

## NashOpt: a Python library for computing generalized Nash equilibria

Alberto Bemporad

To cite this article: Alberto Bemporad (28 Apr 2026): NashOpt: a Python library for computing generalized Nash equilibria, Optimization Methods and Software, DOI: [10.1080/10556788.2026.2661736](https://doi.org/10.1080/10556788.2026.2661736)

To link to this article: <https://doi.org/10.1080/10556788.2026.2661736>



© 2026 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.



Published online: 28 Apr 2026.



Submit your article to this journal [↗](#)



View related articles [↗](#)



View Crossmark data [↗](#)

# NashOpt: a Python library for computing generalized Nash equilibria

Alberto Bemporad

IMT School for Advanced Studies, Lucca, Italy

## ABSTRACT

NashOpt is an open-source Python library for computing and designing generalized Nash equilibria (GNEs) in noncooperative games with shared constraints and real-valued decision variables. The library exploits the joint Karush–Kuhn–Tucker (KKT) conditions of all players to handle both general nonlinear GNEs and linear-quadratic games, including their variational versions. Nonlinear games are solved via nonlinear least-squares formulations, relying on JAX for automatic differentiation. Linear-quadratic GNEs are reformulated as mixed-integer linear programs, enabling efficient computation of multiple equilibria. The framework also supports inverse-game and Stackelberg game-design problems. The capabilities of NashOpt are demonstrated through several examples, including noncooperative game-theoretic control problems of linear quadratic regulation and model predictive control. The library is available at <https://github.com/bemporad/nashopt>.

## ARTICLE HISTORY

Received 30 December 2025  
Accepted 13 April 2026

## 2020 MATHEMATICS SUBJECT CLASSIFICATION

91-04; 91-08; 90-04; 93-04;  
93-08

## 1. Introduction

Generalized games describe situations where several agents take decisions by optimizing individual objective functions under local and coupling constraints on their decision variables. Examples include congestion games, where traffic participants share the same road network, wireless systems sharing limited bandwidth, energy systems where prosumers share transmission capacity and balance constraints [28,40], or firms competing in a market [3]. The solution concept for these problems, in which the cost function and the feasible set of each agent depend on the decisions taken by the other agents, is the generalized Nash equilibrium (GNE), a state in which no agent has an incentive or the possibility to change their decision [2].

GNEPs have been widely studied in the literature, with various algorithms proposed for their solution, including splitting-based operator frameworks [3,41], approaches built on augmented Lagrangian formulations [9,27,29,36], interior-point optimization strategies [13], Newton-type algorithms [17], distributed algorithms [3,39,41], and learning-based methods only relying on measuring best responses [14]. For the specific

**CONTACT** Alberto Bemporad  alberto.bemporad@imtlucca.it  IMT School for Advanced Studies, Piazza San Francesco 19, Lucca, Italy

© 2026 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

class of GNEPs in which the shared constraints are linear, efficient solution methods exist for linear objectives [12,20], multiparametric quadratic programming for strictly convex quadratic objectives [23], and distributed payoff-based algorithms for convex games [38].

### 1.1. Contribution

Motivated by the relative scarcity of software packages for solving rather general classes of GNEPs and their application to game-theoretic control problems, in this paper we describe the mathematical formulations of our package `NashOpt`, a Python library for solving GNEPs based on satisfying the KKT conditions of each player jointly. The library supports nonlinear GNEPs, which are solved via nonlinear least-squares methods, and linear-quadratic GNEPs, which can be solved more efficiently via mixed-integer linear programming (MILP) and can be used to enumerate multiple equilibria for the same game corresponding to different combinations of active constraints at optimality. We also address game-design problems, where a central authority can optimize some parameters of the game to achieve a desired equilibrium in accordance with the optimistic Single-Leader-Multi-Follower (SLMF) Stackelberg game setting [1]. Moreover, we leverage the GNE solution methods to solve game-theoretic control problems, including non-cooperative linear quadratic regulation (LQR) and model predictive control (MPC) problems [24,31]. The main purpose of the tool is to analyze and predict the effect of different agents pursuing their own objectives in a shared environment, and to design the game to promote desired equilibrium behaviours.

For nonlinear GNEs, we leverage JAX [8] for automatic differentiation and just-in-time compilation to achieve high performance, and on the open-source MILP solver HiGHS [26] or Gurobi [22] for the linear-quadratic case, using the so-called ‘big-M’ formulation to encode the complementary slackness conditions of the KKT system [19,25].

The library is open-source and can be easily installed via `pip install nashopt`. We refer the reader to the online documentation at <https://github.com/bemporad/nashopt> for a detailed description of the syntax and usage of the library.

### 1.2. Notation

We denote by  $\mathbb{R}$  the set of real numbers. Given  $N$  vectors  $x_i \in \mathbb{R}^{n_i}$ ,  $i = 1, \dots, N$ , we denote by  $x = \text{col}(x_1, \dots, x_N) \in \mathbb{R}^n$  their vertical concatenation, where  $n = \sum_{i=1}^N n_i$ . We denote by  $\mathbf{1}$  the vector of all ones of appropriate dimension, by  $I$  the identity matrix, and by  $e_i$  the  $i$ th column of  $I$ . Given a matrix  $A$ , we denote by  $A^\top$  its transpose and by  $\|A\|_F$  its Frobenius norm.

## 2. Problem formulation

Consider the following game-design problem based on a multiparametric generalized Nash equilibrium problem (GNEP) with  $N$  players:

$$\begin{aligned}
 & \min_{x^*, p \in P} && J(x^*, p) \\
 & \text{s.t.} && x_i^* \in \arg \min_{x_i} f_i(x, p) \\
 & && \text{s.t.} \quad g(x, p) \leq 0 \\
 & && \quad h(x, p) = 0 \\
 & && \quad \ell_i \leq x \leq u_i \\
 & && \quad x_{-i} = x_{-i}^* \\
 & && \quad i = 1, \dots, N
 \end{aligned} \tag{1}$$

where  $x_i \in \mathbb{R}^{n_i}$  collects the decision variables optimized by agent  $i$ ,  $i = 1, \dots, N$ ,  $x_{-i}$  collects those of all other agents,  $x = \text{col}(x_1, \dots, x_N) \in \mathbb{R}^n$  is the stacked vector of all agents' decision variables,  $x^*$  is constrained to be a generalized Nash equilibrium (GNE) of the game corresponding to the parameter vector  $p \in \mathbb{R}^{n_p}$ ,  $n_p \geq 0$ , where by  $n_p = 0$  we mean that no parameter enters (1). In (1),  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$  is the objective of agent  $i$ ,  $g : \mathbb{R}^n \rightarrow \mathbb{R}^{n_g}$  and  $h : \mathbb{R}^n \rightarrow \mathbb{R}^{n_h}$  define, respectively, shared inequality and equality constraints, and  $\ell_i \in (\mathbb{R} \cup \{-\infty\})^{n_i}$ ,  $u_i \in (\mathbb{R} \cup \{+\infty\})^{n_i}$  define the (local) box constraints for each player  $i$ . Moreover,  $J : \mathbb{R}^n \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}$  is the objective function used to shape the desired equilibrium of the game, and  $P \subseteq \mathbb{R}^{n_p}$  is a given feasible set for the parameter vector  $p$ , such as the hyper-box  $P = \{p \in \mathbb{R}^{n_p} : \ell_p \leq p \leq u_p\}$  where  $\ell_p, u_p$  are (possibly infinite) bounds. Problem (1) is also referred to as an optimistic equilibrium of Single-Leader-Multi-Follower (Stackelberg) game [1], with the leader optimizing  $p$  and the followers playing the GNEP determined by  $p$ .

For any given  $p$ , a GNE  $x^*$  is a vector for which no agent can reduce their cost given the strategies of the other players and feasibility constraints, i.e.

$$f_i(\text{col}(x_i^*, x_{-i}^*), p) \leq f_i(\text{col}(x_i, x_{-i}^*), p) \quad \forall x_i \in X_i(x_{-i}^*, p)$$

where, with a slight abuse of notation, we denoted by  $\text{col}(x_i, x_{-i})$  the vector obtained by collecting the components of  $x_i$  and  $x_{-i}$  in the right order, and  $X_i(x_{-i}^*, p) \subseteq \mathbb{R}^{n_i}$  is the feasible set of player  $i$  defined by the constraints in (1) given the strategies  $x_{-i}$  of the other players. We highlight special cases of (1) in the next sections.

### 2.1. Solving a specific GNEP

To just solve a given game, we set  $J(x^*, p) \equiv 0$  and  $P = \{p_0\}$  a singleton, leading to the standard GNEP:

$$\begin{aligned}
 x_i^* &= \arg \min_{\ell_i \leq x_i \leq u_i} f_i(x, p_0) \\
 & \text{s.t.} \quad g(x, p_0) \leq 0, \quad h(x, p_0) = 0 \\
 & \quad x_{-i} = x_{-i}^*, \quad i = 1, \dots, N.
 \end{aligned} \tag{2}$$

## 2.2. Finding a parameter vector for which a GNE exists

If no specific equilibrium is desired, we can set  $J(x^*, p) \equiv 0$  and let  $P$  be a non-singleton set, leading to the problem of finding a parameter vector  $p \in P$  for which an equilibrium exists.

## 2.3. Inverse game problem

Given an observed agents' equilibrium  $x_{\text{des}}$  (or a desired one we wish to achieve), by setting  $J(x^*, p) = \|x^* - x_{\text{des}}\|_2^2$  (or  $J(x^*, p) = \|x^* - x_{\text{des}}\|_\infty, \|x^* - x_{\text{des}}\|_1$ ), we solve the *inverse game* of finding a vector  $p$  (if one exists) such that  $x^* \approx x_{\text{des}}$ .

## 3. KKT conditions and nonlinear least-squares formulation

Let us assume that  $f_i, g$ , and  $h$  are continuously differentiable with respect to  $x_i$  and that suitable constraint qualifications hold, such as the Linear Independence Constraint Qualification (LICQ) condition at the GNE  $x^*(p)$ .<sup>1</sup> Then, the necessary KKT conditions of optimality for each player  $i$  read as follows [34, Theorem 12.1]:

$$\nabla_{x_i} f_i(x, p) + \nabla_{x_i} g(x, p)^\top \lambda_i + \nabla_{x_i} h(x, p)^\top \mu_i - v_i + y_i = 0 \quad (3a)$$

$$h(x, p) = 0 \quad (3b)$$

$$g(x, p) \leq 0, \quad \ell \leq x \leq u \quad (3c)$$

$$\lambda_i \geq 0, \quad v_i \geq 0, \quad y_i \geq 0 \quad (3d)$$

$$\lambda_{i,j} g_j(x, p) = 0, \quad v_{i,k} (x_{i,k} - \ell_{i,k}) = 0, \quad y_{i,k} (u_{i,k} - x_{i,k}) = 0 \\ j = 1, \dots, n_g, \quad k = 1, \dots, n_i, \quad i = 1, \dots, N \quad (3e)$$

where  $\lambda_i \in \mathbb{R}^{m_i}$  denotes the vector of Lagrange multipliers of player  $i$  associated with the shared inequality constraints  $g(x, p) \leq 0$ ,  $\mu_i$  the multipliers corresponding to the shared equality constraints  $h(x, p) = 0$ , and  $v_i, y_i$  the multipliers associated with bound constraints on  $x_i$ . Note that the problem setting in (1) could be immediately extended to allow for *player-specific* constraints with coupling of the form  $g_i(x, p) \leq 0, h_i(x, p) = 0$  in the KKT system (3a).

A generalized Nash equilibrium is a point  $x^*$  for which the KKT system (3a) holds for *all players simultaneously*. The KKT conditions can be expressed as the following zero-finding problem:

$$\begin{aligned} 0 &= \nabla_{x_i} f_i(x, p) + \nabla_{x_i} g(x, p)^\top \lambda_i + \nabla_{x_i} h(x, p)^\top \mu_i \\ 0 &= h(x, p) \\ 0 &= \sqrt{\lambda_{i,j}^2 + g_j(x, p)^2} - \lambda_{i,j} + g_j(x, p) \\ 0 &= \sqrt{v_{i,k}^2 + (\ell_{i,k} - x_{i,k})^2} - v_{i,k} + \ell_{i,k} - x_{i,k} \\ 0 &= \sqrt{y_{i,k}^2 + (x_{i,k} - u_{i,k})^2} - y_{i,k} + x_{i,k} - u_{i,k} \\ &k = 1, \dots, n_i, \quad i = 1, \dots, N, \quad j = 1, \dots, n_g \end{aligned} \quad (4)$$

where the last three residuals encode the primal feasibility (3c), dual feasibility (3d), and complementary slackness conditions (3e) using the Fischer-Burmeister nonlinear complementarity problem (NCP) function  $\phi : \mathbb{R} \rightarrow \mathbb{R}$  [16,18] componentwise

$$\phi(\alpha, \beta) \triangleq \sqrt{\alpha^2 + \beta^2} - \alpha - \beta = 0 \iff \alpha\beta = 0, \alpha \geq 0, \beta \geq 0.$$

By letting  $z$  be the vector collecting  $x$ ,  $\{\lambda_i\}$ ,  $\{\mu_i\}$ ,  $\{v_i\}$ , and  $\{y_i\}$ , we can express the joint KKT conditions as

$$R(z, p) = 0 \tag{5}$$

where  $R(z, p)$  is the residual function defined in (4). In the sequel, we will denote by  $v = \text{col}(\lambda, \mu, v, y)$  the stacked vector of all dual variables, so that  $z = \text{col}(x, v)$  is the overall vector of primal and dual variables. Given a solution  $z^*(p)$  satisfying (5), we will implicitly denote by  $x^*(p)$  the corresponding GNE is obtained by extracting from  $z^*(p)$  the components corresponding to the primal variables. We will also denote by  $R(z)$  the residual function when  $p$  is omitted or fixed.

For conditions of existence of a GNE for a given  $p$ , such as nonemptiness and compactness of the set of feasible decision vectors, convexity assumptions on  $f$ ,  $g$ , and linearity of  $h$ , and constraint qualifications, we refer the reader to [15]. We remark that only relying on the joint KKT conditions (3a) to characterize GNEs would require additional regularity assumptions. In `NashOpt`, these conditions are not verified numerically by the software. Consequently, when the assumptions above are violated, solving the KKT system should be interpreted as a heuristic computational approach for computing stationary points of the joint KKT system that, however, are not guaranteed a priori to be GNEs. On the other hand, in the case a zero residual  $R$  is achieved within a given tolerance at a certain  $x^*$ , whether this corresponds to a GNE or not can be immediately checked a posteriori by verifying that, for each player  $i$ , the best responses  $\bar{x}_i(x_{-i}^*, p) \approx x_i^*$ , or that the associated value function  $f_i(\text{col}(\bar{x}_i, x_{-i}^*), p) \approx f_i(x^*, p)$ .

### 3.1. Determining a GNE via nonlinear least-squares

For a given parameter  $p_0$ , we can compute the corresponding  $z^*(p_0)$  by minimizing the squared residual

$$z^*(p_0) \in \arg \min_z \frac{1}{2} \|R(z, p_0)\|_2^2 \tag{6}$$

that serves as a *merit function* for the zero-finding problem (5) [13,16]. The nonlinear least-squares problem (6) can be solved via standard methods, such as Levenberg–Marquardt (LM) algorithms [30,33], possibly exploiting sparsity in the Jacobian of  $R(z, p_0)$  with respect to  $z$ . If the minimum residual  $R(z^*(p_0), p_0)$  of the nonlinear least-squares problem (6) is zero (or, more realistically, below a given positive tolerance), then  $z^*(p_0)$  contains a GNE solution  $x^*(p_0)$  and the associated Lagrange multipliers for each agent's problem associated with the parameter vector  $p_0$ .

### 3.2. Variational GNE

Variational GNEs (vGNEs) are obtained by simply enforcing that the Lagrange multipliers associated with the shared constraints are the same for all players, i.e. by replacing  $\{\lambda_i\}$  with a single vector  $\lambda$  and  $\{\mu_i\}$  with a single vector  $\mu$ , which further reduces the number of variables in the zero-finding problem (5).

### 3.3. Game design

To solve the game-design problem (1), we can embed the KKT conditions (3a) in the optimization of  $p$ , leading to the following mathematical program with equilibrium constraints (MPEC) [21,32]:

$$\begin{aligned} \min_{p,z} \quad & J([I \ 0]z, p) \\ \text{s.t.} \quad & R(z, p) = 0, \quad p \in P. \end{aligned} \quad (7)$$

Problem (7) can be solved via constrained nonlinear programming methods with respect to both  $z$  and  $p$ .

To avoid the complexity of dealing with the equilibrium constraints, in our approach we solve the game-design problem (1) by relaxing them and solving instead the following nonlinear programming problem:

$$\begin{aligned} \min_{p,z} \quad & J([I \ 0]z, p) + \frac{\rho}{2} \|R(z, p)\|_2^2 \\ \text{s.t.} \quad & p \in P \end{aligned} \quad (8)$$

where  $\rho > 0$  is a penalty parameter. By letting  $\rho \rightarrow \infty$ , the solution of (8) approaches that of (7). When  $P$  is a hyper-box (with possibly infinite lower and/or upper bounds on some parameters), Problem (8) can be solved very efficiently via bound-constrained nonlinear optimization methods, such as L-BFGS-B [10].

The choice of the initial point  $(z, p)$  and of the penalty  $\rho$  are crucial to achieve convergence to a meaningful solution of (8); in particular, excessively small values of  $\rho$  may lead to solutions that are far from satisfying the KKT conditions, while very large values of  $\rho$  may cause the numerical solver to ignore the game-design objective  $J(x, p)$ . In practice, we can solve (8) with a moderate value of  $\rho$  to get a value  $(\bar{z}^*, \bar{p}^*)$  and then refine the GNE by solving the nonlinear least-squares problem (6) with hot-start at  $(\bar{z}^*, \bar{p}^*)$ .

### 3.4. Non-smooth regularization

When designing a parameter vector  $p$ , or finding an equilibrium among many existing ones for a given  $p_0$ , we can add the regularization term

$$J_{\text{reg}}(x, p) = \alpha_1 \|x\|_1 + \alpha_2 \|x\|_2^2, \quad \alpha_1, \alpha_2 \geq 0$$

in the game-design objective  $J$ ; for example, we can promote sparsity of the GNE  $x^*$  by choosing a sufficiently large  $\alpha_1$ . When  $\alpha_1 > 0$ , due to the non-differentiability of the  $\ell_1$ -norm at zero, we can reformulate the problem by splitting  $x$  into positive and negative parts,

$x = x_p - x_m$ , with  $x_p, x_m \geq 0$  and apply the result in [5, Corollary 1] to minimize

$$\begin{aligned} \min_{p, x_p, x_m, v} \quad & J(x_p - x_m, p) + \frac{\rho}{2} \|R(\text{col}(x_p - x_m, v), p)\|_2^2 \\ & + \alpha_1 \mathbf{1}^\top (x_p + x_m) + \alpha_2 (\|x_p\|_2^2 + \|x_m\|_2^2) \\ \text{s.t.} \quad & p \in P, \quad x_p \geq 0, \quad x_m \geq 0 \end{aligned} \quad (9)$$

where  $v$  collects all dual variables.

In the special case we are just looking for a sparse GNE with no game-design objective (i.e.  $J(x, p) \equiv 0$ ) and  $\alpha_2 > 0$ , by letting  $\alpha_3 \triangleq \sqrt{2\alpha_2}$ , we have

$$\begin{aligned} J_{\text{reg}}(x, p) &= \alpha_1 \mathbf{1}^\top (x_p + x_m) + \alpha_2 (\|x_p\|_2^2 + \|x_m\|_2^2) \\ &= \frac{1}{2} \left\| \alpha_3 x_p + \frac{\alpha_1}{\alpha_3} \mathbf{1} \right\|_2^2 + \frac{1}{2} \left\| \alpha_3 x_m + \frac{\alpha_1}{\alpha_3} \mathbf{1} \right\|_2^2 + \text{const.} \end{aligned}$$

Then, we can equivalently solve

$$\begin{aligned} \min_{p, x_p, x_m, v} \quad & \frac{1}{2} \left\| \begin{bmatrix} \alpha_3 x_p + \frac{\alpha_1}{\alpha_3} \mathbf{1} \\ \alpha_3 x_m + \frac{\alpha_1}{\alpha_3} \mathbf{1} \\ \sqrt{\rho} R(\text{col}(x_p - x_m, v), p) \end{bmatrix} \right\|_2^2 \\ \text{s.t.} \quad & p \in P, \quad x_p \geq 0, \quad x_m \geq 0. \end{aligned} \quad (10)$$

When  $P$  is a hyper-box (with possibly infinite lower and/or upper bounds on some parameters), or a singleton  $\{p_0\}$  as a special case, Problem (9) can be solved as a bound-constrained nonlinear least-squares problem, such as via a trust-region reflective algorithm [11]. Lower and upper bounds  $\ell \leq x \leq u$  can be explicitly included in (9) and (10) by tightening the nonnegativity constraints on  $x_p$  and  $x_m$ , i.e. by setting

$$\max\{0, \ell\} \leq x_p \leq \max\{0, u\}, \quad \max\{0, -u\} \leq x_m \leq \max\{0, -\ell\}.$$

The approach can be immediately extended to the case  $J(x, p) = \frac{1}{2} \|f_j(x, p)\|_2^2$  for some vector function  $f_j : \mathbb{R}^n \times \mathbb{R}^{n_p} \rightarrow n_f$  by extending the residual vector in (10) with  $f_j(x_p - x_m, p)$ .

#### 4. Linear quadratic games

Consider the following special case of (1) where the agents' cost functions are quadratic and the constraints are linear:

$$\begin{aligned} \min_{p, x^*} \quad & J(x^*, p) \\ \text{s.t.} \quad & x_i^* \in \arg \min_{x_i} f_i(x, p) = \frac{1}{2} x^\top Q^i x + (c^i + F^i p)^\top x \\ & \text{s.t.} \quad Ax \leq b + Sp \\ & \quad A_{\text{eq}} x = b_{\text{eq}} + S_{\text{eq}} p \\ & \quad \ell_i \leq x \leq u_i \\ & \quad x_{-i} = x_{-i}^* \\ & \quad i = 1, \dots, N \end{aligned} \quad (11)$$

where  $x^*$  is the generalized Nash equilibrium of the Linear-Quadratic (LQ) game with  $N$  players,  $(Q^i, c^i, F^i)$  define the quadratic cost function for agent  $i$ , with  $Q^i = (Q^i)^\top \in \mathbb{R}^{n \times n}$ , and the diagonal block  $Q_{ii}^i \succeq 0$ ,  $c^i \in \mathbb{R}^n$ ,  $F^i \in \mathbb{R}^{n \times n_p}$ , matrices  $A \in \mathbb{R}^{n_g \times n}$ ,  $b \in \mathbb{R}^{n_g}$ ,  $S \in \mathbb{R}^{n_g \times n_p}$  define the shared linear inequality constraints,  $A_{\text{eq}} \in \mathbb{R}^{n_h \times n}$ ,  $b_{\text{eq}} \in \mathbb{R}^{n_h}$ ,  $S_{\text{eq}} \in \mathbb{R}^{n_h \times n_p}$  define the shared linear equality constraints.

We consider as game-design function  $J(x, p)$ , if any is given, the sum of convex piecewise affine functions

$$J_{\text{PWA}}(x, p) = \sum_{j=1}^{n_j} \max_{k=1, \dots, n_j} D_{jk}x + E_{jk}p + h_{jk} \quad (12a)$$

or the convex quadratic function

$$J_{\text{Q}}(x, p) = \frac{1}{2} [x^\top \ p^\top] Q_J \begin{bmatrix} x \\ p \end{bmatrix} + c_J^\top \begin{bmatrix} x \\ p \end{bmatrix} \quad (12b)$$

where  $Q_J = Q_J^\top \succeq 0$ ,  $Q_J \neq 0$ ,  $Q_J \in \mathbb{R}^{(n+n_p) \times (n+n_p)}$ ,  $c_J \in \mathbb{R}^{n+n_p}$ , or the sum of both, i.e.

$$J(x, p) = J_{\text{PWA}}(x, p) + J_{\text{Q}}(x, p). \quad (12c)$$

Special cases are: (i)  $J_{\text{PWA}}(x, p) = J_{\text{Q}}(x, p) \equiv 0$  for no game design; (ii)  $J_{\text{PWA}}(x, p) = \|x - x_{\text{des}}\|_\infty$  (i.e.  $n_j = 1$ ,  $n_1 = 2n$ ,  $D_1 = [I \ -I]^\top$ ,  $E_1 = 0$ ,  $h_1 = \text{col}(-x_{\text{des}}, x_{\text{des}})$ ), or  $J_{\text{PWA}}(x, p) = \|x - x_{\text{des}}\|_1$  (i.e.  $n_j = n$ ,  $n_j = 2$ ,  $D_{j,1} = e_j^\top$ ,  $D_{j,2} = -e_j^\top$ ,  $E_{j,1} = E_{j,2} = 0$ ,  $h_{j,1} = -x_{\text{des},j}$ ,  $h_{j,2} = x_{\text{des},j}$ ), or  $J_{\text{Q}}(x, p) = \frac{1}{2} \|x - x_{\text{des}}\|_2^2$  for inverse game problems; (iii) the *social welfare* quadratic objective  $J_{\text{Q}}(x, p) = \sum_{i=1}^N f_i(x, p)$ .

For convenience, let us embed finite lower and upper bounds on  $x$  into the inequality constraints by defining

$$\bar{A} = \begin{bmatrix} A \\ I_u \\ -I_\ell \end{bmatrix}, \quad \bar{b} = \begin{bmatrix} b \\ u_u \\ -\ell_\ell \end{bmatrix}, \quad \bar{S} = \begin{bmatrix} S \\ 0 \\ 0 \end{bmatrix}$$

where  $I_\ell$  and  $I_u$  collect the rows of the identity matrix corresponding, respectively, to finite upper and lower bound, and  $\ell_\ell$  and  $u_u$  are the corresponding bounds. Let  $m$  be the total number of resulting inequality constraints.

The KKT conditions for each player's optimization problem are necessary and sufficient for optimality, due to the convexity of the cost functions and constraints, and can be used to characterize the generalized Nash equilibria of the game:

$$\begin{aligned} Q_i^i x + c_i^i + F_i^i p + \bar{A}_i^\top \lambda_i + A_{\text{eq},i}^\top \mu_i &= 0 \\ \bar{A}x &\leq \bar{b} + \bar{S}p \\ A_{\text{eq}}x &= b_{\text{eq}} + S_{\text{eq}}p \\ \lambda_i &\geq 0 \\ \lambda_{i,j}(\bar{A}_j x - \bar{b}_j - \bar{S}_j p) &= 0, \quad i = 1, \dots, N \end{aligned} \quad (13)$$

where  $Q_i^i$  collects the rows of  $Q^i$  corresponding to player  $i$ 's variables,  $\bar{A}_i$  collects the columns and rows of  $\bar{A}$  where  $x_i$  is involved,  $A_{\text{eq},i}$  is defined similarly, and  $\lambda_i$ ,  $\mu_i$ , are

the Lagrange multipliers associated with the inequality and equality constraints, respectively, where the variables in  $x_i$  appear. Note that the LQ problem (11) could be extended to handle player-specific linear constraints with coupling of the form  $A^i x \leq b^i + S^i p$ ,  $A_{\text{eq}}^i x = b_{\text{eq}}^i + S_{\text{eq}}^i p$ , by changing (13) accordingly.

By stacking the KKT conditions (13) for all players, we can rewrite the generalized Nash equilibrium conditions as a single mixed-integer linear program (MILP) using standard big-M constraints to model the complementarity conditions (see, e.g. [25, Proposition 2]):

$$\begin{aligned}
\min_{p,x,\lambda,\mu,\delta,\sigma} \quad & \frac{1}{2} [x^T \ p^T] Q_J \begin{bmatrix} x \\ p \end{bmatrix} + c_J^T \begin{bmatrix} x \\ p \end{bmatrix} + \sum_{j=1}^{n_j} \sigma_j \\
\text{s.t.} \quad & \sigma_j \geq D_{jk} x + E_{jk} p + h_{jk}, \quad k = 1, \dots, n_j \\
& Q_i^i x + c_i^i + F_i^i p + \bar{A}_i^T \lambda_i + A_{\text{eq},i}^T \mu_i = 0 \\
& \bar{A} x \leq \bar{b} + \bar{S} p \\
& A_{\text{eq}} x = b_{\text{eq}} + S_{\text{eq}} p \\
& \lambda_i \geq 0, \quad i = 1, \dots, N \\
& \bar{A} x - \bar{b} - \bar{S} p \leq M(1 - \delta) \\
& \lambda_i \leq M\delta, \quad i = 1, \dots, N \\
& \delta \in \{0, 1\}^m
\end{aligned} \tag{14}$$

where  $M \in \mathbb{R}$  is a sufficiently large positive constant,  $\delta$  is a vector of binary variables introduced to model the complementarity conditions, and  $\sigma$  is a vector of auxiliary variables used to define an upper-bound the objective function,  $\sigma \in \mathbb{R}^{n_j}$ . Clearly, at optimality  $\sum_{i=1}^{n_j} \sigma_j = J_{\text{PWA}}(x, p)$ , as the first  $n_j$  inequalities in (14) become all active. Note that using the same vector  $\delta$  for all players ensures that shared constraints are active/inactive for all players simultaneously.

Problem (14) can be either solved with respect to  $p, x, \lambda, \mu, \delta, \sigma$  by MILP when  $J_Q(x, p) \equiv 0$ , or by mixed-integer quadratic programming (MIQP) otherwise.

#### 4.1. Extracting multiple equilibria

For the same parameter vector  $p$ , multiple generalized Nash equilibria may exist that correspond to different combinations  $\bar{\delta}$  of active inequality constraints, as recently explicitly characterized in [23]. To extract multiple equilibria, we can iteratively solve the MILP above after adding the following ‘no-good’ constraint

$$\sum_{i:\bar{\delta}_i=1} \delta_i - \sum_{i:\bar{\delta}_i=0} \delta_i \leq \left( \sum_{i=1}^m \bar{\delta}_i \right) - 1 \tag{15}$$

which prevents the MILP from returning the same combination  $\bar{\delta}$  of active constraints in all subsequent iterations.

## 4.2. Variational GNE of LQ games

As for the nonlinear case, a vGNE can be obtained by enforcing that the Lagrange multipliers associated with the shared constraints are the same for all players, which further reduces the number of variables in (14).

### 4.2.1. Proximal ADMM solver

For vGNEs of LQ games with fixed parameter  $p = p_0$  and no game objective, we also consider the proximal-ADMM algorithm in [7] as an alternative approach to compute a vGNE. To apply the method, that only handles equality constraints, we reformulate the problem as the following LQ game with  $N + 1$  players:

$$\begin{aligned}
 x_i^* &\in \arg \min_{x_i} \frac{1}{2} x_i' Q^i x_i + (c^i)' x_i \\
 &\text{s.t. } \bar{A} x + s^* = \bar{b}, \quad A_{\text{eq}} x = b_{\text{eq}} \\
 &\quad x_{-i} = x_{-i}^*, \quad i = 1, \dots, N \\
 s^* &\in \arg \min_s \frac{1}{2} \|\bar{A} x^* + s - \bar{b}\|_2^2 \\
 &\text{s.t. } \bar{A} x^* + s = \bar{b}, s \geq 0
 \end{aligned} \tag{16}$$

where, without loss of generality, we have taken  $p_0 = 0$ . The proximal-ADMM iterations for the reformulation (16) are as follows:

$$\begin{aligned}
 x_i^{k+1} &\in \arg \min_{x_i} \frac{1}{2} x_i' Q_{ii}^i x_i + (c_i^i + Q_{i,-i}^i \bar{x}_{-i}^k + (\lambda^k)^\top \bar{A}_i + (\mu^k)^\top A_{\text{eq},i}) x_i + \frac{\gamma}{2} \|x_i - x_i^k\|_2^2 \\
 &\quad + \frac{\rho}{2} (\|\bar{A}_i x_i^k + \bar{A}_{-i} \bar{x}_{-i}^k + s^k - \bar{b}\|_2^2 + \|A_{\text{eq},i} x_i + A_{\text{eq},-i} \bar{x}_{-i}^k - d\|_2^2)
 \end{aligned} \tag{17a}$$

$$s^{k+1} \in \arg \min_{s \geq 0} \frac{1 + \rho}{2} \|\bar{A} x^{k+1} + s - \bar{b}\|_2^2 + (\lambda^k)^\top s + \frac{\gamma}{2} \|s - s^k\|_2^2 \tag{17b}$$

$$\lambda^{k+1} = \lambda^k + \rho (\bar{A} x^{k+1} + s^{k+1} - \bar{b}) \tag{17c}$$

$$\mu^{k+1} = \mu^k + \rho (A_{\text{eq}} x^{k+1} - b_{\text{eq}}) \tag{17d}$$

where  $\bar{x}_{-i}^k$  collects the components of  $x_1^{k+1}, \dots, x_{i-1}^{k+1}, x_{i+1}^k, \dots, x_N^k$ ,  $\gamma > 0$  is the proximal parameter, and  $\rho$  is the augmented Lagrangian parameter. The iterations (17) admit a closed-form solution: the  $x_i$ -update in (17a) is an unconstrained strongly convex quadratic program (QP) that can be solved by caching the Cholesky decompositions of  $Q_{ii}^i + \gamma I + \rho (\bar{A}_i^\top \bar{A}_i + A_{\text{eq},i}^\top A_{\text{eq},i})$  for each player  $i$  and then, at each iteration  $k$ , solving two triangular linear systems; the evaluation of  $s^{k+1}$  also admits the closed-form solution

$$s^{k+1} = \frac{1}{1 + \rho + \gamma} \max \left( (1 + \rho) (\bar{b} - \bar{A} x^{k+1}) + \gamma s^k - \mu^k, 0 \right)$$

as, by completing the squares, (17b) is equivalent to the projection of a vector onto the nonnegative orthant.

## 5. Game-theoretic control

### 5.1. Game-theoretic linear quadratic regulation

We consider a non-cooperative multi-agent linear-quadratic regulator (LQR) problem where  $N$  agents control a shared discrete-time linear dynamical system

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \quad u(t) = \text{col}(u_1(t), \dots, u_N(t)) \\ y(t) &= Cx(t) \end{aligned} \quad (18)$$

where  $x(t) \in \mathbb{R}^{n_x}$  is the state vector at step  $t = 0, 1, \dots$ ,  $y(t)$  is the output vector, and  $u(t) \in \mathbb{R}^{n_u}$  is the input vector partitioned among the  $N$  agents, with  $u_i(t) \in \mathbb{R}^{n_i}$  being the input applied by agent  $i$  at time  $t$ , and  $n_u = \sum_{i=1}^N n_i$ . Accordingly, we set  $B = [B_1 \ \dots \ B_N]$ , where  $B_i \in \mathbb{R}^{n_x \times n_i}$  is the input matrix associated with agent  $i$ . Each agent applies their control input  $u_i(t)$  according to the state-feedback law

$$u_i(t) = -K_i x(t)$$

where  $K_i \in \mathbb{R}^{n_i \times n_x}$  is the LQR gain corresponding to weight matrices  $Q_i \in \mathbb{R}^{n_x \times n_x} \succeq 0$ , such as  $Q_i = C^\top Q_{yi} C$  with  $Q_{yi} \succeq 0$ , and  $R_i \in \mathbb{R}^{n_i}, R_i \succ 0$ . We approximate the desired LQR gain  $K_i$  by defining the function  $K_i^{\text{NLQR}} : \mathbb{R}^{n_x \times (n_u - n_i)} \rightarrow \mathbb{R}^{n_x \times n_i}$  that maps the other agents' gains  $K_{-i}$  into  $K_i$  via the following Riccati-based iterations:

$$\begin{aligned} K_i^k &= (R_i + B_i^T X_i^k B_i)^{-1} B_i^T X_i^k (A - B_{-i} K_{-i}) \\ A_k &= A - B_{-i} K_{-i} - B_i K_i^k \\ X_i^{k+1} &= Q_i + A_k^T X_i^k A_k + (K_i^k)^T R_i K_i^k \end{aligned} \quad (19)$$

starting from  $X_i^0 = Q_i$ , for  $k = 1, \dots, N_{\text{LQR}}$ . Then, we define each agent's cost function as the squared Frobenius norm of the deviation from its optimal gain  $K_i^{\text{NLQR}}(K_{-i})$ :

$$f_i(K_i, K_{-i}) = \frac{1}{2} \|K_i^{\text{NLQR}}(K_{-i}) - K_i\|_F^2. \quad (20)$$

Let  $K = [K_1^\top \ \dots \ K_N^\top]^\top$  denote the stacked feedback gain. A Nash equilibrium is a feedback gain matrix  $K^*$  such that

$$K_i^* \in \arg \min_{K_i} f_i(K_i, K_{-i}^*) \quad \forall i = 1, \dots, N.$$

The equilibrium can be computed as the solution of a Nash equilibrium problem by using the nonlinear least-squares approach (6) (without parameter  $p_0$ ), using the centralized LQR solution  $K_{\text{LQR}}$  associated with  $(A, B, \sum_i Q_i, \text{blkdiag}(R_1, \dots, R_N))$  as initial guess. Note that, in this case, the optimality conditions (4) reduce to  $\nabla_{K_i} f_i(K_i, K_{-i}) = K_i^{\text{NLQR}}(K_{-i}) - K_i = 0$  for all  $i = 1, \dots, N$ , where with a slight abuse of notation, we considered  $K_i$  as its vectorized version; hence, the nonlinear least-squares problem (6) reduces to minimizing the sum of the squared deviations from the optimal gains, i.e.  $\sum_{i=1}^N \|K_i^{\text{NLQR}}(K_{-i}) - K_i\|_F^2$ .

Alternatively, a feedback Nash equilibrium solution to the posed dynamic game can be determined by solving a sequence of discrete algebraic Riccati equations as described

in [35, Section III.B] until convergence is reached within a given tolerance. With this approach, each agent does not have to disclose their costs  $Q_i$ ,  $R_i$ , only the intermediate gains  $K_i$  need to be communicated at each iteration of the procedure. Instead, solving the game centrally by minimizing the sum of the residuals  $f_i(K_i, K_{-i})$  in (20) requires the agents to disclose their costs. Clearly, both methods are offline procedures to compute the equilibrium feedback gain  $K^*$  to apply in closed loop.

## 5.2. Game-theoretic model predictive control

Consider again the discrete-time linear system (18) with  $N$  agents competing to make the output  $y(t)$  track a given set-point  $r(t) \in \mathbb{R}^{n_y}$ . By letting  $\Delta u(t) = u(t) - u(t-1)$  denote the input increment at time  $t$ , each agent  $i$  chooses the sequence  $\Delta u_i \triangleq \{\Delta u_{i,k}\}_{k=0}^{T-1}$  of input increments over a prediction horizon of  $T$  by solving the following standard discrete-time finite-horizon linear optimal control problem [4,6]:

$$\begin{aligned}
 (\Delta u_i, \epsilon_i) \in \arg \min & \quad \left( \sum_{k=0}^{T-1} (y_{k+1} - r(t))^{\top} Q_{y,i} (y_{k+1} - r(t)) + \Delta u_{i,k}^{\top} Q_{\Delta u,i} \Delta u_{i,k} \right) + q_{\epsilon,i} \epsilon_i \\
 \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k \quad y_{k+1} = Cx_{k+1} \\
 & u_{k,i} = u_{k-1,i} + \Delta u_{k,i} \quad u_{-1} = u(t-1) \\
 & \Delta u_{\min} \leq \Delta u_k \leq \Delta u_{\max} \quad u_{\min} \leq u_k \leq u_{\max} \\
 & y_{\min} - \sum_{i=1}^N \epsilon_i \leq y_{k+1} \leq y_{\max} + \sum_{i=1}^N \epsilon_i \quad \epsilon_i \geq 0 \\
 & i = 1, \dots, N, \quad k = 0, \dots, T-1.
 \end{aligned} \tag{21}$$

In (21),  $Q_{\Delta u,i} \succ 0$  and  $Q_{y,i} \succeq 0$ ; for example, if an agent is interested in controlling only a subset  $I_i$  of the components of the output vector with unit weights, we have  $Q_{y,i} = \sum_{j \in I_i} e_j e_j^{\top}$ . The slack variables  $\epsilon_i \geq 0$  are used to soften the shared output constraints (with linear penalty  $q_{\epsilon,i} \geq 0$ ).

The predicted trajectories  $\{x_k, y_k, u_k\}$  in (21) satisfy the dynamics (18) over the prediction horizon  $k = 0, \dots, T$ , with initial condition  $x_0 = x(t)$  and  $u_{-1} = u(t-1)$ . Note that each agent optimizes the sequence of input increments  $\Delta u_i$  and the slack variable  $\epsilon_i$ , so that constraints on inputs, input increments, and slacks are local to each agent, while output constraints are shared among all agents, each one softening them with possibly different slacks  $\epsilon_i$ . Hence, the resulting game-theoretic MPC problem is the GNE problem of finding sequences  $\{\Delta u_i^*\}_{i=1}^N$  (and slacks  $\{\epsilon_i^*\}_{i=1}^N$ ) optimizing (21) for all agents  $i = 1, \dots, N$  simultaneously. In a receding-horizon fashion, only the first inputs  $u_i(t) = u_i(t-1) + \Delta u_{i,0}^*$  are applied for  $i = 1, \dots, N$ , and the entire procedure is repeated at  $t+1$ .

The number of constraints in each agent's problem (21) can be reduced by imposing the constraints only on a shorter constraint horizon of  $T_c < T$  steps. This can be especially useful when  $T$  is large but we want to limit the number of Lagrange multipliers and binary variables involved in the MILP reformulation (14) of the GNEP, in which we remove  $p$ ,  $\sigma$ , and the objective function (12a), as we are only interested in computing the GNE of the noncooperative MPC problem.

If a *variational* equilibrium is requested, the additional equalities enforcing common Lagrange multipliers for the shared output constraints in (21) can be imposed. In Section 6.5, we will consider an example of noncooperative MPC and show that it can deviate significantly from the *cooperative and centralized* MPC solution in which all moves are optimized jointly by solving a quadratic programming (QP) problem with decision variables  $\Delta u = \text{col}(\Delta u_1, \dots, \Delta u_N)$ ,  $\epsilon = \text{col}(\epsilon_1, \dots, \epsilon_N)$  and cost function given by the sum of all agents' costs in (21).

## 6. Examples

We report several numerical tests to illustrate the methods described in the previous sections for computing and designing generalized Nash equilibria. All tests are run on a Macbook Pro with Apple M4 Max chip and 64 GB RAM using JAX [8] for automatic differentiation and just-in-time compilation of the involved functions when required.

### 6.1. Linear-quadratic game

Let us first consider a simple linear-quadratic game with  $N = 3$  agents, each controlling two decision variables  $x_i \in \mathbb{R}^2$ ,  $i = 1, 2, 3$ , with cost functions

$$f_i(x) = \frac{1}{2}x^\top x + i \mathbf{1}^\top x \quad (22)$$

where  $x = \text{col}(x_1, x_2, x_3) \in \mathbb{R}^6$ , and shared constraints

$$\begin{bmatrix} -0.4 & -0.1 & -2.1 & 1.6 & -1.8 & -0.8 \\ 0.5 & -1.2 & -1.1 & -0.9 & 0.6 & 2.3 \\ 0 & -1.1 & 0.5 & -0.6 & 0.0 & 1.2 \\ -0.7 & 0 & -0.9 & -0.2 & 0.3 & -1 \end{bmatrix} x \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

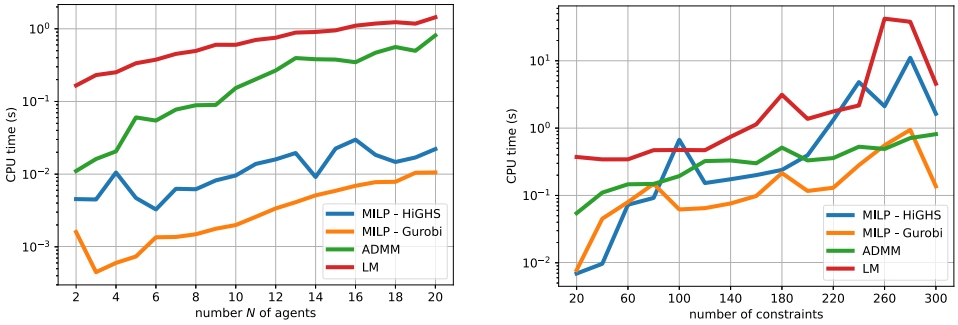
The GNE is computed by solving the MILP (14) with no parameter  $p$ , and optimal combinations extracted by iteratively adding the no-good constraint (15), leading to the three equilibrium solutions:

$$\begin{aligned} x_1^* &= [11.0588 \ 2.7647 \ -1 \ -1 \ -2 \ -2]^\top \\ x_2^* &= [0 \ 0 \ -0.3436 \ -1.5001 \ -1.1599 \ -0.7387]^\top \\ x_3^* &= [0 \ 0 \ -0.7966 \ -0.8336 \ -0.2783 \ -0.1998]^\top. \end{aligned}$$

These correspond to the optimal active constraint combinations (1), (1, 4), and (1, 2, 4), respectively. By imposing the additional constraint that the Lagrange multipliers associated with the shared constraints are the same for all agents, we get the vGNE

$$x_v^* = [0.3553 \ 0.037 \ 0.0431 \ -1.5324 \ -1.4232 \ -1.408]^\top$$

corresponding to the active constraint combination (1, 4). The CPU time for computing each equilibrium ranges between 4.7 and 10.1 ms for the non-variational case and is 8.3 ms for the variational one using the HiGHS solver [26]. Note that infinitely many different GNEs may exist for the same combination of active constraints, as clearly shown in [23].



(a) CPU time vs number  $N$  of agents, under  $n_g = 2N$  shared constraints.

(b) CPU time vs number of shared constraints, with  $N = 5$  agents.

**Figure 1.** CPU time for computing a GNE via MILP vs LM for increasing number of agents and constraints. (a) CPU time vs number  $N$  of agents, under  $n_g = 2N$  shared constraints and (b) CPU time vs number of shared constraints, with  $N = 5$  agents.

For comparison, we also use the nonlinear least-squares approach (6) and compute the same vGNE  $x_v^*$  as above in 229.9 ms (10 LM iterations) starting from the initial guess  $x^0 = 0$ . The same vGNE is found again by running the proximal ADMM approach described in Section 4.2.1, where we set  $\gamma = 1, \rho = 1, x^0 = 0, \lambda^0 = 0$ , which takes 13.9 ms (229 ADMM iterations).

Next, we further compare the MILP, LM, and proximal ADMM approaches on random GNE problems of increasing size. We consider linear-quadratic games with  $N$  agents, each controlling  $n_i = 2$  decision variables, with cost functions  $f$  as in (22) and shared constraints defined by  $n_g = 2N$  random linear inequalities with unit right-hand side. The results are shown in Figure 1(a), where, for an increasing number  $N$  of agents, we report the CPU time for computing a GNE via MILP and for computing a vGNE by LM and proximal ADMM. Computing GNEs on LQ games via MILP with HiGHS is about two orders of magnitude more efficient than by LM, and even more by using Gurobi's state-of-the-art MILP solver; it also outperforms proximal ADMM (which takes between 258 and 1794 iterations). Figure 1(b) shows instead the CPU time taken when fixing  $N = 5$  agents and increasing the number of shared constraints  $n_g$  up to 300, showing that MILP methods degrade more in performance compared to proximal ADMM due to the increasing number of binary variables in (14).

## 6.2. Inverse linear-quadratic game design

We consider an inverse game-design problem for a GNEP with  $N = 10$  agents, each optimizing a vector  $x_i \in \mathbb{R}^{n_i}$  of  $n_i = 10$  variables ( $x \in \mathbb{R}^{100}$ ), parameterized by a vector  $p \in \mathbb{R}^{n_p}$ , with  $n_p = 5$ , with  $\|p\|_\infty \leq 100$ . For each given  $p$ , each agent  $i = 1, \dots, N$  solves the convex quadratic optimization problem defined in (11) with  $b \in \mathbb{R}^{50}, b_{\text{eq}} \in \mathbb{R}^5$ . All matrices and vectors are randomly generated with appropriate dimensions, with  $Q^i \succ 0$ , and the box constraints are defined by  $u = -\ell = 10 \cdot \mathbf{1}$ .

To formulate the inverse game problem in an ideal setting, we first compute a reference equilibrium  $x_{\text{des}}$  and a corresponding parameter vector  $\hat{p}$  by solving (11) with zero objective  $J(x^*, p) = 0$ . Then, we solve the inverse game-design problem (14) by using either

$J(x^*, p) = \|x^* - x_{\text{des}}\|_\infty$  (MILP problem) or  $J(x^*, p) = \frac{1}{2}\|x^* - x_{\text{des}}\|_2^2$  (MIQP problem) to retrieve a parameter vector  $p^*$  that yields an equilibrium  $x^*(p^*) \approx x_{\text{des}}$ . We use Gurobi's solvers in both cases.

The parameter vector  $p^*$  is retrieved, respectively, in 0.0394 s by MILP with  $\|x^*(p^*) - x_{\text{des}}\|_\infty = 5.4712 \cdot 10^{-13}$ , and 0.0660 s by MIQP with  $\|x^*(p^*) - x_{\text{des}}\|_2 = 1.2434 \cdot 10^{-14}$ .

### 6.3. Stackelberg game

Consider a problem with  $N$  agents (the followers), each controlling a decision variable  $x_i \geq 0, i = 1, \dots, N$ , subject to the shared constraint  $\sum_{i=1}^N x_i \leq C$ . Each agent  $i$  minimizes:

$$f_i(x, p) = a_i x_i^2 + \sum_{j=1}^N \Gamma_{ij} x_i x_j + \pi_i(x, p) x_i$$

where  $a_i > 0$  is a self-cost coefficient,  $\Gamma \in \mathbb{R}^{N \times N}$  is a symmetric interaction matrix, and  $\pi_i(x, p)$  is the nonlinear price function

$$\pi_i(x, p) = p_{1,i} + p_2 \left( \sum_{j=1}^N x_j \right)^2$$

Here  $p = \text{col}(p_{1,1}, \dots, p_{1,N}, p_2)$  is the vector of game parameters chosen by the leader to minimize the loss function

$$J(x, p) = \left( \sum_{i=1}^N x - D \right)^2 + \eta \sum_{i=1}^N (p_{1,i} - \bar{p}_{1,i})^2 - \rho \sum_{i=1}^N \pi_i(x, p) x_i$$

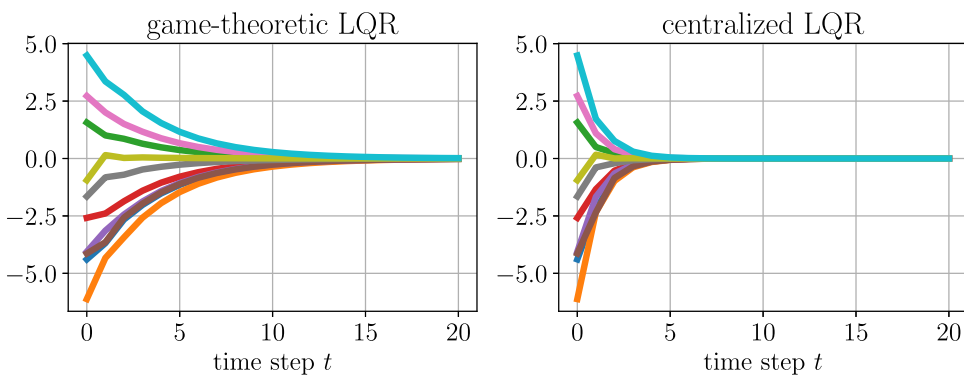
with  $p_{1,i} \in [p_{1,\min}, p_{1,\max}]$  and  $p_2 \in [p_{2,\min}, p_{2,\max}]$ . The numerical values are set as follows:  $N = 8, C = 1, D = 0.9, p_{1,\min} = -5, p_{1,\max} = 0, p_{2,\min} = 0, p_{2,\max} = 0.2, \bar{p}_{1,i} = -2, \eta = 0.1, \rho = 0.3$ , and

$$\Gamma = \begin{bmatrix} 0 & 0.2 & 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0.2 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0.15 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.15 & 0 & 0.1 & 0 & 0 & 0 \\ 0.1 & 0 & 0 & 0.1 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0 \end{bmatrix}, \quad a = \begin{bmatrix} 1.0 \\ 1.5 \\ 0.8 \\ 1.2 \\ 2.0 \\ 0.9 \\ 1.8 \\ 1.1 \end{bmatrix}.$$

By setting  $\rho = 10^8$ , (8) is solved with 83 LM iterations in 1.057 s starting from the initial guess  $p_1^{(0)} = \frac{p_{1,\min} + p_{1,\max}}{2}$ ,  $p_2^{(0)} = \frac{p_{2,\min} + p_{2,\max}}{2}$ , and  $x_i^{(0)} = \frac{C}{N}$ . The optimal parameters and related GNE are

$$p_1^* = -[1.6174 \ 1.8083 \ 1.553 \ 1.6970 \ 1.9523 \ 1.5737 \ 1.9009 \ 1.6576]^\top$$

$$p_2^* = 0.1990$$



**Figure 2.** Game-theoretic vs centralized LQR.

$$x^*(p^*) = [0.0882 \ 0.1745 \ 0.0413 \ 0.1325 \ 0.2002 \ 0.0565 \ 0.1945 \ 0.1124]^T$$

where  $\sum_{i=1}^N x_i^*(p^*) \approx 1 = C$ . The optimal value of the leader's loss function is  $J(x^*(p^*), p^*) = 0.5637$ . We observed that different initializations lead to different GNEs with different values of the leader's loss function, due to the nonconvexity of the optimization problem (8).

#### 6.4. LQR game

We consider a linear system as in (18) with  $n_x = 10$  states and  $n_u = 10$  inputs, each one controlled by a different agent ( $n_i = 1$  for all  $i = 1, \dots, N$ ,  $N = 10$  agents), with  $A, B \in \mathbb{R}^{10 \times 10}$  randomly generated and scaled so that  $A$  is unstable with spectral radius equal to 1.1. We consider state-weighting matrices  $Q_i = e_i e_i^T$  and  $R_i = 10$  for all  $i = 1, \dots, N$ .

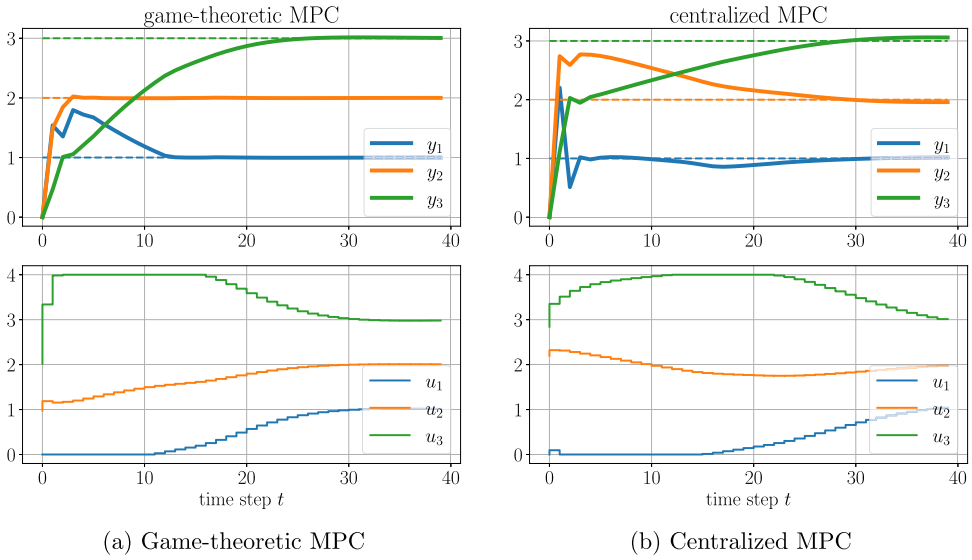
In (19), we set  $N_{\text{LQR}} = 50$ , which is a long-enough horizon used to approximate the infinite-horizon cost, therefore providing a good approximation of the solution of the algebraic Riccati equation associated with the quadruple  $((A - B_{-i}K_{-i}), B_i, Q_i, R_i)$  for each agent  $i$ .

The problem is solved as in (6) (without parameter  $p_0$ ) in 6.05 s (6 LM iterations), leading to an asymptotically closed-loop matrix  $A - BK^*$  with spectral radius 0.7525. For comparison, the centralized (cooperative) LQR solution leads to a spectral radius of 0.3965. Solving the same problem by using the Riccati-based iterations described in [35, Section III.B] leads to the same solution in 0.11 s (7 Riccati iterations).

Figure 2 compares the closed-loop step responses of the combined output vector  $y(t) = \mathbf{1}^T x(t)$  from various initial conditions  $x(0)$  for the noncooperative LQR gain  $K^*$  and the centralized LQR gain  $K_{\text{LQR}}$ .

#### 6.5. Game-theoretic MPC

We consider a linear system as in (18) with  $n_x = 6$  states,  $n_u = 3$  inputs, and  $n_y = 3$  outputs, each one controlled by a different agent ( $n_i = 1$  for all  $i = 1, 2, 3$ ,  $N = 3$  agents), with  $A, B \in \mathbb{R}^{6 \times 3}$  randomly generated and scaled so that  $A$  is stable with spectral radius equal to 0.95. The output matrix  $C \in \mathbb{R}^{3 \times 6}$  is scaled so that the steady-state gain from inputs



**Figure 3.** Closed-loop trajectories for linear MPC: comparison between game-theoretic (competitive) and centralized (cooperative) formulations. (a) Game-theoretic MPC and (b) Centralized MPC.

to outputs is the identity matrix. Each agent  $i$  has state-weighting matrix  $Q_i = C^\top Q_{y,i} C$ , with  $Q_{y,i} = e_i e_i^\top$ ,  $Q_{y,i} \in \mathbb{R}^{3 \times 3}$ , and input-increment weighting  $Q_{\Delta u,i} = 0.5$  for  $i = 1, 2, 3$ . The prediction horizon is  $T = 10$  and the constraint horizon is  $T_c = 3$ . For all  $i = 1, 2, 3$ , the input constraints are  $0 \leq u_i \leq 4$  and the output constraints are  $0 \leq y_i \leq 5$ , with slack penalty  $q_{\epsilon,i} = 10^3$ .

The noncooperative MPC problem is solved at each time step  $t = 0, \dots, T_{\text{sim}} - 1$ , with  $T_{\text{sim}} = 40$ , by solving the MILP reformulation (14) of the GNEP (without parameter  $p$  and objective function (12)) using the HiGHS solver [26]. The resulting closed-loop trajectories are shown in Figure 3(a). For comparison, we also show the closed-loop trajectories obtained by solving a cooperative and centralized MPC problem at each time step  $t$  by jointly optimizing all agents' moves via the QP solver osQP [37].

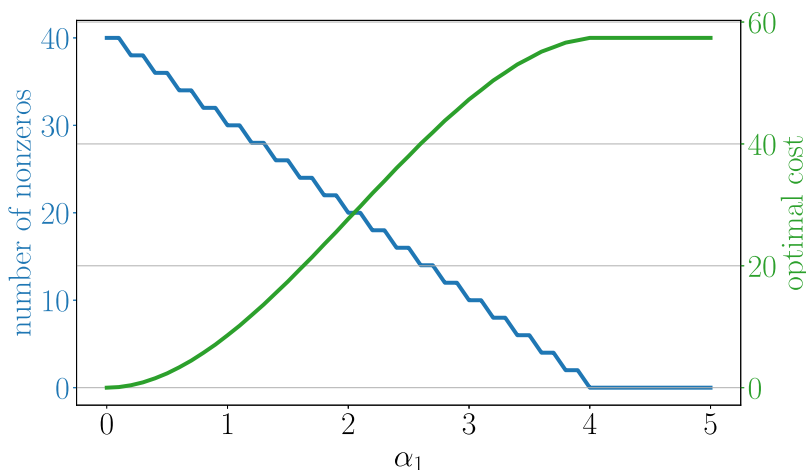
The CPU time for solving each GNEP ranges between 4.36 and 65.62 ms (HiGHS MILP), while the CPU time for solving each centralized MPC problem ranges between 0.57 and 2.80 ms (QP).

## 6.6. Computing sparse equilibria

Consider a Nash equilibrium problem with  $N = 40$  agents, each one controlling a scalar decision variable  $x_i$  and no free parameter (e.g.  $p = 0$ ). The agents are grouped into pairs  $(2k - 1, 2k)$ ,  $k = 1, \dots, 20$ , with the agents' objectives defined as

$$f_i(x) = (x_{2k-1} - x_{2k})^2, \quad k = \lfloor (i + 1)/2 \rfloor.$$

Clearly, if for every agent-pair  $k$  the corresponding variables  $x_{2k-1} = x_{2k}$ , we have a Nash equilibrium  $x^* \in \mathbb{R}^N$ . Among the infinitely many Nash equilibria, we want to minimize



**Figure 4.** Number of nonzeros in  $x^*$  and optimal cost  $J(x^*, 0)$  as a function of the  $\ell_1$ -regularization parameter  $\alpha_1$ .

the game-design objective

$$J(x, 0) = \sum_{i=1}^N (x_i - x_i^{\text{ref}})^2 + \alpha_1 \|x\|_1$$

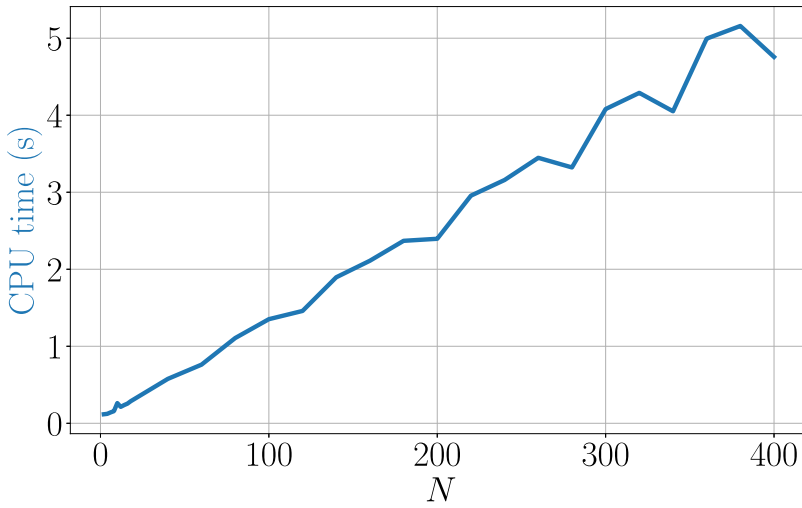
where the components of the reference vector  $x_i^{\text{ref}} = \frac{\lceil (i+1)/2 \rceil}{10}$ ,  $i = 1, \dots, N$ , and  $\alpha_1 > 0$  promotes sparsity in the equilibrium solution.

Figure 4 shows the number of nonzeros in the computed Nash equilibrium  $x^*$  and the optimal cost  $J(x^*, 0)$  as a function of the  $\ell_1$ -regularization parameter  $\alpha_1$ , obtained by solving Problem (10) with  $\rho = 10^4$  and  $P = \{0\}$ . The CPU time is 1.4350 s on the first run due to JAX compilation time, and 29.9 ms on average on the subsequent runs. As expected, as  $\alpha_1$  increases, the number of zeros in  $x^*$  increases, leading to sparser Nash equilibria, at the expense of a larger optimal cost  $J(x^*, 0)$ . Clearly, due to the equilibrium condition  $x_{2k-1}^* = x_{2k}^*$ , the number of nonzeros in  $x^*$  can only decrease by two at a time as  $\alpha_1$  increases.

Figure 5 shows the CPU time required to compute the sparse solution as a function of the number  $N$  of agents for  $\alpha_1 = 2$  using L-BFGS-B. The CPU time grows roughly linearly with  $N$ .

## 7. Conclusions

In this paper, we have proposed different simple methods for designing and computing generalized Nash equilibria in noncooperative games with local and shared constraints. We have presented a mixed-integer linear or quadratic programming reformulation for computing all generalized Nash equilibria of a linear-quadratic game, as well as a nonlinear least-squares approach for computing generalized Nash equilibria of a rather general class of nonlinear games. We have illustrated the proposed methods on several numerical examples, including linear-quadratic games, game-theoretic LQR and MPC control, inverse



**Figure 5.** CPU time to compute the Nash equilibrium as a function of the number  $N$  of agents.

games, and sparsity-promoting GNE computations. We believe that the proposed methods and the associated open-source library can be a useful tool for researchers and practitioners working in the field of game-theory and game-theoretic control and its applications to engineering and economic systems.

## Note

1. The LICQ condition is verified if the gradients of the active constraints at  $x^*(p)$

$$\{\nabla_{x_i} g_j(x^*, p) : j \in \mathcal{A}(x^*)\} \cup \{\nabla_{x_i} h_k(x^*, p)\} \cup \{\pm e_k : x_{i,k}^* = \ell_{i,k} \text{ or } u_{i,k}\}$$

are linearly independent, where  $e_k$  is the  $k$ th unit vector of appropriate dimension,  $\mathcal{A}(x^*) = \{j : g_j(x^*, p) = 0\}$  is the set of active inequality constraints at  $x^*$ , and  $x_{i,k}$ ,  $\ell_{i,k}$ , and  $u_{i,k}$  are the  $k$ th components of  $x_i$ ,  $\ell_i$ , and  $u_i$ , respectively.

## Acknowledgments

We thank the anonymous reviewers for their constructive comments and suggestions.

## Author contributions

CRedit: **Alberto Bemporad**: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

This work was funded by the European Union (ERC Advanced Research Grant COMPACT, No. 101141351). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

## Notes on contributor

*Prof. Alberto Bemporad* received his M.S. degree cum laude in Electrical Engineering (1993) and his Ph.D. in Control Engineering (1997) from the University of Florence, Italy. He held research positions at Washington University in St. Louis (1996–1997) and ETH Zurich (1997–2002). He served at the University of Siena (1999–2009) and at the University of Trento (2010–2011). Since 2011, he has been a Full Professor at the IMT School for Advanced Studies Lucca, where he served as Director from 2012 to 2015. He co-founded ODYS S.r.l. in 2011, a company specializing in industrial model predictive control systems, and has held visiting appointments at Stanford University, the University of Michigan and Zhejiang University. He has authored over 400 publications and 21 patents in model predictive control, quadratic optimization, nonlinear system identification, machine learning and automotive controls, and co-developed the Model Predictive Control Toolbox (MathWorks) for MATLAB. He was an Associate Editor of the IEEE Transactions on Automatic Control from 2001 to 2004 and the Chair of the IEEE CSS Technical Committee on Hybrid Systems from 2002 to 2010. He is an IEEE Fellow (since 2010) and IFAC Fellow (since 2025). His awards include the IFAC High-Impact Paper Award (2011–2014), IEEE CSS Transition to Practice Award (2019), SAE Environmental Excellence in Transportation Award (2021), the Beale-Orchard-Hays Prize (2024) and an ERC Advanced Grant (2024).

## References

- [1] D. Aussel and A. Svensson, *A short state of the art on multi-leader-follower games*, in *Bilevel Optimization*, volume 161, S. Dempe and J. Zemkoho, eds., Springer, 2020, pp. 53–76.
- [2] D. Bauso, *Game Theory with Engineering Applications*, SIAM, Philadelphia, 2016.
- [3] G. Belgioioso, P. Yi, S. Grammatico, and L. Pavel, *Distributed generalized Nash equilibrium seeking: An operator-theoretic perspective*, IEEE Control Syst. Mag. 42(4) (2022), pp. 87–102.
- [4] A. Bemporad, *Hybrid toolbox – user’s guide* (2004). Available at <http://cse.lab.imtlucca.it/bemporad/hybrid/toolbox>.
- [5] A. Bemporad, *An L-BFGS-B approach for linear and nonlinear system identification under  $\ell_1$  and group-lasso regularization*, IEEE Trans. Automat. Contr. 70(7) (2025), pp. 4857–4864.
- [6] A. Bemporad, M. Morari, and N.L. Ricker, *Model Predictive Control Toolbox for MATLAB – User’s Guide*, The Mathworks, Inc., Natick, MA, 2004.
- [7] E. Börgens and C. Kanzow, *ADMM-type methods for generalized Nash equilibrium problems in Hilbert spaces*, SIAM J. Optim. 31(1) (2021), pp. 377–403.
- [8] J. Bradbury, R. Frostig, P. Hawkins, M.J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, *JAX: Composable transformations of Python+NumPy programs* (2018).
- [9] L.F. Bueno, G. Haeser, and F.N. Rojas, *Optimality conditions and constraint qualifications for generalized Nash equilibrium problems and their practical implications*, SIAM J. Optim. 29(1) (2019), pp. 31–54.
- [10] R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM J. Sci. Comput. 16(5) (1995), pp. 1190–1208.
- [11] T.F. Coleman and Y. Li, *An interior trust region approach for nonlinear minimization subject to bounds*, SIAM J. Optim. 6(2) (1996), pp. 418–445.
- [12] A. Dreves and N. Sudermann-Merx, *Solving linear generalized Nash equilibrium problems numerically*, Optim. Methods Softw. 31(5) (2016), pp. 1036–1063.

- [13] A. Dreves, F. Facchinei, C. Kanzow, and S. Sagratella, *On the solution of the KKT conditions of generalized Nash equilibrium problems*, SIAM J. Optim. 21(3) (2011), pp. 1082–1108.
- [14] F. Fabiani and A. Bemporad, *An active learning method for solving competitive multi-agent decision-making and control problems*, IEEE Trans. Automat. Contr. 70(4) (2025), pp. 2374–2389.
- [15] F. Facchinei and C. Kanzow, *Generalized Nash equilibrium problems*, Ann. Oper. Res. 175 (2010), pp. 177–211.
- [16] F. Facchinei, A. Fischer, and C. Kanzow, *Regularity properties of a semismooth reformulation of variational inequalities*, SIAM J. Optim. 8(3) (1998), pp. 850–869.
- [17] F. Facchinei, A. Fischer, and V. Piccialli, *Generalized Nash equilibrium problems and Newton methods*, Math. Program. 117(1-2) (2007), pp. 163–194.
- [18] A. Fischer, *A special Newton-type optimization method*, Optimization 24(3–4) (1992), pp. 269–284.
- [19] B. Franci, F. Fabiani, M. Schmidt, and M. Staudigl, *A Gauss–Seidel method for solving multi-leader-multi-follower games*, in *European Control Conference*, 2025. pp. 2775–2780.
- [20] G. Graser, T. Kreimeier, and A. Walther, *Solving linear generalized Nash games using an active signature method*, Optim. Methods Softw. (2025), pp. 1–24.
- [21] L. Guo, G.-H. Lin, and J.J. Ye, *Solving mathematical programs with equilibrium constraints*, J. Optim. Theory Appl. 166 (2015), pp. 234–256.
- [22] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual* (2024).
- [23] S. Hall and M. Bemporad, *Solving multiparametric generalized Nash equilibrium problems and explicit game-theoretic model predictive control* (2025). Available on arXiv at <https://arxiv.org/abs/2512.05505>. Code available at [https://github.com/bemporad/nash\\_mppq](https://github.com/bemporad/nash_mppq).
- [24] S. Hall, G. Belgioioso, D. Liao-McPherson, and F. Dörfler, *Receding horizon games with coupling constraints for demand-side management*, in *Proceedings of the IEEE 61st Conference on Decision and Control*, 2022. pp. 3795–3800.
- [25] W.P.M.H. Heemels, B. De Schutter, and A. Bemporad, *Equivalence of hybrid dynamical models*, Automatica 37(7) (2001), pp. 1085–1091.
- [26] Q. Huangfu and J.A.J. Hall, *Parallelizing the dual revised simplex method*, Math. Program. Comput. 10(1) (2018), pp. 119–142.
- [27] C. Kanzow and D. Steck, *Augmented Lagrangian methods for the solution of generalized Nash equilibrium problems*, SIAM J. Optim. 26(4) (2016), pp. 2034–2058.
- [28] H. Le Cadre, P. Jacquot abd, C. Wan, and C. Alasseur, *Peer-to-peer electricity market analysis: From variational to generalized Nash equilibrium*, Eur. J. Oper. Res. 282 (2020), pp. 753–771.
- [29] S. Le Cleac’h, M. Schwager, and Z. Manchester, *ALGAMES: A fast augmented Lagrangian solver for constrained dynamic games*, Auton. Robots 46(1) (2022), pp. 201–215.
- [30] K. Levenberg, *A method for the solution of certain non-linear problems in least squares*, Quart. Appl. Math. 2(2) (1944), pp. 164–168.
- [31] M. Liu and I.V. Kolmanovsky, *Input-to-state stability of Newton methods in Nash equilibrium problems with applications to game-theoretic model predictive control*, preprint (2024). Available at arXiv:2412.06186.
- [32] Z.-Q. Luo, J.-S. Pang, and D. Ralph, *Mathematical Programs with Equilibrium Constraints*, Cambridge University Press, 1996.
- [33] D.W. Marquardt, *An algorithm for least-squares estimation of nonlinear parameters*, J. Soc. Ind. Appl. Math. 11(2) (1963), pp. 431–441.
- [34] J. Nocedal and S.J. Wright, *Numerical Optimization*. 2nd ed. Springer, New York, NY, 2006.
- [35] B. Nortmann, A. Monti, M. Sassano, and T. Mylvaganam, *Nash equilibria for linear quadratic discrete-time dynamic games via iterative and data-driven algorithms*, IEEE Trans. Automat. Contr. 69(10) (2024), pp. 6561–6575.
- [36] J.-P. Pang and M. Fukushima, *Quasi-variational inequalities, generalized Nash equilibria, and multi-leader-follower games*, Comput. Manage. Sci. 2(1) (2005), pp. 21–56.
- [37] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, *OSQP: An operator splitting solver for quadratic programs*, Math. Program. Comput. 12 (2020), pp. 637–672.

- [38] T. Tatarenko and M. Kamgarpour, *Learning generalized Nash equilibria in a class of convex games*, IEEE Trans. Automat. Contr. 64(4) (2018), pp. 1426–1439.
- [39] T. Tatarenko, W. Shi, and A. Nedić, *Geometric convergence of gradient play algorithms for distributed Nash equilibrium seeking*, IEEE Trans. Automat. Contr. 66(11) (2021), pp. 5342–5353.
- [40] Z. Wang, F. Liu, Z. Ma, Y. Chen, W. Wei, and Q. Wu, *Distributed generalized Nash equilibrium seeking for energy sharing games in prosumers*, IEEE Trans. Power Syst. 36(6) (2021), pp. 3973–3986.
- [41] P. Yi and L. Pavel, *An operator splitting approach for distributed generalized Nash equilibria computation*, Automatica 102 (2019), pp. 111–121.