This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

Nash Equilibria for linear quadratic discrete-time dynamic games via iterative and data-driven algorithms

B. Nortmann, *Student Member, IEEE*, A. Monti, *Member, IEEE*, M. Sassano, *Senior Member, IEEE* and T. Mylvaganam, *Senior Member, IEEE*

***Abstract*— Determining feedback Nash equilibrium solutions of nonzero-sum dynamic games is generally challenging. In this paper, we propose four different iterative algorithms to find Nash equilibrium strategies for discrete-time linear quadratic games. The strategy update laws are based on the solution of either Lyapunov or Riccati equations for each player. Local convergence criteria are discussed. Motivated by the fact that in many practical scenarios each player in the game may have access to different (incomplete) information, we also introduce purely data-driven implementations of the algorithms. This allows the players to reach a Nash equilibrium solution of the game via scheduled experiments and without knowledge of each other's performance criteria or of the system dynamics. The efficacy of the presented algorithms is illustrated via numerical examples and a practical example involving human-robot interaction.**

***Index Terms*— Game theory, Linear systems, Optimisation algorithms, Data-driven methods**

I. Introduction

DYNAMIC game theory, also referred to as differential game theory in the continuous-time case, provides powerful tools to model dynamic interactions between strategic decision makers [1]. Due to its flexibility in capturing a variety of potentially competitive scenarios, it has, for example, found applications in economics [2], military strategy [3], politics [4] and ecology [5]. While different solution concepts exist, the so-called *Nash equilibrium* solutions are naturally of interest in noncoopertive settings [6]. At a Nash equilibrium, no decision maker - referred to as player - has an incentive to unilaterally deviate from its equilibrium strategy, since such an action would result in a higher cost for the deviating player. Nash equilibrium solutions, and in particular those characterised by feedback strategies, have been extensively studied in the literature, see *e.g.* [1], [7], [8], [9], [10], [11]. However, the set of coupled equations associated with feedback Nash equilibrium solutions is generally difficult to solve, even for games characterised by a quadratic cost and linear time-invariant (LTI) dynamics [6]. Consequently, approaches based on approximate or numerical solutions are of interest. Approximate Nash equilibrium solution concepts have been considered in [12], [13]. Iterative solution schemes to solve the coupled equations associated with Nash equilibrium strategies of (continuous-time) differential games have been suggested in [11], [14], [15], [16]. Most algorithms are presented without a proof of convergence or convergence guarantees are limited to the special case in which there exists a unique feedback Nash equilibrium. Note that in general the number of feedback Nash equilibria can range from zero to infinity [11]. In the discrete-time case, peculiar challenges and difficulties arise compared to the continuous-time counterpart, due to additional mixed product terms of the decision variables appearing in the coupled algebraic equations associated with linear quadratic (LQ) dynamic games [17]. While the literature in the context of iterative solution methods focuses mainly on the continuous-time setting, methods in the discrete-time setting include [18] for finite-horizon games and [19], [20] in the context of reinforcement learning, with similar limitations on convergence guarantees as in the continuous-time case. In [21] a policy iteration algorithm is provided with conditions ensuring convergence to a Nash equilibrium.

Focusing on discrete-time LQ dynamic games, in the first part of this paper, we propose four algorithms - which, to the best of our knowledge, have not previously appeared in the literature - to find a Nash equilibrium solution of the game. The presented iterative schemes rely on solutions of *uncoupled* either Lyapunov or Riccati equations to update the actions of each player. The first two can be interpreted as policy iteration algorithms and the latter two as value iteration algorithms. Criteria for local convergence of the algorithms to a Nash equilibrium solution are discussed. More precisely, conditions are provided under which a set of Nash equilibrium strategies of the LQ dynamic game, which may admit several of such equilibrium solutions with different outcomes, constitutes a locally asymptotically stable (LAS) equilibrium of the iterative

B. Nortmann and T. Mylvaganam are with the Department of Aeronautics, Imperial College London, London SW7 2AZ, UK (e-mail: {benita.nortmann15, t.mylvaganam}@imperial.ac.uk)

A. Monti is with the Institute for Software Technology at the German Aerospace Center (DLR), 38108 Braunschweig, Germany. This work has been carried out while he was a Ph.D. student at the Dipartimento di Ingegneria Civile ed Ingegneria Informatica (DICII), Università degli Studi di Roma "Tor Vegata", Via del Politecnico 1, 00133, Roma, Italy (e-mail:andrea.monti@uniroma2.it)

M. Sassano is with the Dipartimento di Ingegneria Civile ed Ingegneria Informatica (DICII), Università degli Studi di Roma "Tor Vegata", Via del Politecnico 1, 00133, Roma, Italy (e-mail: mario.sassano@uniroma2.it)

The work of B. Nortmann is supported by UKRI DTP 2019 grant number EP/R513052/1.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

schemes.

In the context of dynamic games it is important to specify the information available to each player. In the "classical" game formulation, each player has full knowledge of all system parameters and the performance criteria of all players [6]. However, in many settings, different and incomplete information may be available to each player. Algorithms to determine Nash equilibrium solutions for games with *unknown system dynamics* have been proposed in [22], [23], [24] for the continuous-time case and [19], [20], [21] for the discrete-time case. In a competitive environment, the assumption that each player knows in advance the objective (performance criterion) of all the other players is hardly motivated. Examples include distributed control systems [25], cyber-physical systems [26] or human-robot systems [27]. In [28] a reinforcement learning algorithm for graphical games, in which only local information is available to each agent, is introduced. Inverse dynamic games [29], on the other hand, focus on learning the performance criteria of all players from "expert data" corresponding to a solution of the game. In [30], a special class of LQ dynamic games with asymmetric information structure is considered, in which one player only knows its own performance objective, but not the functions the other players are aiming to optimise. A direct data-driven method (based on non-expert data) extending the results in [31] is proposed to overcome this lack of information and determine the Nash equilibrium strategy for the uninformed player. The method additionally allows to account for the system dynamics being unknown to the player lacking cost information.

In the second part of this paper, we focus on games with incomplete information, in the sense that each player only knows its own objective function, but not the objectives of all other players. We show that in the context of the iterative algorithms presented in the first part of this paper, data can be used in a similar way as in [30] to determine Nash equilibrium strategies even in the event that *all* players have limited information available to them. We demonstrate that the players can jointly converge to a Nash equilibrium solution by scheduling experiments and taking turns to collect finite data sequences of the state and their own input only. More precisely, during each update interval allocated to each player, the strategies implemented by the other players are fixed at their current guesses. As a result, the (partially closed-loop) system dynamics perceived by the updating player include the strategies of the other players and are hence (at least partially) unknown. This is overcome by representing or recovering the dynamics using data. Hence, the presented algorithms are also applicable to games in which the system dynamics are unknown.

Preliminary results regarding the proposed methods have been presented in [32] although limited to the setting of scalar dynamics. In this paper we extend the results to general linear systems, while also providing a more detailed and insightful analysis and discussion of the proposed algorithms. We also consider the application of the algorithms to a practical example involving human-robot interaction.

The remainder of this paper is organised as follows. In Section II, we recall some preliminaries related to LQ dynamic games and data-driven control and formalise the considered problems. In Section III, we propose and analyse iterative algorithms to find Nash equilibria in the presence of complete model and cost information. The results are demonstrated on two illustrative numerical examples. In Section IV, we introduce data-driven implementations of the algorithms that are applicable when cost and/or model information is missing. The efficacy of the results presented in this paper is then illustrated via a practically relevant example involving human-robot interaction in Section V. Finally, some concluding remarks are provided in Section VI.

Notation: The set of real numbers is denoted by $\mathbb{R}$, the set of natural numbers by $\mathbb{N}$ and $\mathbb{N}\setminus\{0\}$ by $\mathbb{N}_{>0}$. Given a matrix A, its vectorisation is denoted by $\mathrm{vec}(A)$, its transpose by $A^\top$, and its induced 2-norm by $\|A\|$. Given a square matrix B, $B \succ 0$ ($B \succeq 0$) denotes that B is positive definite (positive semi-definite), and $\mathbf{Tr}(B)$ denotes its trace. The spectral radius of B is denoted by $\rho(B)$. The set of square positive definite matrices is denoted by $\mathbb{S}^+$. Given a positive semi-definite matrix M the notation $\|x\|_M^2$ describes the weighted semi-norm $x^\top M x$. Given a matrix C of full row rank, $C^\dagger$ denotes its right inverse. The $n \times n$ identity matrix is indicated by I_n. Given a signal $z : \mathbb{N} \to \mathbb{R}^\sigma$ the sequence $\{z(k), \dots, z(k+T)\}$ is denoted by $z_{[k,k+T]}$ with $k, T \in \mathbb{N}$. Given a vector v, the diagonal matrix with the elements of v on its diagonal is denoted by $\mathrm{diag}(v)$.

II. Problem Statements and Preliminaries

Consider a discrete-time system influenced by the control actions of $N \in \mathbb{N}_{>0}$ players and described by the difference equation

$$x(k+1) = Ax(k) + \sum_{j=1}^{N} B_j u_j(k), \tag{1}$$

for $k \in \mathbb{N}$, where $x(k) \in \mathbb{R}^n$ denotes the state of the system and $u_j(k) \in \mathbb{R}^{m_j}$ denotes the control input of the j-th player, for $j = 1, \dots, N$, and A and B_j, $j = 1, \dots, N$, denote constant matrices of appropriate dimension. Let u_{-j} denote the set of control inputs of all players except player j, namely

$$u_{-j}(k) = \{u_1(k), \dots, u_{j-1}(k), u_{j+1}(k), \dots, u_N(k)\},$$

for all $k \in \mathbb{N}$, $j = 1, \dots, N$. Consider the cost functional

$$J_j(x(0), u_j(\cdot), u_{-j}(\cdot)) = \sum_{k=0}^{\infty} \left(\|x(k)\|_{Q_j}^2 + \sum_{l=1}^{N} \|u_l(k)\|_{R_{jl}}^2 \right), \tag{2}$$

associated with player j, for $j = 1, \dots, N$, where $Q_j = Q_j^\top \succeq 0$, $R_{jl} = R_{jl}^\top \succeq 0$ and $R_{jj} = R_{jj}^\top \succ 0$, for $l = 1, \dots, N$, $l \neq j$. Consider the multi-player optimisation problem in which each player seeks to minimise, via its control input, its associated cost functional. Then, the dynamics (1) and cost functionals (2), for $j = 1, \dots, N$, constitute an LQ dynamic game. In this paper, we focus on so-called *feedback Nash-equilibrium solutions* of the dynamic game, as recalled in the following definition (see *e.g.* [1]).

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

Definition 1. (Feedback Nash equilibrium) Consider the dynamic game defined by the system (1) and the cost functionals (2), for $j = 1, \dots, N$. An admissible[1] set of strategies $\{u_j^\star, u_{-j}^\star\}$ constitutes a feedback Nash equilibrium solution of the game if the inequalities

$$J_j(x(0), u_j^\star(\cdot), u_{-j}^\star(\cdot)) \leq J_j(x(0), u_j(\cdot), u_{-j}^\star(\cdot)), \quad (3)$$

hold for all admissible $\{u_j, u_{-j}^\star\}$ and for $j = 1, \dots, N$.

It can be shown, via the Dynamic Programming principle [33], that the LQ dynamic game defined by the dynamics (1) and the cost functionals (2), for $j = 1, \dots, N$, admits linear state-feedback strategies of the form

$$u_j(k) = K_j x(k),$$

for $K_j \in \mathbb{R}^{m_j \times n}$ and $j = 1, \dots, N$, as Nash equilibrium strategies if and only if there exist $K_j^\star$ and $P_j^\star = P_j^{\star\top} \succeq 0 \in \mathbb{R}^{n \times n}$, such that

$$\rho\left(A + \sum_{j=1}^{N} B_j K_j^\star\right) < 1, \quad (4)$$

satisfying[2]

$$P_j^\star = Q_j + \sum_{l=1}^{N} K_l^{\star\top} R_{jl} K_l^\star + \left(A + \sum_{l=1}^{N} B_l K_l^\star\right)^\top P_j^\star \left(A + \sum_{l=1}^{N} B_l K_l^\star\right), \quad (5)$$

for $j = 1, \dots, N$, and

$$M \begin{bmatrix} K_1^\star \\ \vdots \\ K_N^\star \end{bmatrix} = - \begin{bmatrix} B_1^\top P_1^\star \\ \vdots \\ B_N^\top P_N^\star \end{bmatrix} A, \quad (6)$$

where

$$M = \begin{bmatrix} R_{11} + B_1^\top P_1^\star B_1 & \dots & B_1^\top P_1^\star B_N \\ \vdots & \ddots & \vdots \\ B_N^\top P_N^\star B_1 & \dots & R_{NN} + B_N^\top P_N^\star B_N \end{bmatrix}. \quad (7)$$

Hence, provided a solution of (5)-(7), $j = 1, \dots, N$, such that (4) holds, can be obtained, the corresponding set of strategies

$$u_j(k) = K_j^\star x(k), \quad (8)$$

for $j = 1, \dots, N$, constitutes a feedback Nash equilibrium solution of the game.

Remark 1. The conditions (4)-(7) show that feedback Nash equilibria of LQ discrete-time dynamic games are characterised by the stabilising solutions of a set of coupled algebraic equations. Note that if the players' performance criteria are such that $\sum_{j=1}^{N} Q_j \succ 0$, then any solution to (5)-(7) which is such that $\sum_{j=1}^{N} P_j \succ 0$, is stabilising, *i.e.* such that (4) holds.

[1]A set of feedback strategies $\{u_j, u_{-j}\}$ is said to be admissible if it belongs to the subset of state-feedback control laws that render the zero equilibrium of system (1) asymptotically stable.

[2]See e.g. [34], similar conditions can also be found in [1, Section 6.2.3] and [35, Sec. 6.7.2].

Remark 2. It is interesting to put the conditions (5)-(7) into perspective with respect to their continuous-time counterpart. In the continuous-time case (see *e.g.* [11, Chapter 8] for more details), feedback Nash equilibrium solutions to an N-player LQ differential game are characterised by the solution of N coupled algebraic Riccati equations. The feedback equilibrium strategy of player j, $K_j^\star$, depends explicitly only on the solution $P_j^\star$ of the j-th equation, for $j = 1, \dots, N$. The dependency on the strategies of the other players $l = 1, \dots, N$, $l \neq j$, is instead only implicit through the coupling of the N equations. In the discrete-time case, on the other hand, it is evident in (6) (by inspecting the structure of (7)) that the feedback equilibrium strategy of player j explicitly depends on all $P_l^\star$, $l = 1, \dots, N$. Furthermore, combining (5) and (6) and eliminating one set of variables (either $K_j^\star$ or $P_j^\star$ for $j = 1, \dots, N$) results in N coupled algebraic equations, which are not quadratic in the decision variables unlike the (Riccati) matrix equations arising in differential games and (discrete- or continuous-time) infinite-horizon linear quadratic optimal control. It is worth observing that the two differences mentioned above render the iterative and data-driven computation of solutions to the equations (5)-(7) more challenging than the continuous-time setting.

As discussed in Remark 2, solving (5)-(7) and thereby determining Nash equilibrium solutions of the game (1), (2), for $j = 1, \dots, N$, is generally challenging. In the following sections, we propose and discuss four algorithms to find feedback Nash equilibrium solutions of LQ dynamic games iteratively, with the objective of solving, at each step of the iterations, matrix equations of reduced complexity with respect to (5)-(7). We further introduce a data-driven implementation of the algorithms, which allows to account for missing (or partial) information about the system dynamics and the cost parameters by using measured data. The addressed problems are formally stated in the following subsection.

A. Problem statements

Consider the problem of iteratively determining a set of feedback Nash equilibrium strategies for an LQ discrete-time dynamic game, as formalised in the following statement.

Problem 1. Consider the dynamic game defined by the system (1) and the cost functionals (2), for $j = 1, \dots, N$. Given an initial guess $K_j^{(0)}$, for $j = 1, \dots, N$, iteratively determine gains $K_j^\star$, for $j = 1, \dots, N$, such that the corresponding set of feedback strategies (8), for $j = 1, \dots, N$, constitutes a Nash equilibrium solution of the dynamic game.

In Section III we provide four iterative algorithms which address Problem 1, thus allowing us to find an admissible set of linear feedback strategies that constitutes a Nash equilibrium solution of the game, provided the system dynamics (1) and cost functionals (2), $j = 1, \dots, N$, are known, and analyse local convergence of the algorithms. However, dynamic games can be used to model a variety of scenarios in which it is likely that each player has access to different and typically partial information. Motivated by this, we also consider a data-driven version of Problem 1. Namely, we consider the problem

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

in which each player only has information regarding its own cost, but not about the cost functionals associated with all other players, and may not know the system matrices A and B_j, for $j = 1, \dots, N$. To address this, we present methods to iteratively determine Nash equilibrium strategies using input and state data from sequential experiments, as formalised in the following statement.

Problem 2. Consider the dynamic game defined by the system (1) and the cost functionals (2), for $j = 1, \dots, N$. Let the cost matrices Q_w and R_{wg}, for $g = 1, \dots, N$, associated with players w, for $w = 1, \dots, N$, $w \neq j$, and the system matrices A, B_l, for $l = 1, \dots, N$, be unknown[3] to player j, for $j = 1, \dots, N$. Suppose that the players are able to schedule experiments and recursively collect finite-length data sequences of the state x and their own input u_j to update their estimated strategy gains K_j. Then, given an initial guess $K_j^{(0)}$, for $j = 1, ..., N$, iteratively determine a Nash equilibrium solution of the game.

It is worth mentioning that an algorithmic solution to Problem 2 possesses a feature that has recently become particularly desirable in modern applications, especially for problems involving a large number of players. Namely, in solving Problem 2, a Nash equilibrium is obtained based solely on the partial information available to each player. Apart from the preliminary agreement on scheduling sequential experiments on the basis of certain identification labels associated to each player, the computation of the Nash equilibrium strategies is therefore *distributed*, in the sense that each player solves only *its own matrix equation*, by relying only on measured input-state data, even if the equation in principle belongs to a system of coupled matrix equations. This appealing feature is a result of the coupling being *captured* by the collected input-state data, and may lead to a reduction of the overall computational complexity in practice.

In the following subsection we recall some well-known results related to data-driven control, which are instrumental for the solutions of Problem 2 proposed in Section IV.

B. Preliminaries on data-driven control

For illustrative purposes, consider an LTI system influenced by the control action $u(k) \in \mathbb{R}^m$ of a single player, described by the dynamics

$$x(k+1) = \bar{A}x(k) + \bar{B}u(k). \tag{9}$$

Suppose $\bar{A}$ and $\bar{B}$ are unknown and assume T-length data sequences of the input $u_{d,[k_0,k_0+T-1]}$ and the corresponding state response $x_{d,[k_0,k_0+T]}$ can be collected via an open-loop experiment or simulation. The subscript d indicates measured data samples, whereas the subscript $[k_0, k_f]$ indicates the time interval over which the data sequences are collected. The collected data sequences are used to construct the matrices

$$\begin{aligned} U_- &= \begin{bmatrix} u_d(k_0) & \dots & u_d(k_0+T-1) \end{bmatrix}, \\ X_- &= \begin{bmatrix} x_d(k_0) & \dots & x_d(k_0+T-1) \end{bmatrix}, \\ X_+ &= \begin{bmatrix} x_d(k_0+1) & \dots & x_d(k_0+T) \end{bmatrix}. \end{aligned} \tag{10}$$

Assumption 1. The matrix $\begin{bmatrix} X_- \\ U_- \end{bmatrix}$ has full row rank.

If Assumption 1 holds, then the data matrices (10) can be used to design controllers for the unknown system (9), either by identifying $\bar{A}$ and $\bar{B}$ and using any classical control design method (indirect method) or directly from the data bypassing any explicit identification step (direct method).

1) Indirect method: Following [31, Theorem 1] the open-loop system (9) can equivalently be represented as

$$x(k+1) = \begin{bmatrix} \bar{A} & \bar{B} \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} = X_+ \begin{bmatrix} X_- \\ U_- \end{bmatrix}^\dagger \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}, \tag{11}$$

involving the pseudo-inverse of the collected input-state data. This is a system identification-type result, related to dynamic mode decomposition.

2) Direct method: In [31, Theorem 2] it has been shown that (10) can be used to elegantly represent and design state-feedback controllers of the form $u(k) = Kx(k)$ for the system (9) directly using data. Namely, a data-based representation of the closed-loop system is given by

$$x(k+1) = (\bar{A} + \bar{B}K)x(k) = X_+ G x(k), \tag{12}$$

with $G \in \mathbb{R}^{T \times n}$ satisfying

$$\begin{bmatrix} I_n \\ K \end{bmatrix} = \begin{bmatrix} X_- \\ U_- \end{bmatrix} G. \tag{13}$$

Using this system representation, G becomes the decision variable for control design, *i.e.* G can be designed such that system (9) in closed loop with

$$u(k) = U_- G x(k), \tag{14}$$

satisfies certain control objectives.

In Section IV we address Problem 2 by combining these indirect and direct data-driven methods with the Nash equilibrium finding algorithms (*i.e.* solutions of Problem 1) introduced in the following section.

III. Iterative Algorithms for Nash-equilibria

Consider the dynamic game defined by the dynamics (1) and the cost functionals (2), for $j = 1, \dots, N$. Typically, finding Nash equilibrium strategies of the form (8), *i.e.* determining $K_j^\star$ and $P_j^\star$, for $j = 1, \dots, N$, which satisfy (4)-(7), is not a straightforward task. In this section, we propose four different algorithms to determine a solution $K_j^\star$, $P_j^\star$, starting from an initial guess $K_j^{(0)}$, by iteratively updating the solution guesses $K_j^{(i)}$, $P_j^{(i)}$, for $j = 1, \dots, N$ and $i \in \mathbb{N}$. The first two algorithms, the so-called "Lyapunov iterations", involve update laws based on the solution of Lyapunov equations, the latter two, referred to as "Riccati iterations", are based on the solution of Riccati equations. Each type of algorithm is proposed in a synchronous as well as in an asynchronous fashion, as detailed below and illustrated in Figure 1.

[3]Note that the presented results are applicable and relevant if the system dynamics are known and only the cost of the other players is unknown. To consider the most general case we also treat the system dynamics as unknown.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

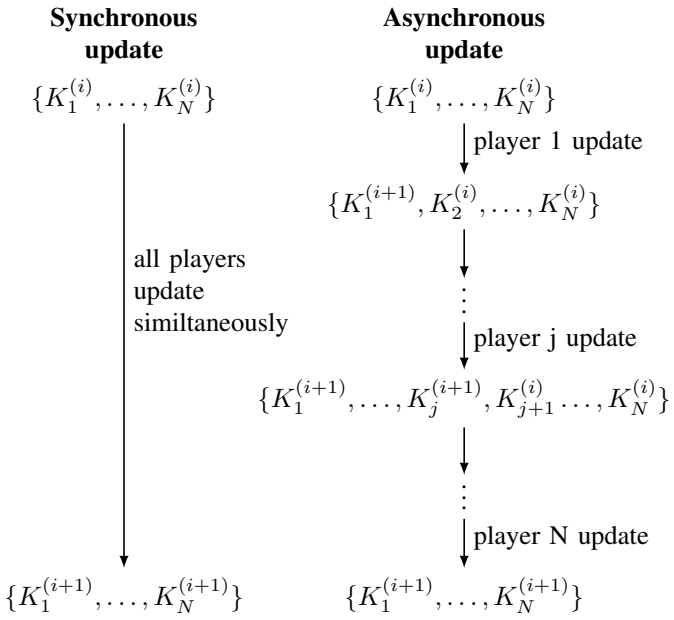

Fig. 1: Illustration of the synchronous and asynchronous strategy update types.

1) Synchronous updates: The strategy update of player j at iteration $(i+1)$ is defined on the premise that the actions of all other players remain fixed at feedback strategies with the gains corresponding to the previous iteration step (i), *i.e.* $u_l(k) = K_l^{(i)} x(k)$, for $l = 1, \dots, N$, $l \neq j$. Hence, the system dynamics perceived by player j at iteration (i) are $x(k+1) = \hat{A}_{s,j}^{(i)} x(k) + B_j u_j(k)$, with $\hat{A}_{s,j}^{(i)} = A + \sum_{l=1, l\neq j}^{N} B_l K_l^{(i-1)}$, and the cost functional of player j becomes $J_j(x(0), u_j) = \sum_{k=0}^{\infty} \left(\|x(k)\|_{\hat{Q}_{s,j}^{(i)}} + \|u_j\|_{R_{jj}} \right)$, with $\hat{Q}_{s,j}^{(i)} = Q_j + \sum_{l=1, l\neq j}^{N} {K_l^{(i-1)}}^\top R_{jl} K_l^{(i-1)}$.

2) Asynchronous updates: The strategy update of player j at iteration $(i+1)$ takes into account the updates of players $w = 1, \dots, j-1$ at the same iteration. Hence, player j treats the strategies of players w fixed at $u_w(k) = K_w^{(i+1)} x(k)$, for $w = 1, \dots, j-1$ and players l fixed at $u_l(k) = K_l^{(i)} x(k)$, for $l = j+1, \dots, N$. The system dynamics perceived by player j at iteration (i) are thus $x(k+1) = \hat{A}_{a,j}^{(i)} x(k) + B_j u_j(k)$, with $\hat{A}_{a,j}^{(i)} = A + \sum_{w=1}^{j-1} B_w K_w^{(i)} + \sum_{l=j+1}^{N} B_l K_l^{(i-1)}$, and the cost functional of player j becomes $J_j(x(0), u_j) = \sum_{k=0}^{\infty} \left(\|x(k)\|_{\hat{Q}_{a,j}^{(i)}} + \|u_j\|_{R_{jj}} \right)$, with $\hat{Q}_{a,j}^{(i)} = Q_j + \sum_{w=1}^{j-1} {K_w^{(i)}}^\top R_{jw} K_w^{(i)} + \sum_{l=j+1}^{N} {K_l^{(i-1)}}^\top R_{jl} K_l^{(i-1)}$.

To streamline the presentation, let $\hat{A}_{\sigma,j}^{(i)}$ and $\hat{Q}_{\sigma,j}^{(i)}$ denote the dynamic matrix and state cost weight, respectively, associated with a generic update rule, where $\sigma = s$ denotes the *synchronous update* and $\sigma = a$ denotes the *asynchronous update*. Moreover, we use the generic notation defined in Appendix I to analyse convergence of the presented algorithms. To this end, consider the following statement introducing conditions related to the solution of the set of coupled matrix equations (5)-(7).

Lemma 1. *Let* $\{K_1^\star, \dots, K_N^\star\}$, *satisfying* (5)-(7), $j = 1, \dots, N$, *for some* $\{P_1^\star, \dots, P_N^\star\}$, *be the set of gains corresponding to any feedback Nash equilibrium solution of the game* (1), (2), $j = 1, \dots, N$. *Let* $z^\star = \begin{bmatrix} \mathrm{vec}(K_1^\star)^\top & \dots & \mathrm{vec}(K_N^\star)^\top \end{bmatrix}^\top$. *The vector* $z^\star$ *is such that* $L(z^\star) = 0$, *with* $L(z^\star) = \begin{bmatrix} L_1^\top & \dots & L_N^\top \end{bmatrix}^\top$, *and*

$$L_j = \mathrm{vec}\left(R_{jj} K_j^\star\right) - \left((A + \sum_{l=1}^{N} B_l K_l^\star)^\top \otimes B_j^\top \right) \times \left((A + \sum_{l=1}^{N} B_l K_l^\star)^\top \otimes (A + \sum_{l=1}^{N} B_l K_l^\star)^\top - I_{n^2} \right)^{-1} \times \mathrm{vec}\left(Q_j + \sum_{l=1}^{N} {K_l^\star}^\top R_{jl} K_l^\star \right), \quad (15)$$

for $j = 1, \dots, N$. *Conversely, any* $\{K_1^\star, \dots, K_N^\star\}$ *satisfying* (15), $j = 1, \dots, N$, *and* (4) *is a set of feedback gains corresponding to a Nash equilibrium solution of the game* (1), (2), $j = 1, \dots, N$.

Proof. The entries of the mapping L given in (15) are derived from the vectorisation of (5), as well as the vectorisation of a single row block of (6), (7) and by eliminating $\mathrm{vec}\left(P_j^\star\right)$. Hence, any $K_j^\star$, $j = 1, \dots, N$ satisfying (5)-(7) satisfies (15), $j = 1, \dots, N$. Note that by Definition 1, the gains $\{K_1^\star, \dots, K_N^\star\}$ are such that the corresponding state-feedback laws render the zero equilibrium asymptotically stable. Hence, for fixed $K_j^\star$, $j = 1, \dots, N$, (5) has a unique solution $P_j^\star$, $j = 1, \dots, N$. Moreover, vectorisation is a linear transformation. Hence, conversely, any $K_j^\star$, $j = 1, \dots, N$, such that (4) holds and satisfying (15) satisfy (5)-(7) for some $P_j^\star$, for $j = 1, \dots, N$. □

A. Lyapunov iterations

In this subsection we introduce iterative algorithms to compute Nash equilibrium strategies for the discrete-time LQ game (1), (2), $j = 1, \dots, N$, via the solution of a sequence of *Lyapunov* equations. Consider the following iterative update law, defined with respect to the unified notation for synchronous ($\sigma = s$) and asynchronous ($\sigma = a$) algorithms introduced above,

$$0 = \hat{Q}_{\sigma,j}^{(i+1)} + \left(K_j^{(i)}\right)^\top R_{jj} K_j^{(i)} - P_j^{(i+1)} + \left(\hat{A}_{\sigma,j}^{(i+1)} + B_j K_j^{(i)}\right)^\top P_j^{(i+1)} \left(\hat{A}_{\sigma,j}^{(i+1)} + B_j K_j^{(i)}\right), \quad (16a)$$

$$K_j^{(i+1)} = -\left(R_{jj} + B_j^\top P_j^{(i+1)} B_j\right)^{-1} B_j^\top P_j^{(i+1)} \hat{A}_{\sigma,j}^{(i+1)}, \quad (16b)$$

for $j = 1, \dots, N$, $i \in \mathbb{N}$. Note that (16a) is a Lyapunov equation, whereas (16b) simply assigns a value to K_j at each step. In the following statement we provide conditions for local asymptotic convergence of the update law (16), for $j = 1, \dots, N$, to a solution $\{K_1^\star, \dots, K_N^\star\}$, $\{P_1^\star, \dots, P_N^\star\}$ of (4)-(7), $j = 1, \dots, N$.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

Proposition 1. *Consider Problem 1 and let* $\{K_1^\star, \dots, K_N^\star\}$, *satisfying* (4)-(7), $j = 1, \dots, N$, *for some* $\{P_1^\star, \dots, P_N^\star\}$, *be the set of gains corresponding to any Nash equilibrium solution of the game. Consider the iterative update law* (16), *for* $j = 1, \dots, N$. *Let* $z^{(i)} = \begin{bmatrix} \mathrm{vec}\left(K_1^{(i)}\right)^\top & \dots & \mathrm{vec}\left(K_N^{(i)}\right)^\top \end{bmatrix}^\top$ *and* $F(z^{(i)}, z^{(i+1)}) = \begin{bmatrix} F_1^\top & \dots & F_N^\top \end{bmatrix}^\top$, *with*

$$F_j = \mathrm{vec}\left(R_{jj}K_j^{(i+1)}\right) - \left((\hat{A}_{\sigma,j}^{(i+1)} + B_jK_j^{(i+1)})^\top \otimes B_j^\top\right) \times \left((\hat{A}_{\sigma,j}^{(i+1)} + B_jK_j^{(i)})^\top \otimes (\hat{A}_{\sigma,j}^{(i+1)} + B_jK_j^{(i)})^\top - I_{n^2}\right)^{-1} \times \mathrm{vec}\left(\hat{Q}_{\sigma,j}^{(i+1)} + {K_j^{(i)}}^\top R_{jj}K_j^{(i)}\right), \quad (17)$$

for $j = 1, ..., N$. *Suppose that*

i. the eigenvalues λ_l, $l = 1, \dots, p$, $p \leq n$ *of* $\left(\hat{A}_{\sigma,j}^{(i+1)} + B_jK_j^{(i)}\right)$ *are such that* $\lambda_l\lambda_q \neq 1$ *for all* $l = 1, \dots, p$, $q = 1, \dots, p$ *and* $j = 1, \dots, N$;

ii. the matrix $H = -M_n^{-1}M_c$ *is Schur, where* M_n *and* M_c *are constructed as in* (26a) *and* (26b) *(see Appendix I) with respect to* F *with entries* F_j *as defined by* (17).

Then $\{K_1^\star, \dots, K_N^\star\}$ *is a LAS equilibrium of the synchronous* ($\sigma = s$) *or asynchronous* ($\sigma = a$) *Lyapunov iterations algorithm* (16), *for* $j = 1, \dots, N$.

Proof. Note that the entries of the mapping F given in (17), are derived from the vectorisation of the (matrix) difference equations (16), for $j = 1, \dots, N$, and by eliminating $\mathrm{vec}\left(P_j^{(i+1)}\right)$. Hence, any $K_j^{(i)}$, $K_j^{(i+1)}$, $j = 1, \dots, N$, satisfying (16) satisfy (17), for $j = 1, \dots, N$. Condition *i.* ensures that there exists a unique solution $P_j^{(i+1)}$ to (16a) [36]. Hence, since vectorisation is a linear transformation, any $K_j^{(i)}$, $K_j^{(i+1)}$, $j = 1, \dots, N$, satisfying (17) satisfy (16), for $j = 1, \dots, N$. Note that $0 = F(z^{(i+1)}, z^{(i)})$ and $0 = F(z^\star, z^\star)$. With this notation in place, LAS follows directly from Lemma 1 and Lemma 3. □

The statement of Proposition 1 ensures the existence of a non-empty basin of attraction with respect to the discrete-time nonlinear system (16) for all Nash equilibrium strategies that satisfy certain conditions. Note that the conditions in Proposition 1 cannot in fact be verified *a priori* since the conditions in item *i.* depend on the strategy updates and the conditions in item *ii.* depend on the knowledge of the Nash equilibrium solution sought for.

Remark 3. The synchronous version of the Lyapunov iterations (16) can be interpreted as a discrete-time, N-player equivalent of the algorithm presented in [14]. Note that due to the additional product terms of the decision variables arising in (6)-(7) compared to the continuous-time case, different interpretations of the discrete-time synchronous Lyapunov iterations are possible. In an alternative version to the one we present, the j-th player updates $P_j^{(i+1)}$ using (16a), however, (6) (in place of (16b)), suitably interpreted within the iterative framework, is used to compute the update of $K_j^{(i+1)}$ for all j simultaneously. This implies that $P_j^{(i+1)}$ is first computed for all $j = 1, \dots, N$, and then used to compute $K_j^{(i+1)}$ for $j = 1, \dots, N$, which is inherently a centralised approach. This version has been presented in [19, Algorithm 1] for the two-player case. Note that in contrast to the version presented herein, the scheme in [19, Algorithm 1] requires the invertibility of a matrix similar to M in (7) (with $P_j^\star$ replaced with $P_j^{(i+1)}$, $j = 1, \dots, N$). Another similar algorithm has been presented in [21, Algorithm 1]. While the update law of this algorithm is equivalent to (16) with $\sigma = s$, the algorithm in [21] is also implemented in a centralised fashion, namely, $P_j^{(i+1)}$ is first computed for all $j = 1, \dots, N$, and then used to compute $K_j^{(i+1)}$ for $j = 1, \dots, N$. In contrast, in the version presented herein, the j-th player updates $P_j^{(i+1)}$ and $K_j^{(i+1)}$ together by solving (16) and the updates of the different players happen sequentially. This choice is motivated by that it allows a data-driven implementation as discussed in Section IV in which each player updates its strategy in a distributed fashion.

B. Riccati iterations

In this subsection we introduce iterative algorithms to compute Nash equilibrium strategies for the discrete-time LQ game (1), (2), $j = 1, \dots, N$, via the solution of a sequence of *independent* Riccati equations. Namely, each player j solves a linear quadratic regulator (LQR) problem at each iteration to update its feedback gain K_j, for $j = 1, \dots, N$, minimising its own cost functional (2) subject to the dynamics (1), with the actions of the other players $l = 1, \dots, N$, $l \neq j$, fixed at feedback strategies (which do not necessarily correspond to a Nash equilibrium solution of the game). Namely, consider the iterative update law given by the stabilising solution of the set of algebraic equations

$$0 = \hat{Q}_{\sigma,j}^{(i+1)} + {K_j^{(i+1)}}^\top R_{jj}K_j^{(i+1)} - P_j^{(i+1)} + \left(\hat{A}_{\sigma,j}^{(i+1)} + B_jK_j^{(i+1)}\right)^\top P_j^{(i+1)} \left(\hat{A}_{\sigma,j}^{(i+1)} + B_jK_j^{(i+1)}\right), \quad (18a)$$

$$K_j^{(i+1)} = -\left(R_{jj} + B_j^\top P_j^{(i+1)} B_j\right)^{-1} B_j^\top P_j^{(i+1)} \hat{A}_{\sigma,j}^{(i+1)}, \quad (18b)$$

for $j = 1, \dots, N$, $i \in \mathbb{N}$. As in Section III-A, we discuss two update scenarios: a synchronous one ($\sigma = s$) and an asynchronous one ($\sigma = a$). Note that (18a) is a Riccati equation, whereas (18b) relates $K_j^{(i+1)}$ and $P_j^{(i+1)}$. In the following statement we provide conditions for local asymptotic convergence of the update law (18), for $j = 1, \dots, N$, to a solution $\{K_1^\star, \dots, K_N^\star\}$, $\{P_1^\star, \dots, P_N^\star\}$ of (4)-(7), for $j = 1, \dots, N$.

Proposition 2. *Consider Problem 1 and let* $\{K_1^\star, \dots, K_N^\star\}$, *satisfying* (4)-(7), $j = 1, \dots, N$, *for some* $\{P_1^\star, \dots, P_N^\star\}$, *be the set of gains corresponding to any Nash equilibrium solution of the game. Consider the iterative update law* (18), *for* $j = 1, \dots, N$. *Let* $z^{(i)} = \begin{bmatrix} \mathrm{vec}(K_1^{(i)})^\top & \dots & \mathrm{vec}(K_N^{(i)})^\top \end{bmatrix}^\top$ *and* $F(z^{(i)}, z^{(i+1)}) = \begin{bmatrix} F_1^\top & \dots & F_N^\top \end{bmatrix}^\top$, *with*

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

$$F_j = \operatorname{vec}\left(R_{jj}K_j^{(i+1)}\right) - \left((\hat{A}_{\sigma,j}^{(i+1)} + B_jK_j^{(i+1)})^\top \otimes B_j^\top\right) \times \left((\hat{A}_{\sigma,j}^{(i+1)} + B_jK_j^{(i+1)})^\top \otimes (\hat{A}_{\sigma,j}^{(i+1)} + B_jK_j^{(i+1)})^\top - I_{n^2}\right)^{-1} \times \operatorname{vec}\left(\hat{Q}_{\sigma,j}^{(i+1)} + {K_j^{(i+1)}}^\top R_{jj}K_j^{(i+1)}\right), \quad (19)$$

for $j = 1, \dots, N$. If the matrix $H = -M_n^{-1}M_c$ is Schur, where M_n and M_c are constructed as in (26a) *and* (26b) *(see Appendix I) with respect to F with entries F_j as defined by* (19)*, then $\{K_1^\star, \dots, K_N^\star\}$ is a LAS equilibrium of the synchronous ($\sigma = s$) or asynchronous ($\sigma = a$) Riccati iterations algorithm* (18)*, for $j = 1, \dots, N$.*

Proof. Note that the entries of the mapping F given in (19), are derived from the vectorisation of the (matrix) difference equations (18), for $j = 1, \dots, N$, and by eliminating $\operatorname{vec}(P_j^{(i+1)})$. Hence, any $K_j^{(i)}$, $K_j^{(i+1)}$, $j = 1, \dots, N$, satisfying (18) satisfy (19), for $j = 1, \dots, N$. By construction, the strategy updates are such that the zero equilibrium of $x(k+1) = \left(\hat{A}_{\sigma,j}^{(i+1)} + B_jK_j^{(i+1)}\right)x(k)$ is stable. Thus, for fixed $K_j^{(i)}$, $K_j^{(i+1)}$, $j = 1, \dots, N$, there exists a unique solution $P_j^{(i+1)}$ to (18a). Hence, since vectorisation is a linear transformation, any $K_j^{(i)}$, $K_j^{(i+1)}$, $j = 1, \dots, N$, satisfying (19) satisfy (18), for $j = 1, \dots, N$. Note that $0 = F\left(z^{(i+1)}, z^{(i)}\right)$ and $0 = F(z^\star, z^\star)$. With this notation in place, LAS follows directly from Lemma 1 and Lemma 3. □

The statement of Proposition 2 ensures the existence of a non-empty basin of attraction with respect to the discrete-time nonlinear system (18) for all Nash equilibrium strategies that satisfy the stated conditions. As in the case of the Lyapunov iterations, the conditions that ensure local convergence in Proposition 2 cannot be verified *a priori* since they depend on the knowledge of the Nash equilibrium solution sought for. Differently from several iterative schemes in the literature (*e.g.* [14], [19], [20], [37]), the iterations (18), $j = 1, \dots, N$, do not require the initial guess $K_j^{(0)}$, for $j = 1, \dots, N$, to be stabilising to converge to a stabilising solution of (5)-(7), $j = 1, \dots, N$. Note that in the asynchronous case ($\sigma = a$) no initial guess $K_1^{(0)}$ is needed for player 1.

Remark 4. The asynchronous version of the Riccati iterations (18) is the discrete-time and N-player equivalent to [16, Algorithm 4.7]. The result therein is provided without any (*a priori* or *a posteriori*) certificate of convergence.

C. Discussion and examples

The conditions in Propositions 1 and 2 ensure that the Lyapunov iterations (16) and Riccati iterations (18), respectively, are locally asymptotically convergent to a Nash equilibrium. However, no comment has been made so far regarding the *recursive feasibility of the update laws* and *stability properties* of the system (1) in closed loop with the current strategy guesses of players $j = 1, \dots, N$. These properties are particularly relevant if the strategy updates are implemented online, as is the case in the data-driven versions of the algorithms, which are introduced in Section IV. In the following we address these important points, before providing two illustrative numerical examples, which include a comparison with the alternative approach presented in [19, Algorithm 1].

1) Recursive feasibility: The Lyapunov updates (16) are feasible at iteration $(i+1)$ if there exists a unique solution $P_j^{(i+1)}$ to (16a). Such a solution exists if condition *i.* of Proposition 1 holds [36].

On the other hand, the Riccati updates (18) are feasible at iteration $(i+1)$ if (18) admits a unique stabilising solution $K_j^{(i+1)}$, $P_j^{(i+1)} = {P_j^{(i+1)}}^\top \succeq 0$. Such a solution exists (recall that by definition $R_{jj} \succ 0$, $Q_j \succeq 0$, and let $\hat{Q}_{\sigma,j}^{(i+1)} = {C_{\sigma,j}^{(i+1)}}^\top C_{\sigma,j}^{(i+1)}$) if (see, e.g., [38])

(RI 1) The pair $\left(\hat{A}_{\sigma,j}^{(i+1)}, B_j\right)$ is stabilisable for $j = 1, \dots, N$.

(RI 2) The pair $\left(\hat{A}_{\sigma,j}^{(i+1)}, C_{\sigma,j}^{(i+1)}\right)$ is detectable for $j = 1, \dots, N$.

Note that the condition (RI 2) is always satisfied, for $j = 1, \dots, N$, if $Q_j \succ 0$, or if $R_{jl} \succ 0$ for all l and the pair (A, Q_j) is observable. These stronger conditions are more easily verified than (RI 2), as they do not depend on strategy guesses and can be checked *a priori*.

In the case of the asynchronous Riccati iterations (18), $\sigma = a$, condition (RI 1) is always satisfied if the pair (A, B_j), for $j = 1, \dots, N$, is stabilisable. Hence, if this is the case and condition (RI 2) holds, then the asynchronous Riccati iterations are guaranteed to be recursively feasible.

2) Stability: Consider system (1) in closed loop with the players' current strategy guesses at iteration (i). In the case of synchronous updates this results in the dynamics $x(k+1) = \left(A + \sum_{j=1}^N B_jK_j^{(i)}\right)x(k) = A_{\mathrm{cl},s}^{(i)}x(k)$. In the case of asynchronous updates this results in the dynamics $x(k+1) = \left(\hat{A}_{a,j}^{(i)} + B_jK_j^{(i)}\right)x(k) = A_{\mathrm{cl},a,j}^{(i)}x(k)$ after the update of player j, for $j = 1, \dots, N$, since the players update their strategy sequentially, as illustrated in Figure 1.

Consider first the (synchronous or asynchronous) Lyapunov iterations. If there exists $P_j^{(i+1)} \in \mathbb{S}^+$ satisfying (16a), with $\sigma = s$ for any $j = 1, \dots, N$ and for all $i \in \mathbb{N}$, it follows that the recursive strategy updates obtained via the *synchronous* Lyapunov iterations are stabilising, namely $\rho\left(A_{\mathrm{cl},s}^{(i)}\right) < 1$, for all $i \in \mathbb{N}$. Similarly, if there exists $P_j^{(i+1)} \in \mathbb{S}^+$ satisfying (16a) with $\sigma = a$ for all $j = 1, \dots, N$, and for all $i \in \mathbb{N}$, it follows that the recursive strategy updates obtained via the *asynchronous* Lyapunov iterations are stabilising, namely $\rho\left(A_{\mathrm{cl},a,j}^{(i)}\right) < 1$, for $j = 1, \dots, N$, for all $i \in \mathbb{N}$.

Consider now the Riccati iterations. By construction, (18) ensures that $\rho\left(\hat{A}_{\sigma,j}^{(i+1)} + B_jK_j^{(i+1)}\right) < 1$ at iteration $(i+1)$ for player j. However, in the case of synchronous updates ($\sigma = s$) this does not provide any guarantees for $A_{\mathrm{cl},s}^{(i+1)}$. For the asynchronous Riccati iterations, on the other hand, this ensures that $\rho\left(A_{\mathrm{cl},a,j}^{(i+1)}\right) < 1$ after the update of player j. Hence, if (18) (with $\sigma = a$) is feasible for all $j = 1, \dots, N$ and for all $i \in \mathbb{N}$, then the recursive strategy updates obtained via the *asynchronous* Riccati iterations are stabilising.

Remark 5. In practice, it is desirable for the initial guess of the matrices $K_j^{(0)}$, for $j = 1, \dots, N$, for the Lyapunov iterations

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

	$P_1^\star$	$P_2^\star$	$K_1^\star$	$K_2^\star$
NE 1	2.0066	29.1606	-0.4771	-2.1339
NE 2	0.9197	44.4309	-0.2138	-3.1793
NE 3	7.9681	1.4540	-1.8084	-0.1016

TABLE I: Values corresponding to the three feedback Nash equilibria of the considered game.

(16) to be such that the resulting closed-loop system is asymptotically stable to search for a positive semi-definite solution to the Lyapunov equation (16a). Albeit involving the solution of more complex algebraic equations at each iteration, the Riccati iterations (18) admit general (non-stabilising) initial guesses of the matrices $K_j^{(0)}$, for $j = 1, \dots, N$.

Remark 6. Note that in the context of reinforcement learning (see e.g. [38], [39]), the Lyapunov iterations (both synchronous and asynchronous) can be interpreted as a policy iteration algorithm, with (16a) representing the policy evaluation step and (16b) the policy update. The (synchronous and asynchronous) Riccati iterations, on the other hand, can be interpreted as value iteration algorithms, since the update law involving the solution of a Riccati equation corresponds to minimising the value function for each player characterised by P_j, $j = 1, \dots, N$, at each update step. The specific nature of this minimisation problem makes it possible to formulate the update law in terms of the gains K_j, $j = 1, \dots, N$, characterising the control strategies (policies) analogous to the policy iteration algorithms.

To illustrate the efficacy of the proposed algorithms, consider the scalar numerical example described by system (1), with $N = 2$, $A = 1.0947$, $B_1 = 0.10254$, $B_2 = 0.045934$, and the cost functionals (2), $j = 1, 2$, with $Q_1 = 0.11112$, $Q_2 = 0.25806$, $R_{11} = 0.40872$, $R_{22} = 0.5949$, $R_{12} = R_{21} = 0$. For the given system and cost parameters, there exist three stabilising solutions to (5)-(7). The corresponding values of $P_j^\star$ and $K_j^\star$, for $j = 1, 2$, are reported in Table I. Figure 2 shows the update histories for the four proposed algorithms starting from initial guess $K_1^{(0)} = -1$, $K_2^{(0)} = -2$. All four algorithms converge to NE 3. The Lyapunov iterations converge to $\max_j \left(\left\| K_j^{(i+1)} - K_j^{(i)} \right\| \right) \leq 10^{-5}$ within 15 iterations (synchronous) and 10 iterations (asynchronous), whereas the Riccati iterations take 19 iterations (synchronous) and 11 iterations (asynchronous). Note that in both cases the asynchronous update converges faster than the synchronous update. Figure 3 gives an insight into the regions of attraction of the three Nash equilibrium solutions. It is worth noting that NE 1 is not attractive for any of the algorithms and that there are significant differences in the regions of attraction for NE 2 and NE 3 between the four algorithms. Apart from a few exceptions, the Lyapunov iterations require a stabilising initial guess to converge to a (stabilising) Nash equilibrium solution, whereas the Riccati iterations do not require a stabilising initial guess. In fact, for the considered example and range of initial conditions the asynchronous Riccati iterations always converge to either NE 3 or NE 2, with the regions of attraction separated by the line $K_2 = K_2^\star$(NE 1). For the synchronous Riccati iterations on the other hand, an interesting behaviour

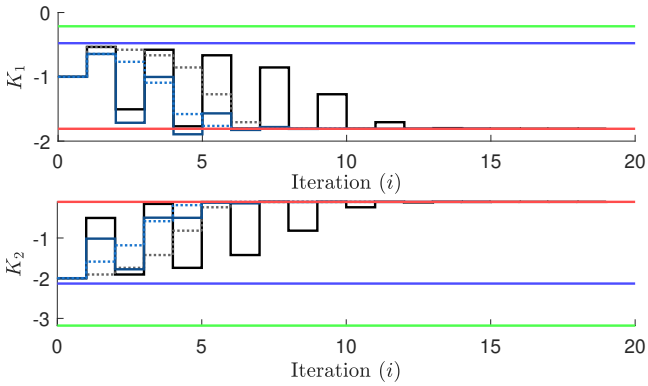

Fig. 2: Update history of the *Lyapunov iterations* (16) (dark blue lines) and the *Riccati iterations* (18) (black lines). Both the synchronous (solid lines) and the asynchronous (dotted lines) versions are shown. The Nash equilibrium strategies in Table I are highlighted in blue (NE 1), green (NE 2) and red (NE 3).

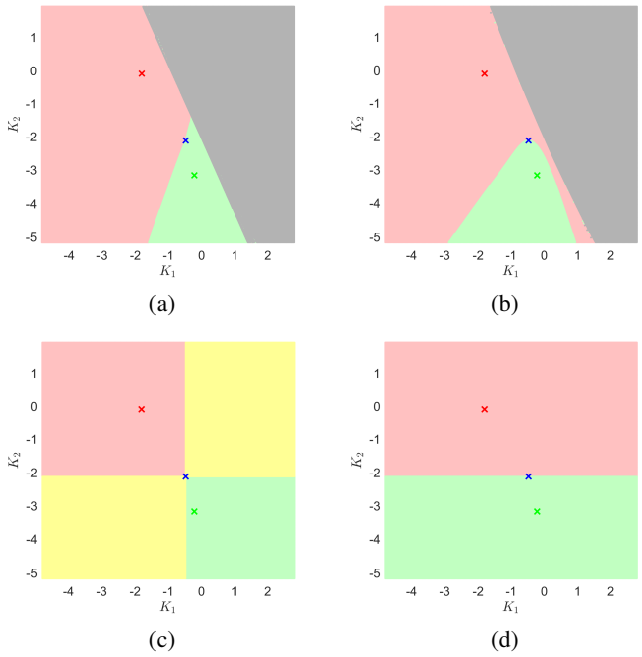

Fig. 3: Regions of the K_1-K_2 space starting from which the synchronous Lyapunov iterations (a), asynchronous Lyapunov iterations (b), synchronous Riccati iterations (c) and asynchronous Riccati iterations (d) converge to NE 1 (blue), NE 2 (green) and NE 3 (red). The yellow regions highlight initial conditions for which the synchronous Riccati iterations converge to a limit cycle. The grey regions highlight initial conditions that do not converge to any of the three equilibria or a limit cycle.

is observed for initial conditions in the regions highlighted in yellow, for which the algorithm converges to a limit cycle with K_1 corresponding to NE 3 and K_2 corresponding to NE 2 and *vice versa*, as shown in Figure 4.

Next, consider the numerical example used in [21], [28],

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

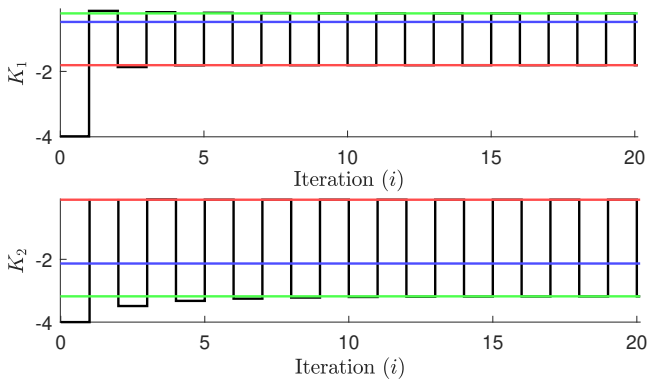

Fig. 4: Update history of the synchronous Riccati iterations (18), $\sigma = s$, with initial conditions $K_1^{(0)} = K_2^{(0)} = -4$. The Nash equilibrium strategies in Table I are highlighted in blue (NE 1), green (NE 2) and red (NE 3).

Algorithm	[19, Alg. 1]	(16), $\sigma = s$	(16), $\sigma = a$	(18), $\sigma = s$	(18), $\sigma = a$
Iterations	12	30	11	31	10

TABLE II: Number of iterations until convergence for the different Nash equilibrium-finding algorithms.

namely, consider the game defined by the dynamics (1) with $N = 4$, $A = \begin{bmatrix} 0.995 & 0.09983 \\ -0.09983 & 0.995 \end{bmatrix}$, $B_1 = \begin{bmatrix} 0.2047 \\ 0.08984 \end{bmatrix}$, $B_2 = \begin{bmatrix} 0.2147 \\ 0.2895 \end{bmatrix}$, $B_3 = \begin{bmatrix} 0.2097 \\ 0.1897 \end{bmatrix}$, $B_4 = \begin{bmatrix} 0.2 \\ 0.1 \end{bmatrix}$ and the cost functionals (2), for $j = 1, \dots, 4$ with $Q_{11} = Q_{22} = Q_{33} = Q_{44} = I_2$, $R_{11} = R_{22} = R_{33} = R_{44} = R_{12} = R_{14} = R_{23} = R_{31} = 1$ and $R_{13} = R_{21} = R_{24} = R_{32} = R_{34} = R_{41} = R_{42} = R_{43} = 0$. Consider the initial guess $K_1^{(0)} = \begin{bmatrix} -2.2570 & 1.1761 \end{bmatrix}$, $K_2^{(0)} = \begin{bmatrix} 1.1629 & -2.5437 \end{bmatrix}$, $K_3^{(0)} = \begin{bmatrix} -0.5465 & -0.6844 \end{bmatrix}$ and $K_4^{(0)} = \begin{bmatrix} -1.9842 & 0.9127 \end{bmatrix}$ as in [21]. The four algorithms presented in Sections III-A and III-B are compared to [19, Algorithm 1] (extended to the N-player case). Note that the synchronous Lyapunov iterations (16), $\sigma = s$, are equivalent to [21, Algorithm 1]. All five algorithms converge to $K_1^\star = \begin{bmatrix} -0.6918 & 0.1601 \end{bmatrix}$, $K_2^\star = \begin{bmatrix} 0.0052 & -0.6095 \end{bmatrix}$, $K_3^\star = \begin{bmatrix} -0.3953 & -0.1560 \end{bmatrix}$ and $K_4^\star = \begin{bmatrix} -0.4239 & 0.0442 \end{bmatrix}$, which satisfy (4)-(7), $j = 1, \dots, 4$, with $P_1^\star = \begin{bmatrix} 5.6345 & -2.6678 \\ -2.6678 & 4.2977 \end{bmatrix}$, $P_2^\star = \begin{bmatrix} 4.3227 & -2.1610 \\ -2.1610 & 4.2364 \end{bmatrix}$, $P_3^\star = \begin{bmatrix} 4.8907 & -1.9054 \\ -1.9054 & 3.2167 \end{bmatrix}$ and $P_4^\star = \begin{bmatrix} 3.8844 & -1.7237 \\ -1.7237 & 3.0592 \end{bmatrix}$. The number of iterations taken by the algorithms until the convergence criterion $\max_j \left(\left\| K_j^{(i+1)} - K_j^{(i)} \right\| \right) \leq 10^{-5}$ is reached is reported in Table II.

IV. Data-driven Iterative Algorithms

In this section, we introduce a data-driven implementation of the asynchronous versions of the algorithms presented in Section III. This enables the players to iteratively converge to a Nash equilibrium solution of a game despite having only limited information regarding the opponents' performance criteria and system dynamics. For ease of exposition we consider the following assumption throughout this section.

Assumption 2. The cost functional (2), which player j, $j = 1, \dots, N$, aims to minimise, is such that $R_{jl} = 0$, for $l = 1, \dots, N$, $l \neq j$.

However, the data-driven results can be extended to include cost cross-terms $R_{jl} \neq 0$ in a straightforward manner by following analogous steps, at the cost of more cumbersome notation and the requirement to collect additional data of a performance variable, as detailed in [30]. In accordance with the definition of Problem 2, further consider the following assumptions.

Assumption 3. Each player j knows the pair (Q_j, R_{jj}) corresponding to its own cost functional, but not the cost weights (Q_l, R_{ll}), for $l = 1, \dots, N$, $l \neq j$ corresponding to the functionals the other players are aiming to minimise. Moreover, the system matrices A, B_j, for $j = 1, \dots, N$, are assumed to be unknown to all players.

Assumption 4. The signals $x(k)$ and $u_j(k)$, are available for measurement for player j, $j = 1, \dots, N$. The players are able and willing to schedule experiments, taking turns to recursively collect sequences of data. During its turn to update from $K_j^{(i)}$ to $K_j^{(i+1)}$, player j collects[4] the state-response $x_{d,[k_0, k_0 + T_j^{(i+1)}]}$ to the sequence of exploring inputs $u_{j,d,[k_0, k_0 + T_j^{(i+1)} - 1]}$, assuming all other players, $l = 1, \dots, N$, $l \neq j$, stick to a constant state-feedback strategy corresponding to their current guess of the Nash equilibrium strategy (*i.e.* at iteration $(i+1)$, $u_w(k) = K_w^{(i+1)} x(k)$, for $w = 1, \dots, j-1$ and $u_l(k) = K_l^{(i)} x(k)$, for $l = j+1, \dots, N$, for $k = k_0, \dots, k_0 + T_j^{(i+1)} - 1$).

The scheduling of data collection and strategy updates described in Assumption 4 is illustrated in Figure 5 for the two-player case, *i.e.* $N = 2$.

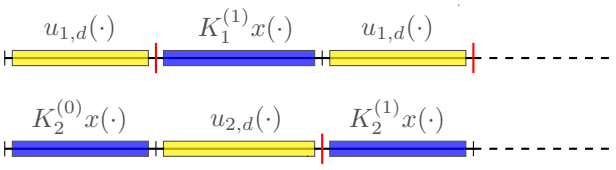

Fig. 5: Illustration of the experiment scheduling with two players. The yellow blocks represent phases in which a player gives "exploring inputs" to excite the system dynamics and collects data, while the blue blocks represent the phases in which a player applies a constant feedback action. Strategy updates are illustrated by the red vertical lines.

Remark 7. Due to lack of knowledge regarding the cost function weights of the other players, player j does not have

[4]The subscripts indicate the sequences correspond to data measured by the j-th player at iteration (i), over the interval $[k_0, k_f]$, with $k_f = k_0 + T_j^{(i)}$ or $k_f = k_0 + T_j^{(i)} - 1$ as appropriate, where $T_j^{(i)}$ is the length of the experiment.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

enough information to solve (5)-(7), for $j = 1, \dots, N$, nor to apply the model-based algorithms introduced in Section III. If Assumption 4 holds, the strategies of the other players are fixed at state-feedback strategies, and the dynamics perceived by player j when updating its strategy guess from iteration (i) to $(i+1)$ are hence $x(k+1) = \hat{A}_{a,j}^{(i+1)} x(k) + B_j u_j(k)$, with $\hat{A}_{a,j}^{(i+1)}$ as defined in Section III. That is, the current guesses of the Nash equilibrium strategies for all other players l, $l = 1, \dots, N$, $l \neq j$, which depend on their cost function weights Q_l, R_{ll}, and are unknown to player j, are encapsulated in the dynamics matrix $\hat{A}_{a,j}^{(i+1)}$. The lack of knowledge is overcome in this section by recovering or representing the perceived dynamics using data. This approach also allows to account for lack of knowledge of the system dynamics. To address the most general case, we hence consider A, B_j, for $j = 1, \dots, N$ unknown, as specified in Assumption 3. Note, however, that the presented results are equally relevant if the system parameters are known.

In the following subsections, we discuss the data-driven implementations of the Lyapunov iterations and the Riccati iterations, by using the indirect method and the direct method introduced in Section II, respectively. To this end, consider at each iteration (i) the matrices X_-, X_+ as defined in (10) with $T = T_j^{(i)}$ and the matrix

$$U_{j-} = \begin{bmatrix} u_{j,d}(k_0) & \dots & u_{j,d}(k_0 + T_j^{(i)} - 1) \end{bmatrix}, \qquad (20)$$

concatenating the exploring inputs of player j, and the corresponding assumption (equivalent to Assumption 1).

Assumption 5. The matrix $\begin{bmatrix} X_- \\ U_{j-} \end{bmatrix}$ has full row rank.

Remark 8. We focus our attention on the asynchronous versions of the algorithms, since it seems natural in practice for each player to update its strategy immediately after collecting data, rather than waiting for all players to finish their experiments. However, the synchronous versions can be implemented using data following analogous steps.

A. Data-driven asynchronous Lyapunov iterations

Towards the design of a data-based version of the iterations (16), let $\bar{B}_j$ and $\bar{A}_j$ denote the estimates of the values B_j and $\hat{A}_{a,j}^{(\cdot)}$, respectively.

Proposition 3. *Consider the game defined by the dynamics* (1) *and the cost functionals* (2)*, for $j = 1, \dots, N$. Let Assumption 2, Assumption 3 and Assumption 4 hold, and suppose the collected input-state data is such that the matrices X_-, X_+ and U_{j-} as defined in* (10) *and* (20) *satisfy Assumption 5. Then, if the conditions stated in Proposition 1 hold for $\{K_1^\star, \dots, K_N^\star\}$, Algorithm 1 solves Problem 2.*

Proof. If Assumptions 2, 4 and 5 hold, then it follows from (11) that the pair $(\bar{A}_j, \bar{B}_j)$ identified in STEPS 19-23 of Algorithm 1 coincides with the pair $(\hat{A}_{a,j}^{(i+1)}, B_j)$. Hence, STEP 24 and STEP 25 coincide with (16). Convergence to a Nash equilibrium follows directly from Proposition 1. □

Algorithm 1 - Data-driven Lyapunov iterations

1: **Initialise:** $x(0) = x_0$, $i = 0$, $k = 0$, $k_0 = k$, $\varepsilon = 1$
2: **Specify:** tolerance $\bar{\varepsilon}$, stabilising $K_l^{(0)}$, for $l = 1, \dots, N$, and time horizon T_f
3: **while** $\varepsilon > \bar{\varepsilon}$ **do**
4: **for** $j = 1$ to N **do**
5: **Assume:** $u_w(k) = K_w^{(i+1)} x(k)$, for $w = 1, \dots, j-1$, and $u_l(k) = K_l^{(i)} x(k)$, for $l = j+1, \dots, N$.
6: **Data collection:**
7: clear X_-, X_+, U_{j-}
8: **while** rank $\left(\begin{bmatrix} X_-^\top & U_{j-}^\top \end{bmatrix}^\top \right) < n + m_j$ **do**
9: give exploring input $u_{j,d}(k)$
10: measure $x_d(k)$
11: $X_- = \begin{bmatrix} x_d(k_0) & \dots & x_d(k) \end{bmatrix}$
12: $U_{j-} = \begin{bmatrix} u_{j,d}(k_0) & \dots & u_{j,d}(k) \end{bmatrix}$
13: $k \leftarrow k + 1$
14: **end while**
15: measure $x_d(k)$
16: $X_+ = \begin{bmatrix} x_d(k_0 + 1) & \dots & x_d(k) \end{bmatrix}$
17: $k_0 \leftarrow k$
18: **Policy update:**
19: **if** $i = 0$ **then**
20: compute $\begin{bmatrix} \bar{A}_j & \bar{B}_j \end{bmatrix} = X_+ (\begin{bmatrix} X_-^\top & U_{j-}^\top \end{bmatrix}^\top)^\dagger$
21: **else**
22: compute $\bar{A}_j = \left(X^+ - \bar{B}_j U_{j-} \right) X_-^\dagger$
23: **end if**
24:

$$P_j^{(i+1)} = Q_j + \left(K_j^i\right)^\top R_{jj} K_j^i + \left(\bar{A}_j + \bar{B}_j K_j^{(i)}\right)^\top P_j^{(i+1)} \left(\bar{A}_j + \bar{B}_j K_j^{(i)}\right)$$

25: $$K_j^{(i+1)} = -\left(R_{jj} + \bar{B}_j^\top P_j^{(i+1)} \bar{B}_j\right)^{-1} \bar{B}_j^\top P_j^{(i+1)} \bar{A}_j$$

26: let $u_j(k) = K_j^{(i+1)} x(k)$
27: **end for**
28: $i \leftarrow i + 1$
29: $\varepsilon \leftarrow \max_j \left(\left\| K_j^{(i)} - K_j^{(i-1)} \right\| \right)$
30: **end while**
31: $K_j^\star = K_j^{(i)}$, for $j = 1, \dots, N$
32: **while** $k < T_f$ **do**
33: let $u_j(k) = K_j^\star x(k)$, for $j = 1, \dots, N$,
34: $k \leftarrow k + 1$
35: **end while**

Remark 9. There exist multiple strategies to tackle the requirement, in STEP 2 of Algorithm 1, for the preliminary computation of a stabilising set of gains $K_l^{(0)}, l = 1, \dots, N$, in a data-driven framework. A feasible solution consists in selecting arbitrary $K_l^{(0)} = 0$, for $l = 1, \dots, N-1$, and in letting player N, without loss of generality, perform, in advance and only once as a preliminary initialisation, STEPS 5 to 25 of Algorithm 1. As a consequence, the computed $K_N^{(0)}$ has the property that $\rho\left(\bar{A}_N + \bar{B}_N K_N^{(0)}\right) < 1$, and hence the overall selection of the initial control gains is (collectively)

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

stabilising for the closed-loop system. Finally, note that STEPS 19-23 in Algorithm 1 could also be replaced with alternative system identification techniques.

B. Data-driven asynchronous Riccati iterations

To formulate the data-driven equivalent to the Riccati iterations update law (18) using the *direct method* introduced in Section II, recall the following result, which introduces a method to design optimal controllers (for linear systems, with the aim of minimising a quadratic cost function) directly using data, without requiring knowledge of the system dynamics.

Lemma 2. *[31, Theorem 4] Consider system* (9) *and assume input-state data is available to form the matrices* (10)*, such that Assumption 1 holds. The optimal state-feedback control gain* $\bar{K}^\star$*, such that* $u(k) = \bar{K}^\star x(k)$ *minimises the cost function*

$$J(x(0), u(\cdot)) = \sum_{k=0}^{\infty} \|x(k)\|_{\bar{Q}}^2 + \|u(k)\|_{\bar{R}}^2 \tag{21}$$

with $\bar{R} \succ 0$ *and* $\bar{Q} \succeq 0$*, is given by* $\bar{K}^\star = U_- G^\star$*, where* $G^\star = YS^{-1}$*, with* Y *and* S *a solution of the convex optimisation problem*

$$\begin{aligned} \min_{S,V,Y} \quad & \left(\mathbf{Tr}\left(\bar{Q}S\right) + \mathbf{Tr}\left(V\right)\right) \\ \text{s.t.} \quad & \begin{bmatrix} S - I_n & X_+ Y \\ Y^\top X_+^\top & S \end{bmatrix} \succeq 0, \\ & \begin{bmatrix} V & \bar{R}^{\frac{1}{2}} U_- Y \\ Y^\top U_-^\top \bar{R}^{\frac{1}{2}} & S \end{bmatrix} \succeq 0, \\ & X_- Y = S. \end{aligned} \tag{22}$$

As a consequence of Lemma 2, solving the data-driven optimisation problem (22) is equivalent to solving the algebraic Riccati equation

$$\left(\bar{A} + \bar{B}\bar{K}^\star\right)^\top \bar{P}^\star \left(\bar{A} + \bar{B}\bar{K}^\star\right) - \bar{P}^\star + \bar{Q} + \bar{K}^{\star\top} \bar{R} \bar{K}^\star = 0,$$

where

$$\bar{K}^\star = -\left(\bar{R} + \bar{B}^\top \bar{P}^\star \bar{B}\right)^{-1} \bar{B}^\top \bar{P}^\star \bar{A},$$

associated with the LQR problem defined by (21) subject to the dynamics (9). Recalling that the update law (18) to iteratively find a Nash equilibrium solution to the game (1), (2), $i = 1, \ldots, N$, corresponds to solving an LQR problem for player j with the strategies of the other players l, for $l = 1, \ldots, N$, $l \neq j$ fixed, for $j = 1, \ldots, N$, Lemma 2 enables a data-driven equivalent of Proposition 2. By construction, Lemma 2 assumes $\bar{P}^\star \succ 0$, hence we consider the following assumption.

Assumption 6. The coupled algebraic equations (5)-(7), associated with the game (1), (2), admit positive definite solutions $P_j^\star \succ 0$, for $j = 1, \ldots, N$.

Proposition 4. *Consider the game defined by the dynamics* (1) *and the cost functionals* (2)*,* $j = 1, \ldots, N$*. Let Assumption 2, Assumption 3, Assumption 4 and Assumption 6 hold, and suppose the collected input-state data is such that the matrices* X_-*,* X_+ *and* U_{j-} *as defined in* (10) *and* (20) *satisfy Assumption 5. Then, if the conditions stated in Proposition 2 hold for* $\{K_1^\star, \ldots, K_N^\star\}$*, Algorithm 2 solves Problem 2.*

Proof. If Assumptions 2, 4, 5 and 6 hold, then by Lemma 2, STEP 9 and STEP 10 of Algorithm 2 are equivalent to (18). Convergence to a Nash equilibrium follows directly from Proposition 2. □

Note that the right hand side of STEP 10 in Algorithm 2 does not explicitly depend on $K_j^{(i)}$ or the strategies of the other players. This dependency is given implicitly via the data matrices X_+, X_- and U_{j-} used to solve (22), which are repopulated for each player j at each iteration (i).

Algorithm 2 - Data-driven Riccati Iterations

1: **Initialise:** $x(0) = x_0$, $i = 0$, $k = 0$, $k_0 = k$, $\varepsilon = 1$
2: **Specify:** tolerance $\bar{\varepsilon}$, $K_l^{(0)}$, for $l = 2, \ldots, N$, and time horizon T_f
3: **while** $\varepsilon > \bar{\varepsilon}$ **do**
4: **for** $j = 1$ to N **do**
5: **Assume:** $u_w(k) = K_w^{(i+1)} x(k)$, for $w = 1, \ldots, j-1$, and $u_l(k) = K_l^{(i)} x(k)$, for $l = j+1, \ldots, N$.
6: **Data collection:**
7: follow steps 7 - 17 of Algorithm 1
8: **Policy update:**
9: solve (22) with $\bar{Q} = Q_j$, $\bar{R} = R_{jj}$, $U_- = U_{j-}$ for Y, S
10: $K_j^{(i+1)} = U_{j-} Y S^{-1}$
11: let $u_j(k) = K_j^{(i+1)} x(k)$
12: **end for**
13: $i \leftarrow i + 1$
14: $\varepsilon \leftarrow \max_j \left(\left\| K_j^{(i)} - K_j^{(i-1)} \right\|\right)$
15: **end while**
16: $K_j^\star = K_j^{(i)}$, for $j = 1, \ldots, N$
17: follow steps 32 - 35 of Algorithm 1

C. Discussion

In Subsections IV-A and IV-B we present data-driven implementations of the asynchronous versions of the algorithms presented in Section III. Note that an indirect data-driven method is used for the Lyapunov iterations and a direct data-driven method is used for the Riccati iterations. The indirect (system identification) method can be readily applied in combination with any model-based control technique and is computationally cheaper for noise-free linear systems as considered herein. In the given setting, the indirect data-driven method further has the advantage that B_j, $j = 1, \ldots, N$, only needs to be identified once (see STEPS 19-23 of Algorithm 1). Thus, the indirect method allows to incorporate available partial system knowledge in a more straightforward way than the direct data-driven method, in which the closed-loop dynamics are entirely represented using data. While the indirect method could be used in combination with both the Lyapunov iterations (16) and the Riccati iterations (18), it is chosen here for the Lyaponov iterations only. This choice is

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

motivated by the fact that the structure of the Riccati iterations (18) allows to readily apply recent direct data-driven results introduced in [31]. Such methods have potential for systems for which system identification is difficult or involved [31]. Hence, the presented results may serve as a basis for future work on games involving more complicated systems.

A benefit of both presented data-driven algorithms is that they are distributed in the sense that a Nash equilibrium is obtained by each player solving only its own matrix equation based on (limited) available information. No knowledge of the cost parameters of the other players nor the system dynamics is required. Apart from the scheduling of experiments, no information exchange between the players is required, *i.e.* player j does not know the strategy guesses of the other players nor measure their inputs.

By capturing the actions of the other players in the dynamics and replacing these with data, each player only updates its own strategy and does not need to explicitly estimate the strategies of the other players. Another benefit of this method is that the computational complexity of the algorithms for each player does not depend on the total number of players in the game. Namely, at each iteration step (i) player j collects $T_j^{(i)}+1$ samples of the state response to its own $T_j^{(i)}$ exploring inputs, where $T_j^{(i)}$ is such that Assumption 5 holds, which implies $T_j^{(i)} \geq n+m_j$. Hence, if the exploring input signal is chosen well, only $n+m_j+1$ data points are required per iteration. The total number of data points a player needs to collect depends on the number of iterations until convergence to the specified tolerance, which in turn depends on the system and cost parameters of the specific problem and the chosen initial conditions as illustrated in Section III-C. In Algorithm 1 the update law at each iteration involves the solution of two linear matrix equations for the system identification step and the policy evaluation step, respectively. The former is of dimension $n \times T_j^{(i)}$ with $n(n+m_j)$ unknowns if $i=0$ and n^2 unknowns if $i>0$. The latter is of dimension $n \times n$ with $(n^2+n)/2$ unknowns. In Algorithm 2, system representation and policy update at each iteration are combined (see STEPS 9 and 10) and involve the solution of a convex programme with two linear matrix inequality constraints of dimension $2n \times 2n$ and $(n+m_j) \times (n+m_j)$ and a linear equality constraint of dimension $n \times n$ with a total number of $T_j^{(i)} n + (n^2+n)/2 + (m_j^2+m_j)/2$ decision variables.

Finally, recall that Algorithm 1 is also applicable for games for which Assumption 6 does not hold. However, Algorithm 2 allows a non-stabilising initial guess, which may be beneficial (particularly in the context of unknown system dynamics).

V. Practical example: Human-robot interaction

We illustrate the efficacy of Algorithms 1 and 2 via a practically motivated example involving human-robot interaction, similar to the example considered in [27], [30]. Consider the interaction dynamics of a contact robot, described by

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -J_c^{-1} D_c \end{bmatrix} x + \begin{bmatrix} 0 \\ J_c^{-1} \end{bmatrix} (u_1 + u_2),$$

with $x = \begin{bmatrix} x_e - x_t & v_e \end{bmatrix}^\top$, where x_e is the position of the robot's end effector, x_t is the target position, v_e is the end effector speed and u_1 and u_2 are the force inputs of the human and the robot, respectively. The inertia and damping coefficients are chosen as $J_c = 6$ kg and $D_c = -0.2$ N/m (as in [27]). The dynamics are discretised using zero-order hold with time step $\Delta = 0.1$ s. However, for the purpose of control design we consider the system dynamics as unknown. The considered task involves arm reaching movements of the human operator to guide the end effector from an initial position to the target position, supported by the robot. This may be relevant in a rehabilitation setting to train a patient to perform reaching movements, or in a manufacturing setting to support an operator in lifting heavy objects. As in [27], [30] we model the human-robot interaction as a two player dynamic game. This is inspired by evidence that human behaviour in such interaction settings can be modelled as Nash equilibrium strategies of a game [40]. Assume the human operator is aiming to minimise a quadratic cost functional (2), for $j=1$ with $Q_1 = \text{diag}([15, 0.1])$, $R_{11} = 0.5$ and $R_{12} = 0$, whereas the robot is aiming to minimise (2), for $j=2$, with $Q_2 = \text{diag}([25, 0.1])$, $R_{21} = 0$ and $R_{22} = 0.1$. Note that each of the two players only has knowledge of its own cost parameters, but not of those of the other player, *i.e.* the human operator only knows Q_1, R_{11}, and the robot only knows Q_2, R_{22}. In contrast to [30], the human operator's control action is not fixed, but both human and robot iteratively update their strategies. To this end, the players take turns to collect data. While the robot's data collection experiments last for $n+m_2 = 3$ steps, *i.e.* 0.3 s, the human collects data for 5 steps, *i.e.* 0.5 s. During its turn, each player gives exploring force inputs sampled randomly from a uniform distribution in the interval (-1 N, 1 N) to excite the system while the other player sticks to its current strategy. The time histories of the states and inputs (for both Algorithm 1 and Algorithm 2) are shown in Figure 6 and Figure 7, respectively. The chosen initial conditions are $K_1^{(0)} = \begin{bmatrix} 0 & 0 \end{bmatrix}$, $K_2^{(0)} = \begin{bmatrix} -0.97 & -3.75 \end{bmatrix}$ and $x_0 = \begin{bmatrix} -0.3 & 0 \end{bmatrix}^\top$. The dotted grey vertical lines indicate the start of each new experiment for data collection. The solid grey vertical lines highlight the time instance at which the algorithms have converged with tolerance $\bar{\varepsilon} = 10^{-5}$. Both Algorithm 1 and Algorithm 2 converge to $K_1^\star = \begin{bmatrix} -1.03 & -0.51 \end{bmatrix}$, $K_2^\star = \begin{bmatrix} -13.18 & -12.39 \end{bmatrix}$, which satisfy (5)-(7), for $j=1,2$, with

$$P_1^\star = \begin{bmatrix} 106.96 & 29.41 \\ 29.41 & 13.80 \end{bmatrix}, \quad P_2^\star = \begin{bmatrix} 224.10 & 88.55 \\ 88.55 & 78.42 \end{bmatrix}.$$

For the remainder of the time horizon ($T_f = 200$, corresponding to 20 s) the players follow their determined equilibrium strategies $K_1^\star$ and $K_2^\star$. Both algorithms converge to the specified tolerance in approximately 7 s, and despite the iterative probing and strategy updates the considered reaching task is completed successfully. At $k = 100$ (10 s) the target position changes from $x_t = 0$ to $x_t = -0.3$, initiating a second arm reaching movement back to the initial condition. The time histories in Figures 6 and 7 show how the human and the robot collaborate to achieve this second reaching movement while following their Nash equilibrium strategies.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

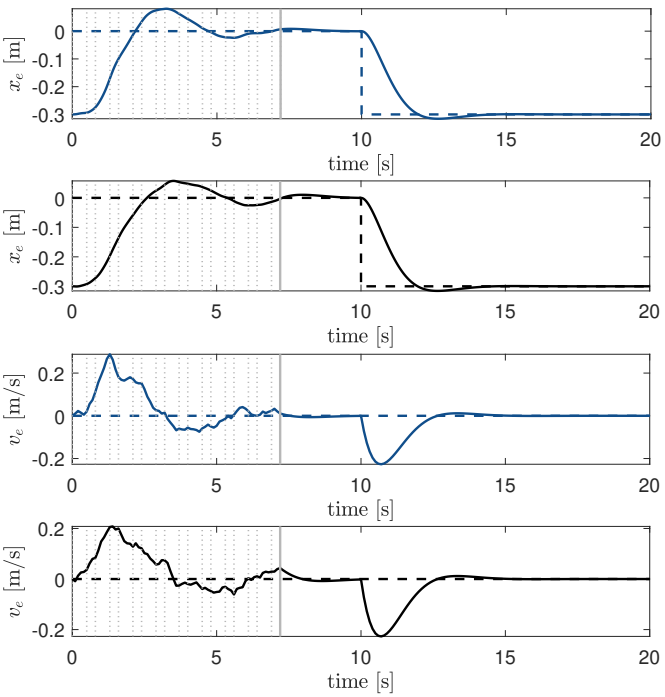

Fig. 6: Time histories of the states using Algorithm 1 (dark blue) and Algorithm 2 (black). The dashed blue and black lines indicate the target values. The dotted grey lines indicate the start of each new experiment, whereas the solid grey line indicates the end of the scheduled experiments.

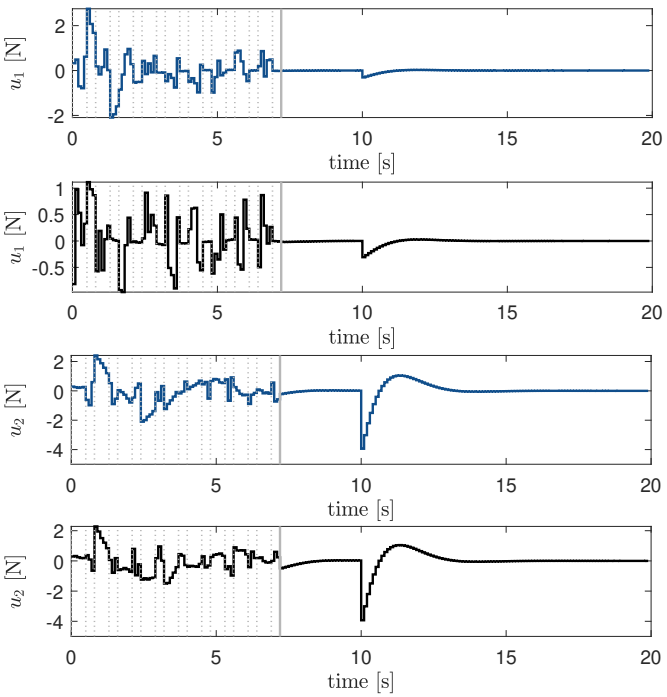

Fig. 7: Time histories of the inputs using Algorithm 1 (dark blue) and Algorithm 2 (black). The dotted grey lines indicate the start of each new experiment, whereas the solid grey line indicates the end of the scheduled experiments.

VI. Conclusions

Considering discrete-time LQ dynamic games, we propose four iterative algorithms for finding feedback Nash equilibrium strategies. The algorithms are based on update laws involving the solution of uncoupled Lyapunov or Riccati equations. For each type a synchronous (*i.e.* all players update their strategy simultaneously) and an asynchronous (*i.e.* each player's update takes the previous players' updates at the same iteration step into account) version are presented. Local convergence conditions are provided. In the second part of the paper, purely data-driven implementations of the asynchronous algorithms are introduced. This allows players to iteratively converge to a Nash equilibrium by scheduling experiments. The approach is distributed in the sense that each player only requires information regarding its own cost functional, but not the cost functionals of the other players nor the system dynamics. The results are demonstrated on illustrative numerical examples and on a practical example involving human-robot interaction. Directions for future work include a more in-depth comparison of the different algorithms for specific applications, as well as studying what affects the regions of convergence and how to strategically initialise the algorithms.

Appendix I
Local Stability Analysis

To streamline the presentation and analysis of the algorithms presented in Section III, let us formulate the local stability analysis for a generic equilibrium finding algorithm. To this end, consider the problem of finding $z^\star \in \mathbb{R}^p$ satisfying the set of algebraic equations

$$0 = L(z^\star), \tag{23}$$

for some $L : \mathbb{R}^p \to \mathbb{R}^p$. Further, consider an iterative update law for the solution guess $z^{(i)}$, $i \in \mathbb{N}$, satisfying the implicit relation

$$0 = F\left(z^{(i+1)}, z^{(i)}\right), \tag{24}$$

with $F : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^p$ differentiable and such that $0 = F(z^\star, z^\star)$. To analyse local convergence of the iterative technique, consider the first order approximation of (24) around the equilibrium $z^\star$, namely

$$0 = M_n\left(z^{(i+1)} - z^\star\right) + M_c\left(z^{(i)} - z^\star\right), \tag{25}$$

where

$$M_n = \left.\frac{\partial F\left(z^{(i+1)}, z^{(i)}\right)}{\partial z^{(i+1)}}\right|_{z^\star, z^\star}, \tag{26a}$$

$$M_c = \left.\frac{\partial F\left(z^{(i+1)}, z^{(i)}\right)}{\partial z^{(i)}}\right|_{z^\star, z^\star}. \tag{26b}$$

Letting $\delta z^{(i)} = \left(z^{(i)} - z^\star\right)$ the dynamics of the iterative update law satisfying (24) can be described by

$$\delta z^{(i+1)} = H \delta z^{(i)}, \tag{27}$$

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

in a neighbourhood of $z^\star$, with $H = -M_n^{-1}M_c$, assuming the matrix M_n is invertible.

Lemma 3. *If H is Schur and $z^{(0)}$ lies in a neighbourhood of $z^\star$, the iterative algorithm with update law satisfying* (24) *asymptotically converges to the equilibrium $z^\star$ of* (23).

Proof. The claim follows from Lyapunov's indirect method. □

REFERENCES

[1] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. SIAM, 1998.

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

[3] R. Isaacs, *Differential games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. Dover Publications, 1999.

[4] E. Calvo, "Dynamic models of international environmental agreements: A differential game approach," *International Review of Environmental and Resource Economics*, vol. 6, no. 4, pp. 289–339, 2013.

[5] G. Neumann and S. Schuster, "Continuous model for the rock–scissors–paper game between bacteriocin producing bacteria," *Journal of Mathematical Biology*, vol. 54, no. 6, pp. 815–846, 2007.

[6] A. W. Starr and Y. C. Ho, "Nonzero-sum differential games," *Journal of Optimization Theory and Applications*, vol. 3, pp. 184–206, 1969.

[7] T. Başar, "On the uniqueness of the Nash solution in linear-quadratic differential games," *International Journal of Game Theory*, vol. 5, no. 2-3, pp. 65–90, 1976.

[8] J. C. Engwerda and Salmah, "Necessary and sufficient conditions for feedback Nash equilibria for the affine-quadratic differential game," *Journal of Optimization Theory and Applications*, vol. 157, no. 2, pp. 552–563, 2012.

[9] C. Possieri and M. Sassano, "An algebraic geometry approach for the computation of all linear feedback Nash equilibria in LQ differential games," in *IEEE Conference on Decision and Control*. IEEE, 2015.

[10] G. P. Papavassilopoulos, J. V. Medanic, and J. B. Cruz, "On the existence of Nash strategies and solutions to coupled riccati equations in linear-quadratic games," *Journal of Optimization Theory and Applications*, vol. 28, no. 1, pp. 49–76, May 1979.

[11] J. Engwerda, *LQ Dynamic Optimization and Differential Games*. John Wiley & Sons, 2005.

[12] T. Mylvaganam, M. Sassano, and A. Astolfi, "Constructive ϵ-Nash equilibria for nonzero-sum differential games," *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 950–965, 2015.

[13] D. Cappello and T. Mylvaganam, "Approximate Nash equilibrium solutions of linear quadratic differential games," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6685–6690, 2020.

[14] T.-Y. 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*. Birkhäuser Boston, 1995, pp. 333–351.

[15] G. Freiling, G. Jank, and H. Abou-Kandil, "On global existence of solutions to coupled matrix Riccati equations in closed-loop Nash games," *IEEE Transactions on Automatic Control*, vol. 41, no. 2, pp. 264–269, 1996.

[16] J. Engwerda, "Algorithms for computing Nash equilibria in deterministic LQ games," *Computational Management Science*, vol. 4, no. 2, pp. 113–140, 2007.

[17] M. Pachter and K. D. Pham, "Discrete-time linear-quadratic dynamic games," *Journal of Optimization Theory and Applications*, vol. 146, no. 1, pp. 151–179, 2010.

[18] D. A. Behrens and R. Neck, "Optgame: An algorithm approximating solutions for multi-player difference games," in *Advances and Innovations in Systems, Computing Sciences and Software Engineering*, K. Elleithy, Ed. Springer Netherlands, 2007, pp. 93–98.

[19] Y. Yang, S. Zhang, J. Dong, and Y. Yin, "Data-driven nonzero-sum game for discrete-time systems using off-policy reinforcement learning," *IEEE Access*, vol. 8, pp. 14 074–14 088, 2020.

[20] J. Li, Z. Xiao, and P. Li, "Discrete-time multi-player games based on off-policy q-learning," *IEEE Access*, vol. 7, pp. 134 647–134 659, 2019.

[21] R. Song, Q. Wei, H. Zhang, and F. L. Lewis, "Discrete-time non-zero-sum games with completely unknown dynamics," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 2929–2943, 2021.

[22] R. Song, F. L. Lewis, and Q. Wei, "Off-policy integral reinforcement learning method to solve nonlinear continuous-time multiplayer nonzero-sum games," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 704–713, 2016.

[23] A. Odekunle, W. Gao, M. Davari, and Z.-P. Jiang, "Reinforcement learning and non-zero-sum game output regulation for multi-player linear uncertain systems," *Automatica*, vol. 112, p. 108672, 2020.

[24] D. Vrabie and F. Lewis, "Integral reinforcement learning for online computation of feedback Nash strategies of nonzero-sum differential games," in *IEEE Conference on Decision and Control*. IEEE, 2010, pp. 3066–3071.

[25] D. Cappello and T. Mylvaganam, "Distributed differential games for control of multi-agent systems," *IEEE Transactions on Control of Network Systems*, 2021.

[26] H. Orojloo and M. A. Azgomi, "A game-theoretic approach to model and quantify the security of cyber-physical systems," *Computers in Industry*, vol. 88, pp. 44–57, 2017.

[27] Y. Li, G. Carboni, F. Gonzalez, D. Campolo, and E. Burdet, "Differential game theory for versatile physical human–robot interaction," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 36–43, 2019.

[28] M. I. Abouheaf, F. L. Lewis, K. G. Vamvoudakis, S. Haesaert, and R. Babuska, "Multi-agent discrete-time graphical games and reinforcement learning solutions," *Automatica*, vol. 50, no. 12, pp. 3038–3053, 2014.

[29] T. L. Molloy, J. I. Charaja, S. Hohmann, and T. Perez, "Inverse noncooperative dynamic games," in *Inverse Optimal Control and Inverse Noncooperative Dynamic Game Theory*. Springer International Publishing, 2022, pp. 143–187.

[30] B. Nortmann and T. Mylvaganam, "Data-driven cost representation for optimal control and its relevance to a class of asymmetric linear quadratic dynamic games," in *European Control Conference*, 2022.

[31] C. De Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality, and robustness," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 909–924, 2020.

[32] B. Nortmann, A. Monti, T. Mylvaganam, and M. Sassano, "Nash equilibria for scalar linear quadratic dynamic games via iterative and data-driven algorithms," in *IEEE Conference on Decision and Control*. IEEE, 2022.

[33] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[34] A. Monti, B. Nortmann, T. Mylvaganam, and M. Sassano, "Feedback and open-loop Nash equilibria for LQ infinite-horizon discrete-time dynamic games," *arXiv preprint:* 10.48550/ARXIV.2307.14898, 2023.

[35] A. Haurie, J. B. Krawczyk, and G. Zaccour, *Games and dynamic games*. World Scientific Publishing Company, 2012, vol. 1.

[36] A. Barraud, "A numerical algorithm to solve $A^T X A$-X= Q," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 883–885, 1977.

[37] K. G. Vamvoudakis and F. L. Lewis, "Multi-player non-zero-sum games: Online adaptive learning solution of coupled hamilton–jacobi equations," *Automatica*, vol. 47, no. 8, pp. 1556–1569, 2011.

[38] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[39] D. P. Bertsekas, *Reinforcement learning and optimal control*. Belmont, Massachusetts: Athena Scientific, 2019.

[40] D. A. Braun, P. A. Ortega, and D. M. Wolpert, "Nash equilibria in multi-agent motor interactions," *PLoS Computational Biology*, vol. 5, no. 8, p. e1000468, 2009.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3375249

Benita Nortmann is a PhD student in control theory at the Department of Aeronautics, Imperial College London, London, UK, where she received the MEng degree in Aeronautical Engineering in 2019. She is a 2022 Zonta International Amelia Earhart Fellow. Her research interests include data-driven control, optimal control and dynamic game theory.

Andrea Monti received the B.S. degree in Computer Science Engineering and the M.S. degree in Control Systems Engineering from the University of Rome Tor Vergata, Italy, in 2016 and 2019, respectively. In 2022 he visited Imperial College London (UK). In 2023 he was awarded a Ph.D. degree in Computer Science, Control and Geoinformation by the University of Rome Tor Vergata, Italy. He is currently a Scientific Researcher in the Institute for Software Technology at the German Aerospace Center (DLR).

Mario Sassano was born in Rome, Italy, in 1985. He received the B.S degree in Automation Systems Engineering and the M.S degree in Systems and Control Engineering from the University of Rome "La Sapienza",Italy, in 2006 and 2008, respectively. In 2012 he was awarded a Ph.D. degree by Imperial College London,UK, where he had been a Research Assistant in the Department of Electrical and Electronic Engineering since 2009. Currently he is an Associate Professor at the University of Rome Tor Vergata, Italy. His research interests are focused on nonlinear observer design, optimal control and differential game theory with applications to mechatronic systems and output regulation for hybrid systems. He is Associate Editor of the IEEE Control Systems Letters, of the European Journal of Control, of IEEE CSS Conference Editorial Board and of the EUCA Conference Editorial Board.

Thulasi Mylvaganam was born in Bergen, Norway, in 1988. She received the M.Eng. degree in Electrical and Electronic Engineering and the Ph.D. degree in Nonlinear Control and Differential Games from Imperial College London, London, U.K., in 2010 and 2014, respectively. From 2014 to 2016, she was a Research Associate with the Department of Electrical and Electronic Engineering, Imperial College London. From 2016 to 2017, she was a Research Fellow with the Department of Aeronautics, Imperial College London, UK, where she is currently a Senior Lecturer. Her research interests include nonlinear control, optimal control, game theory, distributed control and data-driven control. She is Associate Editor of the IEEE Control Systems Letters, of the European Journal of Control, of the IEEE CSS Conference Editorial Board and of the EUCA Conference Editorial Board. She is also a Member of the IFAC Technical Committee 2.4 (Optimal Control) for which she is Vice Chair of Education.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/