify as the NE equilibrium in this game [2]. This restriction is essential because, as shown in [31], without this restriction it is possible to construct an example where a non-stabilizing feedback yields a lower cost for one of the player while other players stick to the stabilizing feedback law. Thus, besides satisfying (6), K_i for $i = 1,\dots,N$ should also be stabilizing to lead to an NE [2]. Thus, the system (1) in the LQ differential games is always assumed to be stabilizable, i.e., $(A,[B_1,\dots,B_N])$ is stabilizable.

B. Inverse LQ Nonzero-sum Differential Game

We formulate the inverse problem for LQ nonzero-sum differential games in this subsection.

Consider an LQ differential game (referred to as the observed LQ game) with continuous-time system dynamics

$$\dot{x}_d = Ax_d + \sum_{i=1}^{N} B_i u_{i,d}, \quad x_d(0) = x_{0,d} \tag{10}$$

where $x_d \in \mathbb{R}^n$, $u_{i,d} \in \mathbb{R}^{m_i}$ are the demonstrated NE trajectories of the observed LQ game with $u_{i,d}$ being the trajectory of player i for $i = 1,\dots,N$; A, B_i have appropriate dimensions. The cost functions of the game have the following known quadratic structure

$$J_i(x_0,u_i,u_{-i}) = \int_0^\infty \Big(x^\top Q_{i,d} x + \sum_{j=1}^{N} u_j^\top R_{ij,d} u_j) \Big) dt, \tag{11}$$

with the *unknown* symmetric matrices $Q_{i,d}$ and $R_{ij,d}$ where $R_{ii,d} > 0$ for $i, j = 1, \ldots, N$. Considering that $(x_d, u_{1,d}, \ldots, u_{N,d})$ are NE trajectories, we have

$$u_{i,d} = -F_{i,d}x_d = -R_{ii,d}^{-1}B_i^\top K_{i,d}x_d, \tag{12}$$

where $K_{i,d}$ is the stabilizing symmetric solution of the following AREs

$$\begin{aligned} &A^\top K_{i,d} + K_{i,d}A + Q_{i,d} + \sum_{j=1}^{N} F_{j,d}^\top R_{ij,d}F_{j,d} - \\ &\big(\sum_{j=1}^{N} F_{j,d}^\top B_j^\top\big)K_{i,d} - K_{i,d}\big(\sum_{j=1}^{N} B_j F_{j,d}\big) = 0. \end{aligned} \tag{13}$$

Remark 1. *Note that we do not make any assumption on stabilizability of the system because it follows from the existence of demonstrated NE trajectories.*

We use the $(A, \{B_i\}_{i=1}^N, \{Q_{i,d}\}_{i=1}^N, \{R_{ij,d}\}_{i,j=1}^N)$ tuple to describe an LQ differential game with the dynamics' matrices $A, B_1, \ldots, B_N$ and the cost function parameters $Q_{1,d}, \ldots, Q_{N,d}$ and $R_{11,d}, \ldots, R_{1N,d}, R_{21,d}, \ldots, R_{NN,d}$.

Definition II.1. *(Equivalent Game). The $(A, \{B_i\}_{i=1}^N, \{Q_i\}_{i=1}^N, \{R_{ij}\}_{i,j=1}^N)$ game is said to be equivalent to the observed game $(A, \{B_i\}_{i=1}^N, \{Q_{i,d}\}_{i=1}^N, \{R_{ij,d}\}_{i,j=1}^N)$ if its AREs* (6) *has a stabilizing solution $\{K_i\}_{i=1}^N$ such that $R_{ii}^{-1}B_i^\top K_i = R_{ii,d}^{-1}B_i^\top K_{i,d}$ (i.e., $F_i = F_{i,d}$) where $\{K_{i,d}\}_{i=1}^N$ is a solution of AREs* (13) *associated with the observed game.*

In other words, the games are equivalent if they share the same equilibrium feedback laws $F_{i,d} = F_i$ for all player $i = 1, \ldots, N$.

Now, we are ready to formulate the inverse problem to be addressed in this paper.

Inverse Differential Game Problem: Given the demonstrated trajectories $(x_d, u_{1,d}, \ldots, u_{N,D})$ of the observed game $(A, \{B_i\}_{i=1}^N, \{Q_{i,d}\}_{i=1}^N, \{R_{ij,d}\}_{i,j=1}^N)$, find the cost function parameters $\{Q_i, R_{ij}\}_{i,j=1}^N$ that synthesize a game $(A, \{B_i\}_{i=1}^N, \{Q_i\}_{i=1}^N, \{R_i j\}_{i,j=1}^N)$ which is equivalent to the observed game.

We solve the problem using model-based and model-free algorithms presented in the following sections.

III. Model-based Inverse Reinforcement Learning Algorithm

This section describes the algorithm that uses the demonstrated equilibrium trajectories $(x_d, \{u_{i,d}\}_{i=1}^N)$ generated by the *known* dynamics $(A, \{B_i\}_{i=1}^N)$ for learning a set of cost function parameters equivalent to $(Q_{i,d}, R_{ij,d})$ for $i, j = 1, \ldots, N$.

The algorithm consists of the following steps - firstly, we use the demonstrated data to estimate the set of target feedback laws $\hat{F}_i = F_{i,d}$ that are supposed to be a set of equilibrium feedback laws both for the original game and the one generated by the algorithm. The next step is the initialization of the parameters $(Q_i^0,\ R_{ij})$ for $i, j = 1, \ldots, N$. Note that the algorithm only updates Q_i's parameters while R_{ij}'s remain the same during the iterative procedure. Then, using the initialized parameters and the known dynamics, we calculate the set of stabilizing solutions of the resulting set of AREs and the corresponding feedback laws. Using the initialized feedback laws $F_i^{(k)}$, we apply the gradient descent method [32] to update $K_i^{(k)}$ in the direction of the minimization of the difference between $F_i^{(k)}$ and $\hat{F}_i$ for $k = 0, 1, \ldots$. After each iteration, using the inverse optimal control [33], we update Q_i^{k+1} substituting the result of the gradient descent update $K_i^{(k+1)}$.

The model-based algorithm requires to know matrices of the game dynamics. Hence, in this section we make the following assumption.

Assumption 1. *The game dynamics matrices $(A, B_1, \ldots, B_N)$ are known.*

A. Feedback Law Estimation

In this step we aim to track the difference between the the current iteration k feedback laws and the desired ones. Using the observed data $(x_d, \{u_{i,d}\}_{i=1}^N)$, we derive the estimation $\hat{F}_i$ of the target feedback law $F_{i,d}$ by applying the batch least-square (LS) method [34]. To implement the estimation procedure we sample the demonstrated trajectories to obtain

$$\begin{aligned} \hat{x}_d &= [x_d(t_1), \ldots, x_d(t_s)] \in \mathbb{R}^{n \times s}, \\ \hat{u}_{i,d} &= [u_{i,d}(t_1), \ldots, u_{i,d}(t_s)] \in \mathbb{R}^{m_i \times s}, \end{aligned} \tag{14}$$

for $i = 1, \ldots, N$ where $s \geq n$, $s \in \mathbb{Z}_+$. Using (12), we estimate $\hat{F}_i$ by calculating

$$\hat{F}_i = -\hat{u}_{i,d}\hat{x}_d^\top(\hat{x}_d\hat{x}_d^\top)^{-1}, \tag{15}$$

for $i = 1, \ldots, N$. Note that the sampling should guarantee that $\hat{x}_d\hat{x}_d$ is full rank, i.e., the rank should be n.

B. Initialized Game

In the next step, we generate an initial set of parameters $\{Q_i^{(0,0)}, \{R_{ij}\}_{j=1}^N\}_{i=1}^N$. Together with the matrices $(A, \{B_i\}_{i=1}^N)$ we have a nonzero-sum linear quadratic differential game. To find the equilibrium set $\{F_i^{(\infty,0)}\}_{i=1}^N$ for the generated game, one needs to solve the following set of equations

$$\begin{aligned} &A^\top K_i^{(0,0)} + K_i^{(0,0)}A + Q_i + \sum_{j=1}^{N} F_j^{(0,0)\top} R_{ij} F_j^{(0,0)} - \\ &\big(\sum_{j=1}^{N} F_j^{(0,0)\top} B_j^\top\big)K_i^{(0,0)} - K_i^{(0,0)}\big(\sum_{j=1}^{N} B_j F_j^{(0,0)}\big) = 0, \end{aligned} \tag{16}$$

where $F_i^{(0,0)} = R_{ii}^{-1}B_i^\top K_i^{(0,0)}$. This set of AREs can be solved using a modified version, for the multiplayer case, of the algorithm of the Lyapunov Iterations presented in [26]. The algorithm includes initialization of $F_i^{(0,0)}$ that should form stable dynamics, i.e.,

$$A - \sum_{i=1}^{N} B_i F_i^{(0,0)} \quad \text{is stable.} \tag{17}$$

However, since $\{\hat{F}_i\}_{i=1}^N$ is derived using the estimation procedure and known to be a set of equilibrium feedback laws, one can skip the initialization step for solving the set of AREs by

setting $F_i^{(0,0)} = \hat{F}_i$. Thus, the algorithm used to solve initialized game is the following

$$\begin{aligned}
&(A - \sum_{j=1}^{N} B_j F_j^{(k,0)})^\top K_i^{(k+1,0)} + K_i^{(k+1,0)}(A - \sum_{j=1}^{N} B_j F_j^{(k,0)}) = \\
&-\tilde{Q}_i = -(Q_i + \sum_{j=1}^{N} F_j^{(k,0)\top} R_{ij} F_j^{(k,0)}), \\
&F_i^{(k+1,0)} = R_{ii}^{-1} B_i^\top K_i^{(k+1,0)}
\end{aligned} \tag{18}$$

for iterations $k = 0, 1, \ldots$. The procedure continues until $\|K_i^{(k+1,0)} - K_i^{(k,0)}\| \leq \varepsilon_i$ where ε_i is some positive constant for $i = 1, \ldots, N$. From [26], we know that under some mild conditions, the algorithm converges to a set of *positive definite stabilizing* solutions $\{K_i^{(\infty,0)}\}_{i=1}^N$, i.e.,

$$\lim_{k \to \infty} K_i^{(k,0)} = K_i^{\infty}, \tag{19}$$

where $K_i^{(\infty,0)}$ are such that $A - \sum_{i=1}^N K_i^{(\infty,0)}$ is stable. Because (18) is a set of the Lyapunov Equations, the conditions that ensure the uniqueness of the set are the following

$$Q_i > 0, R_{ii} > 0, R_{ij} \geq 0 \tag{20}$$

for $i, j = 1, \ldots, N$ and $i \neq j$.

Thus, we set the initialized parameters $Q_i^{(0)}$ and R_{ij} as positive definite for $i = j$ and positive semi-definite for $i \neq j$, $i, j = 1, \ldots, N$. Note that further, in Section IV-C, dedicated to the solution characterization, we show that R_{ij} and the resulting Q_i's can be adjusted relaxing the imposed constraint.

After solving the initialized game, we set $K_i^{(\infty,0)} = K_i^{(0)}$ and correspondingly $F_i^{(\infty,0)} = F_i^{(0)}$.

C. Gradient Descent Update

In this section we present the way we track the difference between the estimated controller $\hat{F}_i$ and $F_i^{(p)} = R_{ii}^{-1} B_i^\top K_i^{(p)}$, where $p = 0, 1, 2, \ldots$ is an iteration step and $K_i^{(0)}$ are the solution of the initialized problem (19) for $i = 1, \ldots, N$. This step is performed using the gradient descent algorithm [32]. We define the following functions

$$d_i^{(p)}(K_i) \coloneqq F_i^{(p)} - \hat{F}_i = R_{ii}^{-1} B_i^\top K_i^{(p)} - \hat{F}_i \tag{21}$$

which track the difference between the target feedback law $\hat{F}_i$ and the current iteration feedback law $F_i^{(p)}$ for player $i = 1, \ldots, N$. Next, we introduce the function $D_i^{(p)}$ of K_i as follows

$$D_i^{(p)}(K_i) = \operatorname{tr}\left(d_i^{(p)\top} d_i^{(p)}\right) \geq 0, \quad i = 1, \ldots, N \tag{22}$$

which we minimize with respect to $K_i^{(p)}$. The update rule is the following

$$K_i^{(p+1)} = K_i^{(p)} - \alpha_i \frac{\partial D_i^{(p)}}{\partial K_i}, \tag{23}$$

for $i = 1, \ldots, N$ where $\alpha_i \geq 0$ is the learning rate for player i. Considering (7), (21) and (22), we compute the partial derivative as follows

$$\begin{aligned}
\frac{\partial D_i^{(p)}}{\partial K_i} &= K_i^{(p)} B_i R_{ii}^{-1} R_{ii}^{-1} B_i^\top + B_i R_{ii}^{-1} R_{ii}^{-1} B_i^\top K_i^{(p)} - \\
&\quad \hat{F}_i^\top R_{ii}^{-1} B_i^\top - B_i R_{ii}^{-1} \hat{F}_i \\
&= (F_i^{(p)} - \hat{F}_i)^\top R_{ii}^{-1} B_i^\top + B_i R_{ii}^{-1} (F_i^{(p)} - \hat{F}_i) \\
&= d_i^{(p)\top} R_{ii}^{-1} B_i^\top + B_i R_{ii}^{-1} d_i^{(p)}.
\end{aligned} \tag{24}$$

At each iteration $p = 0, 1, \ldots$ we have bounded $d_i^{(p)}$ for $i = 1, \ldots, N$ because $d_i^{(0)} = F_i^{(\infty,0)} - \hat{F}_i$ where $F_i^{(0)}$ is the solution of the initialized problem and the following

$$C_i = \|d_i^{(0)}\| > \|d_i^{(1)}\| > \cdots \geq 0 \tag{25}$$

as the result of the minimization procedure where $C_i \geq 0$ is a constant for $i = 1, \ldots, N$.

D. Inverse Update of the Parameters

After the update (23), we use $K_i^{(p+1)}$ to evaluate $Q_i^{(p+1)}$ for $i = 1, \ldots, N$. This is done via substituting the derived values into

$$\begin{aligned}
&Q_i^{(p+1)} = -A^\top K_i^{(p+1)} - K_i^{(p+1)} A - \\
&\sum_{j=1}^{N} F_j^{(p+1)\top} R_{ij} F_j^{(p+1)} + \big(\sum_{j=1}^{N} F_j^{(p+1)\top} B_j^\top\big) K_i^{(p+1)} + \\
&K_i^{(p+1)} \big(\sum_{j=1}^{N} B_j F_j^{(p+1)}\big),
\end{aligned} \tag{26}$$

where $F_i^{(p+1)} = R_{ii}^{-1} B_i^\top K_i^{(p+1)}$ for $i = 1, \ldots, N$.

The described iterative procedure is repeated till for some δ_i, $0 \leq D_i^{(p)} \leq \delta_i$ is achieved where δ_i are desired precision measures that describe how close the generated parameters are to the desired result for each player $i = 1, \ldots, N$. The resulting Q_i^*, together with the initialized R_{ij} for $i, j = 1, \ldots, N$ and the known dynamics $(A, \{B_i\}_{i=1}^N)$ form an equivalent LQ nonzero-sum game as described in Definition II.1. Hence, we have a new set of Algebraic Riccati Equations

$$\begin{aligned}
&Q_i^* = -A^\top K_i^* - K_i^* A - \sum_{j=1}^{N} F_j^{*\top} R_{ij} F_j^* + \\
&\big(\sum_{j=1}^{N} F_j^{*\top} B_j^\top\big) K_i^* + K_i^* \big(\sum_{j=1}^{N} B_j F_j^*\big),
\end{aligned} \tag{27}$$

where K_i^* is the final result of (23) and $F_i^* = R_{ii}^{-1} B_i^\top K_i^* = \hat{F}_i$ for $i = 1, \ldots, N$.

Remark 2. *From the complexity point of view, the demanding parts of the algorithm are finding solutions of the game with the initialized parameters $\{Q_i^{(0)}, R_{ij}\}_{i,j=1}^N$ and matrix multiplication done in the following steps. Implementing the Lyapunov Iterations with respect to $K_i \in \mathbb{R}^{n \times n}$ usually has complexity $\mathcal{O}(n^3)$ [35]. The steps of the algorithm that require performing matrix multiplication via standard methods have complexity $\mathcal{O}(n^3 + n^2 m + n m^2)$ where $m = \max(m_1, \ldots, m_N)$. Hence, the overall computational complexity is $\mathcal{O}(n^3 + n^2 m + n m^2)$.*

Remark 3. *In fact, the implementation of **Algorithm 1** does not necessarily require the iterative update of $Q_i^{(p+1)}$ in step 6. This update might be done only* once *after the desired precision δ_i is achieved, i.e., after getting $K_i^{(p+1)}$ in step 5 such that $\mathrm{tr}(d_i^{(p)\top} d_i^{(p)}) < \delta_i$ for $i = 1,\dots,N$. This would reduce the computational cost of the algorithm. On the other hand, steps* 4, 5 *and* 6 *can be combined by substituting* (31) *and* (32) *into* (33)*.*

Remark 4. *Step* 3 *is only necessary to derive solutions for the game with the initialized parameters. Suppose we are given a set of game parameters $\{Q_i', R_{ij}'\}_{i,j=1}^N$ and the solution for that game $\{K_i', F_i'\}_{i=1}^N$ is known to have the same dynamics as the observed game. Then, considering Remark 3, we only need to perform iterative optimization via steps 4-5 and a single update in step 6. The same applies if A is known to be stable. In that case, one can skip Step* 3 *and set $K_i^{(0)} = \mathbf{0}_{n\times n} \in \mathbb{R}^{n\times n}$ and $F_i^{(0)} = \mathbf{0}_{m_i\times n} \in \mathbb{R}^{m_i\times n}$ for $i = 1,\dots,N$ where $\mathbf{0}$ denotes a zero matrix of particular dimension.*

IV. Analysis of the Model-based Algorithm

This section is dedicated to the analysis of the model-based algorithm – **Algorithm 1**. Firstly, we show the convergence of the algorithm. Next, we prove that the output of the algorithm to solve the problem, i.e., $F_{i,d}$, the feedback laws used to generate the equilibrium trajectories $(x_d, \{u_{i,d}\}_{i=1}^N$, are equilibrium trajectories for the synthesized game. In the end, we give the characterization of the solutions that allows to create other equivalent games.

We need to introduce the following notations

$$A_{cl}^{(k,0)} \coloneqq A - \sum_{i=1}^N B_i F_i^{(k,0)}, \tag{34}$$

$$g_i(d_i^{(p)}) \coloneqq d_i^{(p)\top} R_{ii}^{-1} B_i^\top + B_i R_{ii}^{-1} d_i^{(p)} = g_i^{(p)}, \tag{35}$$

where $g_i^{(p)}$ is the symmetric matrix for $p = 0,1,\dots$ and $i = 1,\dots,N$.

A. Convergence Analysis

The result on the convergence is formulated in the theorem below.

Theorem IV.1. *In **Algorithm 1**, the state reward parameters $Q_i^{(p)}$ converge to Q_i^* for $i = 1,\dots,N$. Furthermore, Q_i^* together with the initialized R_{ij}, $i,j = 1,\dots,N$, and the dynamics matrices $(A, \{B_i\}_{i=1}^N)$ form AREs with the stabilizing solution K_i^* such that*

$$R_{ii}^{-1} B_i^\top K_i^* = R_{ii,d}^{-1} B_i^\top K_{i,d} = F_{i,d}. \tag{36}$$

Proof. After the initialization procedure, we get $\{F_i^{(\infty,0)}\}_{i=1}^N$ such that $A_{cl}^{(\infty,0)}$ is stable. Consider the update rule (32). The gradient descent update drives the initialized $F_i^{(0)} = F_i^{(\infty,0)}$ to the estimation of the target feedback law $\hat{F}_i$ for $i = 1,\dots,N$. Hence, the function that is optimized satisfies

$$0 \le D_i^{(p+1)} < D_i^{(p)}, \quad i = 1,\dots,N,\ p = 0,1,\dots. \tag{37}$$

Algorithm 1 Model-based Inverse Reinforcement Learning Algorithm

1) Initialize $R_{ii} > 0$, $R_{ij} \ge 0$ and $Q_i^{(0)} > 0$ for $i,j = 1,\dots,N$, $i \ne j$. Sample data from demonstrated $(x, \{u_{i,d}\}_{i=1}^N)$ to generate $(\hat{x}, \{\hat{u}_{i,d}\}_{i=1}^N)$. Set $k = 0$ and $p = 0$.
2) Derive the estimation of $F_{i,d}$ using the sampled data as
$$\hat{F}_i = -\hat{u}_{i,d}\hat{x}_d^\top (\hat{x}_d \hat{x}_d^\top)^{-1}. \tag{28}$$
3) Set $F_i^{(0,0)} = \hat{F}_i$, compute $K_i^{(k+1,0)}$ from
$$(A - \sum_{i=1}^N B_i F_i^{(k,0)})^\top K_i^{(k+1,0)} + K_i^{(k+1,0)}(A - \sum_{i=1}^N B_i F_i^{(k,0)}) = -(Q_i^{(0)} + \sum_{j=1}^N F_j^{(k,0)\top} R_{ij} F_j^{(k,0)}), \tag{29}$$
update
$$F_i^{(k+1,0)} = R_{ii}^{-1} B_i^\top K_i^{(k+1,0)}, \tag{30}$$
and set $k = k+1$ till $\|K_i^{(k+1,0)} - K_i^{(k,0)}\| < \varepsilon_i$ where ε_i is a small positive constant for $i = 1,\dots,N$.
4) Set $K_i^{(0)} = K_i^{(k+1,0)}$, $F_i^{(0)} = F_i^{(k+1,0)}$. Evaluate the difference
$$d_i^{(p)} = F_i^{(p)} - \hat{F}_i. \tag{31}$$
5) Update $K_i^{(p+1)}$ and $F_i^{(p+1)}$ for $i = 1,\dots,N$ as
$$\begin{aligned} K_i^{(p+1)} &= K_i^{(p)} - \alpha_i \Big(d_i^{(p)\top} R_{ii}^{-1} B_i^\top + B_i R_{ii}^{-1} d_i^{(p)} \Big), \\ F_i^{(p+1)} &= R_{ii}^{-1} B_i^\top K_i^{(p+1)}. \end{aligned} \tag{32}$$
6) Perform evaluation of $Q^{(p+1)}$ as
$$\begin{aligned} Q_i^{(p+1)} = &-A^\top K_i^{(p+1)} - K_i^{(p+1)} A - \sum_{j=1}^N F_j^{(p+1)\top} R_{ij} F_j^{(p+1)} \\ &+ \big(\sum_{j=1}^N F_j^{(p+1)\top} B_j^\top \big) K_i^{(p+1)} + K_i^{(p+1)} \big(\sum_{j=1}^N B_j F_j^{(p+1)} \big). \end{aligned} \tag{33}$$
7) Set $p = p+1$. Perform steps 4-6 till $\mathrm{tr}(d_i^{(p)\top} d_i^{(p)}) < \delta_i$ where δ_i is a small positive constant for $i = 1,\dots,N$.

Thus, the following can be deduced

$$\begin{aligned} &\lim_{p\to\infty} D_i^{(p)} = 0, \quad \lim_{p\to\infty} d_i^{(p)} = 0 \\ &\lim_{p\to\infty} g_i^{(p)} = 0, \quad i = 1,\dots,N. \end{aligned} \tag{38}$$

Thus,

$$\lim_{p\to\infty} K_i^{(p+1)} = \lim_{p\to\infty} (K_i^{(p)} - \alpha_i g_i^{(p)}) = \lim_{p\to\infty} K_i^{(p)} \tag{39}$$

and, since $\hat{F}_i = F_{i,d}$, one can conclude

$$\lim_{p\to\infty} R_{ii}^{-1} B_i^\top K_i^{(p)} = \lim_{p\to\infty} F_i^{(p)} = F_{i,d} = R_{ii,d}^{-1} B_i^\top K_{i,d}. \tag{40}$$

for $i = 1,\dots,N$.

The result of the convergence is denoted by K_i^* for $i = 1,\dots,N$. Substituting $F_i^{(p+1)} = R_{ii}^{-1}B_i^\top K_i^{(p+1)}$ and the gradient descent update (32) in the form

$$K_i^{(p+1)} = K_i^{(p)} - \alpha_i g_i^{(p)} \tag{41}$$

into (33), we get

$$\begin{aligned} Q_i^{(p+1)} &= A^\top(K_i^{(p)} - \alpha_i g_i^{(p)}) + (K_i^{(p)} - \alpha_i g_i^{(p)})A + \\ &\sum_{j=1}^{N}(K_i^{(p)} - \alpha_i g_i^{(p)})B_j R_{jj}^{-1}R_{ij}R_{jj}^{-1}B_j^\top(K_i^{(p)} - \alpha_i g_i^{(p)}) - \\ &\big(\sum_{j=1}^{N}(K_i^{(p)} - \alpha_i g_i^{(p)})B_j R_{jj}^{-1}B_j^\top\big)(K_i^{(p)} - \alpha_i g_i^{(p)}) - \\ &(K_i^{(p)} - \alpha_i g_i^{(p)})\big(\sum_{j=1}^{N} B_j R_{jj}^{-1}B_j^\top(K_i^{(p)} - \alpha_i g_i^{(p)})\big). \end{aligned} \tag{42}$$

Taking the limit and using $F_i^{(p)} = R_{ii}^{-1}B_i^\top K_i^{(p)}$, we get

$$\begin{aligned} \lim_{p\to\infty} Q_i^{(p+1)} &= \lim_{p\to\infty}(A^\top K_i^{(p)} + K_i^{(p)}A + \sum_{j=1}^{N} F_j^{(p)\top}R_{ij}F_j^{(p)} \\ &- \big(\sum_{j=1}^{N} F_j^{(p+1)\top}B_j^\top\big)K_i^{(p+1)} - K_i^{(p)}\big(\sum_{j=1}^{N} B_j F_j^{(p)}\big)\big). \end{aligned} \tag{43}$$

and, as a result,

$$\lim_{p\to\infty} Q_i^{(p+1)} = \lim_{p\to\infty} Q_i^{(p)}, \quad i = 1,\dots,N. \tag{44}$$

Denoting the result of convergence as Q_i^*, we obtain

$$\begin{aligned} Q_i^* &= A^\top K_i^* + K_i^* A + \sum_{j=1}^{N} F_j^{*\top}R_{ij}F_j^* \\ &- \big(\sum_{j=1}^{N} F_j^{*\top}B_j^\top\big)K_i^* - K_i^*\big(\sum_{j=1}^{N} B_j F_j^*\big), \end{aligned} \tag{45}$$

for $i = 1,\dots,N$. Thus, we conclude that $\{K_i^*\}_{i=1}^N$ is the solution set for the AREs associated with $\{Q_i^*, R_{ij}\}_{i,j=1}^N$ where R_{ij} are initialized parameters in *Step 1*. Moreover, from (40), once concludes that it is a stabilizing solution set. ■

B. Stability Analysis

In this section, we show that the output of the algorithm is an equivalent game to the game that has the demonstrated NE trajectories, i.e., $(x_d, u_{i,d})$ for $i = 1,\dots,N$.

Firstly, we need to present the following result, extended for the multiplayer case on LQ nonzero-sum differential games from [2].

Theorem IV.2. *Let $(K_1,\dots,K_N)$ be a symmetric stabilizing solution of equations* (6) *and define $F_i^* := R_{ii}^{-1}B_i^\top K_i$ for $i = 1,\dots,N$. Then $(F_1^*,\dots,F_N^*)$ is the feedback NE. Conversely, if $(F_1^*,\dots,F_N^*)$ is the feedback NE, there exists a symmetric stabilizing solution $(K_1,\dots,K_N)$ of equations* (6) *such that $F_i^* = R_{ii}^{-1}B_i^\top K_i$ for $i = 1,\dots,N$.*

Finally, one can conclude the following for the proposed algorithm.

Theorem IV.3. *Given the demonstrated trajectories $(x_d, u_{i,d})$ for $i = 1,\dots,N$ generated by a game $(A, \{B_i\}_{i=1}^N, \{Q_{i,d}\}_{i=1}^N, \{R_{ij,d}\}_{i,j=1}^N)$ described in Section II, the output of **Algorithm 1** is the tuple $(\{Q_i^*\}_{i=1}^N, \{R_{ij}\}_{i,j=1}^N)$ which combined with the known dynamics matrices $(A, \{B_i\}_{i=1}^N)$, synthesizes a game equivalent to $(A, \{B_i\}_{i=1}^N, \{Q_{i,d}\}_{i=1}^N, \{R_{ij,d}\}_{i,j=1}^N)$, i.e., $F_i^* = F_{i,d}$ for $i = 1,\dots,N$.*

Proof. From (45) we know that K_i^*, $i = 1,\dots,N$ is the solution for AREs with the parameters $(\{Q_i^*\}_{i=1}^N, \{R_{ij}\}_{i,j=1}^N)$ and dynamics $(A, \{B_i\}_{i=1}^N)$. From Theorem IV.1, we know that $F_{i,d} = R_{ii}^{-1}B_i^\top K_i^* = F_i^*$. Since $\{F_{i,d}\}_{i=1}^N$ is the set of stabilizing feedback laws, K_* is the set of stabilizing solutions for AREs with parameters generated by the algorithm and, as a result of Theorem IV.2, one conclude that $\{F_i^*\}_{i=1}^N$ is the feedback NE for the synthesized game $(A, \{B_i\}_{i=1}^N, \{Q_i^*\}_{i=1}^N, \{R_{ij}\}_{i,j=1}^N)$. ■

The next result is the consequence of the previous theoretical results and is important for practical implementation of the algorithm since the results before are valid for infinitely many iterations.

Theorem IV.4. *For each iteration $p = 0,1,\dots$, there exists a set of learning rates $\{\alpha_i\}_{i=1}^N$ such that $\{K_i^{(p+1)}\}_{i=1}^N$ is the stabilizing solution for* (33) *and, as a result, the dynamics $A - \sum_{j=1}^N B_j F_j^{(p+1)}$ is stable.*

Proof. One can check that $K_i^{(p)}$ linearly affects $F_i^{(p)}$. The initial $K_i^{(p)}$ for $p = 0$ is stabilizing as well as the terminal one K_i^* because of (36). Hence, referring to [32], we know that by choosing an appropriate set of $\{\alpha_i\}_{i=1}^N$ one can always have the next iteration of $K_i^{(p)}$, i.e., $K_i^{(p+1)}$ being a stabilizing solution of (33). Thus, at each iteration, a game described by $(A, \{B_i\}_{i=1}^N, \{Q_i^{(p+1)}\}_{i=1}^N, \{R_{ij}\}_{i,j=1}^N)$ has an NE feedback $F_i^{(p+1)} = R_{ii}B_i^\top K_i^{(p+1)}$. ■

C. Characterization of the Solutions

This section provides a result that allows to adjust the output of **Algorithm 1**.

Note that we are looking for $\{Q_i^*, R_{ij}\}_{i,j=1}^N$ such that with the dynamics $(A, B_1,\dots,B_N)$ (6) has a stabilizing solution $\{K_i^*\}_{i=1}^N$ satisfying $R_{ii,d}^{-1}B_i^\top K_{i,d} = R_{ii}^{-1}B_i^\top K_i^*$ for $i = 1,\dots,N$. Since $R_{ii} > 0$, $B_i^\top K_i^* = R_{ii}R_{ii,d}^{-1}B_i^\top K_{i,d}$ for $i = 1,\dots,N$. If any of B_i has no full rank, there might be an infinite number of possible K_i^* [24].

Remark 5. *All possible outputs of **Algorithm 1**, i.e., Q_i^*, R_{ij}, K_i^*, $i,j = 1,\dots,N$, satisfy the following equality*

$$\begin{aligned} &A^\top(K_{i,d} - K_i^*) + (K_{i,d} - K_i^*)A + (Q_{i,d} - Q_i^*) + \\ &\sum_{j=1}^{N} F_j^{*\top}(R_{ij,d} - R_{ij})F_j^* - \\ &\big(\sum_{j=1}^{N} F_j^{*\top}B_j^\top\big)(K_{i,d} - K_i^*) - (K_{i,d} - K_i^*)\big(\sum_{j=1}^{N} B_j F_j^*\big) = 0, \\ &\textit{where} \quad F_i^* = R_{ii}^{-1}B_i^\top K_i^* = F_{i,d}, \quad i = 1,\dots,N. \end{aligned} \tag{46}$$

These equations are obtained by the subtraction of (45) from (13). Let us define

$$\Delta Q_i = Q_i^* - Q_i', \quad \Delta K_i = K_i^* - K_i', \quad \Delta R_{ij} = R_{ij} - R_{ij}', \tag{47}$$

where Q_i^*, R_{ij} and K_i^* are the output of **Algorithm 1** for $i, j = 1,\dots,N$.

Proposition IV.5. *Set $\Delta K_i = 0$ and $\Delta R_{ii} = 0$ for $i = 1,\dots,N$ (i.e., $R_{ii} = R'_{ii}$ and $K_i^* = K'_i$). Then, every Q'_i and R'_{ij} for $i, j = 1,\dots,N$, $j \neq i$ satisfying*

$$(Q_i^* - Q'_i) + \sum_{j=1, j\neq i}^{N} F_j^{*\top}(R_{ij} - R'_{ij})F_j^* = 0 \tag{48}$$

together with R'_{ii} and the dynamics $(A, B_1, \dots, B_N)$ form a new game equivalent to $(A, \{B_i\}_{i=1}^N, \{Q_{i,d}\}_{i=1}^N, \{R_{ij,d}\}_{i,j=1}^N)$.

This is a consequence of a re-scaling of the parameters that does not affect the feedback laws

$$F_i^* = F_{i,d} = F'_i = R'_{ii} B_i^\top K'_i. \tag{49}$$

Hence, we can adjust Q'_i as

$$Q'_i = Q_i^* + \sum_{j=1, j\neq i}^{N} F_j^{*\top}(R_{ij} - R'_{ij})F_j^* \tag{50}$$

or R_{ij} for $i \neq j$ in a desired way scaling Q'_i. Thus, we can generate an infinite number of possible equivalent games and relax the assumption on definiteness of R_{ij}, $i, j = 1,\dots,N$, $j \neq i$.

V. Model-free Inverse Reinforcement Learning Algorithm

This section present the model-free extension of **Algorithm 1**. Real-world applications rarely assume the knowledge of the model of the systems. There are three steps in the algorithm presented before that use the system's dynamics matrices - computation of the solution for the initialized game, gradient descent update and the evaluation of the cost function's parameter upgrade. Although there were a number of works dedicated to partially model-free or model-free methods to solve AREs (e.g. [30], [36], [29], [37]), to extend our algorithm, we use the ideas presented in [28], [27].

A. Model-free Computation of the Initialized Solution

After the initialization of the game parameters R_{ij} and $Q_i^{(0)}$, we need to solve the synthesized game. Using the demonstrated trajectories $(x_d, \{u_{i,d}\}_{i=1}^N)$, we use the auxiliary controls

$$u_i^{(k,0)} = -F_i^{(k,0)} x_d, \quad i = 1,\dots,N \tag{51}$$

where $k = 0, 1, \dots$ is the iteration for the step 3 of the algorithm. Using these controls we rewrite the dynamics

$$\dot{x}_d = Ax_d + \sum_{i=1}^{N} B_i u_{i,d} = Ax_d + \sum_{i=1}^{N} B_i u_i^{(k,0)} + \sum_{i=1}^{N} B_i (u_{i,d} - u_i^{(k,0)}). \tag{52}$$

Using (51), we extend the dynamics as

$$\dot{x}_d = A_{cl}^{(k,0)} x_d + \sum_{i=1}^{N} B_i (u_{i,d} - u_i^{(k,0)}) \tag{53}$$

where $A_{cl}^{(k,0)} = A - \sum_{i=1}^N B_i F_i^{(k,0)}$.

Next, for each $i = 1,\dots,N$ we multiply (18) by $x^\top$ and x to get

$$x_d^\top (A - \sum_{i=1}^{N} B_i F_i^{(k,0)})^\top K_i^{(k+1,0)} x_d + x_d^\top K_i^{(k+1,0)} (A - \sum_{i=1}^{N} B_i F_i^{(k,0)}) x_d = -x_d^\top (Q_i^{(0)} + \sum_{j=1}^{N} F_j^{(k,0)\top} R_{ij} F_j^{(k,0)}) x_d. \tag{54}$$

Rewriting the dynamics term, the following equations hold

$$x_d^\top (A_{cl}^{(k,0)} - \sum_{i=1}^{N} B_i (F_{i,d} - F_i^{(k,0)}))^\top K_i^{(k+1,0)} x_d + x_d^\top K_i^{(k+1,0)} (A_{cl}^{(k,0)} - \sum_{i=1}^{N} B_i (F_{i,d} - F_i^{(k,0)})) x_d = -x_d^\top (Q_i^{(0)} + \sum_{j=1}^{N} F_j^{(k,0)\top} R_{ij} F_j^{(k,0)} + 2\sum_{j=1}^{N} (F_{j,d} - F_j^{(k,0)})^\top B_j^\top K_i^{(k+1,0)}) x_d. \tag{55}$$

Using (10), (51) and (52), we get

$$\dot{x}_d^\top K_i^{(k+1,0)} x_d + x_d^\top K_i^{(k+1,0)} \dot{x}_d = x_d^\top (-Q_i^{(0)} - \sum_{j=1}^{N} F_j^{(k,0)\top} R_{ij} F_j^{(k,0)} - 2(F_{i,d} + F_i^{(k,0)})^\top B_i K_i^{(k+1,0)} - 2\sum_{j\neq i}^{N} (F_{j,d} - F_j^{(k,0)})^\top Y_{ji}^{(k+1)}) x_d. \tag{56}$$

where $Y_{ji}^{(k+1)} = B_j^\top K_i^{(k+1,0)}$ is another auxiliary variable for $j \neq i, i = 1,\dots,N$. Considering (12) and following the ideas of [28] and [27], we integrate the above equation from t to $t+T$ as follows

$$\begin{aligned} &x_d^\top(t+T) K_i^{(k+1,0)} x_d(t+T) - x_d^\top(t) K_i^{(k+1,0)} x_d(t) - \\ &2\int_t^{t+T} (u_{i,d} + F_i^{(k,0)} x_d)^\top R_{ii} F_i^{(k+1,0)} x_d \, d\tau - \\ &2\sum_{j\neq i}^{N} \int_t^{t+T} (u_{j,d} + F_j^{(k,0)} x_d))^\top Y_{ji}^{(k+1)} x_d \, d\tau = \\ &-\int_t^{t+T} x_d^\top (Q_i^{(0)} + \sum_{j=1}^{N} F_j^{(k,0)\top} R_{ij} F_j^{(k,0)}) x_d \, d\tau, \quad i = 1,\dots,N. \end{aligned} \tag{57}$$

Let us consider the above for $k = 0$. Set the initial stabilizable feedback laws $F_i^{(0,0)} = \hat{F}_i$ where $\hat{F}_i$ is estimated in step 2 for $i = 1,\dots,N$. Then, the unknowns in the above equations are

$$K_i^{(k+1,0)}, \quad F_i^{(k+1,0)}, \quad Y_{ji}^{(k+1)}, \quad i, j = 1,\dots,N, i \neq j \tag{58}$$

and each of them is built using $K_i^{(k+1)}$ and $K_j^{(k+1)}$ for i and $j \neq i$. Thus, we solve (57) with respect to the mentioned unknowns till $\|K_i^{(k+1,0)} - K_i^{(k,0)}\| \leq \varepsilon_i$ where $\varepsilon_i > 0$ is a small constant that describes a measure of precision for $i = 1,\dots,N$.

To perform the next steps, matrices B_i, $i = 1,\dots,N$, are needed. One way to evaluate these matrices is to use the computed values of $K_i^{(k+1)}$, $F_i^{(k+1),0}$ and $Y_{ji}^{(k+1)}$. Recall that

$$\begin{aligned} F_i^{(k+1,0)} &= R_{ii}^{-1} B_i^\top K_i^{(k+1,0)}, \\ Y_{ji}^{(k+1)} &= B_j^\top K_i^{(k+1,0)}. \end{aligned} \tag{59}$$

Using the computed values from equation (57) associated with any player $i \in \{1,\dots,N\}$, the control input matrices can be evaluated as

$$\begin{aligned} B_i &= (R_{ii}F_i^{(k+1,0)}(K_i^{(k+1,0)})^{-1})^\top, \\ B_j &= (Y_{ji}^{(k+1)}(K_i^{(k+1,0)})^{-1})^\top, \quad j \neq i. \end{aligned} \tag{60}$$

Note that the inverse $K_i^{(k+1,0)}$ exists because the initialized cost function parameters guarantee $K_i^{(k+1,0)} > 0$ for $i = 1,\dots,N$.

Theorem V.1. *The solution* $\{K_i^{(k+1,0)}\}_{i=1}^N$ *of* (57) *is a unique positive definite stabilizing solution and is the same as the solution of* (29)*.*

Proof. We give a short proof here that follows [28] and [27]. We can reverse engineer (57) taking its $\lim_{T\to 0}$ and using L'Hopital's rule [38] to derive (29). According to [26], the solution of (29) is a unique positive definite solution for the cost function parameters satisfying $Q_i > 0, R_{ii} > 0, R_{ij} \geq 0$ for $i \neq j$, $i, j = 1,\dots,N$. Thus, we conclude that (57) has the same solution as (29) that is a stabilizing positive definite one. ■

After the computation of the initial solution $\{K_i^{(k+1,0)}\}_{i=1}^N$ is accomplished , as it is done in step 4, we drop the iteration counter and set

$$K_i^{(k+1,0)} = K_i^{(0)}, \quad F_i^{(k+1,0)} = F_i^{(0)}, \quad i = 1,\dots,N. \tag{61}$$

B. Mode-free Inverse Update of the Parameters

Since we evaluated B_i Step 5 can be used as it is in Algorithm 1.

Remark 6. *In fact, using* (7) *and the values in* (61)*, one can conclude the following*

$$\begin{aligned} F_i^{(0)} &= R_{ii}^{-1}B_i^\top K_i^{(0)}, \\ F_i^{(0)}(K_i^{(0)})^{-1} &= R_{ii}^{-1}B_i^\top \end{aligned} \tag{62}$$

because $K_i^{(0)}$ *is guaranteed to be a positive definite solution of* (57) *as it is shown in Theorem V.1 for* $i = 1,\dots,N$*. Thus, step 5 can also be rewritten as*

$$\begin{aligned} K_i^{(p+1)} &= K_i^{(p)} - \alpha_i\Big(d_i^{(p)\top}F_i^{(0)}(K_i^{(0)})^{-1} + (K_i^{(0)})^{-1}F_i^{(0)\top}d_i^{(p)}\Big), \\ F_i^{(p+1)} &= F_i^{(0)}(K_i^{(0)})^{-1}K_i^{(p+1)}. \end{aligned} \tag{63}$$

The last update, step 5 in (33), can also modified to avoid using the unknown matrices. Following the approach used in V-A, one can rewrite (33) as

$$\begin{aligned} x_d^\top Q_i^{(p+1)}x_d = x_d^\top(&-\sum_{j=1}^N F_j^{(p+1)\top}R_{ij}F_j^{(p+1)} - \\ &(A_{cl}^{(p+1)} - \sum_{j=1}^N B_j(F_{j,d} - F_j^{(p+1)}))^\top K_i^{(p+1)} - \\ &K_i^{(p+1)}(A_{cl}^{(p+1)} - \sum_{j=1}^N B_j(F_{j,d} - F_j^{(p+1)})) + \\ &2\sum_{j=1}^N (F_{j,d} - F_j^{(p+1)}))^\top B_j^\top K_i^{(p+1)})x_d. \end{aligned} \tag{64}$$

Integrating both sides of the above equation from t to $t+T'$, we get

$$\begin{aligned} &\int_t^{t+T'} x_d^\top Q_i^{(p+1)}x_d\,d\tau = -\int_t^{t+T'} x_d^\top \sum_{j=1}^N F_j^{(p+1)\top}R_{ij}F_j^{(p+1)}x_d\,d\tau - \\ &x_d^\top(t+T')K_i^{(p+1)}x_d(t+T') + x_d^\top(t)K_i^{(p+1)}x_d(t) - \\ &2\int_t^{t+T'} \sum_{j=1}^N (u_{j,d} + F_j^{(p+1)}x_d)^\top B_j^\top K_i^{(p+1)}x_d. \end{aligned} \tag{65}$$

Since (63) provides us $K_i^{(p+1)}$,$F_i^{(p+1)}$ and the trajectories $(x_d, \{u_{i,d}\}_{i=1}^N)$ are given, $Q_i^{(p+1)}$ can be evaluated. The way it can be done is shown in the next section. All the steps for the model-free **Algorithm 2** are shown below.

Algorithm 2 Model-free Inverse Reinforcement Learning Algorithm

1) Initialize $R_{ii} > 0$, $R_{ij} \geq 0$ and $Q_i^{(0)} > 0$ for $i, j = 1,\dots,N$, $i \neq j$. Sample data from demonstrated $(x, \{u_{i,d}\}_{i=1}^N)$ to generate $(\hat{x}, \{\hat{u}_{i,d}\}_{i=1}^N)$. Set $k = 0$ and $p = 0$.
2) Derive estimation of $F_{i,d}$ using the sampled data as
$$\hat{F}_i = -\hat{u}_{i,d}\hat{x}_d^\top(\hat{x}_d\hat{x}_d^\top)^{-1}. \tag{66}$$
3) Set $F_i^{(0,0)} = \hat{F}_i$ for $i = 1,\dots,N$, solve (57) with respect to $K_i^{(k+1)}$, $F_i^{(k+1,0)}$ and $Y_{ji}^{(k+1)}$ for $i, j = 1,\dots,N$, $j \neq i$. Compute B_i for $i = 1,\dots,N$. Set $k = k+1$ till $\|K_i^{(k+1,0)} - K_i^{(k,0)}\| < \varepsilon_i$ where ε_i is a small positive constant for $i = 1,\dots,N$.
4) Set $K_i^{(0)} = K_i^{(k+1,0)}$, $F_i^{(0)} = F_i^{(k+1,0)}$. Compute $F_i^{(0)}(K_i^{(0)})^{-1}$ and evaluate the difference
$$d_i^{(p)} = F_i^{(p)} - \hat{F}_i. \tag{67}$$
5) Update $K_i^{(p+1)}$ and $F_i^{(p+1)}$ for $i = 1,\dots,N$ as in (63).
6) Perform evaluation of $Q_i^{(p+1)}$ from (65).
7) Set $p = p+1$. Perform steps 4-6 till $\mathrm{tr}(d_i^{(p)\top}d_i^{(p)}) < \delta_i$ where δ_i is a small positive constant for $i = 1,\dots,N$.

Remark 7. *As for* ***Algorithm 1****, the implementation of* ***Algorithm 2*** *does not necessarily require the iterative update of* $Q_i^{(p+1)}$ *in step 6. This update might be done only* once *after the desired precision* δ_i *is achieved, i.e., after getting* $K_i^{(p+1)}$ *in step 5 such that* $\mathrm{tr}(d_i^{(p)\top}d_i^{(p)}) < \delta_i$ *for* $i = 1,\dots,N$*.*

C. Implementation of the algorithm

In this section, we show *one* possible way to implement **Algorithm 2** which is partially based on [28]. For other ways to use the proposed algorithm, the reader can check [27] , [29], [39]. To avoid any confusion due to indexes and terms, we show the algorithm implementation for the two-player case, i.e., $N = 2$. We hope the below description of the implementation clarifies for the reader the implementation of the algorithm in the multiplayer case.

Firstly, we show how to perform evaluation of $K_i^{(k+1,0)}$, $F_i^{(k+1,0)}$ and $Y_{ji}^{(k+1,0)}$ in step 3 from (57). Following [28], the following notations are introduced

$$\begin{aligned}\hat{K}_i &= [k_{i,11}, 2k_{i,12}, \ldots, 2k_{i,1n}, k_{i,22}, 2k_{i,23}, \ldots, k_{i,nn}]^\top \in \mathbb{R}^{n(n+1)/2},\\ \hat{x} &= [x_1^2, x_1x_2, \ldots, x_1x_n, x_2^2, x_2x_3, \ldots, x_n^2]^\top \in \mathbb{R}^{n(n+1)/2}.\end{aligned} \tag{68}$$

where k_{i,l_1l_2} is a particular element of matrix K_i, i.e., $(K_i)_{l_1l_2}$ for $l_1, l_2 = 1, \ldots, n$. We use the following property of the Kronecker product

$$(c^\top \otimes a^\top)\text{vec}(B) = a^\top Bc. \tag{69}$$

Thus, one can rewrite terms in (57) as

$$\begin{aligned}&x_d^\top(t+T)K_i^{(k+1,0)}x_d(t+T) - x_d^\top(t)K_i^{(k+1,0)}x(t) =\\ &(\hat{x}(t+T) - \hat{x}(t))\hat{K}_i^{(k+1,0)},\\ &(u_{i,d} + F_i^{(k,0)}x_d)^\top R_{ii}F_i^{(k+1,0)}x_d = ((x_d^\top \otimes u_{i,d}^\top)(I_n \times R_{ii}) +\\ &(x_d \otimes x_d)(I_n \otimes F_i^{(k,0)\top}R_{ii}))\text{vec}(F_i^{(k+1,0)}),\\ &(u_{j,d} + F_j^{(k,0)}x_d))Y_{ji}^{(k+1)}x_d = ((x_d^\top \otimes u_{j,d}^\top) +\\ &(x_d \otimes x_d)(I_n \otimes F_j^{(k,0)\top}))\text{vec}(Y_{ji}^{(k+1)}).\end{aligned} \tag{70}$$

In addition to the above, we define δ_{xx}, I_{xx} and I_{xu_i} as

$$\begin{aligned}\delta_{xx} &= [\hat{x}(t_1) - \hat{x}(t_0), \hat{x}(t_2) - \hat{x}(t_1), \ldots, \hat{x}(t_s) - \hat{x}(t_{s-1})]^\top,\\ I_{xx} &= \int_{t_0}^{t_1}(x_d \otimes x_d)\,d\tau, \int_{t_1}^{t_2}(x_d \otimes x_d)\,d\tau, \ldots, \int_{t_{s-1}}^{t_s}(x_d \otimes x_d)\,d\tau]^\top\\ I_{xu_i} &= \int_{t_0}^{t_1}(x_d \otimes u_{i,d})\,d\tau, \int_{t_1}^{t_2}(x_d \otimes u_{i,d})\,d\tau, \ldots, \int_{t_{s-1)}}^{t_s}(x_d \otimes u_{i,d})\,d\tau]^\top\end{aligned} \tag{71}$$

where $0 \le t_{l-1} \le t_l$ for $l \in \{0, 1, \ldots, s\}$. Although the data intervals do not need to be equal, in our simulation presented further, we use $t_l - t_{l-1} = T$ for $l \in \{0, 1, \ldots, s\}$.

Then, (57) can be rewritten as

$$H_i^{(k)}\begin{pmatrix}\hat{K}_i^{(k+1,0)}\\ \text{vec}(F_i^{(k+1,0)})\\ \text{vec}(Y_{ji}^{(k+1)})\end{pmatrix} = \Xi_i^{(k)} \tag{72}$$

where

$$\begin{aligned}H_i^{(k)} =&[\delta_{xx}, -2I_{xu_i}(I_n \otimes R_{ii}) - I_{xx}(I_n \otimes F_i^{(k,0)^\top}R_{ii},\\ &- 2I_{xu_i} - I_{xx}(I_n \otimes F_i^{(k,0)^\top}],\\ \Xi_i^{(k)} =& -I_{xx}(Q_i^{(0)} + \sum_{j=1}^{2}F_j^{(k,0)\top}R_{ij}F_j^{(k,0)}).\end{aligned} \tag{73}$$

Then, (72) can be solved as

$$\begin{pmatrix}\hat{K}_i^{(k+1,0)}\\ \text{vec}(F_i^{(k+1,0)})\\ \text{vec}(Y_{ji}^{(k+1)})\end{pmatrix} = (H_i^{(k)\top}H_i^{(k)})^{-1}H_i^{(k)\top}\Xi_i^{(k)}. \tag{74}$$

The equation is solved until the convergence of $\hat{K}_i^{(k+1,0)}$ from which one can recover $K_i^{(k+1,0)}$. Note that the vector of unknowns has $n(n+1)/2 + m_in + m_jn$ parameters. Thus, we need enough data to satisfy $s \ge n(n+1)/2 + m_in + m_jn$.

Remark 8. *If $H_i^{(k)}$ is an invertible square matrix, right side of* (74) *can be computed as $(H_i^{(k)})^{-1}\Xi_i^{(k)}$.*

Although step 6 of **Algorithm 2** can be implemented interactively for every new $K_i^{(p+1)}$, one can implement it only once, as suggested in Remark 7 after the feedback laws converged as a result of the gradient updates, i.e., $\text{tr}(d_i^{(p)\top}d_i^{(p)}) < \delta_i$ for $i = 1, \ldots, N$. Then, set $K_i^{(p+1)} = K_i^*$ and $F_i^{p+1} = F_i^*$. For that one use the same data as in (72). We rewrite one of the terms in (65) as

$$\begin{aligned}&(u_{j,d} + F_j^*x_d)^\top B_j^\top K_i^*x_d =\\ &((x_d^\top \otimes u_{j,d}) + (x_d \otimes x_d))\text{vec}(F_j^{*\top}B_j^\top K_i^*),\\ &x_d^\top Q_i^*x_d = I_{xx}\text{vec}(Q_i^*).\end{aligned} \tag{75}$$

In addition to the above, we define $\hat{Q}_i^*$ and I_{qx} as

$$\begin{aligned}\hat{Q}_i^* &= [q_{i,11}, 2q_{i,12}, \ldots, 2q_{i,1n}, q_{i,22}, 2q_{i,23}, \ldots, q_{i,nn}]^\top,\\ I_{qx} &= [\int_{t_0}^{t_1}\hat{x}\,d\tau, \int_{t_1}^{t_2}\hat{x}\,d\tau, \ldots, \int_{t_{s-1}}^{t_s}\hat{x}\,d\tau]^\top.\end{aligned} \tag{76}$$

Then, using (71), (65) can be rewritten as

$$I_{qx}\hat{Q}_i^* = \Omega_i \tag{77}$$

where

$$\begin{aligned}\Omega_i =& -I_{xx}\sum_{j=1}^{2}\text{vec}(F_j^{*\top}R_{ij}F_j^*) -\\ &\delta_{xx}\hat{K}_i^* - 2\sum_{j=1}^{N}(I_{xu_j} + I_{xx})\text{vec}(F_j^{*\top}B_j^\top K_i^*).\end{aligned} \tag{78}$$

Then, (77) can be solved as

$$\hat{Q}_i^* = (I_{qx}^\top I_{qx})^{-1}I_{qx}^\top\Omega_i. \tag{79}$$

Note, (77) has less unknown parameters than (72) because $\hat{Q}_i^* \in \mathbb{R}^{n(n+1)/2}$. Thus, the previous restriction on s is enough, i.e., $s \ge n(n+1)/2 + m_in + m_jn$.

Remark 9. *Proposition IV.5 also valid in the model-free case. Thus, the value of the output of **Algorithm 2** Q_i^* can be adjusted or the restrictions on definiteness of R_{ij} can be relaxed for $i \ne j$, $i, j = 1, \ldots, N$.*

Remark 10. *Since the equations* (72) *and* (77) *are solved as LQ problems, the probing noise should be injected to satisfy persistence of excitation (PE) condition [27], [28], [36], [39]. The noise can be sinusoids of different frequencies or some random noise. We refer the reader to [40] for more details on that matter.*

Thus, we need to make the following assumption

Assumption 2. *One of the following is true*

- *One can use the estimated stabilizable feedback law $\hat{F}_i$ from* (15) *to apply control inputs $\hat{u}_i = -\hat{F}_ix + \omega_i(t)$, where $\omega_i(t)$ is a noise term, for $i = 1, \ldots, N$ to the system for data collection on the range $(t, t_{\bar{N}})$ at $\bar{N} \ge \max(\bar{n}, \bar{m})$ points. The collection of additional data is performed once.*
- *The demonstrated trajectories were generated under the control inputs $u_{i,d} = -F_{i,d}x_d + \omega_i(t)$ where $\omega_i(t)$ is an*

exponentially decaying noise such that (72) *and* (77) *have a solution. In other words, when the noise decayed significantly, the demonstrated trajectory is* $u_{i,d} \approx -F_{i,d}x_d$ *for* $i = 1,\dots,N$.

VI. Simulations

In this section, we present the simulation results of the algorithms developed in this paper.

A. Model-based Algorithm Simulation

Consider the following continuous time system dynamics

$$\dot{x} = Ax + \sum_{i=1}^{3} B_i u_i, \tag{80}$$

where

$$A = \begin{pmatrix} 3 & -2 \\ 4 & -1 \end{pmatrix}, \quad B_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad B_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{81}$$

The demonstrated NE trajectories are generated for the game with the following weight matrices

$$\begin{aligned} &Q_{1,d} = \begin{pmatrix} 7 & 2 \\ 2 & 5 \end{pmatrix}, \quad Q_{2,d} = 3I_{2\times 2}, \quad Q_{3,d} = I_{2\times 2}, \\ &R_{11,d} = 3, \quad R_{12,d} = 1, \quad R_{13,d} = 1, \\ &R_{21,d} = 1, \quad R_{22,d} = 2, \quad R_{23,d} = 0, \\ &R_{31,d} = 0, \quad R_{32,d} = 1, \quad R_{33,d} = 4. \end{aligned} \tag{82}$$

Given this game, $F_{1,d}$, $F_{2,d}$ and $F_{3,d}$ are

$$\begin{aligned} F_{1,d} &= \begin{pmatrix} 4.2499 & -0.9409 \end{pmatrix}, \\ F_{2,d} &= \begin{pmatrix} -0.4108 & 0.9187 \end{pmatrix}, \\ F_{3,d} &= \begin{pmatrix} 0.2334 & 0.1295 \end{pmatrix}, \end{aligned} \tag{83}$$

with the symmetric solution of AREs

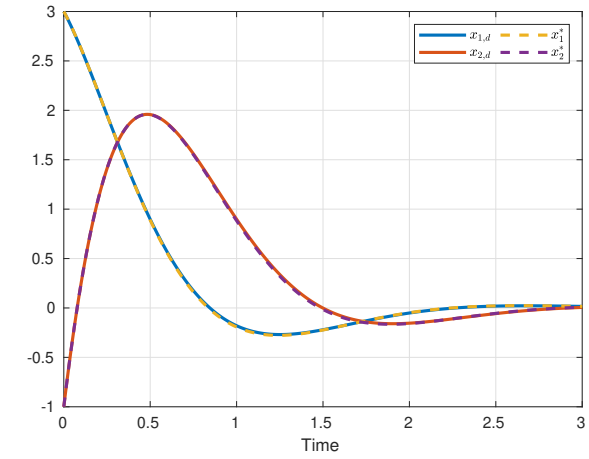
$$\begin{aligned} K_{1,d} &= \begin{pmatrix} 12.7497 & -2.8228 \\ -2.8228 & 3.7172 \end{pmatrix}, \\ K_{2,d} &= \begin{pmatrix} 4.8994 & -0.8216 \\ 0.8216 & 1.8373 \end{pmatrix}, \\ K_{3,d} &= \begin{pmatrix} 0.8116 & 0.1222 \\ 0.1222 & 0.3956 \end{pmatrix}. \end{aligned} \tag{84}$$

The initialized parameters are the following

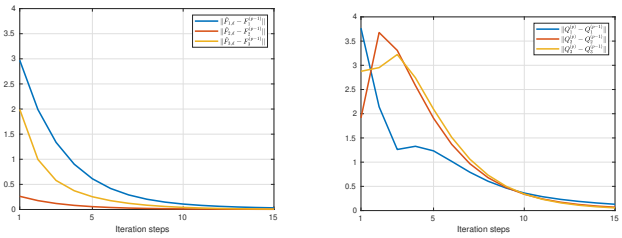
$$\begin{aligned} &Q_1^{(0)} = I_{2\times 2}, \quad Q_2^{(0)} = I_{2\times 2}, \quad Q_3^{(0)} = I_{2\times 2}, \\ &R_{11} = 3, \quad R_{12} = 2, \quad R_{13} = 1, \\ &R_{21} = 2, \quad R_{22} = 3, \quad R_{23} = 1, \\ &R_{31} = 2, \quad R_{32} = 3, \quad R_{33} = 1. \end{aligned} \tag{85}$$

The learning rates are set to $\alpha_1 = 1.5$, $\alpha_2 = 1.5$, $\alpha_3 = 0.15$. The solution generated by the algorithm is

$$\begin{aligned} Q_1^* &= \begin{pmatrix} 6.2118 & 9.0007 \\ 9.0007 & -1.9440 \end{pmatrix}, \\ Q_2^* &= \begin{pmatrix} -15.6574 & -2.6946 \\ -2.6946 & 4.0308 \end{pmatrix}, \\ Q_3^* &= \begin{pmatrix} -13.3547 & -4.2811 \\ -4.2811 & -0.3872 \end{pmatrix}. \end{aligned} \tag{86}$$



(a)



(b) (c)

Fig. 1. Algorithm 1: (a) the stability of the demonstrated and resulting dynamics; (b,c) convergence of the norm for iterations of $F_i^{(p)}$ and $Q_i^{(p)}$, respectively.

with

$$\begin{aligned} F_1^* &= \begin{pmatrix} 4.2398 & -0.9103 \end{pmatrix}, \\ F_2^* &= \begin{pmatrix} -0.4149 & 0.9178 \end{pmatrix}, \\ F_3^* &= \begin{pmatrix} 0.2384 & 0.1245 \end{pmatrix}, \end{aligned} \tag{87}$$

and the symmetric solution of AREs given by

$$\begin{aligned} K_1^* &= \begin{pmatrix} 12.7925 & -2.7461 \\ -2.7461 & 2.1820 \end{pmatrix}, \\ K_2^* &= \begin{pmatrix} 3.5608 & -1.2426 \\ -1.2426 & 2.7543 \end{pmatrix}, \\ K_3^* &= \begin{pmatrix} 2.2276 & -1.9906 \\ -1.9906 & 2.1166 \end{pmatrix}. \end{aligned} \tag{88}$$

The resulting dynamics $A + \sum_{i=1}^{3} B_i F_i^*$ are stable as shown in Figure 1a. The convergence of the iterative procedure is shown in Figures 1b and 1c.

Remark 11. *The reader might notice that the learning rate for players* 1,2 *and player* 3 *differ. The reason is that for* $\alpha_3 = \alpha_1 = \alpha_2$ *the overshooting of the gradient descent method is observed. In fact, an adaptive learning rate might be used, e.g. Polyak step-size and the line search method [41].*

B. Model-free Algorithm Simulation

Consider the following continuous time system dynamics

$$\dot{x} = Ax + \sum_{i=1}^{2} B_i u_i, \tag{89}$$

where

$$A = \begin{pmatrix} 3 & 0 \\ 0 & -4 \end{pmatrix}, \quad B_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{90}$$

The demonstrated NE trajectories are generated for the game with the following weight matrices

$$\begin{aligned} Q_{1,d} &= 2I_2 \quad Q_{2,d} = 3I_2, \\ R_{11,d} &= 2, \quad R_{12,d} = 1, \\ R_{21,d} &= 1, \quad R_{22,d} = 6. \end{aligned} \tag{91}$$

Given this game, $F_{1,d}$ and $F_{2,d}$ are

$$\begin{aligned} F_{1,d} &= \begin{pmatrix} 6.2586 & 0.0186 \end{pmatrix}, \\ F_{2,d} &= \begin{pmatrix} -0.0532 & 0.0620 \end{pmatrix}, \end{aligned} \tag{92}$$

with the symmetric solution of AREs

$$\begin{aligned} K_{1,d} &= \begin{pmatrix} 12.7267 & -0.2095 \\ -0.2095 & 0.2466 \end{pmatrix}, \\ K_{2,d} &= \begin{pmatrix} 7.0811 & -0.3192 \\ -0.3192 & 0.3719 \end{pmatrix}. \end{aligned} \tag{93}$$

Firstly, given the demonstrated trajectories of the game described above, we estimate (15) $\hat{F}_1, \hat{F}_2$. Then, following Assumption 2, for additional data collection we applied the following controller

$$\hat{u}_i = -\hat{F}_i x + \omega_i \tag{94}$$

for $i = 1,2$ where $\omega_i(t) = 100\sum_{k=1}^{100} \sin(c_k t)$ and c_k for $k = 1,\ldots,100$ is a random number selected in the range $[-500, 500]$ [28]. Data are collected at 0.01 sec during 2 seconds. Then, using the collected data and the initialized parameters below

$$\begin{aligned} Q_1^{(0)} &= I_2, \quad Q_2^{(0)} = I_2, \\ R_{11} &= 3, \quad R_{12} = 0, \\ R_{21} &= 0, \quad R_{22} = 3, \end{aligned} \tag{95}$$

we derive solution for the initialized game as

$$\begin{aligned} K_1^{(k+1,0)} &= \begin{pmatrix} 6.3546 & -0.1011 \\ -0.1011 & 0.1212 \end{pmatrix}, \\ K_2^{(k+1,0)} &= \begin{pmatrix} 6.3538 & -0.1050 \\ -0.1050 & 0.1230 \end{pmatrix}, \end{aligned} \tag{96}$$

with the following equilibrium feedback laws

$$\begin{aligned} F_1^{(k+1,0)} &= \begin{pmatrix} 6.2535 & 0.0202 \end{pmatrix}, \\ F_2^{(k+1,0)} &= \begin{pmatrix} -0.1050 & 0.1230 \end{pmatrix}. \end{aligned} \tag{97}$$

The learning rates are set to $\alpha_1 = 0.3$, $\alpha_2 = 0.4$.
As suggested in Remark 7, we perform (74) only once after getting convergence of F_i for $i = 1,2$. The solution generated by the algorithm is

$$\begin{aligned} Q_1^* &= \begin{pmatrix} 1.0284 & 0.0034 \\ 0.0034 & 0.9648 \end{pmatrix}, \\ Q_2^* &= \begin{pmatrix} 1.6420 & 0.0039 \\ 0.0039 & 0.4998 \end{pmatrix}. \end{aligned} \tag{98}$$

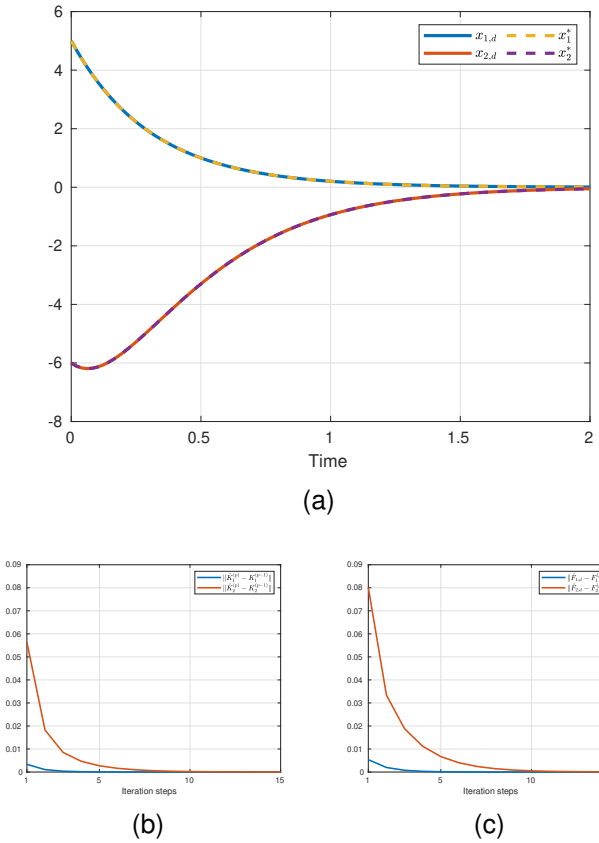


Fig. 2. Algorithm 2: (a) the stability of the observed and resulting dynamics; (b,c) convergence of the norm for iterations of $K_i^{(p)}$ and $F_i^{(p)}$, respectively.

with

$$\begin{aligned} F_1^* &= \begin{pmatrix} 6.2586 & 0.0186 \end{pmatrix}, \\ F_2^* &= \begin{pmatrix} -0.0532 & 0.0620 \end{pmatrix}, \end{aligned} \tag{99}$$

and the symmetric solution of AREs given by

$$\begin{aligned} K_1^* &= \begin{pmatrix} 6.3588 & -0.1002 \\ -0.1002 & 0.1187 \end{pmatrix}, \\ K_2^* &= \begin{pmatrix} 0.3537 & -0.0532 \\ -0.0532 & 0.0620 \end{pmatrix}. \end{aligned} \tag{100}$$

The resulting dynamics $A + \sum_{i=1}^{2} B_i F_i^*$ is stable, as shown in Figure 2a. The convergence of the iterative procedure is shown in Figures 2b and 2c.

Remark 12. *As suggested in the solution characterization section* (50)*, one can change the algorithm output, preserving the game equivalence. For example, set a new* $R'_{21} = -1$ *instead of* $R_{21} = 0$ *used as initialized parameter, relaxing the positive definiteness assumption on* R_{21}*. Then, the game with the same parameters as above, except*

$$Q'_2 = Q_2^* + F_1^{*\top}(R_{21} - R'_{21})F_1^* = \begin{pmatrix} 40.8124 & 0.1200 \\ 0.1200 & 0.5002 \end{pmatrix} \tag{101}$$

instead of Q_2^* *and* $R'_{21} = -1$ *instead of* $R_{21} = 0$ *is also equivalent to the observed game, i.e., it has solution given by* (100) *and* (99)*.*

VII. Conclusion

In this paper, we provide algorithms to solve the inverse problem for linear-quadratic nonzero-sum differential games. Both model-based and model-free versions were introduced. We showed that the algorithms' output is the set of weight matrices that together with the dynamics matrices form an equivalent game for one of the players. After showing the convergence of the algorithms to a desired output, we also provided solution characterizations and showed how the algorithms' output could be adjusted. The effectiveness of the algorithm was demonstrated via simulations. We discussed how the algorithms could be implemented with low (as much as possible) computational cost. The presented algorithms can be extended for the case of non-linear dynamics of the form $f(x)+\sum_{i=1}^{N} g_i(x)u_i$ for an N-player game with necessary assumptions of $f(x)$ and $\{g_i(x)\}_{i=1}^{N}$. This case and consideration of cooperative games or games with some stochastic element in the dynamics can be directions for the further research.

References

[1] T. Başar, *Dynamic games and applications in economics*, vol. 265. Springer Science & Business Media, 1986.

[2] J. Engwerda, *LQ dynamic optimization and differential games*. John Wiley & Sons, 2005.

[3] T. Başar and G. J. Olsder, *Dynamic noncooperative game theory*. SIAM, 1998.

[4] G. J. Mailath and L. Samuelson, *Repeated games and reputations: long-run relationships*. Oxford university press, 2006.

[5] J. Maynard Smith, *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[6] Y. Sannikov, "A continuous-time version of the principal-agent problem," *The Review of Economic Studies*, vol. 75, no. 3, pp. 957–984, 2008.

[7] C. K. Leong and W. Huang, "A stochastic differential game of capitalism," *Journal of Mathematical Economics*, vol. 46, no. 4, pp. 552–561, 2010.

[8] M. Flad, L. Fröhlich, and S. Hohmann, "Cooperative shared control driver assistance systems based on motion primitives and differential games," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 5, pp. 711–722, 2017.

[9] T. Mylvaganam, M. Sassano, and A. Astolfi, "A differential game approach to multi-agent collision avoidance," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 4229–4235, 2017.

[10] D. Gu, "A differential game approach to formation control," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 1, pp. 85–93, 2008.

[11] M. A. Nowak and K. Sigmund, "Evolutionary dynamics of biological games," *science*, vol. 303, no. 5659, pp. 793–799, 2004.

[12] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. 17th International Conf. on Machine Learning*, pp. 663–670, Morgan Kaufmann, 2000.

[13] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *Aaai*, vol. 8, pp. 1433–1438, Chicago, IL, USA, 2008.

[14] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[15] B. D. Anderson, *The inverse problem of optimal control*, vol. 38. Stanford Electronics Laboratories, Stanford University, 1966.

[16] M. Menner and M. N. Zeilinger, "Convex formulations and algebraic solutions for linear quadratic inverse optimal control problems," in *2018 European control conference (ECC)*, pp. 2107–2112, IEEE, 2018.

[17] F. Jean and S. Maslovskaya, "Inverse optimal control problem: the linear-quadratic case," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 888–893, IEEE, 2018.

[18] N. Ab Azar, A. Shahmansoorian, and M. Davoudi, "From inverse optimal control to inverse reinforcement learning: A historical review," *Annual Reviews in Control*, vol. 50, pp. 119–138, 2020.

[19] R. Isaacs, *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1999.

[20] T. L. Molloy, J. J. Ford, and T. Perez, "Inverse noncooperative differential games," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, (Melbourne, VIC), pp. 5602–5608, IEEE, Dec. 2017.

[21] J. Inga, E. Bischoff, T. L. Molloy, M. Flad, and S. Hohmann, "Solution sets for inverse non-cooperative linear-quadratic differential games," *IEEE Control Systems Letters*, vol. 3, no. 4, pp. 871–876, 2019.

[22] F. Köpf, J. Inga, S. Rothfuß, M. Flad, and S. Hohmann, "Inverse reinforcement learning for identification in linear-quadratic dynamic games," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14902–14908, 2017.

[23] T. L. Molloy, J. Inga, M. Flad, J. J. Ford, T. Perez, and S. Hohmann, "Inverse open-loop noncooperative differential games and inverse optimal control," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 897–904, 2019.

[24] B. Lian, W. Xue, F. L. Lewis, and T. Chai, "Robust inverse q-learning for continuous-time linear systems in adversarial environments," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13083–13095, 2021.

[25] K. G. Vamvoudakis, D. Vrabie, and F. L. Lewis, "Online learning algorithm for zero-sum games with integral reinforcement learning," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 1, no. 4, pp. 315–332, 2011.

[26] T. Li and Z. Gajic, "Lyapunov iterations for solving coupled algebraic riccati equations of nash differential games and algebraic riccati equations of zero-sum games," in *New Trends in Dynamic Games and Applications*, pp. 333–351, Springer, 1995.

[27] H. Modares, F. L. Lewis, and Z.-P. Jiang, "H_∞ tracking control of completely unknown continuous-time systems via off-policy reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2550–2562, 2015.

[28] Y. Jiang and Z.-P. Jiang, "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics," *Automatica*, vol. 48, no. 10, pp. 2699–2704, 2012.

[29] K. G. Vamvoudakis, "Non-zero sum Nash Q-learning for unknown deterministic continuous-time linear systems," *Automatica*, vol. 61, pp. 274–281, 2015.

[30] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, no. 2, pp. 477–484, 2009.

[31] E. Mageirou, "Values and strategies for infinite time linear quadratic games," *IEEE Transactions on Automatic Control*, vol. 21, no. 4, pp. 547–550, 1976.

[32] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[33] W. M. Haddad and V. Chellaboina, *Nonlinear dynamical systems and control: a Lyapunov-based approach*. Princeton university press, 2008.

[34] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 2015.

[35] G. Golub and C. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 2013.

[36] D. Vrabie and F. Lewis, "Adaptive dynamic programming for online solution of a zero-sum differential game," *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 353–360, 2011.

[37] K. G. Vamvoudakis, "Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach," *Systems & Control Letters*, vol. 100, pp. 14–20, 2017.

[38] W. Rudin, *Principles of Mathematical Analysis*. New York: McGraw-Hill, 3rd ed., 1976.

[39] W. Xue, P. Kolaric, J. Fan, B. Lian, T. Chai, and F. L. Lewis, "Inverse reinforcement learning in tracking control based on inverse optimal control," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10570–10581, 2021.

[40] P. Ioannou and B. Fidan, *Adaptive control tutorial*. SIAM, 2006.

[41] W. Sun and Y.-X. Yuan, *Optimization theory and methods: nonlinear programming*, vol. 1. Springer Science & Business Media, 2006.