Model-Free Game-Theoretic Feedback Optimization

Anurag Agarwal John W. Simpson-Porco Lacra Pavel
University of Toronto, Department of Electrical and Computer Engineering
email: anurag.agarwal@mail.utoronto.ca, jwsimpson@ece.utoronto.ca, pavel@ece.utoronto.ca

***Abstract*— This paper extends recent work in feedback-based, game-theoretic optimization. We first identify limitations of existing approaches to this problem, often requiring a priori knowledge to construct a nominal sensitivity model. Leveraging zero-order optimization techniques inspired by stochastic perturbation, we develop a model-free algorithm that allows agents to estimate these sensitivities during runtime, rather than a priori. We outline the convergence properties of this algorithm as a forward-backward operator-splitting technique. Finally, we compare this model-free algorithm's performance to existing approaches, outlining its benefits and drawbacks.**

I. Introduction

Game theory studies the strategic interaction between multiple self-interested, rational agents (players), each aiming to optimize their own goals. Modelling agents as selfish, rational players enables the implementation of decentralized optimization techniques, where agents are aware of the dynamics in their learning caused by other agents' behaviours [1, 2]. This framework can be applied to a variety of applications such as swarm robotics, [3–5], wireless communication [6, 7], power systems [8], and smart cities [9].

Recent work in [3, 8] studies the case of game-theoretic optimization problems wherein each agent's decision has an impact on an output generated by the systems within the network [10, 11]. In such cases, existing gradient-play techniques for solving game-theoretic problems [4, 12, 13] require a precise model of the output's sensitivity to the players' actions. To alleviate this limitation, a feedback-based optimization framework (akin to [11, 14–16]) can be used, based on a nominal model of the output sensitivities. The robustness of this approximation is studied in [17, 18].

In this work, we consider the scenario wherein the use of a fixed nominal sensitivity model is insufficient, as it may and result in a quite suboptimal equilibrium point, and an adaptive model-free approach is preferred. Our approach is based on the model-free optimization techniques outlined in [19], where two perturbed evaluations of the cost function are used to estimate its gradient for a time-varying optimization problem where the cost function is affected by an unmodelled, exogenous process. In this approach, the function perturbations are chosen quasi-stochastically, rather than stochastically, as this has been shown to reduce noise during convergence [19, 20]. Leveraging these methods, our novel contributions are that we

- replace the nominal sensitivity approximation made in the game-theoretic framework in [8] with an online estimation of the sensitivity, using quasi-stochastic perturbation techniques developed in [19];
- use operator-theoretic techniques from [12] to prove the convergence of the resulting model-free algorithm, by noting that the model-free estimation technique tracks a "perfect information" forward-backward operator-splitting algorithm;
- validate our results on two examples, a power distribution feeder and an academic example motivated by a swarm robotics application, and compare the performance of the model-free technique to the nominal sensitivity method outlined in [8].

Literature Review

Most prior work in this area assumes that the agents are able to compute their local cost function gradients reliably [3, 4, 12, 13]. This assumption becomes unreasonable if the cost function is dependent on an output generated by an unknown or difficult-to-model process or system, as computing the gradient now requires access to the sensitivity (Jacobian matrix) between the agent decisions and system outputs. This difficulty is often addressed with simplifying assumptions; in [3] the agents are said to have decoupled dynamics (and thus it is reasonable to assume each agent has a model of its own output process), and in [8] the output sensitivity is nominally approximated as a fixed Jacobian (which is often only a good model if the output has a near-linear relation to the control inputs). Our work differs from prior work in this area by lifting the requirement for an a priori model entirely, and using zero-order techniques to approximate the model during run time.

Zero-order (or model-free) techniques have been an ongoing area of research in the optimization, control, and machine learning community. The seminal work in this area is [21], which solves a one-dimensional problem using two perturbations. The Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm devised in [22, 23] uses zero-mean, independent, random perturbations and requires two function evaluations at each step of an l-dimensional problem. In [24, 25] quasi-stochastic perturbations, generated by deterministic processes, were used instead to improve convergence properties. Analyses in [19, 20] also show that the quasi-stochastic approaches reduce noise.

Our work closely relates to the recent work in [19, 26]. Similar to these works, we consider a constrained, time-varying distributed optimization problem, and use a perturbation signal to approximate a gradient based on two function evaluations. Our work differs from these as follows:

1) We consider a game-theoretic setting, rather than the additive distributed optimization problem considered in [19], or the centralized optimization problem considered in [26]. The key benefit of this approach is the ability to encode selfish, competitive behaviour in the network and to minimize communication overhead. In this regard, our work also differs from most existing literature in the area of distributed zero-order optimization [27].
2) It was noted in [19] (in which entire gradient was estimated, similar to [26]) that directly estimating sensitivity was an area of future work. Our work uses techniques developed therein to estimate only the sensitivity model, computing the rest of the gradient precisely.
3) As a consequence of only estimating the sensitivity rather than the whole gradient, we use operator-theoretic techniques inspired by [12] to present the error-bound on the convergence of our approach.

Organization: The rest of this paper is organized as follows. Section II outlines the game-theoretic model studied at the core of this problem. Section III proposes a model-free forward-backward operator splitting algorithm and analyzes its convergence. Section IV validates the convergence of the proposed algorithm and compares it to [8]. Section V concludes the paper and proposes future areas of research.

Notation: The sets $\mathbb{R}, \mathbb{R}_+$ denote the sets of real numbers and nonnegative real numbers respectively. Given a symmetric positive definite matrix $P \succ 0$, $\langle \cdot, \cdot \rangle_P : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ denotes the inner product $\langle x, y \rangle_P = x^\top P y$ and $\| \cdot \|_P : \mathbb{R}^n \to \mathbb{R}_+$ denotes the corresponding induced norm $\|x\|_P = \sqrt{x^\top P x}$. If P is omitted, it is assumed that $P = I$, the identity matrix, inducing the Euclidean 2-norm $\| \cdot \|_2$. A block diagonal matrix A with matrices $A_1, \ldots, A_N$ along its diagonal is denoted $A = \operatorname{diag}(A_1, \ldots, A_N)$. Denote $\operatorname{col}(x_1, \ldots, x_N)$ as the column vector obtained by stacking vectors $x_1, \ldots, x_N$. Given a matrix $A \in \mathbb{R}^{n \times m}$, denote $[a_{ij}]$ or $[A]_{ij}$ to be the component in its i^{th} row and j^{th} column. We denote the n-dimensional vector of ones as $\mathbf{1}_n = \operatorname{col}(1, \ \ldots \ , 1) \in \mathbb{R}^n$, the n-dimension vector of zeroes as $\mathbf{0}_n = \operatorname{col}(0, \ \ldots \ , 0) \in \mathbb{R}^n$ and the n-dimensional identity matrix as $I_n = \operatorname{diag}(\mathbf{1}_n) \in \mathbb{R}^{n \times m}$. We denote the zero matrix $\mathbb{0}_{n \times m} \in \mathbb{R}^{n \times m}$, with each element equal to zero. The zero set of a set-valued operator $\mathfrak{A}$ is $\operatorname{zer}(\mathfrak{A}) = \{x : \mathbf{0} \in \mathfrak{A}(x)\}$. We denote the identity operator as Id where $x = \operatorname{Id}(x)$. Given a closed, convex set $\Omega \subset \mathbb{R}^n$, let $N_\Omega(x) = \{y \in \mathbb{R}^n | y^\top (x' - x) \leq 0 \ \forall x' \in \Omega\}$ denote the normal cone of Ω at $x \in \Omega$.

II. Game-Theoretic Model Setup

A. Problem Formulation

Consider a set of players $\mathcal{N} = \{1, \ldots, N\}$, each with some goals it is trying to fulfill in the game. Each player $i \in \mathcal{N}$ chooses an action $u_i \in \Omega_i \subseteq \mathbb{R}^{n_i}$. The overall action set is denoted $\mathbf{u} \in \Omega = \prod_{i \in \mathcal{N}} \Omega_i$. To explicitly show dependence on agent i's actions and all other agents' actions, we often denote $\mathbf{u}$ as $(u_i, \mathbf{u}_{-i})$, where $\mathbf{u}_{-i} \in \Omega_{-i} = \prod_{j \in \mathcal{N} \setminus \{i\}} \Omega_j$ is the vector of all agents' actions other than agent i's. Each agent has a decision cost function $f_i : \Omega \to \mathbb{R}$ that encodes its goals in the game as a function of *all* agents' actions.

We assume that the agents collectively wish to enforce some constraints on an output from a system impacted by their control decisions (for example, safety constraints). We suppose that the output is generated by a continuous mapping $y_i = \pi_i(\mathbf{u}, w)$, where $w \in \mathcal{W} \subseteq \mathbb{R}^p$ is some exogenous disturbance, and the mapping $\pi_i : \Omega \times \mathcal{W} \to \mathbb{R}^{l_i}$ is unknown to each player. Each agent measures a portion of that output $y_i \in \mathbb{R}^{l_i}$, and we define the stacked vectors $\mathbf{y} \in \mathbb{R}^l$, $l = \sum_{i \in \mathcal{N}} l_i$ and the vector $\mathbf{y}_{-i}$ as we defined $\mathbf{u}_{-i}$ above. We define a stacked mapping $\pi : \Omega \times \mathcal{W} \to \mathbb{R}^l$ such that $\mathbf{y} = \pi(\mathbf{u}, w)$, and a mapping π_{-i} such that $\mathbf{y}_{-i} = \pi_{-i}$. We endow each agent with an output cost function $g_i : \mathbb{R}^l \to \mathbb{R}$ which encodes that agent's goals regarding the outputs. Finally, we define affine constraints on the players' actions:

$$\mathcal{U} := \prod_{i=1}^{N} \Omega_i \cap \left\{ \mathbf{u} \in \mathbb{R}^n \,\middle|\, \sum\nolimits_{i=1}^{N} A_i u_i \geq \sum\nolimits_{i \in \mathcal{N}} b_i \right\}. \quad (1)$$

This corresponds to a global constraint $A\mathbf{u} \geq b$, where

$$A := \begin{bmatrix} A_1 & \cdots & A_N \end{bmatrix} \in \mathbb{R}^{m \times n}, \ b := \sum_{i \in \mathcal{N}} b_i \in \mathbb{R}^m. \quad (2)$$

Let $\mathcal{U}_i(\mathbf{u}_{-i}) := \{u_i \in \Omega_i \mid (u_i, \mathbf{u}_{-i}) \in \mathcal{U}\} \subset \mathbb{R}^{n_i}$ denote the feasible decision set for a given player's action, given each other player's chosen action. With this setup, each player is interested in optimizing the following problem:

$$\begin{aligned} \underset{u_i \in \mathbb{R}^{n_i}}{\text{minimize}} \quad & f_i(u_i, \mathbf{u}_{-i}) + g_i(y_i, \mathbf{y}_{-i}) \\ \text{subject to} \quad & y_i = \pi_i(u_i, \mathbf{u}_{-i}, w) \\ & u_i \in \mathcal{U}_i(\mathbf{u}_{-i}) \subset \mathbb{R}^{n_i}. \end{aligned} \quad (3)$$

Assumption 1: For each player $i \in \mathcal{N}$, the mapping $u_i \mapsto f_i(u_i, \mathbf{u}_{-i})$ is continuously differentiable and convex for fixed $\mathbf{u}_{-i}$. The mapping $\mathbf{y} \mapsto g_i(\mathbf{y})$ is continuously differentiable and convex in $\mathbf{y}$. The feasible decision set Ω_i compact and convex. The constrained decision sets, $\mathcal{U}$ and $\mathcal{U}_i(\mathbf{u}_{-i})$ given fixed $\mathbf{u}_{-i}$, have nonempty interiors. □

B. KKT Conditions and Lagrangian

We introduce the notion of a generalized Nash equilibrium, a point chosen such that no player can unilaterally deviate for a lower cost. We refer to [28] for a deep dive into game theory.

Definition 1: A generalized Nash equilibrium (GNE) of the game (3) is a decision profile $\mathbf{u}^* \in \Omega$ such that for all $i \in \mathcal{N}$

$$\begin{aligned} u_i^* \in \arg\min_{u_i} \ & f_i(u_i, \mathbf{u}_{-i}^*) + g_i(y_i, \mathbf{y}_{-i}^*) \\ \text{s.t. } & y_i = \pi_i(u_i, \mathbf{u}_{-i}^*, w) \\ & u_i \in \mathcal{U}_i(\mathbf{u}_{-i}^*). \end{aligned}$$

We next seek to introduce the KKT conditions for the game (3). For a full analysis of how the KKT conditions relate to a GNE as defined above, we refer the reader to Section 4.1 of [8]. Following the analysis for (6)-(8) in the

paper [8], the *variational* KKT conditions for our problem are given by

$$\begin{aligned} \mathbf{0} &\in H_w(\mathbf{u}^*) - A^\top \lambda^* + N_\Omega(\mathbf{u}^*) \\ \mathbf{0} &\in (A\mathbf{u}^* - b) + N_{\mathbb{R}^m_+}(\lambda^*), \end{aligned} \tag{4}$$

where λ^* is a multiplier such that $\lambda_1^* = \cdots = \lambda_N^* = \lambda^*$ at a KKT point[1], and $H_w(\mathbf{u})$ is the *pseudogradient* of the agents' cost functions, defined as

$$\begin{aligned} H_w(\mathbf{u}) = \operatorname{col}(&\nabla_{u_1}[f_1(u_1, \mathbf{u}_{-1}) + g_1(y_1, \mathbf{y}_{-1})], \\ &\ldots, \nabla_{u_N}[f_N(u_N, \mathbf{u}_{-N}) + g_N(y_N, \mathbf{y}_{-N})]). \end{aligned}$$

Applying the chain rule component wise and grouping components, we can express the pseudogradient as

$$H_w(\mathbf{u}) = F(\mathbf{u}) + \mathfrak{E}(\partial_{\mathbf{u}}\pi(\mathbf{u}, w)^\top \partial_{\mathbf{y}} g(\mathbf{y})^\top), \tag{5}$$

where $F(\mathbf{u}) = \operatorname{col}(\nabla_{u_1} f_1(\mathbf{u}), \ldots, \nabla_{u_N} f_N(\mathbf{u}))$ is the pseudogradient of the agents' decision cost functions and $g(\mathbf{y})$ is the stacked version of the output cost functions g_i. The operator $\mathfrak{E} : \mathbb{R}^{n \times N} \to \mathbb{R}^n$ is a linear operator defined as

$$\mathfrak{E}(M) = \sum\nolimits_{k=1}^{N} \sum\nolimits_{l \in \mathcal{N}_k} (e_l^\top M e_k) e_l, \tag{6}$$

where $\mathcal{N}_k = \left\{1 + \sum_{i=1}^{k-1} n_i \ , \cdots , \ \sum_{i=1}^{k} n_i\right\}$. Under Assumption 1, the existence of a point that satisfies the variational KKT conditions (vKKT point for short) is guaranteed by Corollary 2.2.5 of [29]. We now seek to modify the algorithm presented in [8] to estimate the Jacobian $\partial_{\mathbf{u}}\pi(\mathbf{u}, w)$ during runtime, using zero-order measurements of the output.

III. Model-Free Algorithm

A. Algorithm Development

We begin our analysis by noting that the true Jacobian $\partial_{\mathbf{u}}\pi$ can be expressed row-wise as

$$[\partial_{\mathbf{u}}\pi(\mathbf{u}, w)]_{ij} = \nabla_{\mathbf{u}}^\top \pi_{ij}(\mathbf{u}, w) \tag{7}$$

where for each player $i \in \mathcal{N}$, the mapping π_{ij} represents the j^{th} component of their measured output, with $j \in \{1, \ldots, l_i\}$. We now seek a relation that allows us to approximate these rows using measurements from the vectors $y_i \in \mathbb{R}^{l_i}$. We borrow the following result from [19].

Lemma 1: Let $f : \mathbb{R}^n \to \mathbb{R}$ be a $\mathbb{C}^3$ function, with Lipschitz continuous ∇f and $\nabla^2 f$. Consider a point $x \in \mathbb{R}^n$, a perturbation vector $\phi \in \mathbb{R}^n$, and a perturbation scaling constant $\epsilon \in \mathbb{R}$. Then

$$\frac{1}{2\epsilon}\phi\left[f(x + \epsilon\phi) - f(x - \epsilon\phi)\right] = \phi\phi^\top \nabla f(x) + O(\epsilon^2) \tag{8}$$

We use the left side of (8) as an approximation of the gradient function f, and denote it:

$$\hat{\nabla} f(x; \phi, \epsilon) := \frac{1}{2\epsilon}\phi\left[f(x + \epsilon\phi) - f(x - \epsilon\phi)\right]. \tag{9}$$

The right side of (8) quantifies the error between the two-perturbation gradient evaluation from a true gradient of the function f. The proof for Lemma 1 in [19] requires the Taylor Theorem, thus we outline a differentiability assumption.

Assumption 2: For each agent i, each output mapping $\pi_{ij} : \Omega \times \mathcal{W} \to \mathbb{R}$, $j \in \{1, \ \ldots \ , l_i\}$ is $\mathbb{C}^3$ and has Lipschitz continuous $\nabla\pi_{ij}(\mathbf{u}, w)$ and $\nabla^2\pi_{ij}(\mathbf{u}, w)$.

We follow the quasi-stochastic approach outlined in [19, 20]. To approximate the gradient, our algorithm will iterate over some steps $k \in \{1, \ldots, K\}$ in the same manner as [8]. We seek to provide a technique to sample a perturbation vector $\phi_k \in \mathbb{R}^n$ during each of these steps.

Assumption 3: The perturbation vector $\phi_k \in \mathbb{R}^n$, at the iteration $k \in \{1, \ \ldots \ K\}$, is sampled from a continuous time signal $\phi(t)$ with sampling period T_s as $\phi_k = \phi(kT_s)$. The signal $\phi(t)$ is continuous, periodic with period $T \in \mathbb{R}$, and T_s is chosen such that $T_s > 2T$. The signal $\phi(t)$ satisfies

$$\frac{1}{T}\int_t^{t+T} \phi(\tau)\phi(\tau)^\top d\tau = I_n, \qquad t \in \mathbb{R}. \tag{10}$$

We refer the reader to [19] for a detailed proof on why this assumption minimizes the error between gradient approximations and the true gradient. We next outline the notations used within our algorithm. Each player has access to its local cost function data f_i and g_i and can compute the gradients $\nabla_{u_i} f_i$ and $\nabla_{y_j} g_i$. The approximation of the sensitivity matrix $\partial_{\mathbf{u}}\pi$ at the k^{th} iteration is denoted as $\hat{\Pi}_k \in \mathbb{R}^{l \times n}$, and each player computes the submatrix $\hat{\Pi}_{i,k} \in \mathbb{R}^{l_i \times n}$. The matrix $\hat{\Pi}_{i,k}$ is composed of submatrices $\hat{\Pi}_{iq,k} \in \mathbb{R}^{l_i \times n_q}$, each of which corresponds to the sensitivity of the outputs of player i to the inputs from player $q \in \mathcal{N}$.

At the k-th iteration, the perturbation vector is denoted ϕ_k, and player i's segment of that vector is denoted $\phi_{i,k}$. The perturbed inputs are denoted $u_{i,k+} := u_{i,k} + \epsilon\phi_{i,k}$ and $u_{i,k-} := u_{i,k} - \epsilon\phi_{i,k}$ respectively, and their corresponding output measurements are denoted $y_{i,k+}$ and $y_{i,k-}$.

The global affine constraint $A\mathbf{u} \geq b$ is not known in full by any agent; each agent only knows A_i, b_i, and Ω_i, which characterize its own involvement in the constraint. Agent i controls its local decision $u_i \in \mathbb{R}^{n_i}$ and a local multiplier $\lambda_i \in \mathbb{R}^m_+$. It has a local auxiliary variable $z_i \in \mathbb{R}^m$ to coordinate with neighbours and achieve consensus on λ_i. The *interference graph* $\mathcal{G}_f$ describes communication between agents with cost-decision dependencies, as in [12, 30], with edges (i, j) for any agents i and j that communicate. The weighted *multiplier graph* $\mathcal{G}_\lambda$ similarly describes the (two-way) sharing of λ_i and z_i between agents, with $w_{ij} = w_{ji}$ being edge weights. The neighbour set of agent i is defined as all agents with whom it shares an edge, denoted $\mathcal{N}_{\mathcal{G}_f}(i)$ (similarly $\mathcal{N}_{\mathcal{G}_\lambda}(i)$).

Assumption 4: The graph $\mathcal{G}_\lambda$ is undirected and connected.

With that we are ready to present our algorithm.

[1]The equality of the Lagrange multipliers enforces that each player is equally responsible for obeying the global coupling constraints.

Algorithm 1 Distributed OA vGNE-seeking algorithm with Jacobian Estimation

Initialization: $u_{i,0} \in \Omega_i$, $\lambda_i \in \mathbb{R}^m_+$, and $z_{i,0} \in \mathbb{R}^m_+$

Iteration $k \in \{0, \dots, K\}$**: Player** $i \in \mathcal{N}$

Step 1: Exploration - Applies $u_{i,k}, u_{i,k+}$ and $u_{i,k-}$ in order. Receives corresponding output measurements $y_{i,k}, y_{i,k+}$ and $y_{i,k-}$, and *total* perturbation vector ϕ_k.

Step 2a: Jacobian Approximation - For each $j \in \{1, \dots, l_i\}$, updates:

$$\hat{\nabla}_{\mathbf{u}} \pi_{ij,k} \leftarrow \tfrac{1}{2\epsilon} \phi_k \left[(y_{i,k+})_j - (y_{i,k-})_j \right]$$
$$\hat{\Pi}_{ij,k} \leftarrow \hat{\nabla}^{\top}_{\mathbf{u}} \pi_{ij}$$

Step 2b: Approx. Primal Step and Consensus - Receives $u_{j,k}$, $y_{j,k}$, $j \in \mathcal{N}_{\mathcal{G}_f}(i)$, $\lambda_{j,k}$, $j \in \mathcal{N}_{\mathcal{G}_\lambda}(i)$, $\hat{\Pi}_{ji}$, $j \in \mathcal{N} \backslash \{i\}$, $y_{i,k}$, and updates:

$$u_{i,k+1} \leftarrow P_{\Omega_i}\Big(u_{i,k} - \tau_i (\nabla_{u_{i,k}} f_i(u_{i,k}, \mathbf{u}_{-i,k}) + \sum_{\mathcal{N}_{j \in \mathcal{G}_f}(i)} \hat{\Pi}^{\top}_{ji,k} \nabla_{y_j} g_i(y_{i,k}, \mathbf{y}_{-i,k}) - A_i^{\top} \lambda_{i,k}) \Big)$$

$$z_{i,k+1} \leftarrow z_{i,k} + \nu_i \sum_{j \in \mathcal{N}_{\mathcal{G}_\lambda}(i)} w_{ij} (\lambda_{i,k} - \lambda_{j,k})$$

Step 3: Consensus and Dual Step - Receives $z_{j,k+1}$, $j \in \mathcal{N}_{\mathcal{G}_\lambda}(i)$ and updates:

$$\lambda_{i,k+1} \leftarrow P_{\mathbb{R}^m_+}\Big(\lambda_{i,k} - \sigma_i [A_i(2u_{i,k+1} - u_{i,k}) - b_i + \sum_{j \in \mathcal{N}_{\mathcal{G}_\lambda}(i)} w_{ij} [2(z_{i,k+1} - z_{j,k+1}) - (z_{i,k} - z_{j,k})] + \sum_{j \in \mathcal{N}_{\mathcal{G}_\lambda}(i)} w_{ij} (\lambda_{i,k} - \lambda_{j,k}) \Big)$$

This algorithm is a modification of the forward-backward algorithm outlined in [8, 12], with the difference being the estimation of the Jacobian in steps 1 and 2a. We next present a convergence-analysis for the algorithm. The choice of the scalar step sizes τ_i, ν_i, and σ_i is formalized later.

B. Convergence Analysis

We define a variation of the algorithm presented in [8], where the Jacobian $\partial_{\mathbf{u}} \pi$ is known precisely, as the "golden algorithm". The key idea for the convergence of Algorithm 1 is that it tracks (within bounded error) this golden algorithm. We can show that the golden algorithm converges to a vKKT point of the game, which then leads to the conclusion that Algorithm 1 converges to a limit set around that point. We begin by representing our algorithm as a forward-backward iteration. We first note that a point $\mathbf{u}^*$ and a corresponding Lagrange multiplier λ^* satisfying the vKKT conditions (4) must also satisfy $\mathrm{col}(\mathbf{u}^*, \lambda^*) \in \mathrm{zer}(\mathfrak{A} + \mathfrak{B})$ where

$$\begin{aligned} &\mathfrak{A} : \mathrm{col}(\mathbf{u}, \lambda) \mapsto \mathrm{col}(H_w(\mathbf{u}), -b) \\ &\mathfrak{B} : \mathrm{col}(\mathbf{u}, \lambda) \mapsto \mathrm{col}(-A^{\top}\lambda + N_{\Omega}(\mathbf{u}), A\mathbf{u} + N_{\mathbb{R}^m_+}(\lambda)). \end{aligned} \tag{11}$$

The above result follows from Theorem 2 in [12]. We introduce some additional notation to allow us to express the algorithm in a compact form. Define $\mathbf{u}_k = \mathrm{col}(u_{1,k}, \dots, u_{N,k})$, $\overline{\lambda}_k = \mathrm{col}(\lambda_{1,k}, \dots, \lambda_{N,k})$, with $\overline{z}_k$ and $\overline{b}$ defined similarly. Let $\Lambda = \mathrm{diag}(A_1, \dots, A_N)$ and $\overline{L} = L \otimes I_m$, where L is the Laplacian matrix of the communication graph $\mathcal{G}_\lambda$ (Definition 6.1, [31]). Finally, let $\overline{\tau} = \mathrm{diag}(\tau_1 I_{n_1}, \dots, \tau_N I_{n_N})$ and $\overline{\tau}^{-1} = \mathrm{diag}(\frac{1}{\tau_1} I_{n_1}, \dots, \frac{1}{\tau_N} I_{n_N})$, with $\overline{\nu}$, $\overline{\sigma}$, $\overline{\nu}^{-1}$, $\overline{\sigma}^{-1}$ defined similarly. With this, we can use Lemma 1 from [12] to rewrite each step of the algorithm as

$$\varpi_{k+1} = (\mathrm{Id} + \Phi^{-1}\bar{\mathfrak{B}})^{-1} (\mathrm{Id} - \Phi^{-1}\bar{\mathfrak{A}})(\varpi_k), \tag{12}$$

where $\varpi = \mathrm{col}(\mathbf{u}, \lambda)$, the operators $\bar{\mathfrak{A}}$ and $\bar{\mathfrak{B}}$ are defined as

$$\begin{aligned} &\bar{\mathfrak{A}} : \varpi \mapsto \mathrm{col}(H_w(\mathbf{u}), \mathbf{0}, \bar{L}\bar{\lambda} - \bar{b}) \\ &\bar{\mathfrak{B}} : \varpi \mapsto N_{\Omega}(\mathbf{u}) \times \mathbf{0} \times N_{\mathbb{R}^{mN}_+}(\bar{\lambda}) + \Psi\varpi, \end{aligned} \tag{13}$$

and the matrices Φ and Ψ are defined identically to [12] as

$$\Phi = \begin{bmatrix} \bar{\tau}^{-1} & \mathbb{0} & \Lambda^{\top} \\ \mathbb{0} & \bar{\nu}^{-1} & \bar{L} \\ \Lambda & \bar{L} & \bar{\sigma}^{-1} \end{bmatrix}, \ \Psi = \begin{bmatrix} \mathbb{0} & \mathbb{0} & -\Lambda^{\top} \\ \mathbb{0} & \mathbb{0} & -\bar{L} \\ \Lambda & \bar{L} & L \end{bmatrix}. \tag{14}$$

Finally, we impose an additional assumption so that this forward-backward iteration matches the assumptions in [12].

Assumption 5: The pseudogradient $H_w(\mathbf{u})$ defined in (5) is $\bar{\eta}$-strongly monotone and $\bar{\theta}$-Lipschitz continuous.

Assumption 5 appears rather strict, given that the mapping π is unknown. In Chapter 5 of [32], we developed techniques using matrix inequalities to verify Assumption 5 without full knowledge of the mapping π. These techniques are omitted here for reasons of space.

Proposition 1: Suppose Assumptions 1-5 hold. Take $0 < \beta \leq \min\left\{ \frac{1}{2d^*}, \frac{\bar{\eta}}{\bar{\theta}^2} \right\}$, where d^* is the maximal weighted degree of $\mathcal{G}_\lambda$, and $\bar{\eta}$, $\bar{\theta}$ are the monotonicity and Lipschitz parameters from Assumption 5. Take $\delta > \frac{1}{2\beta}$, and choose step-sizes τ_i, ν_i, σ_i to satisfy the following:

$$\begin{aligned} &0 < \tau_i \leq \left(\max_{j=1,\dots,n_i} \left\{ \sum\nolimits_{k=1}^{m} |[A_i^{\top}]_{jk}| \right\} + \delta \right)^{-1} \\ &0 < \sigma_i \leq \left(\max_{j=1,\dots,m} \left\{ \sum\nolimits_{k=1}^{n_i} |[A_i]_{jk}| \right\} + 2d_i + \delta \right)^{-1} \\ &0 < \nu_i \leq (2d_i + \delta)^{-1}. \end{aligned} \tag{15}$$

Then, using the golden algorithm (as defined in the beginning of this section), each player's local strategy $u_{i,k}$ converges to its corresponding component of a point satisfying the vKKT conditions (4), and their local multipliers $\lambda_{i,k}$ converge to the multiplier satisfying those conditions for that point, i.e., $\lim_{k\to\infty} u_{i,k} = u_i^*$ and $\lim_{k\to\infty} = \lambda^*$, $i \in \mathcal{N}$.

Proof: The assumptions ensure that the operators $\mathfrak{A}$ and $\mathfrak{B}$, as well as the augmented operators $\bar{\mathfrak{A}}$ and $\bar{\mathfrak{B}}$ satisfy Lemmas 1, and 5-7 in [12]. From that, we simply apply the proof for Theorem 3 in [12], with the caveat being that a fixed point of the iteration is not guaranteed to be a vGNE of the game (3), rather it simply fulfills the vKKT conditions (4). This difference is due to the potential non-convexity introduced into the cost function via the mapping π. ■

Next, we aim to represent Algorithm 1 presented here as a similar forward-backward iteration. We define equivalents for $\mathfrak{A}$ and $\bar{\mathfrak{A}}$ (note that $\mathfrak{B}$ and $\bar{\mathfrak{B}}$ remain unchanged for

the two algorithms). First we need an equivalent to the pseudogradient operator H_w to be used in Algorithm 1:

$$H_{w,\hat{\Pi}}(\mathbf{u}) = F(\mathbf{u}) + \mathfrak{E}\big(\hat{\Pi}^\top \partial_{\mathbf{y}} g(\mathbf{y})^\top\big), \tag{16}$$

where $\hat{\Pi}$ is the approximation of the Jacobian matrix $\partial_{\mathbf{u}}\pi$, as computed using the model-free method outlined above. Note here that when applied to a *single* iteration, the matrix $\hat{\Pi}$ is a constant, even though it varies over the course of the algorithm. We also suppress the index k when writing it as $\hat{\Pi}$. Then the equivalent operators can be defined as

$$\hat{\mathfrak{A}} : \mathrm{col}(\mathbf{u}, \lambda) \mapsto \mathrm{col}(H_{w,\hat{\Pi}}(\mathbf{u}), -b) \tag{17}$$

$$\hat{\bar{\mathfrak{A}}} : \varpi \mapsto \mathrm{col}(H_{w,\hat{\Pi}}(\mathbf{u}), \mathbf{0}, \bar{L}\bar{\lambda} - \bar{b}). \tag{18}$$

Steps 2b and 3 of Algorithm 1 can then be similarly written as a forward-backward iteration:

$$\hat{\varpi}_{k+1} = (\mathrm{Id} + \Phi^{-1}\bar{\mathfrak{B}})^{-1}(\mathrm{Id} - \Phi^{-1}\hat{\bar{\mathfrak{A}}})(\hat{\varpi}_k), \tag{19}$$

Consider some arbitrary point ϖ_k, and suppose we apply one iteration of the golden algorithm to obtain ϖ_{k+1}. Similarly apply Algorithm 1 to the original ϖ_k to obtain $\hat{\varpi}_{k+1}$. We denote the backward step of the iteration as $T_2 := (\mathrm{Id} + \Phi^{-1}\bar{\mathfrak{B}})^{-1}$, and the respective forward step as $T_1 := (\mathrm{Id} - \Phi^{-1}\bar{\mathfrak{A}})$ and $\hat{T}_1 := (\mathrm{Id} - \Phi^{-1}\hat{\bar{\mathfrak{A}}})$. From Lemma 5 in [12], $\bar{\mathfrak{B}}$ is maximally monotone. Thus, by Proposition 23.7 in [33], T_2 is firmly nonexpansive. Following the definition of nonexpansive operators (Definition 4.1 in [33]), we say

$$\begin{aligned}\|\varpi_{k+1} - \hat{\varpi}_{k+1}\|_\Phi^2 &= \|T_2 T_1 \varpi_k - T_2 \hat{T}_1 \varpi_k\|_\Phi^2 \\ &\le \|T_1 \varpi_k - \hat{T}_1 \varpi_k\|_\Phi^2.\end{aligned}$$

We expand the definition of T_1 above and simplify to obtain

$$\|T_1 \varpi_k - \hat{T}_1 \varpi_k\|_\Phi^2 = \|\Phi^{-1}(\bar{\mathfrak{A}} - \hat{\bar{\mathfrak{A}}})(\varpi_k)\|_\Phi^2.$$

Assuming that the steps-sizes for both algorithms are chosen as per Proposition 1 above, we can apply Lemma 6 from [12] to conclude that $\Phi - \delta I_{n+2mN}$ is positive semidefinite. Let $\sigma_{\max}(\Phi)$ and $\sigma_{\min}(\Phi)$ be the maximal and minimal eigenvalues of Φ respectively, then $\sigma_{\max}(\Phi) \ge \sigma_{\min}(\Phi) \ge \delta$. Further, since $\|\Phi\|_2 = \sigma_{\max}(\Phi)$, $\frac{1}{\|\Phi^{-1}\|_2} = \sigma_{\min}(\Phi)$, we know that $\|\Phi^{-1}\|_2 \le \frac{1}{\delta}$. Noting that $\bar{\mathfrak{A}}$ and $\hat{\bar{\mathfrak{A}}}$ are both single-valued (for a given value of $\hat{\Pi}$), and the matrix Φ^{-1} is nonsingular, we know that for any $x, y \in \bar{\Omega}$

$$\begin{aligned}&\|\Phi^{-1}(\bar{\mathfrak{A}} - \hat{\bar{\mathfrak{A}}})(\varpi_k)\|_\Phi^2 \\ =\ & \left[(\bar{\mathfrak{A}} - \hat{\bar{\mathfrak{A}}})(\varpi_k)\right]^\top \Phi^{-1} \left[(\bar{\mathfrak{A}} - \hat{\bar{\mathfrak{A}}})(\varpi_k)\right] \\ =\ & \|(\bar{\mathfrak{A}} - \hat{\bar{\mathfrak{A}}})(\varpi_k)\|_{\Phi^{-1}}^2 \\ \le\ & \frac{1}{\delta}\|(\bar{\mathfrak{A}} - \hat{\bar{\mathfrak{A}}})(\varpi_k)\|_2^2.\end{aligned}$$

We can then expand the definitions of $\bar{\mathfrak{A}}$ and $\hat{\bar{\mathfrak{A}}}$ to obtain

$$\frac{1}{\delta}\|(\bar{\mathfrak{A}} - \hat{\bar{\mathfrak{A}}})(\varpi_k)\|_2^2 = \frac{1}{\delta}\left\|\mathfrak{E}\left[\left(\partial_{\mathbf{u}}\pi(\mathbf{u}, w) - \hat{\Pi}\right)^\top \partial_{\mathbf{y}} g(\mathbf{y})^\top\right]\right\|_2^2,$$

where $\mathbf{y} = \pi(\mathbf{u}, w)$ as before. Since $\mathfrak{E}$ is linear and bounded, and stating its upper bound to be $E_{\max} \in \mathbb{R}$,

$$\begin{aligned}&\|\varpi_{k+1} - \hat{\varpi}_{k+1}\|_\Phi \\ \le\ & \frac{E_{\max}}{\delta}\left\|\left(\partial_{\mathbf{u}}\pi(\mathbf{u}, w) - \hat{\Pi}\right)^\top \partial_{\mathbf{y}} g(\pi(\mathbf{u}, w))^\top\right\|_2^2,\end{aligned} \tag{20}$$

i.e., the distance between an iterate of the golden algorithm and an iterate of Algorithm 1 is bounded by the expression on the right-hand side, if we can show that the norm of the difference between the true Jacobian and estimated Jacobian is bounded. Note that $\hat{\Pi}$ is a value that is unique to a specific iteration, and varies quasi-stochastically during runtime. We note that at each iteration, a single row of the Jacobian is constructed as per (9), and thus satisfies Lemma 1:

$$\hat{\nabla}_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w) = \phi\phi^\top \nabla_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w) + O(\epsilon^2),$$

Hence the error between a row of the estimate and that of the true Jacobian can be written as

$$\begin{aligned}&\|\hat{\nabla}_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w) - \nabla_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w)\|_2 \\ \le\ & \|(\phi\phi^\top - I_n)\|_2 \|\nabla_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w)\|_2 + O(\epsilon^2).\end{aligned}$$

Since the perturbations are sinusoidal, there exists a worst-case vector $\bar{\phi}$ that maximizes the error between the estimated row and the true row. Then, for any iteration,

$$\begin{aligned}&\|\hat{\nabla}_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w) - \nabla_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w)\|_2 \\ \le\ & \|(\bar{\phi}\bar{\phi}^\top - I_n)\|_2 \|\nabla_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w)\|_2 + O(\epsilon^2).\end{aligned} \tag{21}$$

We introduce an additional assumption now, to ensure that this estimation error remains bounded for the disturbance.

Assumption 6: Given an output mapping $\pi(\mathbf{u}, w)$, defined in (3), it has a finite sensitivity to all possible action profiles $\mathbf{u}$ for any disturbance w, i.e., there exists some $S_{\max} < \infty$ such that for each $j \in \{1, \ \dots \ , l_i\}$, and for each $i \in \mathcal{N}$

$$\|\nabla_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w)\|_2 \le S_{\max} \ \forall \mathbf{u} \in \Omega, \ \forall w \in \mathcal{W}.$$

Combining the bound (21) with Assumption 6, we obtain

$$\begin{aligned}&\|\hat{\nabla}_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w) - \nabla_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w)\|_2 \\ \le\ & \sigma_{\max}(\bar{\phi}\bar{\phi}^\top - I_n) S_{\max} + O(\epsilon^2),\end{aligned} \tag{22}$$

where $\sigma_{\max}(\bar{\phi}\bar{\phi}^\top - I_n)$ is the maximal eigenvalue of the matrix $(\bar{\phi}\bar{\phi}^\top - I_n)$. Thus each row of the estimated Jacobian $\hat{\Pi}$ is within a bounded distance of the corresponding row of the true Jacobian $\partial_{\mathbf{u}}\pi(\mathbf{u}, w)$. To combine (20) and (22), we note that the induced 2-norm of a matrix is upper-bounded by its Frobenius norm, which is in turn equal to the sum of the 2-norms of its rows. Summing the 2-norms of each row is upper-bounded by multiplying the right side of (22) by the number of rows (l), and thus

$$\begin{aligned}\|\varpi_{k+1} - \hat{\varpi}_{k+1}\|_\Phi^2 \le \frac{E_{\max} l^2}{\delta} \|\partial_{\mathbf{y}} g(\pi(\mathbf{u}, w))\|_2^2 (\sigma_{\max}(\bar{\phi}\bar{\phi}^\top \\ - I_n) S_{\max} + O(\epsilon^2))^2.\end{aligned} \tag{23}$$

Combining with Proposition 1, we conclude that Algorithm 1 converges to a limit set around a KKT point of game (3).

Theorem 1: Suppose that Assumptions 1-6 are satisfied. Take $0 < \beta \le \min\left\{\frac{1}{2d^*}, \frac{\bar{\eta}}{\theta^2}\right\}$, where d^* is the maximal

weighted degree of $\mathcal{G}_\lambda$, and $\bar{\eta}$, $\bar{\theta}$ are the monotonicity and Lipschitz parameters from Assumption 5. Take $\delta > \frac{1}{2\beta}$. Suppose that τ_i, ν_i, σ_i are chosen to satisfy Proposition 1. Then with Algorithm 1, the overall action profile $\mathbf{u}$ converges to an open ball $\mathcal{B}^n_\kappa(\mathbf{u}^*)$ where $\mathbf{u}^*$ is a point satisfying the vKKT conditions (4) (for some consensus value of the multipliers λ^*), and the radius κ is defined by

$$\kappa = \frac{E_{\max} l^2}{\delta} \left\| \partial_{\mathbf{y}} g(\pi(\mathbf{u}^*, w)) \right\|_2^2 \left(\sigma_{\max} + S_{\max} + O(\epsilon^2) \right)^2 .$$

IV. Simulation Studies

A. Control of a Distribution Feeder

We first consider a practical example, arising in control of renewable power systems. Our setup is identical to that of [8], with a power distribution feeder whose details can be found in [17]. The agents in this game-theoretic problem are the nodes equipped with photovoltaic (PV) panels. Variations in solar irradiance can cause voltage to rise above acceptable safety levels. Each node's goal is to maximize power production (minimize curtailment) while enforcing safety limits.

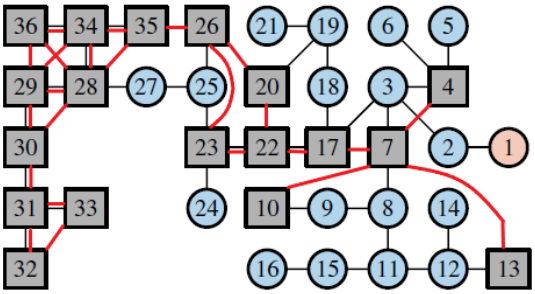

Fig. 1. IEEE 37-node feeder [11]. Node 1 is the Point of Common Coupling (PCC), with all other nodes connected to a load and a voltage sensor. The square nodes are equipped with PV systems. Black lines denote electrical connections in the feeder and red lines are edges of the multiplier graph $\mathcal{G}_\lambda$.

We let the players in this game be the set $\mathcal{N} = \{4, 7, \ldots, 36\}$, the PV-equipped, controllable nodes within this distribution feeder (the grey nodes in Figure 1). The voltage levels throughout the network are dictated by the power flow equations of the distribution feeder. We assume the nodes can measure the voltages, but do not have access to a system model to compute them with. We define the safety constraints on each output to be the set $\mathcal{Y} = \prod_{i \in \mathcal{V}} \mathcal{Y}_i$, where $\mathcal{Y}_i = \{y_i \mid \underline{y} \leq y_i \leq \bar{y}\}$, with $\underline{y} = 0.95$ p.u. and $\bar{y} = 1.05$ p.u. Then each PV-equipped node, $i \in \mathcal{N}$, aims to solve the following optimization problem:

$$\begin{aligned} \min_{u_i} \; & \|(u_{\text{ref}})_i - u_i\|^2 + \frac{1}{2} \sum_{j \in \mathcal{N}} [\max(0, \underline{y} - y_j, y_j - \bar{y})]^2 \\ \text{s.t. } & y_i = \pi_i(\mathbf{u}, w) \\ & u_i \in \mathcal{U}_i(\mathbf{u}_{-i}). \end{aligned} \tag{24}$$

The coupling constraints set is chosen to enforce an upper bound on the total power curtailed throughout the feeder, i.e., $\sum_{i \in \mathcal{N}} (p_i^{\max} - p_i) \leq l$ where we use $l = 0.02$ p.u. We use a perturbation distance of $\epsilon = 0.0001\sqrt{2}$, and we use the perturbation signals $\phi_i(t) = \sqrt{2}[sin(2\pi f_{ip} t)\ sin(2\pi f_{iq} t)]^\top$ $i \in \{1, \ldots, 18\}$, with f_{ip} and f_{iq} sampled uniquely from the ranges $[2.2\ 3.9]$ Hz and $[4.1\ 5.8]$ Hz respectively. The frequency ranges were tuned to yield the least noisy results, similar to [19]. A standard, algorithmic approach for performing this tuning is an area of future research.

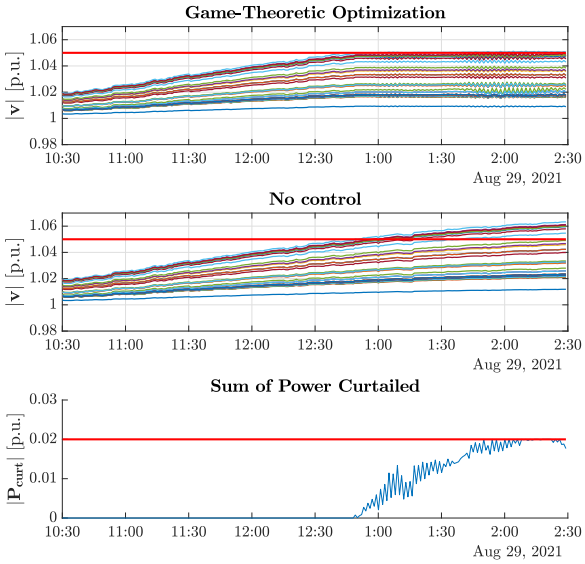

Fig. 2. Simulation results for 4 hours of solar irradiance data of an IEEE37 distribution feeder.

Figure 2 illustrates the results of the optimization over 4 hours of real-time solar irradiance data (with a resolution of 1 second). As solar irradiance causes power fluctuations (peaking at 2:30 pm), the model-free controllers curtail the requisite power to ensure that voltages are maintained within safety limits. Note that due to the uncertainty and stochasticity in the sensitivity estimation, power fluctuates to a larger degree than in the nominal approach in [8]. The model-free approach thus requires less overhead, as a nominal model needn't be calculated, but has poorer convergence properties due to the uncertainties in the model-estimation framework.

B. Simple Swarm Robotics Example

We next consider a problem inspired by swarm robotics, where the robots $i \in \{1, 2\}$ each seek to approach a target position $\bar{r}_i \in \mathbb{R}^2$. We assume that the robots are equipped with a position controller that asymptotically tracks position set-point $u_i \in \mathbb{R}^2$, in this instance a PD controller. Thus the robot position $r_i(t)$ asymptotically approaches the chosen control input u_i. We suppose that each robot measures the squared distance between the two, biased by a disturbance $w_i \in \mathbb{R}$ (e.g., noise on the channel that they communicate with each other while localizing). Each robot seeks to drive to the target position $\bar{r}_i$. Further, the robots wish to reach their targets while ensuring that they do not go further than

a specified distance away from one another. This constraint could be motivated, for example, by a communication constraint for the robots, the distance marking a point of failure.

Finally we outline the coupling constraints between the robots. We aim to prevent collisions between the robots by constraining the 1-norm of the difference of their positions by some lower bound $|u_{1a}-u_{2a}|+|u_{1b}-u_{2b}| \geq l \in \mathbb{R}$, where a and b respectively denote the first and second components of vectors in $\mathbb{R}^2$. Using the triangle inequality, we can relax this constraint as $|u_{1a} - u_{2a} + u_{1b} - u_{2b}| \geq l$, which is a non-convex constraint.

To resolve the non-convexity, we follow the method outlined in [34] to create a Mixed Integer Program (MIP). In our framework the introduction of the binary independent variable (denoted $\delta \in \{0,1\}$) in the MIP is resolved by introducing a third player whose goal is enforcing the other players' coupling constraints. Thus we have a 3-player game with the matrices from (1) defined as

$$A_1 = A_2 := \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \; A_3 := \begin{bmatrix} M \\ -M \\ 1 \\ -1 \end{bmatrix},$$
$$b_1 = b_2 := \frac{1}{2}\begin{bmatrix} l & l-M & 0 & -1 \end{bmatrix}^\top, \; b_3 := \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^\top, \tag{25}$$

where $M \in \mathbb{R}$ is a constant significantly larger than l. Each robot $i \in \{1,2\}$ then aims to solve

$$\begin{aligned} \min_{u_i \in \mathbb{R}^2} \; & \frac{1}{2}\|u_i - \bar{r}_i\|^2 - \log(d - y_i) \\ \text{s.t. } & y_i = \|u_1 - u_2\|^2 + w_i \\ & u_i \in \mathcal{U}_i(u_{-i}, \delta), \end{aligned} \tag{26}$$

where u_{-i} denotes the other robot's action (from robot i's perspective), δ denotes the coordination variable, and $\mathcal{U}_i(u_{-i})$ is as defined under (1). The aforementioned third player [2] attempts to solve the following feasibility problem:

$$\min_{\delta \in [0,1]} \; 0 \quad \text{s.t. } \delta \in \Delta(\mathbf{u}), \tag{27}$$

where $\Delta(\mathbf{u}) = \{\delta \in \mathbb{R} : (\mathbf{u}, \delta) \in \mathcal{U}\}$. The sets $\mathcal{U}$ and Δ are defined as per (1) with the matrices A_i and b_i chosen in (25). Note that $\delta \in [0,1]$ is a relaxation of the binary variable $\delta \in \{0,1\}$ introduced earlier.

We run two different simulations to validate Algorithm 1:

1) the algorithm in [8] with a nominal Jacobian, calculated by sampling the precise Jacobian along a straight-line path between the starting points and the targets.
2) Algorithm 1, the model-free Jacobian approach.

In Figure 3, the golden 'O' denotes the starting point for the robots, and the green 'X' denotes the target position. The golden 'X' denotes the final position they did converge to, and the blue trajectories are the paths they took to get there. The red circle denotes the final constraint boundary. The dotted purple line connecting the two green 'X' targets is the optimality line: you would intuitively expect the optimum to be when both robots are at the intersection of the optimality line and the final constraint boundary. The figure has two images, corresponding respectively from left to right to the two listed above: nominal Jacobian and model-free Jacobian.

[2]The addition of a third player means this game no longer satisfies Assumption 5. The assumptions we outlined are sufficient, rather than necessary. We believe this example is still worth including as it demonstrates the convergence of the algorithm via an intuitive, easily-visualized application.

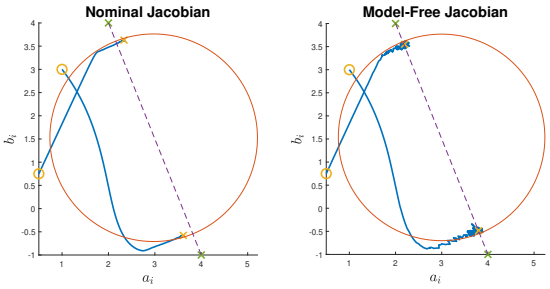

Fig. 3. Visual of the model-free algorithm's performance compared to the algorithm from [8].

Note the performance of the nominal approach compared to the model-free approach in Figure 3. The nominal Jacobian made the assumption of a relatively straight-line path between the robots' starting positions and targets, and this causes inaccuracy and a large deviation from the optimum near the end of the trajectory. The model-free approach converges to a set around the true optimum, since it made no assumptions about the trajectory the robots would take.

V. Conclusion and Future Work

We have presented a novel algorithm for online generalized Nash equilibrium seeking, which combines ideas from game theory, feedback-based optimization, and zero-order optimization. Convergence was established by casting the algorithm as a forward-backward operator-splitting iteration that tracks the algorithm presented in [8], assuming the latter computes the Jacobian perfectly rather than nominally. We presented simulation results for a power distribution feeder example and a simple example motivated by swarm robotics.

A key direction for future work would be a more detailed study of the perturbation signal. We utilize Assumption 3 as a simplification that enables us to minimize the error between rows of the estimated and true Jacobians in a convenient fashion, but there is scope for a better tuned choice that leads to less noisy convergence. In particular, we qualitatively assessed the noise and error while tuning the frequencies for our signal choices. An algorithmic technique to choose the signal is thus an important area of future work.

References

[1] J. S. Shamma, "Game theory, learning, and control systems," *National Science Review*, vol. 7, no. 7, pp. 1118–1119, Nov. 2019.

[2] J. R. Marden and J. S. Shamma, "Chapter 16 - game theory and distributed control," in ser. Handbook of Game Theory with Economic Applications, H. P. Young and S. Zamir, Eds., vol. 4, Elsevier, 2015, pp. 861–899.

[3] G. Belgioioso, D. Liao-McPherson, M. H. de Badyn, S. Bolognani, J. Lygeros, and F. Dörfler, *Sampled-data online feedback equilibrium seeking: Stability and tracking*, 2021.

[4] A. R. Romano and L. Pavel, "Dynamic NE seeking for multi-integrator networked agents with disturbance rejection," *IEEE Trans. Control Net. Syst.*, vol. 7, no. 1, pp. 129–139, Mar. 2020.

[5] S. Givigi and H. Schwartz, "A game theoretic approach to swarm robotics," *Appl. Bionics and Biomechanics*, vol. 3, 2006.

[6] Y. Zhang and M. Guizani. CRC Press, 2019.

[7] A. B. MacKenzie and L. A. DaSilva, *Synthesis Lectures on Communications*. Springer, 2006.

[8] A. Agarwal, J. W. Simpson-Porco, and L. Pavel, "Game-theoretic feedback-based optimization," *IFAC-PapersOnLine*, vol. 55, no. 13, pp. 174–179, 2022, IFAC NecSys Workshop.

[9] E. Ismagilova, L. Hughes, N. Rana, and Y. Dwivedi, "Security, privacy and risks within smart cities: Literature review and development of a smart city interaction framework," *Information Sys. Frontiers*, Jul. 2020.

[10] A. Bernstein, E. Dall'Anese, and A. Simonetto, "Online primal-dual methods with measurement feedback for time-varying convex optimization," *IEEE Trans. Signal Proc.*, vol. 67, no. 8, pp. 1978–1991, Apr. 2019.

[11] M. Colombino, J. W. Simpson-Porco, and A. Bernstein, "Towards robustness guarantees for feedback-based optimization," in *Proc. IEEE CDC*, 2019, pp. 6207–6214.

[12] P. Yi and L. Pavel, "An operator splitting approach for distributed generalized Nash equilibria computation," *Automatica*, vol. 102, pp. 111–121, 2019.

[13] D. Gadjov and L. Pavel, "A passivity-based approach to Nash equilibrium seeking over networks," *IEEE Trans. Autom. Control*, vol. 64, no. 3, pp. 1077–1092, 2019.

[14] A. Hauswirth, S. Bolognani, G. Hug, and F. Dörfler, "Optimization algorithms as robust feedback controllers," Unpublished, 2021. arXiv: 2103.11329 [math.OC].

[15] M. Colombino, E. Dall'Anese, and A. Bernstein, "Online optimization as a feedback controller: Stability and tracking," *IEEE Trans. Control Net. Syst.*, vol. 7, no. 1, pp. 422–432, 2020.

[16] L. S. P. Lawrence, J. W. Simpson-Porco, and E. Mallada, "Linear-convex optimal steady-state control," *IEEE Trans. Autom. Control*, vol. 66, no. 11, pp. 5377–5384, 2021.

[17] E. Dall'Anese and A. Simonetto, "Optimal power flow pursuit," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 942–952, 2018.

[18] A. Hauswirth, S. Bolognani, G. Hug, and F. Dörfler, "Projected gradient descent on riemannian manifolds with applications to online power system optimization," in *Allerton Conf on Comm, Ctrl & Comp*, 2016, pp. 225–232.

[19] Y. Chen, A. Bernstein, A. Devraj, and S. Meyn, *Model-free primal-dual methods for network optimization with application to real-time optimal power flow*, 2019.

[20] D. Shirodkar and S. P. Meyn, "Quasi stochastic approximation," *Proceedings of the 2011 American Control Conference*, pp. 2429–2435, 2011.

[21] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *SIAM Journal on Optimization*, vol. 23, no. 3, pp. 462–466, 1952.

[22] J. C. Spall, "A one-measurement form of simultaneous perturbation stochastic approximation," *Automatica*, vol. 33, no. 1, pp. 109–112, 1997, ISSN: 0005-1098.

[23] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *taes*, vol. 34, no. 3, pp. 817–823, 1998.

[24] S. Bhatnagar, M. C. Fu, S. I. Marcus, and I.-J. Wang, "Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences," vol. 13, no. 2, pp. 180–209, Apr. 2003, ISSN: 1049-3301. DOI: 10.1145/858481.858486. [Online]. Available: https://doi.org/10.1145/858481.858486.

[25] P. L. A, S. Bhatnagar, N. Bhavsar, M. Fu, and S. I. Marcus, *Random directions stochastic approximation with deterministic perturbations*, 2018. DOI: 10.48550/ARXIV.1808.02871.

[26] Z. He, S. Bolognani, J. He, F. Dörfler, and X. Guan, *Model-free nonlinear feedback optimization*, 2022. DOI: 10.48550/ARXIV.2201.02395.

[27] D. Hajinezhad, M. Hong, and A. Garcia, *Zeroth order nonconvex multi-agent optimization over networks*, 2017. DOI: 10.48550/ARXIV.1710.09997.

[28] J. N. Webb, "Game theory, Decisions, interaction and evolution," in New York, USA: Springer, 2007, ch. 4, sec. 1, p. 62.

[29] F. Facchinei and J.-S. Pang, *Finite-Dimensional Variational Inequalities and Complementary Problems*. Springer Science & Business Media, 2007.

[30] H. Yin, U. V. Shanbhag, and P. G. Mehta, "Nash equilibrium problems with scaled congestion costs and shared constraints," *IEEE Trans. Autom. Control*, vol. 56, no. 7, pp. 1702–1708, 2011.

[31] F. Bullo, *Lectures on Network Systems*, 1.6. Kindle Direct Publishing, 2022.

[32] A. Agarwal, "Robust feedback-based nash equilibrium seeking," M.S. thesis, University of Toronto, 27 King's College Cir, Toronto, Canada, 2022.

[33] H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011, p. 468.

[34] M. Chan, Y. Yin, B. Amado, P. Williams, and D. Xiao, *Optimization with absolute values*, 2020.