Robust Feedback-Based Nash Equilibrium Seeking

by

Anurag Agarwal

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

Department of The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
University of Toronto

Robust Feedback-Based Nash Equilibrium Seeking

Anurag Agarwal
Master of Applied Science
Department of The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
University of Toronto
2022

Abstract

We consider the problem of distributed Nash equilibrium seeking over networks. Each agent desires to play a game while constraining a system output generated by a process affected by the agents' control actions as well as some unknown, exogenous disturbances. Each self-interested agent seeks to minimize their cost function within this game, while ensuring that the output obeys the given constraints. We outline the limitations of forecasting the output and the relevant external disturbances, which motivates our feedback-based approach, using measurements to lift the forecasting requirements. We then develop two algorithms: one that makes a nominal, constant approximation of the system's input-output sensitivities a priori, and another that estimates sensitivities in a model-free manner. We exploit operator-theoretic properties to show our algorithms' convergence robust to model uncertainty and unknown disturbances. We then test the developed techniques using academic and real-world examples, validating and outlining the convergence properties of our algorithm.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Game theory studies the strategic interaction between multiple, self-interested agents, each optimizing a cost that is coupled with each other agent's actions. The study of games revolves around seeking a Nash equilibrium, where each agent has chosen an action profile from which no single agent can unilaterally deviate for a better cost or payoff. Consider, for example, the simple case of traffic flow through a city. Each driver would like to minimize the time it takes to reach a destination, but each driver's time taken would depend on each other driver's path too. Thus a Nash equilibrium (NE) would be when each driver chooses a path such that, given each other drivers' paths, no driver could deviate from their path and reach their destination faster.

This problem formulation, illustrated by the simple cases of traffic flow across a city, can be extended to large classes of problems where the decision-makers (henceforth referred to as agents/players) have access to sensors, information, and control inputs such that they can be set up in a distributed manner, e.g. power systems, networks, smart cities, and so on. Such problems however, often involve complex coupling and interaction between players' actions, exogenous disturbances, and their impact on system constraints and/or safety requirements. For example, consider an extension of the aforementioned traffic flow problem: each agent in the network is a self-driving car, each capable of communicating with one another, trying to minimize their travel time to their destination. The travel time is affected by other agents' path selection, hence a game-theoretic framework is applicable. It is *also* affected by weather, traffic from cars not connected to the network, accidents, traffic signals at intersections, construction, and so on. To account for those factors, we want to develop our optimization techniques to be robust to various externalities, accounting for them without having a perfect forecast for their behaviour.

Thus this thesis studies problems at intersection of game theory and robust control, and leverages both to implement a distributed optimization framework. With game theory, our framework enables agents to learn in an environment that is altered by other agents' learning and responses. With robust control, our framework enables agents to learn in an environment while accounting for factors that they do not have a perfect model for.

1.1 Motivation

Classical methods of solving optimization are often *offline*, referring to the significant amount of a priori knowledge and modelling effort needed to correctly represent the behaviour of the relevant costs, and any other factors that affect them. These methods are not well-suited for solving problems pertaining to large, complex, and uncertain systems, as they impose stringent, often infeasible, information requirements, and demand accurate forecasting and modelling of the system. Recent development in the area of centralized, online optimization algorithms [1–4] aims to address that gap by providing a framework for feedback-based optimization, lowering these informational requirements. This research has applications in the areas of communication networks [5], power systems [6–9], and transportation [10]. The advantage of using online optimization techniques over offline ones is the same as that of using feedback over feedforward control: feedback-based optimization techniques tend to be more robust to model uncertainty, and are better able to attenuate or reject exogenous disturbances [1, 3].

The applications discussed above often involve the regulation of outputs that are affected by the actions of several different agents within the same system (e.g. multiple loads in a power distribution system), and thus a centralized, feedback-based optimization algorithm adds communication overhead. A distributed method that enables agents to solve these problems locally, with reduced communication overhead, would allow the development of efficient and scalable optimization techniques, relevant to larger classes of complex problems. We aim to utilize the non-cooperative nature of game theory to develop these distributed optimization techniques. The key difference between a game-theoretic setup (as opposed to a classical distributed reinforcement learning setup) is the dependence of agents' costs on each other agents' actions. This creates a learning environment that evolves as the agents learn, where each self-interested agent is now interested in finding the best possible outcome while being aware that each other agent aims to do the same, to which they respond accordingly [11]. This has led to recent interest in studying the application of game-theoretic techniques to distributed control problems [12] with applications such as swarm robotics [13–15], wireless communication [16, 17], etc.

We propose a novel model to combine game theoretic NE-seeking techniques with robust feedback-based optimization. We consider the case where each player in a game is concerned with an input-to-output mapping (for example, the equilibrium mapping of a dynamical system) affected by unknown external disturbances. Each player seeks to minimize a cost function that represents its goal within the game, while ensuring that the output obeys certain constraints (such as safety constraints in a power system). This thesis' goal is to develop an algorithm to perform this minimization in a distributed, game-theoretic manner, robust to model uncertainties and exogenous disturbances. We then further restrict the information available to each agent, and study the application of model-free techniques to game-theoretic problems.

1.2 Literature Review

Our work primarily builds off of the robust, feedback-based optimization framework in [1]. We extend recent work done in the area of similar online optimization methods [1–4]. As mentioned before, the main benefit of these methods over classical optimization methods is their robustness to external

factors and model uncertainty [3], and they can be applied to a wide variety of engineering problems [5, 6, 10]. Research in this area primarily considers centralized, cooperative decision-making, but often studies systems with distributed learning dynamics (such as the bulk power grid) [1, 18]. In [7], although the decision-making is done in a distributed optimization framework, the focus is on cooperative decision-making. This thesis expands on prior work in this area by leveraging the non-cooperative decision-making that game theory enables, which leads into our novel distributed optimization framework.

Our work is thus also related to the literature on generalized Nash equilibrium (GNE) seeking techniques. Prior works primarily consider games where the cost is impacted solely by the players' actions and their coupling constraints [19–21]. In [13], the effects of cost functions dictated by outputs from a dynamical system are considered, but this is done without any coupling constraints and with the assumption that the agents' equilibrium output mappings are decoupled. Our work differs from [13] in that we allow agents to have inter-dependent output mappings. We also introduce coupling constraints and consensus dynamics, building off the operator-theoretic framework developed in [19].

Finally, our work relates to recent research in model-free optimization. Model-free (or zero-order) techniques have been an ongoing area of research in optimization, control, and machine learning. This thesis primarily builds off the methods developed in [18]. Recent work in this area includes [22–24]. The techniques developed in these works involve the use of stochastic exploration signals, generated from independent, identically distributed random variables, to use two function evaluations per iteration to predict the gradient(s) of the cost function(s). These stochastic models are inspired by the Simultaneous Perturbation Stochastic Approximation technique developed in [25, 26]. Model-free techniques are particularly helpful when agents are optimizing while account for unknown (or difficult to model) factors that affect their costs (and thus their learning). Recent work in the area uses quasi-stochastic exploration signals, rather than stochastic ones, to reduce noise from the perturbations [27]. Similar to [18], we develop a constant step-size method for time-varying optimization problems. Our main extension to this framework is that we use a two-function evaluation to estimate the sensitivity of the output mapping to the control inputs, rather than the gradient of the overall cost function. We note that this isolates the uncertainty of the output's sensitivity, allowing quasi-stochastic, model-free optimization to be utilized within a game-theoretic framework with milder assumptions.

1.3 Contributions

Our problem formulation can be viewed as an extension of robust control and feedback-optimization techniques to non-cooperative, multi-agent decision-making problems. It can also be viewed as extending the techniques of solving N-player games with affine constraints to games played in systems where each agent is concerned with constraining a measured output generated by an unmodelled process, affected by unknown exogenous disturbances. Thus our overarching contribution is a unified framework for feedback-based, robust, game-theoretic optimization. The following provides an outline of our specific contributions:

- We show that under assumptions of strong monotonicity and Lipschitz continuity on the games' "pseudogradient" (a generalization of the gradient which we formalize later on), we can use forward-backward operator splitting techniques to find an approximate GNE which is

a bounded distance away from a true KKT point of the game [19, 28]. We discuss how, similar to [1], a KKT point is a candidate for a local/global optimum if the system admits one. We also discuss what further assumptions can be made to ensure that the game admits a GNE, which would then correspond to the KKT point found by our algorithm. Our analysis is a unification of [1] and [19], by combining the former's online, feedback-based framework and approximate gradient dynamics with the latter's non-cooperative, game-theoretic framework.

- We outline how the above assumptions on the pseudogradient can be hard to verify for an unknown model. We thus leverage a linear fractional transformation (LFT) to parametrize the model [29]. This enables the use of a semidefinite programming approach to numerically verify the criteria outlined above. We provide a specific example of a common class of cost functions, and outline how to use the above matrix inequalities.

- We illustrate the advantages of our framework over offline methods in our simulation results. We show that, even for a simple, academic example, the convergence of simplistic forecasting methods cannot be guaranteed. We compare the online methods to various perfect forecasts and note the numerical verification of our theoretical convergence bounds.

- Since the focus is on developing an information-light framework to robustly control uncertain systems in a distributed manner, our work naturally extends to further relaxing assumptions on the information available to agents. We discuss and explore the possibility of model-free optimization being merged with this framework. We develop a model-free technique, and utilize similar operator-theoretic techniques to prove its convergence to a limit set around a vKKT point of the system.

- We discuss the advantages and disadvantages of the two frameworks developed within this work. We illustrate, with examples, how some systems favour one framework over another, and note that both outperform simplistic forecast based methods.

1.4 Organization

This thesis is organized as follows:

- **Chapter 2:** Background
  Relevant notations and definitions from control theory, graph theory, operator theory are presented. Mathematical preliminaries about sets, functions, vectors, and matrices are outlined. The linear fraction transformation is introduced, and simple examples illustrating its use are provided.

- **Chapter 3:** Generalized Nash Equilibrium Problems
  An outline of prior work in game theory is provided, as a point to build off of for this thesis' focus. We define a relevant solution concept for a game, the generalized Nash equilibrium (GNE). We outline the GNE-seeking algorithm that our thesis seeks to extend. Note that [19] solves the problem without the output mapping, and our later problem formulation is thus a generalization of this framework. We provide intuition for why this algorithm converges, using concepts from operator theory.

- **Chapter 4:** Games Played with Output Mappings
  We introduce the main problem that this thesis explores. We motivate our goal of unifying game-theoretic concepts with the framework from [1] by outlining an offline method, and discussing its limitations. We discuss the difficulty of solving the problem in an online manner, due to the unknown nature of the output model and any disturbances that affect it, leading to a nominal approximation of the Jacobian of the output mapping. We develop a new solution concept for our problem, the *online approximate variational GNE* (OA vGNE) and discuss its relevance as an approximation of one of the system's optimal points (or candidates for such a point), chosen to penalize all players "fairly". We show the existence and uniqueness of the OA vGNE, and develop an algorithm to find the OA vGNE, extending from the one presented in Chapter 4. We prove the algorithm's convergence as a forward-backward operator splitting technique, and provide error bounds comparing the algorithm's limit point to candidate optima.

- **Chapter 5:** Monotonicity and Lipschitz Continuity of Uncertain Operators
  The formulation in the prior chapter considers outputs generated by an uncertain system, affected by unknown externalities. This uncertainty makes verifying the criteria for the algorithm's convergence inherently harder. We propose a method to represent the output mapping as a set-valued, uncertain operator and use that to parametrize its Jacobian as a linear fractional transformation (LFT). We combine the LFT parametrization with semidefinite programming methods to outline conditions on the Jacobian that are sufficient to verify the convergence criteria presented in Chapter 4.

- **Chapter 6:** Jacobian-Free Algorithm
  In this chapter, we start with a motivating example from swarm robotics that outlines how a nominal Jacobian (as used in Chapter 4) may not be sufficient for certain classes of problems. In short, we outline how the sensors available for output measurement could cause the "true" Jacobian (one computed with a perfect model) to be heavily perturbed over time. In this case, the error between an OA vGNE and a true KKT point could be large enough to the point of being a bad candidate for an optimal operating point. We then outline a model-free framework that estimates the Jacobian during runtime, and show that a quasi-stochastic perturbation is sufficient for the algorithm to converge to a limiting point. We outline the convergence of this algorithm to a set within finite bounds of a true vKKT point of the problem.

- **Chapter 7:** Simulation and Results
  Simulation results are provided for various problems. We start with an academic problem that illustrates the advantages of online frameworks over offline ones. We then outline a swarm robotics example with a discussion on various cases, some of which perform better when optimized with the OA vGNE algorithm from Chapter 4, and others which perform better with the Jacobian-free algorithm from Chapter 6. Finally we use data gathered from real-time operation of a distribution feeder, and show robust convergence of the OA vGNE algorithm over the course of a long runtime with varying disturbances.

- **Chapter 8:** Conclusion and Future Work
  We conclude the thesis with a summary of the work done, concluding remarks, and directions for future research.

Chapter 2

Background

In this section, we introduce the notation and mathematical background required to develop the theoretical foundation of this work. Section 2.1 introduces basic mathematical notation. Section 2.2 introduces projections, convex sets, and cones. Section 2.3 outlines the preliminaries of functions, continuity, and relevant properties of functions that we utilize to prove convergence. Section 2.4 outlines similarly useful properties of operators, a generalization of functions. Section 2.5 discusses graphs and their algebraic representation, which are necessary to codify the communication between players within a game-theoretic setting. Finally in Section 2.6, we discuss a framework from robust control theory that allows the parametrization of nonlinearities and uncertainties in terms of a linear system with an uncertain feedback path. We utilize this parametrization to isolate the unknown elements of the problem classes that we study in Chapters 4 and 5, to develop tractable methods to verify if these problems are solvable.

2.1 Mathematical Notations and Linear Algebra

In this section we define vectors, matrices, and preliminary concepts from linear algebra [30]. We also introduce some notation that we use throughout this work.

The sets $\mathbb{R}, \mathbb{R}_+$ denote the sets of real numbers and nonnegative real numbers respectively. The sets $\mathbb{R}^n, \mathbb{R}^n_+$ denote n-dimensional Euclidean space and the corresponding nonnegative space respectively. For a vector $x \in \mathbb{R}^n$ (matrix $A \in \mathbb{R}^{n\times m}$), $x^\top$ ($A^\top$) denotes its transpose. Given a symmetric positive definite matrix $P \succ 0$, the operator $\langle \cdot, \cdot \rangle_P : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ 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 $A = \mathrm{diag}(A_1, \ldots, A_N)$. Denote $\mathrm{col}(x_1, \ldots, x_N)$ 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 $\mathbf{1}_n = \mathrm{col}(1, \ \ldots \ , 1) \in \mathbb{R}^n$, the n-dimension vector of zeroes as $\mathbf{0}_n = \mathrm{col}(0, \ \ldots \ , 0) \in \mathbb{R}^n$ and the n-dimensional identity matrix as $I_n = \mathrm{diag}(\mathbf{1}_n) \in \mathbb{R}^{n\times m}$. We denote the zero matrix $\mathbb{0}_{n\times m} \in \mathbb{R}^{n\times m}$, with each element equal to zero. We sometimes omit the dimensional subscript for these vectors and matrices.

Lemma 2.1 (Cauchy-Schwarz Inequality, [31]). *Given any two vectors $a, b \in \mathbb{R}^n$ and some $P \succ 0$:*

$$|\langle a, b\rangle_P| \leq \|a\|_P \|b\|_P.$$

2.2 Sets and Set Projection

In this section we define sets, cones, and projections onto a set. The following are from [32–34]. We deal with sets when talking about the players' action sets in Chapters 3, 4.1, and 6, and the structure of these sets and their projections is often a consideration for showing their convergence. We also use these definitions throughout this work as foundations for other, more complex concepts.

Given two sets Ω and Φ, we denote the *union* of the sets as $\Omega \cup \Phi$ and we denote their *intersection* as $\Omega \cap \Phi$. If $\Phi \subset \Omega$, we denote $\Omega \backslash \Phi = \{x : x \in \Omega, x \notin \Phi\}$ as the set of all elements in Ω except for those also in Φ. We denote the empty set as $\emptyset = \{\}$.

An *open ball* of radius δ centred at point $p \in \mathbb{R}^n$ is defined as $\mathcal{B}_\delta^n(p) = \{x \in \mathbb{R}^n : \|x - p\| < \delta\}$. A point $p \in \Omega \subset \mathbb{R}^n$ is a limit point of the set Ω if for all $\delta > 0$ there exists $x \in \mathcal{B}_\delta^n(p)$ such that $x \neq p$, $x \in \Omega$. The set Ω is *closed* if it contains all its limit points. The set Ω is *bounded* in $\mathbb{R}^n$ if there exists $p \in \mathbb{R}^n$ and $\delta > 0$ such that $\Omega \subset \mathcal{B}_\delta^n(p)$. A set is *compact* if and only if it is closed and bounded. The set Ω is *convex* if for all $\theta \in [0, 1]$ and for all $x, y \in \Omega$, $\theta x + (1-\theta)y \in \Omega$. The *interior* of the set Ω is defined as $\operatorname{int}\Omega = \{x \in \Omega : \exists \delta > 0,\ \mathcal{B}_\delta^n(x) \subset \Omega\}$.

Definition 2.1 (Cones). *Given a set $\Omega \subset \mathbb{R}^n$,*

- *the set Ω is a cone if for any $x \in \Omega$, $\gamma x \in \Omega$ for all $\gamma > 0$, and*
- *the normal cone of Ω at a point $x \in \mathbb{R}^n$ is defined as*

$$N_\Omega(x) = \begin{cases} \{y \in \mathbb{R}^n | y^\top (x' - x) \leq 0\ \forall x' \in \Omega\},\ x \in \Omega \\ \emptyset,\ x \notin \Omega \end{cases}.$$

Definition 2.2 (Projection). *We define the projection of a point $x \in \mathbb{R}^n$ to the closed, convex set $\Omega \subset \mathbb{R}^n$ as $P_\Omega(x) = \arg\min_{x' \in \Omega} \|x - x'\|$.*

It should be noted that, by Moreau's Decomposition Theorem [34], $x = P_\Omega(x) + v$, for some $v \in N_\Omega(x)$. Later in this work, we develop optimum-seeking algorithms for which each iteration is expressed in an operator-theoretic framework by using this decomposition.

Definition 2.3 (Variational Inequality (VI) Def. 1.1.1. [35]). *Given a closed, convex set $\Omega \subset \mathbb{R}^n$ and a function $F : \Omega \to \mathbb{R}^n$, the variational inequality, denoted $\mathrm{VI}(\Omega, F)$, w.r.t. $\langle \cdot, \cdot \rangle_P$ is to find a vector $x \in \Omega$ such that*

$$\langle F(x), x' - x\rangle_P \geq 0,\ \forall x' \in \Omega.$$

Notice that by rearranging the above equation, we obtain $\langle -F(x), x' - x\rangle \leq 0$. Thus a variational inequality is solved by a vector $x \in \Omega$ such that $-F(x)$ is in the normal cone of Ω at x. We use this concept in Chapter 4 to characterize the error bounds for the solution of our problem.

Definition 2.4 (Power Set [36]). *Given a set $\Omega \subseteq \mathbb{R}^n$, its power set, denoted 2^Ω, is the set of all subsets of Ω, including the empty set $\emptyset$ and the set Ω itself.*

The notion of a power set is utilized in the definition of set-valued operators later in this work.

2.3 Functions and Continuity

In this section we review notions of continuity, Lipschitz continuity, and monotonicity. The following are from [28, 32, 37].

Definition 2.5 (Continuity, [32]). *Consider a function $F : \mathcal{S} \to \mathbb{R}^m$, where $\mathcal{S} \subseteq \mathbb{R}^n$. The function F is*

1. *continuous at $x_0 \in \mathcal{S}$ if for all $\delta > 0$ there exists $\epsilon > 0$ such that $\forall x \in \mathcal{S},\ x \in \mathcal{B}^n_\delta(x_0) \implies F(x) \in \mathcal{B}^m_\epsilon(F(x_0))$,*
2. *continuous if it is continuous for all points in $\mathcal{S}$,*
3. *Lipschitz continuous at $x_0 \in \mathcal{S}$, w.r.t. $\langle\cdot,\cdot\rangle_P$, if there exist $\delta, L > 0$ such that $\forall x, y \in \mathcal{B}^n_\delta(x_0),\ \|F(x) - F(y)\|_P \leq L\|x - y\|_P$,*
4. *locally Lipschitz on $\mathcal{S}$ if it is Lipschitz for all points in $\mathcal{S}$,*
5. *Lipschitz on $\mathcal{D} \subseteq \mathcal{S}$, w.r.t. $\langle\cdot,\cdot\rangle_P$, if there exists $L > 0$ such that $\forall x, y \in \mathcal{D},\ \|F(x) - F(y)\|_P \leq L\|x - y\|_P$, and*
6. *globally Lipschitz if $\mathcal{D} = \mathcal{S}$.*

Continuity and Lipschitz continuity are standard assumptions made for analyzing optimization problems.

Definition 2.6 (Cocoercivity, Def. 4.4 [28]). *Let $F : \Omega \subset \mathbb{R}^n \to \mathbb{R}^n$. The function F is μ-cocoercive, $\mu > 0$, w.r.t. $\langle\cdot,\cdot\rangle_P$ if*

$$\langle x - y, F(x) - F(y)\rangle_P \geq \mu\|F(x) - F(y)\|^2_P \ \forall x, y \in \Omega.$$

Definition 2.7 (Monotone Functions, [28]). *A function $F : \mathcal{S} \to \mathbb{R}^n$ where $\mathcal{S} \subseteq \mathbb{R}^n$ is said to be*

1. *monotone w.r.t. $\langle\cdot,\cdot\rangle_P$ if $\langle x - y, F(x) - F(y)\rangle_P \geq 0$, for all $x, y \in \mathcal{S}$, and*
2. *ρ-strongly monotone w.r.t. $\langle\cdot,\cdot\rangle_P$ if there exists $\rho > 0$ such that $\langle x - y, F(x) - F(y)\rangle_P \geq \rho\|x - y\|^2_P$, for all $x, y \in \mathcal{S}$.*

Definition 2.8 (Convex Functions, [28]). *A function $F : \mathcal{S} \to \mathbb{R}$ where $\mathcal{S} \subseteq \mathbb{R}^n$ is said to be*

1. *convex if $F(\alpha x + (1-\alpha)y) \leq \alpha F(x) + (1-\alpha)F(y)$, for all $x, y \in \mathcal{S}$, and for all $\alpha \in (0, 1)$, and*
2. *ρ-strongly convex if there exists $\rho > 0$ such that $F(\alpha x + (1-\alpha)y) \leq \alpha F(x) + (1-\alpha)F(y) - \frac{\rho}{2}\alpha(1-\alpha)\|x - y\|^2$, for all $x, y \in \mathcal{S}$, and for all $\alpha \in (0, 1)$.*

We use the notions of monotonicity and convexity to show that a game admits a solution. Additionally, strong monotonicity is an important property in proving the convergence of many game-theoretic algorithms, and we use this in Chapter 4.

Definition 2.9 (Differentiable Functions, [28]). *Consider a function $f : \mathcal{S} \to \mathcal{V}$ where $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{V} \subseteq \mathbb{R}^m$. The function f is said to be differentiable at $x \in \mathcal{S}$ if there exists a map $Df : \mathbb{R}^n \to \mathbb{R}^m$ (the Fréchet derivative) of f at x such that*

$$\lim_{\|y\| \to 0} \frac{\|f(x+y) - f(x) - Df(x)y\|}{\|y\|} = 0.$$

If f is differentiable for all $x \in \mathcal{S}$ then f is said to be differentiable. If the derivative of f is continuous, then f is said to be continuously differentiable, denoted $\mathbb{C}^1$. If the first n derivatives of f are continuous, it is said to be $\mathbb{C}^n$.

Let $f : \mathcal{S} \to \mathcal{V}$ be a differentiable function where $\mathcal{S} \in \mathbb{R}^n$ and $\mathcal{V} \in \mathbb{R}^m$. Denote the i-th component of the the function f as f_i, and denote the natural basis of $\mathbb{R}^n$ as $\{e_1, \ \dots \ , e_n\}$. Then, for any $i \in \{1, \ \dots \ , m\}$ and $j \in \{1, \ \dots \ , n\}$ we define

$$\frac{\partial f_i(x)}{\partial x_j} = \lim_{t \to 0} \frac{f_i(x + te_i) - f_i(x)}{t}.$$

If the limit exists, this function is termed the j-th partial derivative of f_i at x [32]. In finite-dimensional spaces, the Fréchet derivative can be represented in coordinate form with the partial derivatives.

Definition 2.10 (Jacobian Matrix). *Given a vector-valued function $f : \mathcal{S} \to \mathbb{R}^m$, $\mathbb{C}^1$ on its domain $\mathcal{S} \subseteq \mathbb{R}^n$, its Jacobian matrix at x is defined as*

$$\partial f(x) := \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(x) & \cdots & \frac{\partial f_m}{\partial x_n}(x) \end{bmatrix}.$$

Each row of the Jacobian matrix can be denoted $\nabla_x f_i^\top(x), \ i \in \{1, \dots, m\}$, termed the gradient of the scalar-valued function $f_i : \mathcal{S} \to \mathbb{R}$ at x.

When expressing the gradient $\nabla_x f(x)$ we sometimes omit the subscript and denote it $\nabla f(x)$. In such instances, we simply mean that the gradient is taken across the entire vector of arguments to the function f. We similarly omit the subscript for $\partial F(x)$ when referencing the Jacobian of F. We next generalize the concept of the Jacobian to functions that are *not* $\mathbb{C}^1$. We use this to analyze uncertain functions where the derivatives cannot be computed precisely.

Definition 2.11 (Clarke Generalized Jacobian, [37]). *Given a closed, convex set $S \subseteq \mathbb{R}^n$, and a map $F : \mathcal{S} \to \mathbb{R}^n$ such that F is locally Lipschitz on $\mathcal{S}$, the Clarke generalized Jacobian of F at $x \in \mathcal{S}$ is defined as the set*

$$\partial_C F(x) = \text{co} \left\{ J \in \mathbb{R}^{n \times n} : J = \lim_{\substack{x_i \to x \\ F(x_i) \text{differentiable}}} \partial_{x_i} F(x_i) \right\},$$

where co *denotes the convex hull operator, which is the minimum convex set that encloses all the points in its argument [38].*

Next, we introduce the relationship between monotonicity and convexity in differentiable functions.

Proposition 2.1. *Let $F : \Omega \to \mathbb{R}$ be $\mathbb{C}^1$ where $\Omega \subseteq \mathbb{R}^n$ is an open, convex set. Then*

1. *F is convex if and only if $\nabla_x F(x)$ is monotone for all $x \in \Omega$ (Prop. 17.10 [28]), and*
2. *F is ρ-strongly convex if and only if $\nabla_x F$ is ρ-strongly monotone for all $x \in \Omega$ (Ex. 22.3 [28]).*

We do not directly use the above relation between monotonicity and convexity in our optimization framework. Rather, our goal is to optimize *games* (a term we will formalize in Chapter 3), where this notion is *not* applicable. Finally, analogous to how the gradient is a generalization of the first derivative to multivariate functions, we introduce a generalization to the notion of the second derivative.

Definition 2.12 (Hessian Matrix)**.** *Given a $\mathbb{C}^2$ function $F : S \to \mathbb{R}$ where $S \subseteq \mathbb{R}^n$, its Hessian is defined as the Jacobian of its gradient, denoted*

$$\nabla^2 F(x) = \partial_x \left(\nabla_x F(x)\right)(x).$$

2.4 Operator Theory

This section outlines the notation and results from monotone operator theory that are used throughout this thesis. We use the material from this section to prove the convergence of distributed GNE-seeking algorithms in Chapters 4 and 6 by representing the game as a sum of operators with some of the outlined properties. For a thorough reading, we refer the reader to [28].

Let $\mathfrak{A} : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ be a set-valued operator. The domain of $\mathfrak{A}$ is denoted $\operatorname{dom}\mathfrak{A} = \{x : \mathfrak{A}x \neq \emptyset\}$. The range of $\mathfrak{A}$ is denoted $\operatorname{ran}\mathfrak{A} = \{y : \exists x \text{ s.t. } y \in \mathfrak{A}x\}$. The graph of $\mathfrak{A}$ is denoted $\operatorname{gra}\mathfrak{A} = \{(x, u) : u \in \mathfrak{A}x\}$, and the inverse of $\mathfrak{A}$ is defined through its graph as $\operatorname{gra}(\mathfrak{A}^{-1}) = \{(u, x) : (x, u) \in \operatorname{gra}\mathfrak{A}\}$. The zero set of $\mathfrak{A}$ (also known as the kernel) is $\operatorname{zer}\mathfrak{A} = \{x : \mathbf{0} \in \mathfrak{A}x\}$. We denote the identity operator as Id where $x = \mathrm{Id}(x),\ \forall x \in \operatorname{dom}\mathrm{Id}$. In addition to the notions of Lipschitz continuity and (strong) monotonicity which we defined for functions in the previous section, a monotone operator $\mathfrak{A}$ is maximally monotone if $\operatorname{gra}\mathfrak{A}$ is not strictly contained is not strictly contained in the graph of any other monotone operator.

Proposition 2.2 (Minty's Theorem, (Theorem 21.1 [28]))**.** *Let $\mathfrak{A} : \Omega \to 2^{\Omega}$ be a monotone operator. Then $\mathfrak{A}$ is maximally monotone if and only if* $\operatorname{ran}(\mathrm{Id} + \mathfrak{A}) = \Omega$.

One particular operator whose maximal monotonicity is relevant to the work within this thesis is the *normal cone operator* N_Ω defined in Definition 2.1.

Lemma 2.2 ([39, 40])**.** *For a closed, convex set $\Omega \in \mathbb{R}^n$, the normal cone operator N_Ω is maximally monotone.*

Definition 2.13 (Resolvent)**.** *Given an operator $\mathfrak{A} : \mathbb{R}^n \to 2^{\mathbb{R}^n}$, the resolvent of $\mathfrak{A}$ is*

$$R_{\mathfrak{A}} = (\mathrm{Id} + \mathfrak{A})^{-1}.$$

Lemma 2.3 (Proposition 23.2, [28]). *Given a maximally monotone operator $\mathfrak{A} : \mathbb{R}^n \to 2^{\mathbb{R}^n}$, the resolvent of $\mathfrak{A}$ is single-valued and $\operatorname{dom} R_{\mathfrak{A}} = \mathbb{R}^n$.*

Definition 2.14 (Nonexpansive Operators). *An operator $T : \Omega \subset \mathbb{R}^m \to \mathbb{R}^m$ is nonexpansive w.r.t. $P \succ 0$, if it is 1-Lipschitz w.r.t. to $\langle \cdot, \cdot \rangle_P$, i.e., $\|T(x) - T(y)\|_P \leq \|x - y\|_P,\ \forall x, y \in \Omega$. Given $\alpha \in (0, 1)$, T is α-averaged if there exists a nonexpansive operator T' such that $T = (1 - \alpha)\operatorname{Id} + \alpha T'$. Denote the class of α-averaged operators as $\mathcal{A}(\alpha)$. If $T \in \mathcal{A}(\frac{1}{2})$, then T is also called firmly nonexpansive.*

Lemma 2.4 (Proposition 4.25, [28]). *Given any operator $T : \mathbb{R}^n \to 2^{\mathbb{R}^n}$, $\alpha \in (0, 1)$, and $P \succ 0$ then $T \in \mathcal{A}(\alpha)$ w.r.t. $\langle \cdot, \cdot \rangle_P$ is equivalent to any of the following statements:*

1. $\|Tx - Ty\|_P^2 \leq \|x - y\|_P^2 - \frac{1-\alpha}{\alpha}\|x - y - (Tx - Ty)\|_P^2,\ \forall x, y \in \Omega$;
2. $\|Tx - Ty\|_P^2 + (1 - 2\alpha)\|x - y\|_P^2 \leq 2(1 - \alpha)\langle x - y, Tx - Ty\rangle_P, \forall x, y \in \Omega$.

Following from Lemma 2.4 (ii), $T \in \mathcal{A}\left(\frac{1}{2}\right)$ if and only if

$$\|Tx - Ty\|_P^2 \leq \langle x - y, Tx - Ty\rangle_P,\ \forall x, y \in \Omega.$$

Given $\beta > 0$, T is called β-cocoercive if $\beta T \in \mathcal{A}\left(\frac{1}{2}\right)$, i.e.

$$\beta\|Tx - Ty\|_P^2 \leq \langle x - y, Tx - Ty\rangle_P,\ \forall x, y \in \Omega. \tag{2.1}$$

Lemma 2.5 (Proposition 23.7, [28]). *If an operator $\mathfrak{A} : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is maximally monotone, then its resolvent $T = (\operatorname{Id} + \mathfrak{A})^{-1}$ is firmly nonexpansive.*

We noted in Section 2.2 that any point can be viewed as the sum of its projection onto a closed, convex set and a vector from that projected point's normal cone. The next lemma follows from that observation.

Lemma 2.6 (Proposition 6.46, [28]). *The projection of a point $x \in \mathbb{R}^n$ onto a set Ω can be expressed in terms of the normal cone operator as*

$$P_\Omega(x) = (\operatorname{Id} + N_\Omega)^{-1}(x).$$

Following from Definition 2.13, the projection operator is the resolvent of the normal cone operator. From 2.3, we know that the projection operator is thus also single-valued. Next, we define the notion of a fixed point. Solutions to the algorithm presented in Chapter 4 are characterized as fixed points of appropriately defined operators.

Definition 2.15. *For a single-valued operator $T : \mathbb{R}^n \to \mathbb{R}^n$, $x \in \mathbb{R}^n$ is a fixed point of T if $Tx = x$.*

The composition of two operators is denoted $\mathfrak{A} \circ \mathfrak{B}$ and is defined via its graph $\operatorname{gra}(A \circ \mathfrak{B}) = \{(x, z) : \exists y \in \operatorname{ran} \mathfrak{B}, (x, y) \in \operatorname{gra} \mathfrak{B}, (y, z) \in \operatorname{gra} \mathfrak{A}\}$. The sum of two operators is denoted $\mathfrak{A} + \mathfrak{B}$ and defined $\operatorname{gra}(\mathfrak{A} + \mathfrak{B}) = \{(x, y + z) : (x, y) \in \operatorname{gra} \mathfrak{A},\ (x, z) \in \operatorname{gra} \mathfrak{B}\}$.

Lemma 2.7 ([41]). *Given two maximally monotone operators $\mathfrak{A} : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ and $\mathfrak{B} : \mathbb{R}^n \to 2^{\mathbb{R}^n}$, their sum $\mathfrak{A} + \mathfrak{B}$ is maximally monotone if $\operatorname{dom} \mathfrak{A} \cap \operatorname{dom} \mathfrak{B} \neq \emptyset$, i.e., the intersection of their domains is not the empty set.*

2.5 Graph Theory

In this section we outline some basic elements from graph theory. We use these concepts to describe the communication of players over a network while playing the game. The material in this section is from [42–44].

A weighted, directed graph is a 3-tuple $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \{w_{ij}\})$, with node (vertex) set $\mathcal{N} = \{1, \ \dots \ , N\}$ (where N is the number of nodes), and edge set $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$. Each edge $e_k \in \mathcal{E}$ is an ordered pair $e_k = (i, j)$ that represents a connection between nodes. For an undirected graph $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$. Each $w_{ij} > 0$ represents a weight for an edge $(i, j) \in \mathcal{E}, \ i \neq j$. In an undirected graph $w_{ij} = w_{ji} \ \forall (i, j) \in \mathcal{E}$. In the game-theoretic setting we use graphs to describe information sharing between player: each player is a node, and communication between players is represented as an edge. Note that in this work, we assume the communication between agents is always two way, thus we are only concerned with undirected graphs.

The neighbour set of node i, defined as $\mathcal{N}_{\mathcal{G}}(i) = \{j : (i, j) \in \mathcal{E}\}$, is the set of all nodes that node i has an edge to. For an undirected graph, the weighted degree of a node i s defined as $d_i = \sum_{j \in \mathcal{N}_{\mathcal{G}}(i)} w_{ij}$.

Definition 2.16 (Adjacency Matrix)**.** *Given a weighted, undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \{w_{ij}\})$, the adjacency matrix $\text{Adj} \in \mathbb{R}^{N \times N}$ is defined as*

$$[\text{Adj}]_{ij} = \begin{cases} w_{ij} & (i, j) \in \mathcal{E} \\ 0 & (i, j) \notin \mathcal{E}. \end{cases}$$

Definition 2.17 (Degree Matrix)**.** *Given a weighted, undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \{w_{ij}\})$, the degree matrix $\text{Deg} \in \mathbb{R}^{N \times N}$ is defined as $\text{Deg} = \text{diag}(d_1, \ \dots \ , d_N)$.*

Together, the adjacency and degree matrices describe the connectivity of the graph $\mathcal{G}$. We define the notion of a Laplacian to express these in a mathematically compact and useful manner.

Definition 2.18 (Laplacian Matrix)**.** *Given a weighted, undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \{w_{ij}\})$, the Laplacian matrix $L \in \mathbb{R}^{N \times N}$ is defined as*

$$[L]_{ij} = \begin{cases} d_i & i = j \\ -w_{ij} & (i, j) \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases}$$

Equivalently, the Laplacian can be written as $L = \text{Deg} - \text{Adj}$.

A path from node i to node j is a sequence of distinct nodes such that consecutive nodes are connected by an edge, i.e., a set of nodes $\{v_1, \ \dots \ , v_l\} \subseteq \mathcal{N}$ connected by edges, i.e.,

$$\{(v_1, v_2), (v_2, v_3), \ \dots \ , (v_{l-1}, v_l) : v_1 = i, \ v_l = j\} \subseteq \mathcal{E}.$$

A graph is *connected* if there exists a path between every pair of nodes. A graph is *complete* if every pair of nodes is connected by an edge.

When $\mathcal{G}$ is undirected and connected, 0 is a simple eigenvalue of L with corresponding eigenvector $\mathbf{1}_N$ (Theorem 2.1 (c), [45]). It follows from this property, and from L being a symmetric matrix for

undirected graphs, that $L\mathbf{1}_N = \mathbf{1}^\top L = 0$.

2.6 Linear Fractional Representation

In this section, we present the Linear Fractional Transformation (LFT), a framework to parameterize nonlinearities and uncertainties. It is often helpful to express uncertainties and nonlinearities as set-valued operators that perturb a linear, deterministic system. Such a framework enables the isolation of these factors, separate from a linear component of the system. The material from this section is from [29, 46, 47].

Consider an input-output representation of a mapping, $\zeta = \mathfrak{M}(\xi)$, where $\mathfrak{M} : \mathbb{R}^n \to \mathbb{R}^l$. Assume that there are factors within the system that make the output uncertain, i.e., the output ζ is not known precisely for any given input ξ. We then aim to parametrize the input-output relation as a set of mappings, rather than a deterministic mapping.

Definition 2.19 (Linear Fractional Transformation)**.** *Given an input-to-output mapping $\zeta = \mathfrak{M}(\xi)$, where $\mathfrak{M} : \mathbb{R}^n \to \mathbb{R}^l$, a linear fractional transformation (LFT) of the mapping is the following relation:*

$$\left[\begin{array}{c} \zeta \\ \hline q \end{array}\right] = \left[\begin{array}{c|c} M & B \\ \hline C & D \end{array}\right] \left[\begin{array}{c} \xi \\ \hline p \end{array}\right] \qquad (2.2)$$
$$p = \Delta q,$$

where $M \in \mathbb{R}^{l\times n}$, $B \in \mathbb{R}^{l\times s}$, $C \in \mathbb{R}^{z\times n}$, and $D \in \mathbb{R}^{z\times s}$ are fixed matrices. The matrix $\Delta \in \mathbf{\Delta} \subset \mathbb{R}^{s\times z}$ is referred to as the parameter block of the system. We assume $I - D\Delta$ is invertible.

Note here that our definition of the linear fractional transformation applies only to *static maps* $\mathfrak{M}$. In general, the operator $\mathfrak{M}$ can represent a dynamical system, and then the matrices in the LFT become *transfer matrices* of a linear, time-invariant system. We do not need the general framework for this thesis, since we only use this to parametrize a static map in Chapter 5.

Figure 2.1 illustrates the LFT representation. Note how the uncertainty is isolated entirely on the feedback path, and $\Delta = \mathbf{0}_{z\times s}$ recovers a deterministic, linear mapping. Thus the parameter block represents the uncertainties and nonlinearities within the mapping $\mathfrak{M}$. We use the LFT parametrization in Chapter 5 to develop a matrix inequality to verify convergence criteria for optimization problems with uncertainties in their cost function gradients.

Given the matrices outlined in Definition 2.19, we can write an input-output representation of the LFT parametrization by using the fundamentals of feedback loop decomposition:

$$\zeta = M\xi + Bp \qquad (2.3)$$
$$q = C\xi + Dp \qquad (2.4)$$
$$p = \Delta q. \qquad (2.5)$$

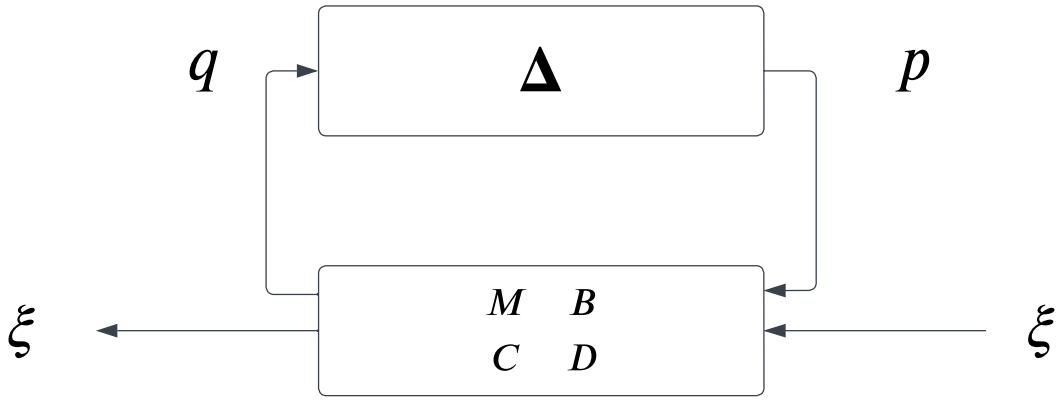

Figure 2.1: A block diagram of the feedback loop of the LTI and the uncertainty in the LFT representation.

By combining the second and third lines of (2.3), we get

$$\begin{aligned} q &= C\xi + D\Delta q \\ (I_z - D\Delta)q &= C\xi \\ q &= (I_z - D\Delta)^{-1}C\xi. \end{aligned}$$

Combining this result with the first and third lines of (2.3), we conclude

$$\zeta = \left[M + B\Delta(I_z - D\Delta)^{-1}\right]\xi. \tag{2.6}$$

Thus the uncertain operator $\mathfrak{M}$ can be represented as an input-output relation between a linear operator defined by matrices $M,\ B,\ C,\ D$, connected in feedback with a nonlinear and/or uncertain operator $\Delta \in \mathbf{\Delta}$. Note that this input-output relation is a *set-valued* operator, parametrized by the set of matrices $\mathbf{\Delta}$. The set-valued nature of this operator will be used in Chapter 5 to develop matrix inequalities for characterizing strong monotonicity and Lipschitz continuity of uncertain operators.

2.6.1 Examples of Linear Fractional Transformation

In this section, we illustrate the LFT parametrization introduced in Definition 2.19 with simple examples, to help build the necessary intuition for the complex use case presented in Chapter 5.

Example 1.

$$\zeta = \begin{bmatrix} -1 & 2\delta_1 \\ 1+\delta_1 & -2 \end{bmatrix}\xi,$$

where $\delta_1 \in [-1, 1]$.

Note that in this example there is only one scalar uncertainty in a mapping $\mathfrak{M} : \mathbb{R}^2 \to \mathbb{R}^2$. We

parametrize the uncertainty as a matrix

$$\Delta = \begin{bmatrix} \delta_1 & 0 \\ 0 & \delta_1 \end{bmatrix}.$$

We can refer to (2.6) and immediately choose

$$M = \begin{bmatrix} -1 & 0 \\ 1 & -2 \end{bmatrix},$$

since it isolates the constant term with no dependence on the uncertainty matrix. For simplicity, we can assume that

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

This simplifies the second term in the equation to

$$B\Delta C = \begin{bmatrix} 0 & 2\delta_1 \\ \delta_1 & 0 \end{bmatrix}.$$

We can achieve this by choosing one of the following two for B and C:

$$B = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix},\ C = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$
$$B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}.$$

With that we have an LFT parametrization for the problem in Example 1. Note that this example illustrated that, depending on the nature of the uncertainties, we have some freedom to choose the matrices within the LFT. A good choice would usually end up with smaller matrices. We refer the reader to Section 6.2 in [29] for a detailed outline on how to iteratively choose a linear fractional transformation for a given matrix.

Example 2. *This example is borrowed from [29].*

$$\zeta = \begin{bmatrix} -1 & 2\delta_1 \\ \frac{-1}{1+\delta_1} & -4+3\delta_2 \end{bmatrix} \xi,$$

Note that we immediately notice that choosing D as a matrix of zeroes is no longer a valid option, since this function is not affine in Δ. We refer to the matrix in this example as $F(\delta)$, where $\delta = (\delta_1, \delta_2)$ for convenience of notation. We note that each element of $F(\delta)$ is a rational function in δ. We start by expressing $F(\delta)$ such that it is decomposed into a polynomial on each side of the

equality:

$$\zeta = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{1+\delta_1} \end{bmatrix} \begin{bmatrix} -1 & 2\delta_1 \\ -1 & (-4+3\delta_2)(1+\delta_1) \end{bmatrix} \xi$$
$$\begin{bmatrix} 1 & 0 \\ 0 & 1+\delta_1 \end{bmatrix} \zeta = \begin{bmatrix} -1 & 2\delta_1 \\ -1 & (-4+3\delta_2)(1+\delta_1) \end{bmatrix} \xi.$$

At this point, we begin isolating the terms from δ iteratively, and then express a relation between p and q that isolates them to the feedback path. Noting that $\zeta = (\zeta_1, \zeta_2)$, on the left side we have

$$\begin{bmatrix} 1 & 0 \\ 0 & 1+\delta_1 \end{bmatrix} \zeta = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \zeta + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \delta_1 \zeta_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \zeta + \begin{bmatrix} 0 \\ 1 \end{bmatrix} p_1,$$

where $p_1 = \delta_1 q_1$ and $q_1 = \zeta_2$. Similarly, with $\xi = (\xi_1, \xi_2)$, the right side can be decomposed as

$$\begin{bmatrix} -1 & 2\delta_1 \\ -1 & (-4+3\delta_2)(1+\delta_1) \end{bmatrix} \xi = \begin{bmatrix} -1 & 0 \\ -1 & -4+3\delta_2 \end{bmatrix} \xi + \begin{bmatrix} 2 \\ -4+3\delta_2 \end{bmatrix} p_2,$$

where $p_2 = \delta_1 q_2$ and $q_2 = \xi_2$. We repeat the process for each of the matrices on the right hand side:

$$\begin{bmatrix} -1 & 0 \\ -1 & -4+3\delta_2 \end{bmatrix} \xi + \begin{bmatrix} 2 \\ -4+3\delta_2 \end{bmatrix} p_2 = \begin{bmatrix} -1 & 0 \\ -1 & -4 \end{bmatrix} \xi + \begin{bmatrix} 2 \\ -4 \end{bmatrix} p_2 + \begin{bmatrix} 0 \\ 3 \end{bmatrix} p_3,$$

where $p_3 = \delta_2 q_3$ and $q_3 = \xi_2 + p_2$. Finally we isolate for ζ again using these decomposed matrices:

$$\begin{aligned} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \zeta + \begin{bmatrix} 0 \\ 1 \end{bmatrix} p_1 &= \begin{bmatrix} -1 & 0 \\ -1 & -4 \end{bmatrix} \xi + \begin{bmatrix} 2 \\ -4 \end{bmatrix} p_2 + \begin{bmatrix} 0 \\ 3 \end{bmatrix} p_3 \\ \zeta &= \begin{bmatrix} -1 & 0 \\ -1 & -4 \end{bmatrix} \xi - \begin{bmatrix} 0 \\ 1 \end{bmatrix} p_1 + \begin{bmatrix} 2 \\ -4 \end{bmatrix} p_2 + \begin{bmatrix} 0 \\ 3 \end{bmatrix} p_3 \\ &= \begin{bmatrix} -1 & 0 \\ -1 & -4 \end{bmatrix} \xi + \begin{bmatrix} 0 & 2 & 0 \\ -1 & -4 & 3 \end{bmatrix} p \\ &:= M\xi + Bp. \end{aligned}$$

We next solve for the C and D by looking at the equations we set out for each element of q above, and obtain

$$\begin{aligned} q &= \begin{bmatrix} \zeta_2 \\ \xi_2 \\ \xi_2 + p_2 \end{bmatrix} \\ &= \begin{bmatrix} -1 & 4 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xi + \begin{bmatrix} -1 & -4 & 3 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} p \\ &:= C\xi + Dp. \end{aligned}$$

Finally we have the relation $p = \Delta q$, with $\Delta = \text{diag}(\delta_1, \delta_1, \delta_2)$. Note here, the repetition of δ_1 along

the diagonal. This is a pattern which can be observed with minimal LFTs of matrices $F(\delta)$ when they are rational function of δ. This notion is formalized in Theorem 6.4 of [29]. This concludes our introduction of LFTs, and with that we have the necessary background required to move on to the problems studied within this thesis.

Chapter 3

Generalized Nash Equilibrium Problems

In this chapter, we define what a game is and describe a relevant solution concept for games, a generalized Nash equilibrium (GNE). Conditions for the existence and uniqueness of GNEs are outlined. Some of the material in this chapter is from [19, 21, 35, 48–51].

3.1 Game Formulation and Nash Equilibria

In this section, we formulate the core game-theoretic problem that this thesis studies, and define the concept of a Nash equilibrium (NE), which we later generalize later as a GNE. We denote a game as $\mathbf{G}(\mathcal{N}, f_i, \Omega_i)$, where $\mathcal{N} = \{1, \ \dots \ , N\}$ is the set of N players (agents) involved. Each player chooses an action $u_i \in \Omega_i$, where $\Omega_i \subseteq \mathbb{R}^{n_i}$ is the action set for the player. Note that each player may have a different dimension n_i for their action set. We can then denote $\Omega = \prod_{i \in \mathcal{N}} \Omega_i$ to be the overall action set for all players in the game, where $\Omega \subseteq \mathbb{R}^n$, $\Omega_i \subseteq \mathbb{R}^{n_i}$, and $n = \sum_{i \in \mathcal{N}} n_i$. The function $f_i : \Omega \to \mathbb{R}$ is a cost function that relates all players' actions to player i's goals in the game.

Let $\mathbf{u} = \text{col}(u_1, \ \dots \ , u_N) \in \Omega$ denote the overall action profile for the players. We often denote $\mathbf{u} = (u_i, \mathbf{u}_{-i})$, where $u_i \in \Omega_i$ is player i's action, and $\mathbf{u}_{-i} \ \in \Omega_{-i}$ is the vector of all other players' actions, where the set $\Omega_{-i} = \prod_{i \in \mathcal{N} \setminus \{i\}} \Omega_i$ is the set of all other players' possible actions.

Definition 3.1 (Nash Equilibrium)**.** *Given a game* $\mathbf{G}(\mathcal{N}, f_i, \Omega_i)$*, an action profile* $\mathbf{u}^* \in \Omega$ *is a Nash Equilibrium (NE) of* $\mathbf{G}$ *if*

$$f_i(u_i^*, \mathbf{u}_{-i}^*) \leq f_i(u_i, \mathbf{u}_{-i}^*) \ \forall u_i \in \Omega_i, \ \forall i \in \mathcal{N}.$$

At a Nash Equilibrium, no player can deviate unilaterally for a lower cost. Note that this does not necessarily correspond to a perfect outcome for any of the players, it is simply a best-case cost reduction such that no player "regrets" the choice they made. As an example, consider the classic Prisoner's Dilemma (Example 4.2, [52]). The setup involves 2 criminals (players), with no communication, given the choice to keep quiet or squeal when questioned on the other's involvement in a crime. If one of the players confesses, they will be set free and the other sentenced for 5 years. If neither confesses, both will be sentenced for 3 years, but if both confess they will both be sentenced

	1	2
1	{3,3}	{5,0}
2	{0,5}	{4,4})

Table 3.1: Action sets for the Prisoner's Dilemma.

to 4 years. Table 3.1 outlines the costs for each player as pairs, indexed by each player's action. We can relate this to our above notation by indexing the players as $\mathcal{N} = \{1, 2\}$, and enumerating the keep quiet action as $u_i = 1$ and the squeal action as $u_i = 2$. Then each entry of Table 3.1 corresponds to the costs $\{f_1(u_1, u_2),\ f_2(u_1, u_2)\}$, where u_1 and u_2 are respectively read from the column and row headers.

The Nash equilibrium in this case is the action (2,2) for a cost of $\{4, 4\}$, i.e., both players squeal. If the action profile was (1,1), each player wishes they had chosen the action 2 to get the lowest possible cost, and if the outcome was (1,2) or (2,1), the player that chose 2 wishes to have deviated and chosen 1 for a lower cost. Note how this is not a "socially optimal" outcome, in that it's not the happiest possible outcome for any of the players. It is simply the one in which neither player could have done better unilaterally.

We now generalize this concept to the case where there are affine coupling constraints defining a relationship between the players' actions. We denote these constraints as $\mathbf{u} \in \mathcal{U}$, where

$$\mathcal{U} = \left\{ \mathbf{u} \in \Omega \ : \ \sum_{i \in \mathcal{N}} A_i u_i \geq \sum_{i \in \mathcal{N}} b_i,\ b_i \in \mathbb{R}^m,\ A_i \in \mathbb{R}^{m \times n_i} \right\}. \tag{3.1}$$

Note that the conditions in the set $\mathcal{U}$ represent m global, affine constraints: $A\mathbf{u} \geq b$ where $A = [A_1,\ \dots\ , A_N]$ and $b = \sum_{i \in \mathcal{N}} b_i$. We denote a game $\mathbf{G}(\mathcal{N}, \mathbf{f_i}, \mathbf{\Omega_i})$ with constraints $\mathcal{U}$ as $\mathbf{G}(\mathcal{N}, f_i, \Omega_i, \mathcal{U})$. We then generalize Definition 3.1 as follows.

Definition 3.2 (Generalized Nash Equilibrium)**.** *Given a game $\mathbf{G}(\mathcal{N}, f_i, \Omega_i, \mathcal{U})$, an action profile $\mathbf{u}^* \in \mathcal{U}$ is a Generalized Nash Equilibrium (GNE) of $\mathbf{G}$ if*

$$f_i(u_i^*, \mathbf{u}_{-i}^*) \leq f_i(u_i, \mathbf{u}_{-i}^*)\ \forall u_i \in \mathcal{U}_i(\mathbf{u}_{-i}^*),\ \forall i \in \mathcal{N},$$

where $\mathcal{U}_i(\mathbf{u}_{-i}^*) = \left\{ u_i \in \Omega_i : (u_i, \mathbf{u}_{-i}^*) \in \mathcal{U} \right\}$.

In other words, a GNE is defined as an action profile with the optimal cost for the player on the feasible set $\mathcal{U}_i(\mathbf{u}_{-i}^*)$. We focus on Generalized Nash Equilibrium Problems for the rest of this work.

Assumption 1. *For each player $i \in \mathcal{N}$, $f_i(u_i, \mathbf{u}_{-i})$ is a differentiable, convex function with respect to $u_i \in \Omega_i$ for any fixed $\mathbf{u}_{-i} \in \Omega_{-i}$. The action set Ω_i is a closed, convex set, and the feasible sets, $\mathcal{U}$ and $\mathcal{U}_i(\mathbf{u}_{-i})$, given any fixed $\mathbf{u}_{-i} \in \Omega_{-i}$, have a nonempty interior.*

Let $\mathbf{u}^*$ be a GNE for a game $\mathbf{G}(\mathcal{N}, f_i, \Omega_i, \mathcal{U})$. For player i, their chosen action u_i^* is the optimal solution to the following convex optimization problem:

$$\min_{u_i} f_i(u_i, \mathbf{u}_{-i}^*),\ \text{s.t. } u_i \in \Omega_i,\ A_i u_i \geq b - \sum_{j \in \mathcal{N} \setminus \{i\}} A_j u_j^*. \tag{3.2}$$

We can define a Lagrangian function for player i as $L_i(u_i, \lambda_i; \mathbf{u}_{-i}) = f_i(u_i, \mathbf{u}_{-i}) + \lambda_i^\top (b - A\mathbf{u})$. When u_i^* is an optimum of (3.2), there exists a multiplier $\lambda_i^* \in \mathbb{R}_+^m$ such that the following KKT conditions

are satisfied:

$$\begin{aligned} &\mathbf{0} \in \nabla_{u_i} f_i(u_i, \mathbf{u}^*_{-i}) - A_i^\top \lambda_i^* + N_{\Omega_i}(u_i^*), \\ &\mathbf{0} \in (A\mathbf{u}^* - b) + N_{\mathbb{R}^m_+}(\lambda_i^*). \end{aligned} \tag{3.3}$$

Denote $\lambda = \text{col}(\lambda_1, \ \dots \ , \lambda_N)$. When $\mathbf{u}^*, \lambda^*$ satisfies the KKT conditions (3.3) for all $i \in \mathcal{N}$, the action profile $\mathbf{u}^*$ is a GNE of the game $\mathbf{G}$ (Theorem 4.6, [21]).

For a given GNE of $\mathbf{G}$, the Lagrange multipliers may be different, that is, $\lambda_i \neq \lambda_j$ may be true for one or more pairs of players $i, \ j, \ i \neq j$. In this thesis, we are concerned with seeking a GNE such that $\lambda_1 = \ldots = \lambda_N$, referred to as a *variational GNE* (vGNE). This ensures a more "socially stable" equilibrium, by penalizing players equally for violating the constraints. From [21] (Theorem 4.8), we know that a vGNE of $\mathbf{G}$ is a solution $\mathbf{u}^* \in \mathcal{U}$ of the variational inequality VI(F,$\mathcal{U}$), as defined in Definition 2.3, where F denotes the pseudogradient of the players' cost functions, defined as

$$F(\mathbf{u}) = \text{col}(\nabla_{u_1} f_1(u_1, \mathbf{u}_{-1}), \ \dots \ , \nabla_{u_N} f_N(u_N, \mathbf{u}_{-N})). \tag{3.4}$$

From Definition 2.3, an action profile $\mathbf{u}^* \in \mathcal{U}$ solves VI(F,$\mathcal{U}$) if and only if $\mathbf{u}^*$ is an optimal solution to

$$\min_{\mathbf{u} \in \Omega} \langle F(\mathbf{u}^*), \mathbf{u} \rangle \text{ s.t. } \mathbf{u} \in \Omega, \ A\mathbf{u} \geq b$$

Hence $\mathbf{u}^*$ solves VI(F,$\mathcal{U}$) if and only if there exists a multiplier $\lambda^* \in \mathbb{R}^m$ such that the following KKT conditions are met [35]:

$$\begin{aligned} &\mathbf{0} \in \nabla_{u_i} f_i(u_i, \mathbf{u}^*_{-i}) - A_i^\top \lambda^* + N_{\Omega_i}(u_i^*), \ i \in \mathcal{N}, \\ &\mathbf{0} \in (A\mathbf{u}^* - b) + N_{\mathbb{R}^m_+}(\lambda^*). \end{aligned} \tag{3.5}$$

Theorem 3.1 (Variational Generalized Nash Equilibrium, Theorem 4.8, [21])**.** *Suppose Assumption 1 holds. Then every solution* $\mathbf{u}^*$ *of the variational inequality* VI(F,$\mathcal{U}$) *is a GNE of the game* $\mathbf{G}(\mathcal{N}, f_i, \Omega_i, \mathcal{U})$*. Further, given that* $\mathbf{u}^*$ *and* λ^* *satisfy the VI KKT conditions* (3.5)*,* $\mathbf{u}^*$ *together with* $\lambda_1^* = \ \cdots \ = \lambda_N^* = \lambda^*$ *satisfy the GNE KKT conditions* (3.3)*.*

3.2 Distributed vGNE-seeking Algorithm

This section develops and outlines an algorithm that can be used to find a variational GNE of the GNE problem described in Section 3.1. For a detailed proof and convergence analysis, refer to [19]. The idea of the proof is to view a point satisfying the KKT conditions (3.5) as a zero of the sum of monotone operators. The algorithm then takes the form of a forward-backward operator-splitting method [28]. In Chapter 4 we adapt this algorithm to cases where agents are concerned with a measured output which they do not have an a priori model for.

Notice in the KKT conditions (3.5), that we can stack all $i \in \mathcal{N}$ elements of the first condition as

$$\mathbf{0} \in F(\mathbf{u}) - A^\top \lambda + N_\Omega(\mathbf{u}).$$

We can then define the following two operators:

$$\begin{aligned} &\mathfrak{A} : \text{col}(\mathbf{u}, \lambda) \mapsto \text{col}(F(\mathbf{u}), -b) \\ &\mathfrak{B} : \text{col}(\mathbf{u}, \lambda) \mapsto \text{col}(-A^\top \lambda + N_\Omega(\mathbf{u}), A\mathbf{u} + N_{\mathbb{R}^m_+}(\lambda)). \end{aligned} \tag{3.6}$$

The KKT conditions (3.5) can then be expressed as $\text{col}(\mathbf{u}^*, \lambda^*) \in \text{zer}(\mathfrak{A} + \mathfrak{B})$. Note that $\mathfrak{B}$ is a maximally monotone operator (proof in Appendix A), so we can place assumptions on F to ensure that solving VI(F,$\mathcal{U}$) can be expressed as a problem of finding zeros of a sum of monotone operators.

Assumption 2. *The pseudogradient $F(\mathbf{u})$ defined in* (3.4) *is η-strongly monotone and θ-Lipschitz continuous.*

Remark 1. *Assumption 2 guarantees the existence of a unique solution to* VI(F,$\mathcal{U}$) *[19, 48, 53]. This ensures that the game has a unique variational GNE even if it may not have a unique GNE. The algorithm from [19] thus aims to find this unique variational GNE, enforcing the coupling constraint.*

We next outline the algorithm from [19], and analyze its convergence properties. Each player has access to its local cost function f_i and, for a given $\text{col}(u_i, \mathbf{u}_{-i})$ they can evaluate the gradient $\nabla_{u_i} f_i$. As in [19, 48, 49, 53], we assume that the player has access to any other player's decision that its cost is dependent on, obtained via an interference graph (i.e., we assume $\mathbf{u}_{-i}$ is known precisely). However, we assume that the global constraint set is not fully known, instead each player only knows its local block of the constraints, i.e., $\Omega_i,\ A_i,\ b_i$.

Agent i controls its local decision $u_i \in \mathbb{R}^{n_i}$ and a local copy of the multiplier $\lambda_i \in \mathbb{R}^m_+$. It has a local auxiliary variable $z_i \in \mathbb{R}^m$ which it uses to coordinate with its neighbours in order to achieve consensus on λ_i, as needed to satisfy the vGNE KKT conditions (3.5). We specify two graphs, $\mathcal{G}_f$ and $\mathcal{G}_\lambda$. The *interference graph* $\mathcal{G}_f$ is defined as in [48], [50]: each agent has an edge connecting it to each other agent whose decision or output directly affects its costs, i.e., $\mathcal{G}_f = (\mathcal{N}, \mathcal{E}_f, \{w_{ij}\}),\ (j,i) \in \mathcal{E}_f$ if $f_i(u_i, \mathbf{u}_{-i})$ *explicitly* depends on u_j. Agent i has the ability to compute its respective gradients as described above, observing the relevant values through $\mathcal{G}_f$. Agents communicate their values for λ_i and z_i through $\mathcal{G}_\lambda$, defined as having an edge between two agents if they communicate their multiplier estimate to one another during the algorithm.

Assumption 3. *The weighted communication graph $\mathcal{G}_\lambda$ is undirected and connected.*

The auxiliary variables z_i are introduced in the problem to help agents arrive to a consensus on the Lagrange multipliers λ_i. We noted in Remark 1 that we are seeking the variational GNE of the problem, where the Lagrange multipliers are all equal, which motivates the inclusion of our consensus variables. Each z_i can be viewed as player i's estimate of the other players' contribution to the constraint satisfaction.

Algorithm 1 Distributed GNE-seeking algorithm

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

Iteration: Player i

Step 1: Primal Step and Consensus - Receives $u_{j,k}$, $j \in \mathcal{N}_{\mathcal{G}_f}(i)$, $\lambda_{j,k}$, $j \in \mathcal{N}_{\mathcal{G}_\lambda}(i)$ and updates:

$$u_{i,k+1} \leftarrow P_{\Omega_i}\Big(u_{i,k} - \tau_i(\nabla_{u_{i,k}} f_i(u_{i,k}, \mathbf{u}_{-i,k}) - A_i^\top \lambda_{i,k})\Big)$$

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

Step 2: Consensus and Dual Step - Receives $z_{j,k+1}$, $j \in \mathcal{N}_{\mathcal{G}_\lambda}(i)$ and updates:

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

Step 1 of the algorithm consists of two parts: the first is simply the primal step of a projected gradient algorithm, and the second is a consensus estimation step, where each agent uses data from its neighbours to update its estimate of other players' contributions to the coupling constraints. Step 2 is a dual step that updates the Lagrange multipliers while accounting incorporating the impact of other agents' contributions through the consensus values from the previous step.

3.2.1 Forward-Backward Algorithm

In this section we outline the development of the distributed GNE-seeking algorithm, Algorithm 1. For a detailed reading, we refer the reader to [19].

We first define some notation to express Algorithm 1 in a compact manner. Define $\mathbf{u}_k = \text{col}(u_{1,k}, \dots, u_{N,k})$, $\overline{\lambda}_k = \text{col}(\lambda_{1,k}, \dots, \lambda_{N,k})$, with $\overline{z}_k$ and $\overline{b}$ defined similarly. Let $\Lambda = \text{diag}(A_1, \dots, A_N)$ and $\overline{L}{=}L \otimes I_m$. Finally, let $\overline{\tau} = \text{diag}(\tau_1 I_{n_1}, \dots, \tau_N I_{n_N})$ and $\overline{\tau}^{-1} = \text{diag}(\frac{1}{\tau_1} I_{n_1}, \dots, \frac{1}{\tau_N} I_{n_N})$, with $\overline{\nu}$, $\overline{\sigma}$, $\overline{\nu}^{-1}$, $\overline{\sigma}^{-1}$ defined similarly. With this, we can rewrite each step of the algorithm as

$$\mathbf{u}_{k+1} = P_\Omega[\mathbf{u}_k - \bar{\tau}(F(\mathbf{u}_k) - \Lambda^\top \bar{\lambda}_k)], \tag{3.7a}$$

$$\bar{z}_{k+1} = \bar{z}_k + \bar{\nu}\bar{L}\bar{\lambda}_k, \tag{3.7b}$$

$$\bar{\lambda}_{k+1} = P_{\mathbb{R}^{mN}_+}\{\bar{\lambda}_k - \bar{\sigma}[\Lambda(2\mathbf{u}_{k+1} - \mathbf{u}_k) - \bar{b} + \bar{L}\bar{\lambda}_k + \bar{L}(2\bar{z}_{k+1} - \bar{z}_k)]\}. \tag{3.7c}$$

By Lemma 2.6, $P_\Omega(x) = (\text{Id} + N_\Omega)^{-1}$, so (3.7a) can be rewritten as

$$\begin{aligned}
(\text{Id} + N_\Omega)^{-1}[\mathbf{u}_k - \bar{\tau}(F(\mathbf{u}_k) - \Lambda^\top \bar{\lambda}_k)] &= \mathbf{u}_{k+1} \\
\mathbf{u}_k - \bar{\tau}(F(\mathbf{u}_k) - \Lambda^\top \bar{\lambda}_k) &\in \mathbf{u}_{k+1} + N_\Omega(\mathbf{u}_{k+1}) \\
-F(\mathbf{u}_k) + \Lambda^\top \bar{\lambda}_k + \Lambda^\top(\bar{\lambda}_{k+1} - \bar{\lambda}_{k+1}) &\in \bar{\tau}^{-1}(\mathbf{u}_{k+1} - \mathbf{u}_k) + N_\Omega(\mathbf{u}_{k+1}) \\
-F(\mathbf{u}_k) &\in N_\Omega(\mathbf{u}_{k+1}) - \Lambda^\top \bar{\lambda}_{k+1} + \bar{\tau}^{-1}(\mathbf{u}_{k+1} - \mathbf{u}_k) + \Lambda^\top(\bar{\lambda}_{k+1} - \bar{\lambda}_k).
\end{aligned} \tag{3.8}$$

With similar arguments, (3.7c) can be rewritten as

$$-[\bar{L}\bar{\lambda}_k - \bar{b}] \in N_{\mathbb{R}^{mN}_+}(\bar{\lambda}_{k+1}) + \Lambda \mathbf{u}_{k+1} + \bar{L}\bar{z}_{k+1} + \Lambda(\mathbf{u}_{k+1} - \mathbf{u}_k) + \bar{L}(\bar{z}_{k+1} - \bar{z}_k) + \bar{\sigma}^{-1}(\bar{\lambda}_{k+1} - \bar{\lambda}_k) \quad (3.9)$$

We define two matrices

$$\Phi = \begin{bmatrix} \bar{\tau}^{-1} & \mathbb{0} & \Lambda^\top \\ \mathbb{0} & \bar{\nu}^{-1} & \bar{L} \\ \Lambda & \bar{L} & \bar{\sigma}^{-1} \end{bmatrix}, \; \Psi = \begin{bmatrix} \mathbb{0} & \mathbb{0} & -\Lambda^\top \\ \mathbb{0} & \mathbb{0} & -\bar{L} \\ \Lambda & \bar{L} & \mathbb{0} \end{bmatrix}. \quad (3.10)$$

Note that Φ is symmetric ($\Phi = \Phi^\top$) and Ψ is skew-symmetric ($\Psi = -\Psi^\top$). Denote $\varpi = \mathrm{col}(\mathbf{u}, \bar{z}, \bar{\lambda})$. Then (3.7b), (3.8), (3.9) can be expressed as

$$-\bar{\mathfrak{A}}(\varpi_k) \in \bar{\mathfrak{B}}(\varpi_{k+1}) + \Phi(\varpi_{k+1} - \varpi_k), \quad (3.11)$$

with

$$\begin{aligned} \bar{\mathfrak{A}} &: \varpi \mapsto \mathrm{col}(F(\mathbf{u}), \mathbf{0}, \bar{L}\bar{\lambda} - \bar{b}) \\ \bar{\mathfrak{B}} &: \varpi \mapsto N_\Omega(\mathbf{u}) \times \mathbf{0} \times N_{\mathbb{R}^{mN}_+}(\bar{\lambda}) + \Psi\varpi. \end{aligned} \quad (3.12)$$

Rearranging (3.11), we obtain

$$\varpi_{k+1} = (\mathrm{Id} + \Phi^{-1}\bar{\mathfrak{B}})^{-1}(\mathrm{Id} - \Phi^{-1}\bar{\mathfrak{A}})(\varpi_k). \quad (3.13)$$

The iteration (3.13) is a compact representation of a single step of Algorithm 1, referred to as Picard's iteration [19]. We go into more detail about this iteration later, in the proof for Lemma 4.1. Each iteration can be viewed as having a forward step evaluating $\bar{\mathfrak{A}}$ and a backward step evaluating the resolvent of $\bar{\mathfrak{B}}$, hence the algorithm being referred to as a forward-backward operator-splitting algorithm. For more background on this class of algorithms, refer to [28]. The traditional forward-backward algorithm presented therein uses $\Phi = I$, and evaluates a cocoercive operator in the forward step and the resolvent (Definition 2.13) of a monotone operator in the backward step. The key problem is that the resolvent cannot be computed without computing the inverse of Ψ, which contains the constraint matrix and thus cannot be computed in a distributed manner (since we assume each agent only knows a local block of the constraints). This difficulty is resolved by introducing the metric matrix Φ, which enables the distributed evaluation of the backward step using local information as in (3.7c).

We do not present the rest of the convergence analysis in this section, as we would repeat large parts of it for the proof in the convergence analysis we perform for our later adaptation of this algorithm. For a full treatment, refer to [19], or to the later Sections 4.3.1 and 4.3.2 where we outline the application of this proof to our developed algorithm. Intuitively, the convergence of the algorithm relies on these key factors:

1. The operators $\bar{\mathfrak{A}}$ and $\bar{\mathfrak{B}}$ are higher dimensional augmentations of the operators $\mathfrak{A}$ and $\mathfrak{B}$ described in (3.6), introducing the auxiliary variables $\bar{z} \in \mathbb{R}^{mN}$, and the augmented variables and matrices defined at the beginning of this section.

2. A limiting point of Algorithm 1 corresponds to a zero of the sum of the operators $\bar{\mathfrak{A}}$ and $\bar{\mathfrak{B}}$, which also corresponds to a point satisfying the variational KKT conditions (3.5).

3. The convergence of Algorithm 1 to a limiting point can be guaranteed by the choice of specific step sizes and assumptions that ensure that each iteration is a sum of maximally monotone, nonexpansive operators.

This forms the basis for the problem space that this thesis intends to explore, which we introduce in the following chapter. Following Theorem 3 in [19], we know that the algorithm converges to a limiting point that corresponds to the unique vGNE of the game $\mathbf{G}$. This provides theoretical guarantees of finding an optimum of the game obeying the global coupling constraints, with each agent only having local knowledge of those constraints.

Chapter 4

Games Played with Output Mappings

In this chapter, we introduce the primary class of problems that this thesis addresses. We begin from the GNE problem introduced in Section 3.1, and modify the problem to account for an input-output mapping, akin to the one in [1]. We briefly discuss offline methods of solving optimization problems and the limitations of these methods, before detailing our online approach for the GNE problem. We redefine our solution concept as a variational KKT point, rather than a vGNE (discussed in Chapter 3), since the unknown nature of the output mapping makes the existence of a vGNE hard to guarantee. Then we reformulate Algorithm 1 for the newly defined class of problems, and provide a detailed convergence analysis.

4.1 Problem Formulation

Consider the GNE problem from Section 3.1. We extend that formulation, and assume that each player $i \in \mathcal{N}$ is endowed with an output $y_i \in \mathbb{R}^{l_i}$, generated by some unknown input-to-output relation. Suppose that they wish to constrain this output while attempting to achieve their respective goals (described via the cost function f_i introduced in Section 3.1). The output is a function of all the players' actions $\mathbf{u} \in \Omega$ and of an unmeasured, exogenous disturbance $w \in \mathcal{W} \subseteq \mathbb{R}^p$. We define the output via

$$y_i := \pi_i(u_i, \mathbf{u}_{-i}, w) \tag{4.1}$$

where $\pi_i : \Omega \times \mathcal{W} \to \mathbb{R}^{l_i}$ is the input-to-output mapping outlined above [1]. We assume that each π_i is $\mathbb{C}^1$ in $\mathbf{u}$. The dependence of the mapping on $\mathbf{u}$ reflects the fact that, as they attempt to optimize f_i, players will cause fluctuations in the system output. We denote $\mathbf{y} = \text{col}(y_1, \ \dots \ , y_N)$ as the overall system output, with dimension $l = \sum_{i \in \mathcal{N}} l_i$. Further, for compactness we denote the stacked output mapping as

$$\mathbf{y} = \pi(\mathbf{u}, w) := \text{col}(\pi_1(u_1, \mathbf{u}_{-1}, w), \ \dots \ , \pi_N(u_N, \mathbf{u}_{-N}, w)).$$

We consider GNE problems where the players are playing within a system with specific output constraints. Suppose that the players are expected to play the game while constraining the

system output to $\mathbf{y} \in \mathcal{Y}$ (for example, a set of safety constraints for voltage levels in a distributed power system). Then for any player, the output constraint can be expressed as $\mathcal{Y}_i(\mathbf{y}^*_{-i}) = \{y_i \in \Omega_i : (y_i, \mathbf{y}_{-i}) \in \mathcal{Y}\}$, and each player is interested in solving an optimization problem of the form

$$\begin{aligned} \underset{u_i \in \mathbb{R}^{n_i}}{\text{minimize}} \quad & f_i(u_i, \mathbf{u}_{-i}) \\ \text{subject to} \quad & u_i \in \mathcal{U}_i(\mathbf{u}_{-i}) \subset \mathbb{R}^{n_i} \\ & y_i \in \mathcal{Y}_i(\mathbf{y}_{-i}) \subset \mathbb{R}^{l_i} \\ & y_i = \pi_i(u_i, \mathbf{u}_{-i}, w), \end{aligned} \tag{4.2}$$

where each player's goal in the game is encoded by the cost function $f_i : \Omega_i \to \mathbb{R}$. Problems of the form (4.2) are a useful representation of problems in many engineering disciplines, such as traffic control, networks, power systems, etc. These problems generally tend to be hard to solve. The mapping π is often non-convex (and nonlinear), and involves many outputs with interdependence on a large number of inputs and disturbances. Offline models of such systems usually require precise knowledge of the mapping π and the disturbance w, which is often not available.

4.1.1 Feedforward Forecast-Based Optimization

In real-world applications, we can often assume that each player has access to a linearized model of the output system:

$$y_i \approx \Pi_i \mathbf{u} + \Pi_{w,i} w, \tag{4.3}$$

where $\Pi_i,\ \Pi_{w,i}$ are fixed matrices extracted from a linearized model of the mapping in question. We also assume that the players have access to a forecast or guess for the disturbance, denoted $\hat{w}$. The player then periodically solves for y_i and attempts to solve the following optimization problem:

$$\begin{aligned} \underset{u_i \in \mathbb{R}^{n_i}}{\text{minimize}} \quad & f_i(u_i, \mathbf{u}_{-i}) \\ \text{subject to} \quad & u_i \in \mathcal{U}_i(\mathbf{u}_{-i}) \subset \mathbb{R}^{n_i} \\ & \hat{y}_i \in \mathcal{Y}_i(\mathbf{y}_{-i}) \subset \mathbb{R}^{l_i} \\ & \hat{y}_i = \Pi_i \mathbf{u} + \Pi_{w,i} \hat{w}. \end{aligned} \tag{4.4}$$

This optimization problem is analogous to a feedforward control law applied to a dynamical system. Figure 4.1 illustrates the algorithm visually. Note that (4.2) is a non-convex optimization problem, and the feedforward approach in (4.4) is a convex relaxation of it, since the cost function is convex and the constraints are affine. Such relaxations are very useful as they enable the optimization of various otherwise intractable problems.

A limitation of this technique is that the linearization is usually only accurate enough when the system stays close to the point around which it was linearized. Large deviations can cause the algorithm to have an inaccurate estimation of the outputs, and this sometimes limits the usefulness of these techniques. As a particular example, consider the use of this technique over the course of long runtime (e.g. the distribution feeder considered in Section 7.4) with large variation in disturbances). A forecast of the output based on the linearized model would be relatively accurate for portions of the runtime where the system is close to the point of linearization, but there is no guarantee of its behaviour when far from that point.

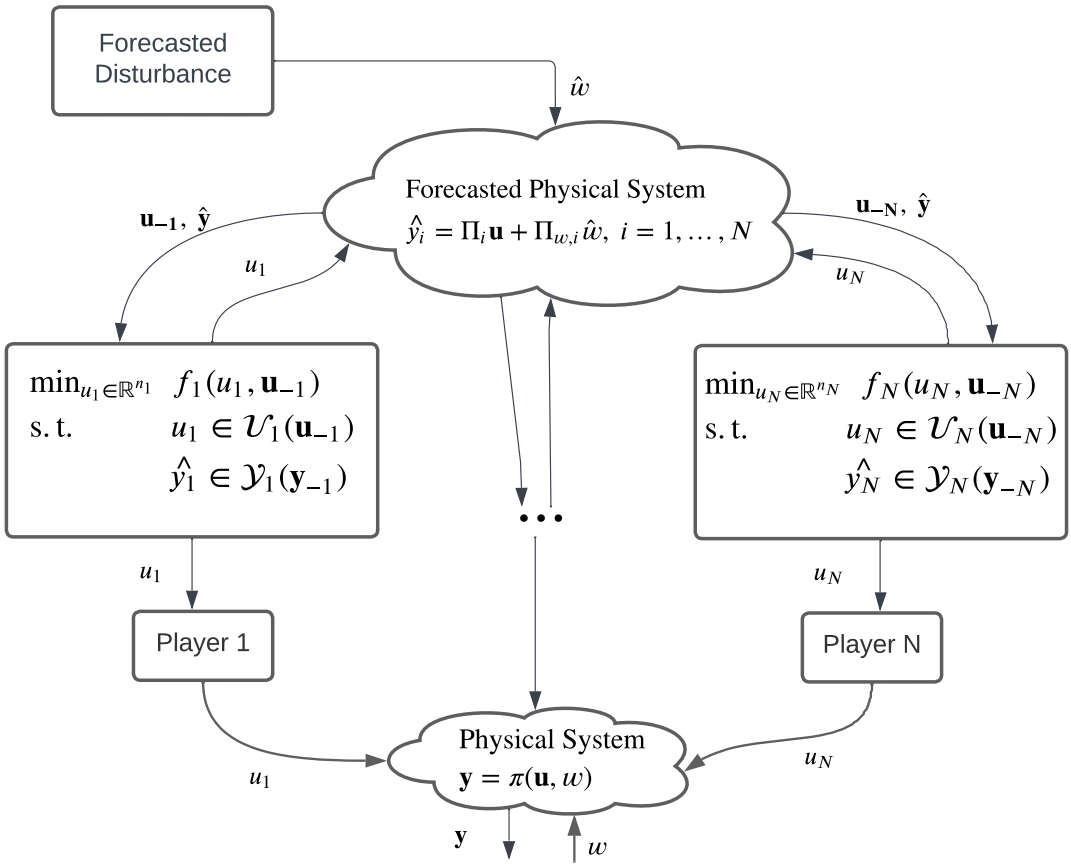

Figure 4.1: Illustration of a forecast-based optimizer.

Further, note the stringent informational requirements of this problem. Suppose that we approach this problem using Algorithm 1. Then each agent needs to receive a forecast for each *iteration* of the algorithm. The more accurate a forecast we need, the more complex a system we would need to generate it. This presents a dilemma where a designer has to choose between accurate convergence, and the implementability of the optimization algorithm.

The use of output feedback can help alleviate some of these concerns. The agent does not need to be able to predict the disturbance or the output if it can, instead, measure the output (or a related quantity). Using the measurement, we can develop decentralized, feedback-based optimization schemes to approach a solution of (4.2).

4.1.2 Online Feedback-Based Optimization

We outlined the reasons for using measurements of y_i instead of a forecast above. In this section we outline our output-feedback based method. First, we address the problem of actually constraining the output to the set $\mathcal{Y}_i(\mathbf{y}_{-i})$. We achieve this by encoding the safety constraint in some appropriately chosen cost function. The cost function can be chosen as a soft constraint (e.g. a quadratic

function penalizing point-to-set distance from $\mathcal{Y}_i(\mathbf{y}_{-i})$) or a hard constraint (e.g. a logarithmic barrier function). A soft constraint can be justified in a variety of applications where the system can tolerate small deviations outside the safety set.

Each player is then interested in solving the following optimization problem:

$$\begin{aligned} \underset{u_i \in \mathbb{R}^{n_i}}{\text{minimize}} \quad & f_i(u_i, \mathbf{u}_{-i}) + g_i(y_i, \mathbf{y}_{-i}) \\ \text{subject to} \quad & y_i = \pi_i(u_i, \mathbf{u}_{-i}, w) \\ & u_i \in \mathcal{U}_i(\mathbf{u}_{-i}) \subset \mathbb{R}^{n_i}, \end{aligned} \tag{4.5}$$

where f_i is the cost function encoding each player's goals in the game, and g_i is an appropriately chosen cost function penalizing them for disobeying the output constraint $y_i \in \mathcal{Y}_i(\mathbf{y}_{-i})$. Games of this form are decentralized versions of the problem described in [1]. The game formulation enables non-cooperative decision-making, allowing more decentralized implementations of controllers and a potential reduction in the communication overhead.

Assumption 4. *For each player $i \in \mathcal{V}$, the cost function $g_i(y_i, \mathbf{y}_{-i})$ is $\mathbb{C}^1$ and convex with respect to $\mathbf{y} = (y_i, \mathbf{y}_{-i})$.*

Note here that we do not yet place any assumptions on the output mapping π or the disturbance w. Later in this chapter, we parametrize our optimum as a function of the unknown disturbance w, and in Chapter 5 we present structural constraints on $\pi(\mathbf{u}, w)$ that ensure that our algorithm is applicable to the system.

4.2 Generalized Nash Equilibrium with Output Mapping

A solution $\mathbf{u}^* \in \mathbb{R}^n$ to (4.5) is a GNE, as described in Section 3.1. We thus follow the same procedure to define the KKT conditions for this game. For player i, their chosen action u_i^* is the optimal solution to the following optimization problem:

$$\begin{aligned} \underset{u_i}{\text{minimize}} \quad & f_i(u_i, \mathbf{u}^*_{-i}) + g_i(y_i, \mathbf{y}_{-i}) \\ \text{subject to} \quad & u_i \in \Omega_i, \\ & y_i = \pi_i(u_i, \mathbf{u}^*_{-i}, w), \\ & A_i u_i \geq b - \sum_{j \in \mathcal{N} \setminus \{i\}} A_j u_j^*. \end{aligned} \tag{4.6}$$

We now define a Lagrangian function for player i as

$$L_i(u_i, \lambda_i; \mathbf{u}_{-i}, w) = f_i(u_i, \mathbf{u}_{-i}) + g_i(\pi_i(u_i, \mathbf{u}_{-i}, w), \pi_{-i}(u_i, \mathbf{u}_{-i}, w)) + \lambda_i^\top (b - A\mathbf{u}).$$

Note that this Lagrangian explicitly denotes the dependence on u_i, and is parametrized by the disturbance w and the actions $\mathbf{u}_{-i}$ of all other agents. Applying the chain rule, we know that the gradient of the Lagrangian can be expressed as

$$\nabla_{u_i} L_i(u_i, \lambda_i; \mathbf{u}_{-i}, w) = \nabla_{u_i} f_i(u_i, \mathbf{u}_{-i}) + \sum_{j=1}^{N} \partial_{u_i} \pi_j(u_i, \mathbf{u}_{-i}, w)^\top \nabla_{y_j} g_i(y_i, \mathbf{y}_{-i}) - A_i^\top \lambda_i,$$

where $\mathbf{y} = \pi(\mathbf{u}, w)$. We can combine this with the results from Section 3.1 to find the variational KKT conditions for the game. We first define the cost functions for each agent as

$$h_{i,w}(u_i, \mathbf{u}_{-i}) = f_i(u_i, \mathbf{u}_{-i}) + g_i(\pi_i(u_i, \mathbf{u}_{-i}, w), \pi_{-i}(u_i, \mathbf{u}_{-i}, w)).$$

We can then define the pseudogradient of these cost functions as

$$H_w(\mathbf{u}) = \operatorname{col}(\nabla_{u_1} h_{1,w}(u_1, \mathbf{u}_{-1}),\ \ \dots\ \ , \nabla_{u_N} h_{N,w}(u_N, \mathbf{u}_{-N})).$$

Here, the subscript w indicates the implicit dependence on the disturbance. We can compactly express the operator H_w as

$$H_w(\mathbf{u}) = F(\mathbf{u}) + \mathfrak{E}\big(\partial_{\mathbf{u}} \pi(\mathbf{u}, w)^\top \partial_{\mathbf{y}} g(\mathbf{y})^\top\big), \tag{4.7}$$

where $F(\mathbf{u})$ is the pseudogradient of the agents' cost functions f_i, and the function $g(\mathbf{y}) = \operatorname{col}(g_1(y_1, \mathbf{y}_{-1}),\ \ \dots\ \ , g_N(y_N, \mathbf{y}_{-N}))$ is the stacked vector of the output-constraint penalties. Note again that $\mathbf{y} = \pi(\mathbf{u}, w)$, hence why the function H_w is only in $\mathbf{u}$. The operator $\mathfrak{E} : \mathbb{R}^{n \times N} \to \mathbb{R}^n$ is a bounded linear operator (we provide a proof for this in Appendix B) that aligns the relevant entries of the Jacobians with the pseudogradient, defined as

$$\mathfrak{E}(M) = \sum_{k=1}^{N} \sum_{l \in \mathcal{N}_k} (e_l^\top M e_k) e_l, \tag{4.8}$$

where $\mathcal{N}_k = \left\{1 + \sum_{i=1}^{k-1} n_i\ \ , \cdots,\ \ \sum_{i=1}^{k} n_i\right\}$.

Remark 2. *The set $\mathcal{N}_k$ defined as part of the extraction operator $\mathfrak{E}$ is a set that contains indices chosen in a way to align the relevant entries of the Jacobian product $\partial_{\mathbf{u}} \pi(\mathbf{u}, w)^\top \partial_{\mathbf{y}} g(\mathbf{y})^\top$. As an example, in a 3-player game where players 1 and 3 choose an action in $\mathbb{R}^2$ and player 2 chooses an action in $\mathbb{R}^3$, the sets would be defined as*

$$\begin{aligned} \mathcal{N}_1 &= \{1, 2\}, \\ \mathcal{N}_2 &= \{3, 4, 5\}, \\ \mathcal{N}_2 &= \{6, 7\}. \end{aligned}$$

This is meant to reflect the fact that in the overall action vector $\mathbf{u}$, the first two entries correspond to $u_1 \in \mathbb{R}^2$, the next three correspond to $u_2 \in \mathbb{R}^3$, and the final two correspond to $u_3 \in \mathbb{R}^2$.

An action profile $\mathbf{u}^*$ solves VI(H_w,$\mathcal{U}$) if and only if the following KKT conditions are met:

$$\begin{aligned} &\mathbf{0} \in \nabla_{u_i} f_i(u_i^*, \mathbf{u}_{-i}^*) + \sum_{j=1}^{N} \partial_{u_i} \pi_j(u_i, \mathbf{u}_{-i}, w)^\top \nabla_{y_j} g_i(y_i^*, \mathbf{y}_{-i}^*) - A_i^\top \lambda^* + N_{\Omega_i}(u_i^*),\ i \in \mathcal{N}, \\ &\mathbf{0} \in (A\mathbf{u}^* - b) + N_{\mathbb{R}^m_+}(\lambda^*), \end{aligned} \tag{4.9}$$

where $\mathbf{y}^* = \pi(\mathbf{u}^*, w)$. The existence of a KKT point that satisfies these variational KKT conditions (a vKKT point, for short) is guaranteed under Assumptions 1 and 4 by Corollary 2.2.5 in [35].

Remark 3. *Due to the potential non-convexity of $g \circ \pi$ in* (4.5)*, we cannot guarantee the existence of of a (v)GNE. Instead, we focus on finding vKKT points that satisfy the conditions* (4.9)*. These points are candidates to be local vGNEs [51]. Under Assumptions 1 and 4, if we additionally assume that π is an affine mapping, then $g_i \circ \pi$ will be convex in u_i for a fixed $\mathbf{u}_{-i}$. Then, by Theorem 4.8 of [21], this point would be a vGNE of* (4.5)*.*

Note that to develop Algorithm 1, we made additional assumptions on the pseudogradient operator to ensure that the problem could be formulated as finding the zeros of a sum of monotone operators. However, this assumption cannot immediately be made regarding the operator H_w, as the mapping π and the disturbance w are unknown. We next make an approximation to alleviate this problem.

The expression for H_w in (4.7) contains both the mapping π and its Jacobian $\partial_{\mathbf{u}}\pi$. In our treatment of the algorithm, we replace the former with a direct measurement of the output variables $y_i = \pi(u_i, \mathbf{u}_{-i}, w)$. However, the Jacobian may not be known exactly, and may depend on the unmeasured disturbance w. To obtain an implementable algorithm, we approximate the Jacobian at some nominal operating point. In [1, 7, 9], algorithms for various online, optimal power flow problems are formulated with an approximate Jacobian, and shown to converge robustly to near-optimal solutions. We consider the simplest approximation of the Jacobian,

$$\partial_{\mathbf{u}}\pi(\mathbf{u}, w) \approx \Pi, \ \forall \mathbf{u} \in \mathcal{U}, \ \forall w \in \mathcal{W} \tag{4.10}$$

where $\Pi \in \mathbb{R}^{l \times n}$ is an appropriately chosen nominal Jacobian matrix. The choice of an "appropriate" matrix is problem-dependent: it can be a linearization of the system with respect to $\mathbf{u}$ or a Jacobian computed experimentally at some nominal operating point. In Chapter 7, where we introduce various applications and simulations, we illustrate methods for computing such Jacobians for the problems we choose. Each entry of the matrix can be viewed as an approximation of the sensitivity of one of the outputs to an agent's control input, and thus we interchangeably refer to this Jacobian as the approximate sensitivity matrix. We can then redefine the pseudogradient operator (4.7) as

$$H_{w,\Pi}(\mathbf{u}) = F(\mathbf{u}) + \mathfrak{E}\big(\Pi^\top \partial_{\mathbf{y}} g(\mathbf{y})^\top\big). \tag{4.11}$$

It should be noted that approximating the Jacobian as a constant is the same as assuming that the output mapping has the form $\pi(\mathbf{u}, w) = \Pi\mathbf{u} + \pi_w(w)$, i.e., it is affine in the agents' control inputs $\mathbf{u}$ and has some unknown relation $\pi_w(w)$ with the disturbance w. As we noted earlier in Remark 3, assuming that π is affine in $\mathbf{u}$ guarantees that a vGNE exists under Theorem 4.8 of [21], since each $\pi_i \circ g_i$ would then be convex. Thus, even though the potential non-convexity of π means that we cannot guarantee the existence of a vGNE for the game (4.5), there is a point which would be a vGNE of the game if the mapping $\pi(\mathbf{u}, w) = \Pi\mathbf{u} + \pi_w(w)$ were locally accurate. To formally define this "vGNE-like" point, we first rewrite our KKT conditions (4.9) in terms of the approximated pseudogradient operator (4.11). An action profile $\breve{\mathbf{u}}$ solves $\mathrm{VI}(H_{w,\Pi}, \mathcal{U})$ if and only if the following

KKT conditions are met for some $\breve{\lambda} \in \mathbb{R}^m_+$:

$$\begin{aligned} &\mathbf{0} \in \nabla_{u_i} f_i(\breve{u}_i, \breve{\mathbf{u}}_{-i}) + \sum_{j=1}^{N} \Pi_{ij}^\top \nabla_{y_j} g_i(\breve{y}_i, \breve{\mathbf{y}}_{-i}) - A_i^\top \breve{\lambda} + N_{\Omega_i}(\breve{u}_i),\ i \in \mathcal{N}, \\ &\mathbf{0} \in (A\breve{\mathbf{u}} - b) + N_{\mathbb{R}^m_+}(\breve{\lambda}), \end{aligned} \tag{4.12}$$

where $\breve{\mathbf{y}} = \pi(\breve{\mathbf{u}}, w)$ and each $\Pi_{ij} \approx \partial_{u_i} \pi_j$ is a submatrix of the nominal Jacobian $\Pi \approx \partial_{\mathbf{u}} \pi$.

Definition 4.1. *Given $\Pi \in \mathbb{R}^{l \times n}$ and a disturbance $w \in \mathbb{R}^p$, a vector $\breve{\mathbf{u}} = \breve{\mathbf{u}}(w) \in \mathbb{R}^n$ is an online approximate variational GNE (OA vGNE) of the game* (4.5) *if*

$$\begin{aligned} &\breve{\mathbf{y}} = \pi(\breve{\mathbf{u}}, w) \\ &\breve{\mathbf{u}} \in \mathcal{U} \\ &\breve{\lambda} \in \mathbb{R}^m_+ \\ &A^\top \breve{\lambda} - H_{w,\Pi}(\breve{\mathbf{u}}) \in N_{\mathcal{U}}(\breve{\mathbf{u}}) \\ &b - A\breve{\mathbf{u}} \in N_{\mathbb{R}^m_+}(\breve{\lambda}). \end{aligned} \tag{4.13}$$

Note that the fourth line of (4.13) is a vectorized form of the first line of (4.12). Accordingly, the point satisfies the vKKT conditions, quantified by the approximate pseudogradient (4.11). Intuitively, this means an OA vGNE is a point which would be a vKKT point if the model of the Jacobian were locally accurate, and would further be a vGNE if $g \circ \pi$ were convex in u_i (Remark 3). Additionally, the approximation resembles a vGNE in that if each agent believes the nominal model of the input-output sensitivity, no agent has any incentive to deviate. Note the dependence of the optimum $\breve{\mathbf{u}}$ on the disturbance w; the larger the impact of the latter, the further the distance of the OA vGNE from a true vKKT point of the original game. We formalize the error between the OA vGNE and a true vKKT point later in this chapter.

Following the logic in Section 3.2, we can compactly express the KKT conditions (4.12) as $\mathrm{col}(\breve{\mathbf{u}}, \breve{\lambda}) \in \mathrm{zer}(\mathfrak{A} + \mathfrak{B})$ where

$$\begin{aligned} &\mathfrak{A} : \mathrm{col}(\mathbf{u}, \lambda) \mapsto \mathrm{col}(H_{w,\Pi}(\mathbf{u}), -b) \\ &\mathfrak{B} : \mathrm{col}(\mathbf{u}, \lambda) \mapsto \mathrm{col}(-A^\top \lambda + N_\Omega(\mathbf{u}), A\mathbf{u} + N_{\mathbb{R}^m_+}(\lambda)). \end{aligned} \tag{4.14}$$

With this, we now introduce our OA vGNE seeking algorithm extended from Algorithm 1.

4.3 Online Approximate vGNE-seeking Algorithm

With the approximated Jacobian, our game has KKT conditions of the same form as discussed in Section 3.1. We thus need an equivalent of Assumption 2 on the pseudogradient operator (4.11).

Assumption 5. *The pseudogradient $H_{w,\Pi}(\mathbf{u})$ defined in* (4.11) *is $\bar{\eta}$-strongly monotone and $\bar{\theta}$-Lipschitz continuous.*

The operator $H_{w,\Pi}$, specifically through the $\partial_{\mathbf{y}} g(\mathbf{y})$ embedded in it, has a dependence on an unknown disturbance w. To verify the monotoncity and Lipschitz continuity of this operator, we can treat it as an uncertain operator, and utilize a semidefinite programming approach to verify

the conditions. We describe and prove this approach in Chapter 5. For now, we assume that these prerequisites are verifiable, and present the algorithm.

We now outline the notations used in the algorithm. As in Algorithm 1, we assume that 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 $\nabla_{y_j} g_i$. Each player additionally knows $\Pi_{i1}, \ldots, \Pi_{iN}$, which are submatrices of the nominal sensitivity matrix Π. All of the remaining notation is identical to that outlined for Algorithm 1, except that we assume that the interference graph $\mathcal{G}_f$ also communicates the outputs y_j, $j \in \mathcal{N}_{\mathcal{G}_f}(i)$ for each agent i. Figure 4.2 depicts the decentralized nature of the algorithm.

Algorithm 2 Distributed Online Approximate vGNE-seeking algorithm

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

Iteration: Player i

Step 1: Approximate Primal Step and Consensus - Receives $u_{j,k}$, $y_{j,k}$, $j \in \mathcal{N}_{\mathcal{G}_f}(i)$, $\lambda_{j,k}$, $j \in \mathcal{N}_{\mathcal{G}_\lambda}(i)$, $y_{i,k}$ and updates:

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

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

Step 2: Consensus and Dual Step - Receives $z_{j,k+1}$, $j \in \mathcal{N}_{\mathcal{G}_\lambda}(i)$ and updates:

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

Remark 4. *The algorithm presented in [19] and this thesis make the assumption that $\mathcal{G}_f$ has an edge for each cost dependence. This assumption is adopted in [48, 49, 53] as well. However, if there is too much dependence between agents' cost functions, this graph becomes very large, and could even be a complete graph as in [54]. Distributed GNE-seeking with only partial or incomplete information obtained from the interference graph is an area of future research [20]. In this thesis, we only consider the impact of the global coupling constraints being optimized via local knowledge of the constraints' structure as well as partial information on the neighbours' estimates, hence the assumption on $\mathcal{G}_\lambda$ being connected rather than complete.*

The main difference between Algorithm 2 and Algorithm 1 is the impact of the output mapping on the update. At any given update step, the agent is using a measurement to calculate the cost function penalizing it for disobeying its output, and using the nominal Jacobian to quantify each agents' contribution to any changes in that measurement. For simplicity, we can assume that the measurement itself is accurate, however the Jacobian is inexact, since we initially assumed that the output mapping π can be nonlinear in the agents' actions. Further, the impact of the disturbance on the output mapping is not known to any agent. From these sources of uncertainty, we would intuitively expect the quality of the approximation to be dependent on the relation between the approximated Jacobian and the "true" Jacobian of the system (computed at the optimum). Additionally, the more heavily this Jacobian varies with the unknown disturbance, the worse we expect the quality of the approximation to be. The following proposition encodes that mathematically.

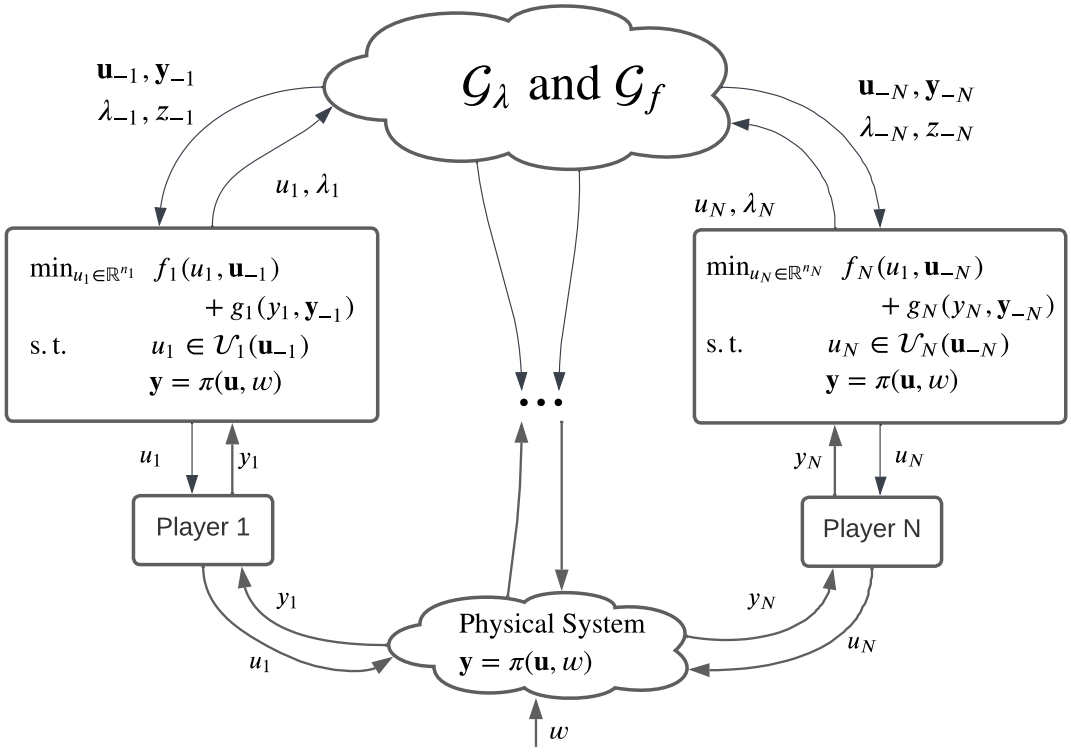

Figure 4.2: Depiction of online feedback-based vGNE-seeking algorithm. Each agent communicates with each other agent through the communication and interference graphs, and locally updates its decision.

Proposition 4.1. *If $H_{w,\Pi}$, defined in* (4.11)*, is η-strongly monotone, then*

$$\|\breve{\mathbf{u}} - \mathbf{u}^*\| \leq \frac{1}{\eta} \left\| \mathfrak{E} \left([\Pi - \partial_{\mathbf{u}} \pi(\breve{\mathbf{u}}, w)]^\top \partial_{\mathbf{y}} g(\pi(\breve{\mathbf{u}}, w))^\top \right) \right\|$$

where $\breve{\mathbf{u}}$ is the unique OA vGNE as defined in Definition 4.1, and $\mathbf{u}^*$ is a point satisfying the vKKT conditions (4.9)*.*

Proof. We note that $\breve{\mathbf{u}}$ solves the variational inequality $\mathrm{VI}(H_{w,\Pi}, \mathcal{U})$ and $\mathbf{u}^*$ solves the variational inequality $\mathrm{VI}(H_w, \mathcal{U})$. Thus, the above follows from Theorem 1.14 in [55]. □

4.3.1 Forward-Backward Algorithm

In this section we show that a limiting point of Algorithm 2 corresponds to an OA vGNE as defined in Definition 4.1. The proof is a modification of the proof in [19], which we provided an abridged, intuitive version of in Section 3.2.1.

We first follow the notation defined in Section 3.2.1 to write the algorithm in a compact manner:

$$\mathbf{u}_{k+1} = P_{\Omega}[\mathbf{u}_k - \bar{\tau} H_{w,\Pi}(\mathbf{u}_k) - \Lambda^\top \bar{\lambda}_k)], \tag{4.15a}$$
$$\bar{z}_{k+1} = \bar{z}_k + \bar{\nu} \bar{L} \bar{\lambda}_k, \tag{4.15b}$$
$$\bar{\lambda}_{k+1} = P_{\mathbb{R}_+^{mN}}\{\bar{\lambda}_k - \bar{\sigma}[\Lambda(2\mathbf{u}_{k+1} - \mathbf{u}_k) - \bar{b} + \bar{L}\bar{\lambda}_k + \bar{L}(2\bar{z}_{k+1} - \bar{z}_k)]\}. \tag{4.15c}$$

Following the same arguments as (3.8) and (3.9),

$$-H_{w,\Pi}(\mathbf{u}) \in N_{\Omega}(\mathbf{u}_{k+1}) - \Lambda^\top \bar{\lambda}_{k+1} + \bar{\tau}^{-1}(\mathbf{u}_{k+1} - \mathbf{u}_k) + \Lambda^\top(\bar{\lambda}_{k+1} - \bar{\lambda}_k) \tag{4.16a}$$
$$-[\bar{L}\bar{\lambda}_k - \bar{b}] \in N_{\mathbb{R}_+^{mN}}(\bar{\lambda}_{k+1}) + \Lambda \mathbf{u}_{k+1} + \bar{L}\bar{z}_{k+1} + \Lambda(\mathbf{u}_{k+1} - \mathbf{u}_k) + \bar{L}(\bar{z}_{k+1} - \bar{z}_k) + \bar{\sigma}^{-1}(\bar{\lambda}_{k+1} - \bar{\lambda}_k) \tag{4.16b}$$

As in (3.10), we define the symmetric matrix Φ and skew-symmetric matrix Ψ:

$$\Phi = \begin{bmatrix} \bar{\tau}^{-1} & \mathbb{0} & \Lambda^\top \\ \mathbb{0} & \bar{\nu}^{-1} & \bar{L} \\ \Lambda & \bar{L} & \bar{\sigma}^{-1} \end{bmatrix}, \; \Psi = \begin{bmatrix} \mathbb{0} & \mathbb{0} & -\Lambda^\top \\ \mathbb{0} & \mathbb{0} & -\bar{L} \\ \Lambda & \bar{L} & \mathbb{0} \end{bmatrix}. \tag{4.17}$$

Denote $\varpi = \mathrm{col}(\mathbf{u}, \bar{z}, \bar{\lambda})$. Then (4.15b), (4.16a), (4.16b) can be expressed as

$$-\bar{\mathfrak{A}}(\varpi_k) \in \bar{\mathfrak{B}}(\varpi_{k+1}) + \Phi(\varpi_{k+1} - \varpi_k), \tag{4.18}$$

with

$$\begin{aligned} \bar{\mathfrak{A}} &: \varpi \mapsto \mathrm{col}(H_{w,\Pi}(\mathbf{u}), \mathbf{0}, \bar{L}\bar{\lambda} - \bar{b}) \\ \bar{\mathfrak{B}} &: \varpi \mapsto N_{\Omega}(\mathbf{u}) \times \mathbf{0} \times N_{\mathbb{R}_+^{mN}}(\bar{\lambda}) + \Psi \varpi. \end{aligned} \tag{4.19}$$

As noted in Section 3.2.1 $\bar{\mathfrak{A}}$ and $\bar{\mathfrak{B}}$ can be viewed as extensions of the operators $\mathfrak{A}$ and $\mathfrak{B}$ described in (4.14), augmenting $\lambda \in \mathbb{R}^m$ to $\bar{\lambda} \in \mathbb{R}^{mN}$ and introducing auxiliary variables $\bar{z} \in \mathbb{R}^{mN}$. The local auxiliary variables z_i can be viewed as each player i estimating the contribution of the other players to the constraint, helping players reach consensus over λ_i.

The formulation in (4.18) is a *forward-backward splitting method* [28], as we outlined in Section 3.2.1, using the metric matrix Φ in a similar manner, to enable distributed evaluation of the backward step. We relate the zeroes of $\bar{\mathfrak{A}}+\bar{\mathfrak{B}}$ and $\mathfrak{A}+\mathfrak{B}$ to a an OA vGNE of the game, as defined in Definition 4.1.

Theorem 4.1. *Suppose that Assumptions 1 and 3-5 hold. Consider the operators $\bar{\mathfrak{A}}$ and $\bar{\mathfrak{B}}$ in* (4.19)*, and $\mathfrak{A}$ and $\mathfrak{B}$ in* (4.14)*. Then the following statements hold:*

1. *Given a $\mathrm{col}(\breve{\mathbf{u}}, \breve{\bar{z}}, \breve{\bar{\lambda}}) \in \mathrm{zer}(\bar{\mathfrak{A}}+\bar{\mathfrak{B}})$, $\breve{\mathbf{u}}$ solves the variational inequality $\mathrm{VI}(H_{w,\Pi}, \mathbf{u})$, thus $\breve{\mathbf{u}}$ is an online approximate variational GNE of the game, as per Definition 4.1. Further, $\mathrm{col}(\breve{\mathbf{u}}, \breve{\lambda}) \in \mathrm{zer}(\mathfrak{A}+\mathfrak{B})$, i.e., $\breve{\mathbf{u}}$ and $\breve{\lambda}_1 = \cdots = \breve{\lambda}_N = \breve{\lambda}$ satisfy the KKT conditions* (4.12)*.*
2. *$\mathrm{zer}(\mathfrak{A}+\mathfrak{B}) \neq \emptyset$ and $\mathrm{zer}(\bar{\mathfrak{A}}+\bar{\mathfrak{B}}) \neq \emptyset$.*

Proof. This proof is a modification of the proof of Theorem 2 in [19].

1. Suppose that $\breve{\mathbf{u}}, \breve{\bar{z}}, \breve{\bar{\lambda}} \in \mathrm{zer}(\bar{\mathfrak{A}}+\bar{\mathfrak{B}})$. By the definition in (4.19) we can add the two operators

and obtain

$$\begin{bmatrix} \mathbb{0} & \mathbb{0} & -\Lambda^\top \\ \mathbb{0} & \mathbb{0} & -\bar{L} \\ \Lambda & \bar{L} & \bar{L} \end{bmatrix} \begin{bmatrix} \breve{\mathbf{u}} \\ \breve{\breve{z}} \\ \breve{\breve{\lambda}} \end{bmatrix} + \begin{bmatrix} H_{w,\Pi}(\breve{\mathbf{u}}) + N_\Omega(\breve{\mathbf{u}}) \\ \mathbf{0} \\ N_{\mathbb{R}_+^{mN}}(\breve{\breve{\lambda}}) - \bar{b} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{4.20}$$

with $H_{w,\Pi}$ as defined in (4.11). From the second line of (4.20) and the definition of $\bar{L}$ in Section 3.2.1, we have $-\bar{L}\breve{\breve{\lambda}} = -L \otimes I_m \breve{\breve{\lambda}} = \mathbf{0}_{mN}$. By Assumption 3 L is the weighted Laplacian of the connected graph $\mathcal{G}_\lambda$, thus $\breve{\breve{\lambda}} = \mathbf{1}_N \otimes \breve{\lambda},\ \breve{\lambda} \in \mathbb{R}^m$.

Then, combining the first line of (4.20) with the definitions $\Lambda^\top = \operatorname{diag}(A_1^\top,\ \dots\ , A_N^\top)$ and $A = [\, A_1\ \cdots\ A_N\,]$, and with $\breve{\lambda}_1 = \ \cdots\ = \breve{\lambda}_N = \breve{\lambda}$, we have $\mathbf{0}_n \in -A^\top \breve{\lambda} + H_{w,\Pi}(\breve{\mathbf{u}}) + N_\Omega(\breve{\mathbf{u}})$. Equivalently, we can write this per agent as

$$\mathbf{0}_{n_i} \in -A_i^\top \breve{\lambda} + \nabla_{u_i} h_{i,w,\Pi}(\breve{u}_i, \breve{\mathbf{u}}_{-i}) + N_{\Omega_i}(\breve{u}_i),\ i \in \mathcal{N}, \tag{4.21}$$

where each $h_{i,w,\Pi}$ denotes agent i's cost function, defined as

$$h_{i,w,\Pi}(u_i, \mathbf{u}_{-i}) = f_i(u_i, \mathbf{u}_{-i}) + g_i(\pi_i(u_i, \mathbf{u}_{-i}), \pi_{-i}(u_i, \mathbf{u}_{-i})),$$

and the subscript Π denotes that when applying the chain rule in the second term while taking the gradient, we use the nominal Jacobian. Given $-\bar{L}\breve{\breve{\lambda}} = \mathbf{0}_{mN}$ from the second line of (4.20), the third line simplifies to

$$\mathbf{0}_{mN} \in \Lambda \breve{\mathbf{u}} - \bar{b} + \bar{L}\breve{\breve{z}} + N_{\mathbb{R}_+^{mN}}(\breve{\breve{\lambda}}).$$

This implies that there exist vectors $v_1,\ \dots\ , v_N \in N_{\mathbb{R}_+^m}(\breve{\breve{\lambda}})$ such that

$$\mathbb{0}_{mN} = \Lambda \breve{\mathbf{u}} - \bar{b} + (L \otimes I_m)\breve{\breve{z}} + \operatorname{col}(v_1,\ \dots\ , v_N).$$

Multiplying both sides of the equation by $\mathbf{1}_N^\top \otimes I_m$ and noting that $\mathbf{1}^\top L = \mathbb{0}^\top$ we obtain

$$\begin{aligned} \mathbf{0}_m &= (\mathbf{1}_N^\top \otimes I_m)(\Lambda \breve{\mathbf{u}} - \bar{b} + (L \otimes I_m)\breve{\breve{z}} + \operatorname{col}(v_1,\ \dots\ , v_N)) \\ &= \sum_{i \in \mathcal{N}} A_i \breve{u}_i - \sum_{i \in \mathcal{N}} b_i + \sum_{i \in \mathcal{N}} v_i \\ \mathbf{0}_m &\in \sum_{i \in \mathcal{N}} A_i \breve{u}_i - \sum_{i \in \mathcal{N}} b_i + \sum_{i \in \mathcal{N}} N_{\mathbb{R}_+^m}(\breve{\lambda}) \end{aligned}$$

By Corollary 16.39 of [28], if $\operatorname{int}(\Omega_i) \neq \emptyset$, $\sum_{i \in \mathcal{N}} N_{\Omega_i} = N_{\cap_{i \in \mathcal{N}} \Omega_i}$, thus

$$\begin{aligned} \mathbf{0}_m &\in \sum_{i \in \mathcal{N}} A_i \breve{u}_i - \sum_{i \in \mathcal{N}} b_i + N_{\cap_{i \in \mathcal{N}} \mathbb{R}_+^m}(\breve{\lambda}) \\ &\in \sum_{i \in \mathcal{N}} A_i \breve{u}_i - \sum_{i \in \mathcal{N}} b_i + N_{\mathbb{R}_+^m}(\breve{\lambda}) \end{aligned} \tag{4.22}$$

Thus, by (4.21) and (4.22), a point $\operatorname{col}(\breve{\mathbf{u}}, \breve{\breve{\lambda}}, \breve{\breve{z}}) \in \operatorname{zer}(\bar{\mathfrak{A}} + \bar{\mathfrak{B}})$ satisfies the VI KKT conditions (4.12) for $\breve{\mathbf{u}}$ with $\breve{\lambda}$. Thus, by Theorem 3.1, $\breve{\mathbf{u}}$ is an online approximate variational GNE of the game, as defined in Definition 4.1. Note further that $\operatorname{col}(\breve{\mathbf{u}}, \breve{\lambda}) \in \operatorname{zer}(\mathfrak{A} + \mathfrak{B})$.

2. Under Assumptions 1, 4 and 5, the game has a unique OA vGNE $\breve{\mathbf{u}}$, and there exists a bounded $\breve{\lambda}$ such that the vKKT conditions (4.12) are satisfied, i.e., $\text{col}(\breve{\mathbf{u}}, \breve{\lambda}) \in \text{zer}(\mathfrak{A} + \mathfrak{B})$, defined in (4.14) [56]. Therefore $\text{zer}(\mathfrak{A} + \mathfrak{B}) \neq \emptyset$. We similarly need to show that there exists $\text{col}(\breve{\mathbf{u}}, \breve{\breve{\lambda}}, \breve{\breve{z}})$ such that (4.20) holds.

   Consider $\breve{\breve{\lambda}} = \mathbf{1}_N \otimes \breve{\lambda}$. Since $L\mathbf{1}_N = 0$, the second line of (4.20) is satisfied. From $\text{col}(\breve{\mathbf{u}}, \breve{\lambda}) \in \text{zer}(\mathfrak{A} + \mathfrak{B})$ we know that $\mathbf{0} \in H_{w,\Pi}(\breve{\mathbf{u}}) - A^\top \breve{\lambda} + N_\Omega(\breve{\mathbf{u}})$. Using $\breve{\lambda}_1 = \ \cdots \ = \breve{\lambda}_N = \breve{\lambda}$ and $\Lambda^\top \breve{\breve{\lambda}} = A^\top \breve{\lambda}$, we conclude that the first line of (4.20) is satisfied.

   Finally, we need to show that there exists $\breve{\breve{z}} \in \mathbb{R}^{mN}$ such that the third line of (4.20) is satisfied. Consider a $v^* \in N_{\mathbb{R}^{mN}_+}(\breve{\lambda})$ such that $\mathbf{0} = A\breve{\mathbf{u}} - b + v^*$. Since $\breve{\breve{\lambda}} = \mathbf{1}_N \otimes \breve{\lambda}$ and $N_{\mathbb{R}^{mN}_+}(\breve{\breve{\lambda}}) = \prod_{i\in\mathcal{N}} N_{\mathbb{R}^m_+}(\breve{\lambda})$, take $v_1^* = \ \cdots \ = v_N^* = \frac{1}{N} v^* \in N_{\mathbb{R}^m_+}$. Then $(\mathbf{1}_N^\top \otimes I_m)(\Lambda\breve{\mathbf{u}} - \bar{b} + \bar{L}\breve{\breve{\lambda}} + \text{col}(v_1^*, \ \ldots \ , v_N^*) = \sum_{i\in\mathcal{N}} A_i u_i^* + \sum_{i\in\mathcal{N}} b_i + v^* = A\breve{\mathbf{u}} - b + v^* = \mathbf{0}_m$. Thus $\Lambda\breve{\mathbf{u}} - b + \bar{L}\breve{\breve{\lambda}} + \text{col}(v_1^*, \ \ldots \ , v_N^*) \in \text{Null}(\mathbf{1}_N^\top \otimes I_m)$. By the fundamental theorem of linear algebra, we know that $\text{Null}(\mathbf{1}_N^\top \otimes I_m) = \text{Range}(\mathbf{1}_N \otimes I_m)^\perp$ and similarly, since $\bar{L}$ is symmetric, $\text{Null}(\bar{L}) = \text{Range}(\bar{L})^\perp$. Notice that, since $\bar{L}(\mathbf{1}_N \otimes \breve{\lambda}) = \mathbf{0}_{mN}$, $\text{Null}(\bar{L}) = \text{Range}(\mathbf{1}_N \otimes I_m)$. Hence $\Lambda\breve{\mathbf{u}} - \bar{b} + \bar{L}\breve{\breve{\lambda}} + \text{col}(v_1^*, \ \ldots \ , v_N^*) \in \text{Range}(\bar{L})$. Since $\text{col}(v_1^*, \ \ldots \ , v_N^*) \in N_{\mathbb{R}^{mN}_+}(\breve{\breve{\lambda}})$, there exists $\breve{\breve{z}} \in \mathbb{R}^{mN}$ such that the third line of (4.20) is satisfied. Thus $\text{zer}(\bar{\mathfrak{A}} + \bar{\mathfrak{B}}) \neq \emptyset$.

□

Having shown that the zeroes of the sum of the augmented operators, we now show that the algorithm can be viewed as a forward-backward splitting method for finding zeros.

Lemma 4.1. *Suppose that Assumptions 1, and 3-5 hold. Suppose that* Φ *in* (4.17) *is positive definite, and the operators* $\bar{\mathfrak{A}}$ *and* $\bar{\mathfrak{B}}$ *in* (4.19) *are maximally monotone. Suppose* $\Phi^{-1}\bar{\mathfrak{A}}$ *and* $\Phi^{-1}\bar{\mathfrak{B}}$ *are maximally monotone. Denote* $T_1 = \text{Id} - \Phi^{-1}\bar{\mathfrak{A}}$ *and* $T_1 = (\text{Id} + \Phi^{-1}\bar{\mathfrak{A}})^{-1}$ *Then any limiting point,* $\text{col}(\breve{\mathbf{u}}, \breve{\breve{z}}, \breve{\breve{\lambda}})$*, of Algorithm 2 or* (4.18) *is a zero of* $\bar{\mathfrak{A}} + \bar{\mathfrak{B}}$ *and a fixed point of* $T_2 \circ T_1$

Proof. This proof follows from the proof for Lemma 1 in [19], applied to our modified definitions for $\bar{\mathfrak{A}}$ and $\bar{\mathfrak{B}}$. Consider the compact form (4.18) of Algorithm 2. Since Φ is symmetric and positive definite, we can rewrite (4.18) as

$$\begin{aligned} &\varpi_k - \Phi^{-1}\bar{\mathfrak{A}}(\varpi_k) \in \varpi_{k+1} + \Phi^{-1}\bar{\mathfrak{B}}(\varpi_{k+1}) \\ &(\text{Id} - \Phi^{-1}\bar{\mathfrak{A}})(\varpi_k) \in (\text{Id} + \Phi^{-1}\bar{\mathfrak{B}})(\varpi_{k+1}) \end{aligned} \tag{4.23}$$

Since $\Phi^{-1}\bar{\mathfrak{B}}$ is maximally monotone, $(\text{Id} + \Phi^{-1}\bar{\mathfrak{B}})^{-1}$ is a single-valued operator (Lemma 2.3). Thus we can rewrite (4.23) as

$$\varpi_{k+1} = (\text{Id} + \Phi^{-1}\bar{\mathfrak{B}})^{-1}(\text{Id} - \Phi^{-1}\bar{\mathfrak{A}})(\varpi_k). \tag{4.24}$$

We denote $T_2 := (\text{Id} + \Phi^{-1}\bar{\mathfrak{B}})^{-1}$ and $T_1 := (\text{Id} - \Phi^{-1}\bar{\mathfrak{A}})$. Suppose that Algorithm 2 (i.e., (4.18) or (4.24)) converges to a limiting point $\breve{\varpi}$. Then from (4.18) we conclude that $\breve{\varpi} \in \text{zer}(\bar{\mathfrak{A}} + \bar{\mathfrak{B}})$. It also follows that $\breve{\varpi} = T_2 \circ T_1 \breve{\varpi}$. Thus any limiting point $\breve{\varpi}$ of Algorithm 1 is a zero of $\bar{\mathfrak{A}} + \bar{\mathfrak{B}}$ and a fixed point of $T_2 \circ T_1$. □

As in (3.13), the iteration (4.24) is Picard's iteration for computing fixed points of $T_2 \circ T_1$. Algorithm 2 is thus a forward-backward operator splitting algorithm, similar to Algorithm 1. Convergence of the algorithm to a vKKT point of the game thus requires showing that the prerequisites for Lemma 4.1 are satisfied.

4.3.2 Convergence Analysis

This section focuses on showing that the prerequisites for Lemma 4.1 can be satisfied for suitable step sizes, and then prove the convergence of Algorithm 2 based on its treatment as a fixed point iteration, outlined in the previous section. We begin by outlining some preliminary lemmas that support the proof of the convergence. These results are taken from [19], and their proofs can be seen in detail there. We present the proof for Lemma 4.4, because the steps taken in the proof are relevant to a proof we do later in the thesis. We omit the proofs for Lemmas 4.2 and 4.3.

Lemma 4.2. *Suppose that Assumptions 1 and 3-5 hold. Then operators $\bar{\mathfrak{A}}$ and $\bar{\mathfrak{B}}$, w.r.t. to the Euclidean norm $\|\cdot\|_2$, satisfy:*

1. *$\bar{\mathfrak{A}}$ is β-cocoercive with $\beta \in \left(0, \min\left\{\frac{1}{2d^*}, \frac{\bar{\eta}}{\bar{\theta}^2}\right\}\right)$, 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.*
2. *$\bar{\mathfrak{B}}$ is maximally monotone.*

Lemma 4.3. *Given any $\delta > 0$, if each player chooses fixed step sizes τ_i, ν_i, σ_i in Algorithm 2 satisfying:*

$$0 < \tau_i \leq \left(\max_{j=1,\dots,n_i} \left\{ \sum_{k=1}^{m} |[A_i^\top]_{jk}| \right\} + \delta \right)^{-1}, \tag{4.25}$$

$$0 < \nu_i \leq \left(\max_{j=1,\dots,m} \left\{ \sum_{k=1}^{n_i} |[A_i]_{jk}| \right\} + 2d_i\delta \right)^{-1}, \tag{4.26}$$

$$0 < \tau_i \leq (2d_i + \delta)^{-1}, \tag{4.27}$$

then Φ in (4.17) is positive definite, and $\Phi - \delta I_{n+2mN}$ is positive semi-definite.

Lemma 4.4. *Suppose Assumptions 1 and 3-5 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 $\delta > \frac{1}{2\beta}$. Suppose that τ_i, ν_i, σ_i are chosen to satisfy (4.25). Then the operators $\Phi^{-1}\bar{\mathfrak{A}}$ and $\Phi^{-1}\bar{\mathfrak{B}}$ defined as in (4.17) and (4.19) and the operators T_1, T_2 defined as in Lemma 4.1 satisfy the following properties w.r.t. to the induced norm $\|\cdot\|_\Phi$:*

1. *$\Phi^{-1}\bar{\mathfrak{A}}$ is $\beta\delta$-cocercive and $T_1 \in \mathcal{A}\left(\frac{1}{2\delta\beta}\right)$;*
2. *$\Phi^{-1}\bar{\mathfrak{B}}$ is maximally monotone and $T_2 \in \mathcal{A}\left(\frac{1}{2}\right)$.*

Proof. 1. By the definition of cocoercivity in (2.1), we need to prove $\langle \Phi^{-1}\bar{\mathfrak{A}}(x) - \Phi^{-1}\bar{\mathfrak{A}}(y), x - y\rangle_\Phi \geq \beta\delta\|\Phi^{-1}\bar{\mathfrak{A}}(x) - \Phi^{-1}\bar{\mathfrak{A}}(y)\|_\Phi^2$, $\forall x, y \in \bar{\Omega}$. Given the choice of the step sizes, we know from Lemma 4.3 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 $\sigma_{\max}(\Phi) \geq \sigma_{\min}(\Phi) \geq \delta$. Further, since $\|\Phi\|_2 = \sigma_{\max}(\Phi)$, $\frac{1}{\|\Phi^{-1}\|_2} = \sigma_{\min}(\Phi)$, we know that $\|\Phi^{-1}\|_2 \leq \frac{1}{\delta}$. Since $\bar{\mathfrak{A}}$ is single-valued

and Φ^{-1} is nonsingular, we know that for any $x, y \in \bar{\Omega}$,

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

Combining this with Lemma 4.2 ($\bar{\mathfrak{A}}$ is β-cocoercive), we get:

$$\begin{aligned}
& \langle \Phi^{-1}\bar{\mathfrak{A}}(x) - \Phi^{-1}\bar{\mathfrak{A}}(y), x - y\rangle_\Phi \\
= \; & \left[\Phi^{-1}\left(\bar{\mathfrak{A}}(x) - \bar{\mathfrak{A}}(y)\right)\right]^\top \Phi\,(x - y) \\
= \; & \left[\bar{\mathfrak{A}}(x) - \bar{\mathfrak{A}}(y)\right]^\top (x - y) \\
= \; & \langle \bar{\mathfrak{A}}(x) - \bar{\mathfrak{A}}(y), x - y\rangle \\
\geq \; & \beta\|\bar{\mathfrak{A}}(x) - \bar{\mathfrak{A}}(y)\|_2^2 \\
\geq \; & \beta\delta\|\Phi^{-1}\bar{\mathfrak{A}}(x) - \Phi^{-1}\bar{\mathfrak{A}}(y)\|_\Phi^2.
\end{aligned}$$

Thus $\Phi^{-1}\bar{\mathfrak{A}}$ is $\beta\delta$-cocoercive under norm $\|\cdot\|_\Phi$, and thus $\beta\delta\Phi^{-1}\bar{\mathfrak{A}}$ is firmly nonexpansive by (2.1). By the definition of nonexpansive operators (Definition 2.14), there exists a nonexpansive $\hat{T}$ such that $\beta\delta\Phi^{-1}\bar{\mathfrak{A}} = \frac{1}{2}\hat{T} + \frac{1}{2}\,\mathrm{Id}$. Then

$$\begin{aligned}
T_1 &= \mathrm{Id} - \Phi^{-1}\bar{\mathfrak{A}} \\
&= \left(1 - \frac{1}{2\beta\delta}\right)\mathrm{Id} + \frac{1}{2\beta\delta}(-\hat{T}).
\end{aligned}$$

Since $\delta > \frac{1}{2\beta} \Rightarrow 1 < 2\beta\delta$, and $-\hat{T}$ nonexpansive, this implies $T_1 \in \mathcal{A}\left(\frac{1}{2\beta\delta}\right)$ by Definition 2.14.

2. Φ is positive definite and nonsingular. For any $(x, u) \in \operatorname{gra}\Phi^{-1}\bar{\mathfrak{B}}$ and $(y, v) \in \operatorname{gra}\Phi^{-1}\bar{\mathfrak{B}}$, we know $\Phi u \in \Phi\Phi^{-1}\bar{\mathfrak{B}}(x) \in \bar{\mathfrak{B}}(x)$ and $\Phi v \in \Phi\Phi^{-1}\bar{\mathfrak{B}}(y) \in \bar{\mathfrak{B}}(y)$. Then

$$\begin{aligned}
& \langle x - y, u - v\rangle_\Phi \\
= \; & \langle x - y, \Phi(u - v)\rangle_2 \\
\geq \; & 0, \;\; \forall x, y \in \operatorname{dom}\mathfrak{B},
\end{aligned}$$

because $\bar{\mathfrak{B}}$ is monotone under $\|\cdot\|_2$ by Lemma 4.2. Thus $\Phi^{-1}\bar{\mathfrak{B}}$ is monotone under $\|\cdot\|_\Phi$.

Take $(y, v) \in \bar{\Omega} \times \mathbb{R}^{n+2mN}$ and $\langle x - y, u - v\rangle_\Phi \geq 0$ for any other $(x, u) \in \operatorname{gra}\Phi^{-1}\bar{\mathfrak{B}}$. For any $(x, \tilde{u}) \in \operatorname{gra}\bar{\mathfrak{B}}$, we have $(x, \Phi^{-1}\tilde{u}) \in \operatorname{gra}\phi^{-1}\bar{\mathfrak{B}}$. Thus $\langle x - y, \Phi(\Phi^{-1}\tilde{u} - v) \geq 0\rangle$, or equivalently, $\langle x - y, \tilde{u} - \Phi v\rangle \geq 0$. Since $\bar{\mathfrak{B}}$ is maximally monotone by Lemma 4.2, $(y, \Phi v) \in \operatorname{gra}\bar{\mathfrak{B}}$. Thus $v \in \Phi^{-1}\bar{\mathfrak{B}}(y)$, and $\Phi^{-1}\bar{\mathfrak{B}}$ is maximally monotone. By Lemma 2.5, $T_2 = (\mathrm{Id} + \Phi^{-1}\bar{\mathfrak{B}})^{-1}$ is firmly nonexpansive w.r.t. $\|\cdot\|_\Phi$.

□

Theorem 4.2. *Suppose Assumptions 1 and 3-5 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* $\delta > \frac{1}{2\beta}$*. Suppose that* τ_i*,* ν_i*,* σ_i *are chosen to satisfy* (4.25)*. Then with Algorithm 2, each player's local strategy* $u_{i,k}$ *converges to its corresponding component of an Online Approximate variational GNE as defined in 4.1, and their local multipliers* $\lambda_{i,k}$ *converge to the multiplier corresponding to the vKKT conditions* (4.12)*, i.e.,* $\lim_{k\to\infty} u_{i,k} = \breve{u}_i$ *and* $\lim_{k\to\infty} = \breve{\lambda}$*,* $i \in \mathcal{N}$*.*

Proof. This proof is an adaptation of the proof for Theorem 3 in [19]. Following from Lemmas 4.2 and 4.3, we can use Lemma 4.1 to rewrite Algorithm 2 as (4.24), so $\varpi_{k+1} = T_2 \circ T_1 \varpi_k$, and any limiting point is also a fixed point of $T_2 \circ T_1$. Note that, under Lemmas 2.4 and 4.4, T_1, T_2 are averaged and nonexpansive operators under $\|\cdot\|_\Phi$. Take any zero $\breve{\varpi} \in \operatorname{zer}(\bar{\mathfrak{A}} + \bar{\mathfrak{B}})$, i.e., any fixed point $\breve{\varpi} = T_2 T_1 \breve{\varpi}$. Then, following from Lemma 4.1 and (4.24),

$$\|\varpi_{k+1} - \breve{\varpi}\|_\Phi = \|T_2 T_1 \varpi_k - T_2 T_1 \breve{\varpi}\|_\Phi \leq \|T_1 \varpi_k - T_1 \breve{\varpi}\|_\Phi \leq \|\varpi_k - \breve{\varpi}\|_\Phi. \tag{4.28}$$

Thus the sequence $\{\|\varpi_k - \breve{\varpi}\|_\Phi\}$ is nonincreasing and bounded from below, and it is bounded and converges for any $\breve{\varpi} \in \operatorname{zer}(\bar{\mathfrak{A}} + \bar{\mathfrak{B}})$. By Lemma 4.4, $T_1 \in \mathcal{A}(\alpha)$, $T_2 \in \mathcal{A}\left(\frac{1}{2}\right)$, where $\alpha = \frac{1}{2\beta\delta} \in (0, 1)$. By applying Lemma 2.4 (i) to $T_2 \in \mathcal{A}(\frac{1}{2})$ in the equality in (4.28), we have

$$\begin{aligned} \|\varpi_{k+1} - \breve{\varpi}\|_\Phi^2 &\leq \|T_1 \varpi_k - T_1 \breve{\varpi}\|_\Phi^2 - \|(T_1 \varpi_k - T_1 \breve{\varpi}) - (T_2 T_1 \varpi_k - T_2 T_1 \breve{\varpi})\|_\Phi^2 \\ &\leq \|\varpi_k - \breve{\varpi}\|_\Phi^2 - \|(T_1 \varpi_k - T_1 \breve{\varpi}) - (T_2 T_1 \varpi_k - T_2 T_1 \breve{\varpi})\|_\Phi^2 \\ &\qquad - \frac{1-\alpha}{\alpha}\|(\varpi_k - \breve{\varpi}) - (T_1 \varpi_k - T_1 \breve{\varpi})\|_\Phi^2, \end{aligned} \tag{4.29}$$

where the second step follows from applying Lemma 2.4 (i) to $T_1 \in \mathcal{A}(\alpha)$ in the right side of the first step. Denote the sum of the last two terms as

$$\Xi := \|(T_1 \varpi_k - T_1 \breve{\varpi}) - (T_2 T_1 \varpi_k - T_2 T_1 \breve{\varpi})\|_\Phi^2 + \frac{1-\alpha}{\alpha}\|(\varpi_k - \breve{\varpi}) - (T_1 \varpi_k - T_1 \breve{\varpi})\|_\Phi^2,$$

Letting

$$\begin{aligned} x &:= (T_1 \varpi_k - T_1 \breve{\varpi}) - (T_2 T_1 \varpi_k - T_2 T_1 \breve{\varpi}), \text{and} \\ y &:= (\varpi_k - \breve{\varpi}) - (T_1 \varpi_k - T_1 \breve{\varpi}), \end{aligned}$$

it follows from $\alpha\|x\|^2 + (1-\alpha)\|y\|^2 \geq \alpha(1-\alpha)\|x - y\|^2$ that

$$\Xi \geq (1-\alpha)\|(\varpi_k - \breve{\varpi}) - (T_2 T_1 \varpi_k - T_2 T_1 \breve{\varpi})\|_\Phi^2 = (1-\alpha)\|\varpi_k - \varpi_{k+1}\|_\Phi^2. \tag{4.30}$$

Note that in both (4.29) and (4.30), the term Ξ occurs on the greater side. We add both inequalities to obtain, for all $k \geq 0$,

$$\|\varpi_{k+1} - \breve{\varpi}\|_\Phi^2 \leq \|\varpi_k - \breve{\varpi}\|_\Phi^2 - (1-\alpha)\|\varpi_k - \varpi_{k+1}\|_\Phi^2. \tag{4.31}$$

This inequality holds for each iterate k. We sum each of the k iterates to obtain

$$\sum_{l=0}^{k} \|\varpi_{l+1} - \breve{\varpi}\|_{\Phi}^2 \leq \sum_{l=0}^{k} \|\varpi_l - \breve{\varpi}\|_{\Phi}^2 - (1-\alpha) \sum_{l=0}^{k} \|\varpi_l - \varpi_{l+1}\|_{\Phi}^2$$

$$\|\varpi_{k+1} - \breve{\varpi}\|_{\Phi}^2 \leq \|\varpi_0 - \breve{\varpi}\|_{\Phi}^2 - (1-\alpha) \sum_{l=0}^{k} \|\varpi_l - \varpi_{l+1}\|_{\Phi}^2.$$

Since $\|\varpi_{k+1} - \breve{\varpi}\|_{\Phi}^2$ converges and is bounded from below by 0, taking the limit as $k \to \infty$ yields

$$0 \leq \lim_{k\to\infty} \|\varpi_{k+1} - \breve{\varpi}\|_{\Phi}^2 \leq \|\varpi_0 - \breve{\varpi}\|_{\Phi}^2 - (1-\alpha) \sum_{l=0}^{\infty} \|\varpi_l - \varpi_{l+1}\|_{\Phi}^2.$$

Rearranging we obtain

$$(1-\alpha) \sum_{l=0}^{\infty} \|\varpi_l - \varpi_{l+1}\|_{\Phi}^2 \leq \|\varpi_0 - \breve{\varpi}\|_{\Phi}^2.$$

Since $1 - \alpha > 0$, the second term must converge, and thus $\lim_{k\to\infty} \varpi_k - \varpi_{k+1} = 0$.

Since the sequence $\{\|\varpi_k - \breve{\varpi}\|_{\Phi}\}$ is bounded and converges, $\{\varpi_k\}$ is bounded and there exists a subsequence $\{\varpi_{n_k}\}$ that converges to some $\tilde{\varpi}$. Thus from $\varpi_{n_{k+1}} = T_2 T_1 \varpi_{n_k}$ and the Lipschitz continuity of $T_2 \circ T_1$, we have that $\tilde{\varpi} = T_2 T_1 \tilde{\varpi}$, i.e., it is a fixed point of $T_2 \circ T_1$. Letting $\breve{\varpi} = \tilde{\varpi}$ in (4.28) yields that $\{\|\varpi_k - \tilde{\varpi}\|_{\Phi}\}$ is bounded and converges. Since there exists a subsequence $\{\varpi_{n_k}\}$ that converges to $\tilde{\varpi}$, it follows that $\{\|\varpi_k - \tilde{\varpi}\|_{\Phi}\}$ converges to 0.

Since the iterates ϖ_k converge to a fixed point $\breve{\varpi}$ of $T_2 \circ T_1$, we know from Lemma 4.1 that $\breve{\varpi}$ is a zero of $\bar{\mathfrak{A}} + \bar{\mathfrak{B}}$. Then we can invoke Theorem 4.1: $\breve{\varpi} = \mathrm{col}(\breve{\mathbf{u}}, \breve{\lambda})$ satisfies the variational KKT conditions 4.12, and is an online approximate variational GNE of the game as per Definition 4.1. □

This concludes the convergence analysis, and thus this chapter . We showed that Algorithm 2 is a fixed-point iteration of averaged operators, and converges to a zero of a sum of monotone operators. We also showed that this zero corresponds to a variational GNE of our approximate game.

Chapter 5

Monotonicity and Lipschitz Continuity of Uncertain Operators

The focus of this chapter is developing numerical guarantees such that Assumption 5 is satisfied. The operator $H_{w,\Pi}$, defined in (4.11), is an approximation of the operator H_w in (4.7), which is an uncertain operator due to its dependence on the Jacobian of the unknown mapping π, and its dependence on the disturbance w. We make a nominal approximation of the Jacobian while running the online approximate algorithm, as discussed in the last chapter, but providing numerical guarantees of convergence would require a proper treatment of the uncertainty within the operator. We parametrize the uncertainty of the pseudogradient by representing it as a set of operators, rather than a precisely known one. This allows us to develop techniques based on linear matrix inequalities (LMI) to test for the Lipschitz continuity and strong monotonicity of the pseudogradient operator.

5.1 Semidefinite Programming Methods to Verify Monotonicity and Lipschitz Continuity

For this section, we assume we have access to the precisely known Clarke generalized Jacobian $\partial_C F$ (Definition 2.11) of an operator F. Note that this is a restrictive assumption, and is relaxed in the following section, but is necessary to begin the development of techniques to verify strong monotonicity and Lipschitz continuity for uncertain operators. Building on [1, 46], we aim to present semidefinite programming methods based on linear matrix inequalities (LMIs) to verify these conditions.

Proposition 5.1. *Let $\Omega \subset \mathbb{R}^n$ be a closed convex set and let F be locally Lipschitz continuous operator on Ω. Given $P \succ 0$, F is ρ-strongly monotone and θ-Lipschitz w.r.t. $\langle \cdot, \cdot \rangle_P$ on Ω if*

$$\begin{bmatrix} J \\ I_n \end{bmatrix}^\top (X_{\rho,\theta} \otimes P) \begin{bmatrix} J \\ I_n \end{bmatrix} \preceq 0 \tag{5.1}$$

for all $J \in \partial_C F(\mathbf{u})$ and all $u \in \Omega$, where

$$X_{\rho,L} := \begin{bmatrix} 2 & -(\rho+\theta) \\ -(\rho+\theta) & 2\rho\theta \end{bmatrix}$$

and where $\partial_C F(\mathbf{u})$ is the Clarke generalized Jacobian of F, as per Definition 2.11.

Proof. We assume that the matrix inequality (5.1) holds for some $P \succ 0$, and $\rho, \theta > 0$. We can expand the matrix inequality and write it out as

$$2J^\top PJ - (\rho+\theta)\left[J^\top P + PJ\right] + 2\rho\theta P \preccurlyeq 0.$$

This matrix inequality (5.1) is equivalent to

$$\|F(x) - F(y)\|_P^2 - (\rho+\theta)\langle F(x) - F(y), x - y\rangle_P + \rho\theta\|x-y\|_P^2 \leq 0. \tag{5.2}$$

This equivalence is proved in Appendix C.1. Following from (5.2) we can prove monotonicity by rearranging it:

$$(\rho+\theta)\langle F(x) - F(y), x - y\rangle_P \geq \|F(x) - F(y)\|_P^2 + \rho\theta\|x-y\|_P^2 \geq 0 \tag{5.3}$$

$$\langle F(x) - F(y), x - y\rangle_P \geq 0. \tag{5.4}$$

Monotonicity implies that we can apply the Cauchy-Schwarz inequality (Lemma 2.1):

$$\begin{aligned} & \langle F(x) - F(y), x - y\rangle_P \\ = \; & |\langle F(x) - F(y), x - y\rangle_P| \\ \leq \; & \|F(x) - F(y)\|_P \|x-y\|_P. \end{aligned}$$

Subtracting a more positive number makes the result more negative. Thus we can replace the second term in (5.2) to obtain

$$\begin{aligned} \|F(x) - F(y)\|_P^2 - (\rho+\theta)\|F(x) - F(y)\|_P\|x-y\|_P + \rho\theta\|x-y\|_P^2 \leq 0 \\ \left[\|F(x) - F(y)\|_P - \rho\|x-y\|_P\right]\left[\|F(x) - F(y)\|_P - \theta\|x-y\|_P\right] \leq 0. \end{aligned}$$

We can assume, without loss of generality, that $\theta \geq \rho > 0$. Then the above is a sector-bounded nonlinearity (proof in Appendix C.2) and is thus equivalent to

$$\rho\|x-y\|_P \leq \|F(x) - F(y)\|_P \leq \theta\|x-y\|_P. \tag{5.5}$$

The right side of (5.5) immediately corresponds to F being θ-Lipschitz continuous, as per Definition 2.5. We now return to the inequality (5.3), and use the left side of (5.5) to obtain

$$\begin{aligned} (\rho+\theta)\langle F(x) - F(y), x - y\rangle_P &\geq \rho^2\|x-y\|_P^2 + \rho\theta\|x-y\|_P^2 \\ &\geq \rho(\rho+\theta)\|x-y\|)P^2 \\ \langle F(x) - F(y), x - y\rangle_P &\geq \rho\|x-y\|_P^2. \end{aligned}$$

This corresponds to F being ρ-strongly monotone as per Definition 2.7 and concludes the proof. □

5.2 Linear Fractional Representation of Uncertain Operators

To apply the techniques developed in Section 5.1, we stated that the Clarke generalized Jacobian $\partial_C H_{w,\Pi}$ is known precisely. This assumption is not reasonable since the output mapping π and the disturbance w are both unknown. We aim to solve this by describing some set $\mathcal{J}$ such that $\partial_C H_{w,\Pi}(\mathbf{u}) \subset \mathcal{J}$, $\forall \mathbf{u} \in \mathbb{R}^n$, i.e., we overbound the Clarke generalized Jacobian with some uncertainty set. Thus if we verify the conditions (5.1) for the uncertainty set, that is sufficient to verify that the conditions apply for $\partial_C H_{w,\Pi}$ and thus the pseudogradient $H_{w,\Pi}$ is strongly monotone and Lipschitz continuous.

For this purpose, we utilize the linear fractional representation from literature on robust control [29, 47, 57], which we presented in Section 2.6. While this method can be less intuitive and presents a sharper learning curve than simpler methods (for example, bounding the uncertainty within a polytope), it has two key advantages:

1. It can be used to parametrize larger, more general classes of uncertainty.
2. It allows the development of a *single* matrix inequality that sufficiently verifies the conditions in (5.1), as opposed to needing to verify it for a large number of Jacobians in the overbounded set. This is outlined later in this section.

With access to a good model of the operator $\partial_C H_{w,\Pi}(\mathbf{u})$, we could use (2.2) and the techniques outlined in Section 2.6 to represent the operator as a linear fractional transformation, with Δ quantifying the variation caused by $\mathbf{u}$ and w. An example of such a parametrization applied to verify our criteria is presented in Section 7.1 too. However, we wish to develop a technique to verify these criteria *without* the significant modelling effort it takes to find a precise representation. As such, our goal is to find a linear fractional transformation that *contains* our mapping $\partial_C H_{w,\Pi}(\mathbf{u})$ with as little conservatism as possible.

Consider, for example, if we were analyzing a system whose precise uncertainty can be quantified as in Example 2. One could use the following overbound: the term $\frac{-1}{1+\delta_1}$ is simply some third uncertainty δ_3 with an appropriately adjusted range (for example, if $\delta_1 \in [0, 1]$ then $\delta_3 \in [-1, -\frac{1}{2}]$). This model now covers a larger set of uncertainties than the precise model presented in Example 2 (constraining δ_3 to be a rational function of δ_1 is akin to taking a slice out of the larger model). However, this is a model one could easily develop from some quick experimentation on the system dictated by these uncertainties, and if we could simply verify our convergence criteria on the more conservative model rather than a precise one, it would make our framework more general, faster to verify, and lighter on overhead. This motivates our goal in the rest of this section: to use (2.2) to overbound the mapping $\zeta = \partial_C H_{w,\Pi}(\mathbf{u})\xi$ within a more conservative representation. We work with the following LFT representation where Δ is no longer a precise parametrization of the uncertainties of the mapping $\partial_C H_{w,\Pi}$:

$$\left[\begin{array}{c} \zeta \\ \hline q \end{array}\right] = \left[\begin{array}{c|c} M & B \\ \hline C & D \end{array}\right] \left[\begin{array}{c} \xi \\ \hline p \end{array}\right] \tag{5.6}$$
$$p = \Delta q,$$

where $M \in \mathbb{R}^{l\times n}$, $B \in \mathbb{R}^{l\times s}$, $C \in \mathbb{R}^{z\times n}$, $D \in \mathbb{R}^{z\times s}$, and $\Delta \in \mathbf{\Delta} \subset \mathbb{R}^{s\times z}$. Solving (5.6) for q we obtain

$$\begin{aligned} q &= C\xi + Dp \\ q &= C\xi + D\Delta q \\ q &= (I_z - D\Delta)^{-1}C\xi. \end{aligned}$$

Note that Definition 2.19 requires that $(I - D\Delta)$ is invertible. Substituting the above line into the first row of the matrix equation in (5.6) yields

$$\begin{aligned} \zeta &= M\xi + Bp \\ \zeta &= M\xi + B\Delta q \\ \zeta &= \left[M + B\Delta(I_z - D\Delta)^{-1}C\right]\xi. \end{aligned}$$

Further, we assume that we have access to a convex cone of matrices $\mathbf{\Theta} \subset \mathbb{R}^{(s+z)\times(s+z)}$ such that

$$p = \Delta q,\ \Delta \in \mathbf{\Delta} \implies \begin{bmatrix} q \\ p \end{bmatrix}^\top \Theta \begin{bmatrix} q \\ p \end{bmatrix} \geq 0,\ \forall \Theta \in \mathbf{\Theta}. \tag{5.7}$$

This assumption is unintuitive as presented, but is key to the relaxation that we use to develop a more tractable matrix inequality later. We can now define the uncertainty set that overbounds the operator $H_{w,\Pi}$.

Definition 5.1 (Jacobian Uncertainty Set)**.** *The uncertainty set of an operator $H : \mathbb{R}^n \to \mathbb{R}^n$ is defined as the set*

$$\mathcal{H}^{\text{lft}} = \{H : \partial_C H(\mathbf{u}) \subseteq \mathcal{J}^{\text{lft}},\ \forall \mathbf{u} \in \Omega\}, \tag{5.8}$$

where the Jacobian set $\mathcal{J}^{\text{lft}}$ is defined as

$$\mathcal{J}^{\text{lft}} := \{M + B\Delta(I_z - D\Delta)^{-1}C : \Delta \in \mathbf{\Delta}\}. \tag{5.9}$$

The matrices M, B, C, D, Δ are defined as per (5.6)*, and there exists a convex cone of matrices $\mathbf{\Theta} \subset \mathbb{R}^{(s+z)\times(s+z)}$ such that Δ satisfies* (5.7)*.*

Thus to verify the strong monotonicity and Lipschitz continuity of the operator $H_{w,\Pi}$ defined in (4.11), it is sufficient to construct the set $\mathcal{H}^{\text{lft}}$ such that $H_{w,\Pi} \in \mathcal{H}^{\text{lft}}$ (by following the above procedure), and to then show that all operators $H \in \mathcal{H}^{\text{lft}}$ fulfill those conditions.

Proposition 5.2. *Given $P \succ 0$, any operator $H \in \mathcal{H}^{\text{lft}}$ is ρ-strongly monotone and θ-Lipschitz continuous w.r.t. $\langle\cdot,\cdot\rangle_P$ on $\mathbb{R}^n$ if there exists $\Theta \in \mathbf{\Theta}$ such that*

$$\begin{bmatrix} M & B \\ I_n & \mathbb{0} \end{bmatrix}^\top (X_{\rho,\theta} \otimes P) \begin{bmatrix} M & B \\ I_n & \mathbb{0} \end{bmatrix} + \begin{bmatrix} C & D \\ \mathbb{0} & I_s \end{bmatrix}^\top \Theta \begin{bmatrix} C & D \\ \mathbb{0} & I_s \end{bmatrix} \preccurlyeq 0 \tag{5.10}$$

where $X_{\rho,\theta}$ is defined as in Proposition 5.1.

Proof. Given an arbitrary $x \in \mathbb{R}^n$ and $\Delta \in \mathbf{\Delta}$, define $q = Cx + Dp$ and $p = \Delta q$. Pre and post

multiplying (5.10) by $\mathrm{col}(x, p)$ we obtain

$$\begin{bmatrix} x \\ p \end{bmatrix}^\top \begin{bmatrix} M & B \\ I_n & \mathbb{0} \end{bmatrix}^\top \mathbb{P} \begin{bmatrix} M & B \\ I_n & \mathbb{0} \end{bmatrix} \begin{bmatrix} x \\ p \end{bmatrix} + \begin{bmatrix} q \\ p \end{bmatrix}^\top \Theta \begin{bmatrix} q \\ p \end{bmatrix} \leq 0 \tag{5.11}$$

where $\mathbb{P} = X_{\rho,L} \otimes P$. Since $p = \Delta q$, (5.7) implies that the first term in (5.11) must be nonpositive. Therefore

$$\begin{bmatrix} Mx + Bp \\ x \end{bmatrix}^\top \mathbb{P} \begin{bmatrix} Mx + Bp \\ x \end{bmatrix} \leq 0, \ \forall x \in \mathbb{R}^n \tag{5.12}$$

From $p = \Delta q$ and $q = Cx + Dp$ we conclude that $p = \Delta(I_z - D\Delta)^{-1}Cx$. Substituting this into (5.12), we find that

$$\begin{bmatrix} J_\Delta \\ I_n \end{bmatrix}^\top \mathbb{P} \begin{bmatrix} J_\Delta \\ I_n \end{bmatrix} \preceq 0$$

where $J_\Delta = M + B\Delta(I_z - D\Delta)^{-1}C \in \mathcal{J}^{\mathrm{lfr}}$. Therefore, we conclude that all elements of $\mathcal{J}^{\mathrm{lft}}$ satisfy (5.1), and thus all operators $H \in \mathcal{H}^{\mathrm{lft}}$ are ρ-strongly monotone and L-Lipschitz continuous w.r.t. $\langle \cdot, \cdot \rangle_P$. □

Note that in Assumption 5, we verify the conditions with regards to the Euclidean norm, and thus we use $P = I_n$ when evaluating these inequalities.

5.2.1 Recipes for LFR Modelling

In this section we outline recipes to select a convex cone $\boldsymbol{\Theta}$ that satisfies the conditions outlined in Proposition 5.2. These recipes are taken from [1], from where we borrowed the idea of using a single matrix inequality to tackle the problem of verifying strong monotonicity and Lipschitz continuity.

1. Unstructured, Norm-Bounded Uncertainty:

   Given $\gamma \geq 0$, let $\boldsymbol{\Delta}(\gamma) := \{\Delta \in \mathbb{R}^{s \times z} : \|\Delta\|_2 = \sigma_{\max}(\Delta) \leq \gamma\}$ be the set of unstructured matrices with induced 2-norm less than or equal to γ. Then a cone $\boldsymbol{\Theta}$ that achieves the positivity condition (5.7) is

   $$\boldsymbol{\Theta} = \left\{ \theta \begin{bmatrix} I_z & \mathbb{0} \\ \mathbb{0} & -\frac{1}{\gamma^2} I_s \end{bmatrix} : \theta \geq 0 \right\}$$

2. Repeated Scalar, Norm-Bounded Uncertainty:

   Given $\gamma \geq 0$, let $\boldsymbol{\Delta}(\gamma) := \{\Delta = \delta I : \delta \in \mathbb{R}, |\delta| \leq \gamma\}$ denote the set of diagonal matrices with each diagonal entry bounded in magnitude by γ. Then a cone that achieves the positivity condition is

   $$\boldsymbol{\Theta} = \left\{ \Theta = \begin{bmatrix} \Phi & \Psi \\ \Psi^\top & -\frac{1}{\gamma^2}\Phi \end{bmatrix} : \Phi \succcurlyeq 0, \ \Psi = -\Psi^\top \right\}.$$

3. Unstructured monotone and Lipschitz uncertainty:

   Given $\mu, L \in \mathbb{R}$ such that $0 \leq \mu \leq L < \infty$, let $\boldsymbol{\Delta} := \{\Delta \in \mathbb{R}^{s \times s} : \mu I \preccurlyeq \Delta \preccurlyeq LI\}$. A cone that

works in this case is

$$\mathbf{\Theta} = \left\{ \Theta = \varphi \begin{bmatrix} -2\mu L & \mu + L \\ \mu + L & -2 \end{bmatrix} \otimes I_s : \varphi \geq 0 \right\}.$$

4. Repeated scalar monotone and Lipschitz uncertainty:

   Given Given $\mu, L \in \mathbb{R}$ such that $0 \leq \mu \leq L < \infty$, let $\mathbf{\Delta} := \{\Delta \in \mathbb{R}^{s \times s} : \Delta = \delta I_s,\ \mu \leq \delta \leq L$. A cone that works in this case is

$$\mathbf{\Theta} = \left\{ \Theta = \varphi \begin{bmatrix} -2\mu L & \mu + L \\ \mu + L & -2 \end{bmatrix} \otimes \Phi : \varphi \geq 0,\ \Phi \succcurlyeq 0 \right\}.$$

5. Block-structured uncertainty:

   Consider the block-diagonal uncertainty set defined as

$$\mathbf{\Delta} := \{\Delta = \text{blkdiag}(\Delta_1,\ \ldots\ , \Delta_r)\},$$

   where each block satisfies one of the previous criteria. We can then construct a cone of matrices by using the previous individually for their corresponding blocks. We provide an example of this construction in Section 5.3.

It should be noted that the more we know about the structure of the parameter block Δ (i.e., the less conservatism we have in our uncertainty parametrization), the lighter the restrictions we have to impose on the constraints of our convex cone. For example, with case 1 we are restricted to identity matrices along the main diagonal, and zeroes along the anti-diagonal. With case 2 we can consider all positive semidefinite matrices along the main diagonal, and any choice of anti-diagonal that makes the matrix skew-symmetric. This means that the optimization problem performed to find a minimal (maximal) θ (ρ) fulfilling Proposition 5.2 would have a larger feasible set and thus is likelier to find relatively tight estimates for these values, which in turn helps with tuning appropriate step sizes as per Lemma 4.3. This ties back to what we led this discussion with: that we are trying to bound the Clarke-generalized Jacobian of the pseudogradient with as little conservatism as possible, while still not relying on a precise model.

5.3 Feedback Optimization Example

In this section, we go through the process of developing an LFT for a common class of feedback optimization problems. Consider the output $\mathbf{y} \in \mathbb{R}^l$ and the goal of constraining it with vectors $\underline{\mathbf{y}}, \bar{\mathbf{y}} \in \mathbb{R}^l$ such that $\mathbf{y}_j \in \left[\underline{\mathbf{y}}_j, \bar{\mathbf{y}}_j\right],\ \forall j \in \{1,\ \ldots\ , l\}$. For each agent i, denote their output constraint vectors as $\underline{y}_i$ and $\bar{y}_i$, stacked similarly to how $\mathbf{y} = \text{col}(y_1,\ \ldots\ , y_N)$. Now we consider the class of problems where each agent is trying to reduce a cost function of the form

$$\min_{u_i \in \Omega_i} u_i^\top Q_i u_i + q_i^\top u_i + \frac{1}{2}\alpha_i \sum_{j=1}^{l} \left[\max(0, \underline{\mathbf{y}}_j - \mathbf{y}_j, \mathbf{y}_j - \bar{\mathbf{y}}_j\right]^2 \tag{5.13}$$

subject to $\mathbf{y} = \pi(\mathbf{u}, w)$.

The max function encodes $\mathbf{y}_j \in \left[\underline{\mathbf{y}}_j, \bar{\mathbf{y}}_j\right]$ as a soft constraint, penalizing the point-to-set distance between $\mathbf{y}$ and a rectangular box in $\mathbb{R}^l$. The hyperparameter α_i forces agents to obey the constraint more stringently if it is set to a larger value. The matrices $Q_i \in \mathbb{R}^{n_i \times n_i}$ and the vectors $q_i \in \mathbb{R}^{n_i}$ are fixed matrices and vectors that impose a quadratic cost on each agent's action. For example, if each agent were concerned with driving the system state towards some set-point $(u_{\text{ref}})_i \in \mathbb{R}^{n_i}$, then the cost would be quantified by

$$\|(u_{\text{ref}})_i - u_i\|^2 = [(u_{\text{ref}})_i - u_i]^\top [(u_{\text{ref}})_i - u_i] = u_i^\top u_i - 2(u_{\text{ref}})_i^\top u_i + (u_{\text{ref}})_i^\top (u_{\text{ref}})_i.$$

For the above cost, we can omit the final constant term since it has no effect on minimization, and then express it as (5.13) with $Q_i = I_{n_i}$ and $q_i = 2(u_{\text{ref}})_i$.

Note that we formulate this problem without coupling constraints, as the Algorithm 2 converges obeying any coupling constraints imposed on the actions, so long as Assumption 5 is satisfied. Therefore we focus on verifying the criteria using the techniques developed in the previous section. Figure 5.1 shows a visualization of the soft constraint. Note that the function is $\mathbb{C}^1$.

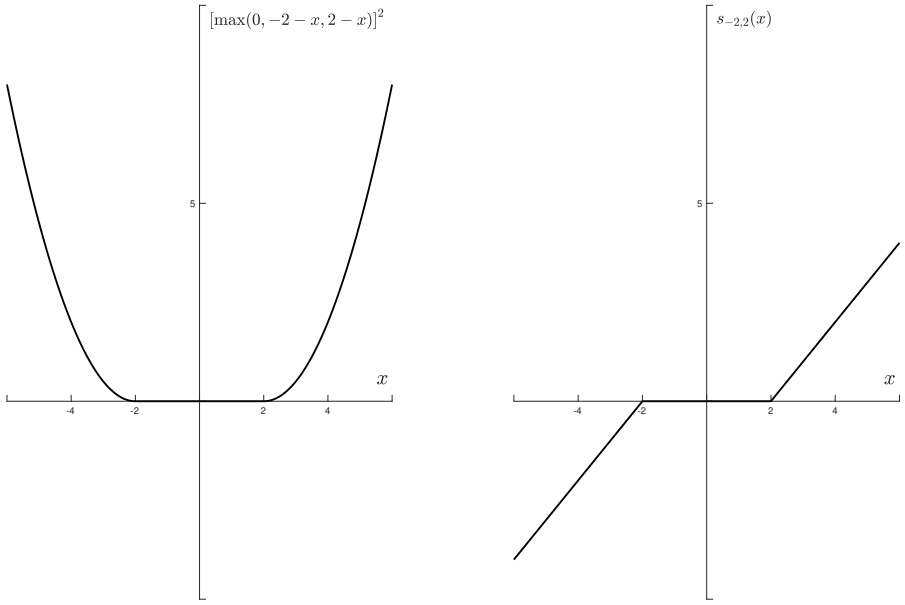

Figure 5.1: The soft constraint function and its derivative with thresholds set to $\underline{y}_i = -2$ and $\bar{y}_i = 2$.

Next, we attempt to compute the gradients of the cost function. Player i's action cost function

is a simple quadratic form, and thus

$$\begin{aligned} f_i(\mathbf{u}) &= u_i^\top Q_i u_i + q_i^\top u_i \\ \partial_{\mathbf{u}} f_i(\mathbf{u}) &= \left[(Q_i + Q_i^\top)u_i + q_i\right]^\top. \end{aligned} \tag{5.14}$$

Thus the gradient $\nabla_{u_i} f_i(\mathbf{u})$ can be expressed as

$$\nabla_{u_i} f_i(\mathbf{u}) = \begin{bmatrix} \mathbb{0} & \cdots & Q_i + Q_i^\top & \cdots \mathbb{0} \end{bmatrix} u_i + q_i. \tag{5.15}$$

For the output cost functions:

$$g_i(\mathbf{y}) = \frac{1}{2}\alpha_i \sum_{j=1}^{l} \left[\max(0, \underline{\mathbf{y}}_j - \mathbf{y}_j, \mathbf{y}_j - \bar{\mathbf{y}}_j)\right]^2$$

$$\nabla_{u_i} g_i(\pi(\mathbf{u}, w)) = \sum_{k=1}^{N} \partial_{u_i} \pi_k(\mathbf{u}, w)^\top \nabla_{y_k} g_i(\mathbf{y}) \tag{5.16a}$$

$$= \alpha_i \begin{bmatrix} \partial_{u_i}\pi_1(\mathbf{u}, w)^\top & \dots & \partial_{u_i}\pi_N(\mathbf{u}, w)^\top \end{bmatrix} s_{\underline{\mathbf{y}}, \bar{\mathbf{y}}}(\mathbf{y}). \tag{5.16b}$$

The function $s_{\underline{\mathbf{y}}, \bar{\mathbf{y}}}(\mathbf{y})$ denotes the vectorized soft-thresholding function, defined for each entry of $\mathbf{y} \in \mathbb{R}^l$. For all $j \in \{1, \dots, l\}$

$$s_{\underline{\mathbf{y}}_j, \bar{\mathbf{y}}_j}(\mathbf{y}_j) = \begin{cases} \mathbf{y}_j - \underline{\mathbf{y}}_j & \mathbf{y}_j < \underline{\mathbf{y}}_j \\ 0 & \underline{\mathbf{y}}_j \le \mathbf{y}_j \le \bar{\mathbf{y}}_j \\ \mathbf{y}_j - \bar{\mathbf{y}}_j & \bar{\mathbf{y}}_j < \mathbf{y}_j \end{cases} \tag{5.17}$$

We can now make an approximation for the Jacobian $\partial_{\mathbf{u}}\pi$ to apply the techniques developed in Chapter 4 and this chapter. We assume we have access to an experimentally obtained nominal Jacobian with an error bound $\gamma \in \mathbb{R}$ such that

$$\partial_{\mathbf{u}}\pi(\mathbf{u}, w) = \Pi_{\text{nom}} + \Delta_\pi, \ \|\Delta_\pi\| \le \gamma, \ \forall \mathbf{u} \in \Omega, w \in \mathcal{W}, \tag{5.18}$$

i.e., the nominal Jacobian is within a bounded error of the instantaneous Jacobian of the system for all possible actions and disturbances. In (5.16b), we replace the true Jacobians $\partial_{u_i}\pi_j$ with submatrices of the fixed approximation $\Pi = \Pi_{\text{nom}} \approx \partial_{\mathbf{u}}\pi$ and thus

$$\nabla_{u_i} g_i(\mathbf{y}) = \alpha_i \begin{bmatrix} \Pi_{i1}^\top & \dots & \Pi_{iN}^\top \end{bmatrix} s_{\underline{\mathbf{y}}, \bar{\mathbf{y}}}(\mathbf{y}). \tag{5.19}$$

We can then construct the pseudogradient as follows:

$$H_{w,\Pi} = \begin{bmatrix} Q_1 + Q_1^\top & \mathbb{0} & \cdots & \mathbb{0} \\ \mathbb{0} & Q_2 + Q_2^\top & \cdots & \mathbb{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{0} & \mathbb{0} & \cdots & Q_N + Q_N^\top \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} + \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{bmatrix}$$
$$+ \begin{bmatrix} \Pi_{11}^\top & \dots & \dots & \Pi_{1N}^\top \\ \Pi_{21}^\top & \dots & \dots & \Pi_{2N}^\top \\ \vdots & \vdots & \vdots & \vdots \\ \Pi_{N1}^\top & \dots & \dots & \Pi_{NN}^\top \end{bmatrix} \begin{bmatrix} s_{\underline{y}_1, \bar{y}_1}(y_1) \\ s_{\underline{y}_2, \bar{y}_2}(y_2) \\ \vdots \\ s_{\underline{y}_N, \bar{y}_N}(y_N) \end{bmatrix}$$
$$H_{w,\Pi} := \mathbf{Q}\mathbf{u} + \mathbf{h} + \Pi^\top s_{\underline{\mathbf{y}}, \bar{\mathbf{y}}}(\mathbf{y}). \tag{5.20}$$

We now wish to characterize the Clarke generalized Jacobian $\partial_C H_{w,\Pi}$, and then apply the techniques developed within Section 5.2 to overbound it in a tractable manner. We start by differentiating (5.20), and by applying the chain rule to the final term, we obtain:

$$\begin{aligned} \partial_{\mathbf{u}} H_{w,\Pi} &= \mathbf{Q} + \Pi^\top \partial_{\mathbf{u}} s_{\underline{\mathbf{y}}, \bar{\mathbf{y}}}(\pi(\mathbf{u}, w)) \\ \partial_{\mathbf{u}} H_{w,\Pi} &= \mathbf{Q} + \Pi^\top \partial_{\mathbf{y}} s_{\underline{\mathbf{y}}, \bar{\mathbf{y}}}(\pi(\mathbf{u}, w)) \, \partial_{\mathbf{u}} \pi(\mathbf{u}, w). \end{aligned} \tag{5.21}$$

The two terms $\partial_{\mathbf{y}} s_{\underline{\mathbf{y}}, \bar{\mathbf{y}}}(\pi(\mathbf{u}, w))$ and $\partial_{\mathbf{u}} \pi(\mathbf{u}, w)$ are the sources of uncertainty in the Jacobian that we seek to characterize. First, we note that the j-th element of $s_{\underline{\mathbf{y}}, \bar{\mathbf{y}}}(\pi(\mathbf{u}, w))$ is only dependent on the output $\mathbf{y}_j$ and not any of the others, so its derivative for all other other outputs would simply be zero. For the aligned output, we can differentiate the piecewise soft-thresholding function (5.17) to obtain

$$\frac{\partial \left(s_{\underline{\mathbf{y}}, \bar{\mathbf{y}}}(\mathbf{y}_j) \right)}{\partial \mathbf{y}_j} = \begin{cases} 1, & \mathbf{y}_j \in \left(-\infty, \underline{\mathbf{y}}_j \right) \bigcup (\bar{\mathbf{y}}_j, \infty) \\ 0, & \mathbf{y}_j \in [\underline{\mathbf{y}}_j, \bar{\mathbf{y}}_j] \end{cases}, \ \forall j \in \{1, \ \dots \ , l\}.$$

The elements of the Jacobian thus vary discontinuously within the set $\{0, 1\}$. Within the LFT framework, we instead vary the uncertainty continuously in the range $[0, 1]$ (note how the true Jacobian is now a subset of the LFT parametrization, since it will never take values in the range $(0, 1)$). We thus define the following uncertainty matrix to represent the Jacobian $\partial_{\mathbf{y}} s_{\underline{\mathbf{y}}, \bar{\mathbf{y}}}(\pi(\mathbf{u}, w))$:

$$\Delta_q = \begin{bmatrix} \delta_1 & \mathbb{0} & \cdots & \mathbb{0} \\ \mathbb{0} & \delta_2 & \cdots & \mathbb{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{0} & \mathbb{0} & \cdots & \delta_l \end{bmatrix}, \ \delta_i \in [0, 1] \ \forall i \in \{1, \ \dots \ , l\}. \tag{5.22}$$

We approximate the uncertain Jacobian $\partial_{\mathbf{u}} \pi$ using (5.18). Then the Jacobian (5.21), can be overbounded as per Definition 5.1 for all $\mathbf{u} \in \mathbb{R}^n$ by the set of Jacobians:

$$\partial_C H_{w,\Pi}(\mathbf{u}) \in \mathcal{J}^{\text{lft}} := \left\{ \mathbf{Q} + \Pi^\top \Delta_q (\Pi_{\text{nom}} + \Delta_\pi) : \Delta_q \in \mathbb{D}^l, \ \mathbb{0} \preccurlyeq \Delta_q \preccurlyeq I_m, \ \|\Delta_\pi\| \leq \gamma \right\}, \tag{5.23}$$

where $\mathbb{D}^l$ is the set of $l \times l$ diagonal matrices. Note that the semi-definiteness constraints on Δ_q

constrain the diagonal elements to range in $[0, 1]$.

We now try to represent this as a linear fractional transformation as in (5.6). For that, we require matrices M, B, C, D, Δ such that $M + B\Delta(I_z - D\Delta)^{-1}C = \mathbf{Q} + \Pi^\top \Delta_q(\Pi_{\text{nom}} + \Delta_\pi)$. We let

$$\Delta = \begin{bmatrix} \Delta_q & \mathbb{0} \\ \mathbb{0} & \Delta_\pi \end{bmatrix}.$$

It immediately follows that $M = \mathbf{Q}$ as the affine term that is not dependent on the uncertainties. Then we let $B\Delta$ correspond to the "left term" in the rest of the parametrization. It follows that

$$\begin{aligned} B\Delta &= \begin{bmatrix} \Pi^\top \Delta_q & \mathbb{0} \end{bmatrix} \\ B &= \begin{bmatrix} \Pi^\top & \mathbb{0} \end{bmatrix}. \end{aligned}$$

To obtain $B\Delta(I_z - D\Delta)^{-1}C = \mathbf{Q} + \Pi^\top \Delta_q(\Pi_{\text{nom}} + \Delta_\pi)$ we then need

$$(I_z - D\Delta)^{-1}C = \begin{bmatrix} \Pi_{\text{nom}} + \Delta_\pi \\ * \end{bmatrix},$$

where the $*$ indicates that the term can take any value (since it gets zeroed out when multiplied). Note here that $z = l + n$. We fix the following matrices:

$$C = \begin{bmatrix} \Pi_{\text{nom}} \\ I_n \end{bmatrix}, \; D = \begin{bmatrix} \mathbb{0} & I_l \\ \mathbb{0} & \mathbb{0} \end{bmatrix}.$$

Using these matrices in the expression $(I_z - D\Delta)^{-1}C$ yields

$$\begin{aligned} & (I_z - D\Delta)^{-1}C \\ = & \begin{bmatrix} I_l & -\Delta_\pi \\ \mathbb{0} & I_n \end{bmatrix}^{-1} \begin{bmatrix} \Pi_{\text{nom}} \\ I_n \end{bmatrix} \\ = & \begin{bmatrix} I_l & \Delta_\pi \\ \mathbb{0} & I_n \end{bmatrix} \begin{bmatrix} \Pi_{\text{nom}} \\ I_n \end{bmatrix} \\ = & \begin{bmatrix} \Pi_{\text{nom}} + \Delta_\pi \\ I_n \end{bmatrix} \end{aligned}$$

Thus every element $J \in \mathcal{J}^{\text{lft}}$, with the set as defined (5.23), can be written as $J = M + B\Delta(I_{l+n} - D\Delta)^{-1}C$ where

$$\left[\begin{array}{c|c} M & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{c|cc} \mathbf{Q} & \Pi^\top & \mathbb{0} \\ \hline \Pi_{\text{nom}} & \mathbb{0} & I_l \\ I_n & \mathbb{0} & \mathbb{0} \end{array}\right], \; \Delta = \begin{bmatrix} \Delta_q & \mathbb{0} \\ \mathbb{0} & \Delta_\pi \end{bmatrix}. \tag{5.24}$$

Remark 5. *In* (5.24)*, we distinguish* Π_{nom} *and* Π*, even though in this specific instance we simply used* $\Pi = \Pi_{\text{nom}}$*. This is to explicitly show the fact that the latter term came from the constant approximation we make in the cost function itself, while the former term appears while parametrizing the uncertainty for the LFT. In this case they were the same matrix, but this may not necessarily be true in general.*

We can now utilize the recipes from the previous section to characterize our uncertainties. We have a block-structured uncertainty with the first block Δ_q being composed of scalars along the diagonal, each of which can be represented as a matrix in $\mathbb{R}^{1\times 1}$ fulfilling the repeated monotone and Lipschitz criteria with $\rho = 0$ and $\theta = 1$. The matrix Δ_π is an unstructured, norm-bounded uncertainty. We can thus construct the set

$$\boldsymbol{\Theta} := \left\{ \sum_{j=1}^{l} \Theta_j : \varphi_j \geq 0,\ \theta \geq 0 \right\}, \tag{5.25}$$

$$\Theta_j := \begin{bmatrix} \mathbb{0} & \mathbb{0} & \varphi_j e_j e_j^\top & \mathbb{0} \\ \mathbb{0} & \frac{\theta}{l} I_n & \mathbb{0} & \mathbb{0} \\ \varphi_j e_j e_j^\top & \mathbb{0} & -2\varphi_j e_j e_j^\top & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & -\frac{\theta}{l\gamma^2} I_l \end{bmatrix}, \tag{5.26}$$

where e_j is the j^{th} canonical vector in $\mathbb{R}^l$. Thus, combining this LFT parametrization with Proposition 5.2, we can verify the convergence of Algorithm 2.

This example concludes this chapter. We developed a semidefinite programming based approach to verify the convergence criteria from the previous chapter, and introduced conservatism to reduce informational requirements and a priori overhead. The final section was designed to illustrate a realistic example of introducing conservatism while developing the LFT representation, and to show a specific choice of the convex cone of matrices.

Chapter 6

Jacobian-Free Algorithm

6.1 Limitations of a Fixed Jacobian

In this section, we motivate the reasoning behind estimating the Jacobian in an online manner. In Section 4.2, we approximated the Jacobian using a fixed matrix as per (4.10). While this approximation has been shown to robustly converge in large classes of problems, its accuracy can be insufficient or unreliable in cases where the instantaneous Jacobian at any given time-step is heavily perturbed from its linear, approximate counterpart. Proposition 4.1 gives us the error bound between the result of the OA vGNE algorithm and a true candidate for an optimum, dependent on the error between the nominal and true Jacobian. We motivate the problem by exploring the case where that error becomes unreasonably large.

We consider a simple problem without disturbances or coupling constraints to demonstrate this. Suppose that we have two robots, $i \in \{1, 2\}$, each moving in 2D space with the goal of approaching a target position $\bar{r}_i \in \mathbb{R}^2$. They are driven by a controller that asymptotically tracks some position input $u_i \in \mathbb{R}^2$. Suppose they also have a connectivity goal, where they are not allowed to be outside of a specified maximum distance d from one another (i.e., $\|r_1 - r_2\| \leq d$). Consider a penalty function $\bar{g} : \mathbb{R} \to \mathbb{R}$, chosen to penalize the argument for being greater than the upper bound d. If each robot is equipped with a sensor that outputs their absolute position $y_i = r_i$, then the game can be encoded as

$$J_i(u_i, \mathbf{y}) = \|u_i - \bar{r}_i\|^2 + \bar{g}(\|y_i - y_{-i}\|),$$

where y_{-i} is the other player's output. Since y_i asymptotically approaches the target position u_i, we expect the steady-state output mapping to be

$$\pi(\mathbf{u}) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

and thus the Jacobian of the outputs with respect to the inputs would be

$$\partial_{\mathbf{u}} \pi(\mathbf{u}) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

A nominal Jacobian for this problem can be relatively easily calculated, and even a bad approxi-

mation can still be expected to converge to an optimum within a bounded distance of a true vKKT point as per Proposition 4.1. Note that the derivatives of $\bar{g}(\|y_i - y_{-i}\|)$ with respect to each y_j can be calculated in a deterministic fashion, as we choose $\bar{g}$ to enforce the boundary constraint.

Now we consider the case where the output is simply the (squared) Euclidean 2-norm distance between the two players, that is, $y_i = \|r_i - r_{-i}\|^2$. Thus at steady-state the output converges to $y_i = \|u_i - u_{-i}\|^2$, and the Jacobian becomes

$$\partial_{\mathbf{u}}\pi(\mathbf{u}) = 2\begin{bmatrix} u_1^\top - u_2^\top & u_2^\top - u_1^\top \\ u_1^\top - u_2^\top & u_2^\top - u_1^\top \end{bmatrix}. \tag{6.1}$$

Note that this Jacobian varies heavily as a function of the control input. It is not reasonable to fix a nominal Jacobian and characterize variation from the nominal as uncertainty as we did in (5.18), as the variation over the course of a game can be quite heavy and lead to large deviation from a true optimum. An illustration of this problem is presented in Figure 6.1.

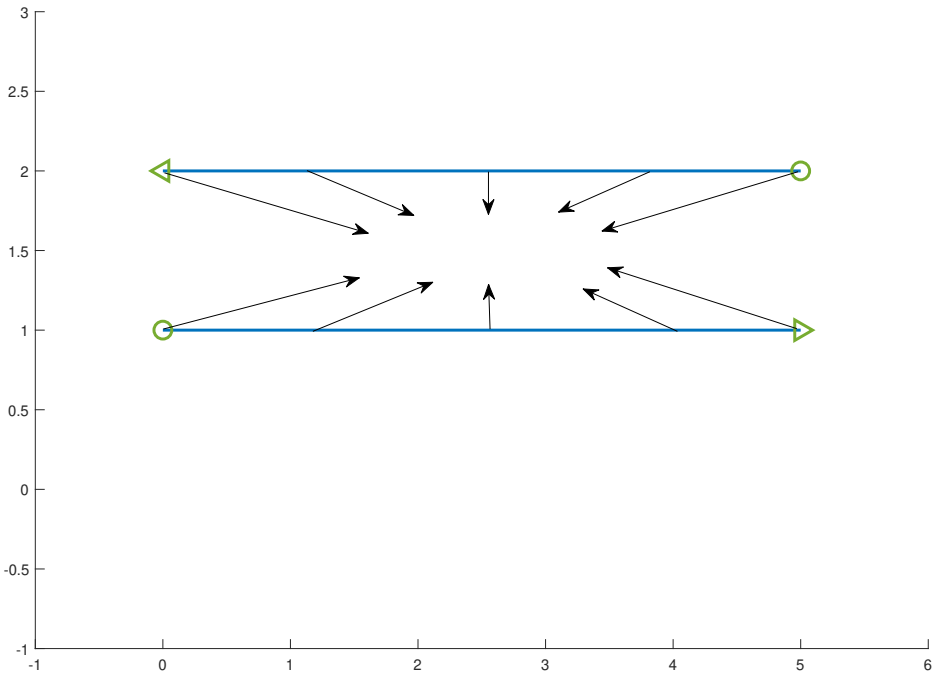

Figure 6.1: Each robot moves from the marked 'O' to the marked arrowhead along the blue trajectory. The black arrows represent the vector $u_i - u_{-i}$.

Inspecting the figure, we could compute gradients at the mid-point of the trajectories to construct the nominal Jacobian Π, and the maximum distance from the gradients at the start and end points to construct a Δ_π similar to (5.18). However, this construction only appears reasonable if we assume that the robots will take a relatively straight path towards their destination.This method would fail

to characterize the Jacobian in cases where the robot has more complex dynamics that lead to a curving path as the robot adjusts its heading (for example, unicycle dynamics), or if the robots' initialization and targets naturally lead to a path that perturbed more heavily from a straight line option. Finally, even if the nominal Jacobian seems to be reasonable in a 2-robot problem, an N-robot problem makes the problem of finding a nominal Jacobian far harder, as each component could now be an average of several gradients, compounding the offline knowledge required as well as approximation errors.

Thus, a simple change in the output feedback available for a problem can cause the difference between a trivially computed nominal Jacobian, and a nominal Jacobian that requires extensive forecasting for a relatively inaccurate result. This illustrates the need for online methods of estimating the Jacobian $\partial_{\mathbf{u}}\pi$ with minimal a priori knowledge. The rest of this chapter focuses on developing this approach.

6.2 Quasi-Stochastic Approximation of Jacobian

This section expands on prior work on model-free optimization [18], where it was noted that directly approximating the Jacobian matrix is a subject of ongoing research. We utilize the quasi-stochastic perturbation approach to estimate the Jacobian.

A Simultaneous Perturbation Stochastic Approximation (SPSA) approach [24, 26] has proven to be effective at approximating gradients effectively, where each agent only needs 2 measurements of the objective function. Contrast this approach with a finite-differences method, which would require each control input to be perturbed for a measurement, requiring $2n$ measurements total in our case. We thus develop our approach based off the stochastic perturbation methods.

We begin by expressing the Jacobian matrix in terms of its rows:

$$\partial_{\mathbf{u}}\pi(\mathbf{u}, w) = \begin{bmatrix} \nabla_{\mathbf{u}}^{\top}\pi_{11} \\ \vdots \\ \nabla_{\mathbf{u}}^{\top}\pi_{1l_1} \\ \vdots \\ \nabla_{\mathbf{u}}^{\top}\pi_{N1} \\ \vdots \\ \nabla_{\mathbf{u}}^{\top}\pi_{Nl_N} \end{bmatrix}, \tag{6.2}$$

where for each player i, the mapping π_{ij}, $j \in \{1, \ \dots \ , l_i\}$ represents the j-th output that they can measure. We aim to utilize the measured outputs $y_i \in \mathbb{R}^{l_i}$ to approximate each of these gradients, and thus form an estimation of the Jacobian.

Lemma 6.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a $\mathbb{C}^3$ function (Definition 2.9), 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*

$$\frac{1}{2\epsilon}\phi\left[f(x + \epsilon\phi) - f(x - \epsilon\phi)\right] = \phi\phi^{\top}\nabla f(x) + O(\epsilon^2) \tag{6.3}$$

Proof. We borrow this result and proof from [18]. The proof follows from Taylor Theorem. Consider

any $x \in \mathbb{R}^n$ and $r \in \mathbb{R}$. Then

$$f(x + r\phi) = f(x) + r\phi^\top \nabla f(x) + \frac{r^2}{2}\phi^\top \nabla^2 f(x)\phi + O(r^3).$$

Taking $r = \epsilon$ and $r = -\epsilon$ as two separate instances of the above equality and then subtracting them yields

$$\begin{aligned} f(x + \epsilon\phi) - f(x - \epsilon\phi) = f(x) + \epsilon\phi^\top \nabla f(x) + \frac{\epsilon^2}{2}\phi^\top \nabla^2 f(x)\phi + O(\epsilon^3) \\ - \left[f(x) - \epsilon\phi^\top \nabla f(x) + \frac{(-\epsilon)^2}{2}\phi^\top \nabla^2 f(x)\ \phi + O(\epsilon^3) \right]. \end{aligned}$$

Cancelling out the like terms and rearranging we conclude:

$$\begin{aligned} f(x + \epsilon\phi) - f(x - \epsilon\phi) &= 2\epsilon\phi^\top \nabla f(x) + O(\epsilon^3) \\ \frac{1}{2\epsilon}\phi\left[f(x + \epsilon\phi) - f(x - \epsilon\phi)\right] &= \phi\phi^\top \nabla f(x) + O(\epsilon^2). \end{aligned}$$

□

We define the final equation in the above proof as the estimated gradient:

$$\hat{\nabla} f(x; \phi, \epsilon) := \frac{1}{2\epsilon}\phi\left[f(x + \epsilon\phi) - f(x - \epsilon\phi)\right] = \phi\phi^\top \nabla f(x) + O(\epsilon^2). \tag{6.4}$$

Note that this equation contains two perturbed, zero-order evaluations of the function $f(x)$, and shows the error between those evaluations and the precise gradient $\nabla f(x)$. With our setup, we can measure these evaluations, and thus use them to compute an approximate gradient to form each row of the Jacobian. Since Lemma 6.1 requires the use of Taylor's Theorem, we need additional assumptions on the output mappings π_i to ensure that this is a valid approach.

Assumption 6. *For each agent i, each output mapping $\pi_{ij} : \Omega \times \mathcal{W} \to \mathbb{R},\ j \in \{1,\ \dots\ , l_i\}$ is $\mathbb{C}^2$ and has Lipschitz continuous $\nabla\pi_{ij}(\mathbf{u}, w)$ and $\nabla^2\pi_{ij}(\mathbf{u}, w)$.*

In classical implementations of SPSA, the vector ϕ is generated by sampling independent and identically distributed random variables [25, 26]. Recent work in model-free methods shows that using a quasi-stochastic approach, one where all the processes generating the perturbation vector are deterministic but "appear" stochastic, helps reduce variance and improve convergence [18, 27]. We note that the algorithm we will develop iterates in the same manner as Algorithm 2, and we denote iterations as $k \in \{1,\ \dots\ , K\}$. We will later formalize how these function evaluations and perturbations fit into Algorithm 2 exactly, for now we assume we are evaluating some function while iterating through the algorithm. At each iteration, we aim to sample a perturbation vector $\phi_k \in \mathbb{R}^n$ that we will use two generate the two function evaluations by replacing ϕ in (6.4).

Assumption 7. *The perturbation vector $\phi_k \in \mathbb{R}^n$, at the iteration $k \in \{1,\ \dots\ K\}$, 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 periodic with period $T \in \mathbb{R}$ and satisfies the following for any $t \in \mathbb{R}$:*

$$\frac{1}{T}\int_t^{t+T} \phi(\tau)\phi(\tau)^\top d\tau = I_n. \tag{6.5}$$

Assumption 7 enforces that the perturbation signal is chosen such that $\phi(t)\phi(t)^\top = I_n$ on average. From the right hand side of (6.4), this would correspond to the average two function evaluation having an error of $O(\epsilon^2)$ from a true, instantaneous gradient evaluation. The goal then, is to construct an approximate Jacobian to replace the one used in step 1 of Algorithm 2 using these two function evaluations. We thus obtain the average error for the gradient by rearranging (6.4):

$$\hat{\nabla} f(x;\phi,\epsilon) - \nabla f(x) = O(\epsilon^2), \tag{6.6}$$

i.e., if we estimate each row of the Jacobian using a model-free, quasi-stochastic perturbation approach, we expect each row to have an average error of $O(\epsilon^2)$ from the corresponding row of the true Jacobian. Note that if the mapping f is quadratic in x then $O(\epsilon^2) = 0$.

We now outline a modification to Algorithm 2, estimating the Jacobian at each time step. In addition to the notation from Algorithm 2, we introduce the following notation. 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 matrix $\hat{\Pi}_{i,k} \in \mathbb{R}^{l_i \times n}$ is the submatrix of the overall approximate Jacobian ($\hat{\Pi} \in \mathbb{R}^{l \times n}$) that player i computes. Note that player i is responsible for computing Jacobians $\hat{\Pi}_{1i,k} \ \cdots \ \hat{\Pi}_{Ni,k}$ (each of these is a submatrix of $\hat{\Pi}_{i,k}$), but receives Jacobians $\hat{\Pi}_{i1,k} \ \cdots \ \hat{\Pi}_{iN,k}$ from the other players for its primal step during each iteration. We now present the algorithm.

Algorithm 3 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: Player $i \in \mathcal{N}$

Step 1: Exploration - Applies control $u_{i,k+}$ and $u_{i,k-}$ to the system. Receives their corresponding output measurements $y_{i,k+}$ and $y_{i,k-}$, and the *total* perturbation vector ϕ_k.

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

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

$\hat{\Pi}_{ji,k} \leftarrow \hat{\nabla}^\top_{\mathbf{u}} \pi_{ij}$

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

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

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

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

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

Remark 6. *We do not formalize the choice of T and T_s in Assumption 7. Developing techniques to choose them appropriately and studying the impact of these choices is an area of future work. We note that we want our sampled perturbation vector to be quasi-stochastic, and thus we can loosely*

impose that

1. *T is significantly larger than T_s so that every period we see during runtime contains a large number of samples,*
2. *K is significantly larger than $\frac{T}{T_s}$ so that we see several periods of $\phi(t)$ over a single run of the algorithm, and*
3. *K is not an integer multiple of $\frac{T}{T_s}$ so that we see variation in periodically chosen values.*

Note that the algorithm does not necessarily converge in its estimate of the Jacobian (unless all derivatives higher than the second are identically zero, for example, if π were a quadratic). The *average* primal step (across any run with a large number of iterations) should be an accurate estimate of the Jacobian, but each instantaneous Jacobian could vary. We introduce a new pseudogradient operator:

$$H_{w,\hat{\Pi}}(\mathbf{u}) = F(\mathbf{u}) + \mathfrak{E}\big(\hat{\Pi}^\top \partial_{\mathbf{y}} g(\mathbf{y})^\top\big). \tag{6.7}$$

It should be noted that the pseudogradient operator $H_{w,\hat{\Pi}}(\mathbf{u})$ is no longer fixed for a given $(\mathbf{u}, w)$. The Jacobian matrix $\hat{\Pi}$ varies after each iteration as a function of the chosen perturbation vector $\phi(t)$, defined in Assumption 7. This variation exists even if we initialize the algorithm at a vKKT point of the game, and thus we would intuitively expect Algorithm 3 to converge to a limit set rather than a single limit point, within a bounded distance of a point satisfying the vKKT conditions (4.9). We develop this bound more formally in the following subsection.

6.2.1 Convergence Analysis

The key idea behind proving the convergence of Algorithm 3 is that the algorithm tracks a hypothetical, "golden" algorithm that knows the Jacobian precisely. We can show that the golden algorithm converges to a vKKT point of the game (i.e., a point that satisfies the variational KKT conditions (4.9)). Since Algorithm 3 tracks the iterates of the golden algorithm, it then converges to a limit set within a bounded distance around the vKKT point that the golden algorithm would converge to.

We first show the convergence of the golden algorithm. The golden algorithm, in this case, would be Algorithm 2, but replacing the nominal Jacobians Π_{ij} with the precisely known Jacobians $\partial_{u_i}\pi_j$ (the nominal approximation was explained in Section 4.2, culminating in (4.10)). To this end, we use the true pseudogradient operator H_w defined in (4.7). We start by representing the golden algorithm in its forward backward form. We first express the KKT conditions. A point $\mathbf{u}^*$ and a corresponding Lagrange multiplier λ^* satisfying the vKKT conditions (4.9) must also satisfy $\mathrm{col}(\mathbf{u}^*, \lambda^*) \in \mathrm{zer}(\mathfrak{A} + \mathfrak{B})$ where

$$\begin{aligned} &\mathfrak{A} : \mathrm{col}(\mathbf{u}, \lambda) \mapsto \mathrm{col}(H_w(\mathbf{u}), -b) \\ &\mathfrak{B} : \mathrm{col}(\mathbf{u}, \lambda) \mapsto \mathrm{col}(-A^\top \lambda + N_\Omega(\mathbf{u}), A\mathbf{u} + N_{\mathbb{R}^m_+}(\lambda)). \end{aligned} \tag{6.8}$$

We can then rewrite the golden algorithm as a forward-backward iteration, similar to (4.24):

$$\varpi_{k+1} = (\mathrm{Id} + \Phi^{-1}\bar{\mathfrak{B}})^{-1}(\mathrm{Id} - \Phi^{-1}\bar{\mathfrak{A}})(\varpi_k), \tag{6.9}$$

where the operators $\bar{\mathfrak{A}}$ and $\bar{\mathfrak{B}}$ are defined as

$$\begin{aligned} \bar{\mathfrak{A}} &: \varpi \mapsto \operatorname{col}(H_w(\mathbf{u}), \mathbf{0}, \bar{L}\bar{\lambda} - \bar{b}) \\ \bar{\mathfrak{B}} &: \varpi \mapsto N_\Omega(\mathbf{u}) \times \mathbf{0} \times N_{\mathbb{R}_+^{mN}}(\bar{\lambda}) + \Psi\varpi, \end{aligned} \tag{6.10}$$

and the matrices Φ and Ψ are defined identically to (3.10) as

$$\Phi = \begin{bmatrix} \bar{\tau}^{-1} & \mathbf{0} & \Lambda^\top \\ \mathbf{0} & \bar{\nu}^{-1} & \bar{L} \\ \Lambda & \bar{L} & \bar{\sigma}^{-1} \end{bmatrix}, \ \Psi = \begin{bmatrix} \mathbf{0} & \mathbf{0} & -\Lambda^\top \\ \mathbf{0} & \mathbf{0} & -\bar{L} \\ \Lambda & \bar{L} & L \end{bmatrix}. \tag{6.11}$$

Note again that the difference between (6.10) and (4.19) is the use of the precisely known pseudogradient H_w. We require an additional assumption to be placed on this pseudogradient to ensure convergence.

Assumption 8. *The pseudogradient $H_w(\mathbf{u})$ defined in* (4.7) *is $\bar{\eta}$-strongly monotone and $\bar{\theta}$-Lipschitz continuous.*

With this added assumption, we can show the convergence of the golden algorithm.

Proposition 6.1. *Suppose Assumptions 1, 3, 4, and 8 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 8. Take $\delta > \frac{1}{2\beta}$. Suppose that τ_i, ν_i, σ_i are chosen to satisfy* (4.25)*. Then with Algorithm 2, modified such that each player has access to their corresponding submatrix of the precise Jacobian $\partial_{\mathbf{u}}\pi(\mathbf{u}, w)$ at each iteration rather than the nominal one, each player's local strategy $u_{i,k}$ converges to its corresponding component of a point satisfying the vKKT conditions* (4.9)*, 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 proof for this is near-identical to the proof for Theorem 4.2, and builds off a similarly modified Theorem 4.1. We simply redefine the operators $\mathfrak{A}$, $\mathfrak{B}$, $\bar{\mathfrak{A}}$, and $\bar{\mathfrak{B}}$, and repeat the proof for the two Theorems and their intermediary Lemmas. An important note is that while Theorem 4.2 showed the convergence to a vGNE of the approximated game, this was because the approximation of the Jacobian as a fixed matrix is akin to assuming that the cost function $f_i(u_i, \mathbf{u}_{-i}) + g_i(\pi_i(u_i, \mathbf{u}_{-i}), \pi_{-i}(u_i, \mathbf{u}_{-i}), w)$ is convex for all u_i given a fixed $\mathbf{u}_{-i}$ and w (elaborated on under (4.11)). By using the precisely known Jacobian, we make no such assumption and thus Proposition 6.1 only guarantees convergence to a point satisfying the vKKT conditions (4.9). Such a point is a candidate for a local vGNE, but it is possible that the game does not admit one. □

This concludes the first step of our convergence analysis. We showed that the golden algorithm converges to a vKKT point. The next phase of this proof is showing that Algorithm 3 approximately follows the trajectory of the golden algorithm. We denote the model-free pseudogradient operator $H_{w,\hat{\Pi}}$ as defined in (6.7). Additionally, we define equivalents of operators $\mathfrak{A}$ and $\bar{\mathfrak{A}}$ for the model-free method as

$$\hat{\mathfrak{A}} : \operatorname{col}(\mathbf{u}, \lambda) \mapsto \operatorname{col}(H_{w,\hat{\Pi}}(\mathbf{u}), -b) \tag{6.12}$$

$$\hat{\bar{\mathfrak{A}}} : \varpi \mapsto \operatorname{col}(H_{w,\hat{\Pi}}(\mathbf{u}), \mathbf{0}, \bar{L}\bar{\lambda} - \bar{b}). \tag{6.13}$$

Algorithm 3 can then be similarly written as a forward-backward iteration:

$$\hat{\varpi}_{k+1} = (\mathrm{Id} + \Phi^{-1}\bar{\mathfrak{B}})^{-1}(\mathrm{Id} - \Phi^{-1}\hat{\bar{\mathfrak{A}}})(\hat{\varpi}_k), \tag{6.14}$$

Now consider some arbitrary point ϖ_k. Suppose that we apply one iteration of the golden algorithm (6.9) to obtain ϖ_{k+1}. Similarly, we apply an iteration of Algorithm 3, i.e., an iteration of (6.14), to obtain $\hat{\varpi}_{k+1}$. Similar to Section 4.3.2, we denote $T_2 := (\mathrm{Id} + \Phi^{-1}\bar{\mathfrak{B}})^{-1}$, $T_1 := (\mathrm{Id} - \Phi^{-1}\bar{\mathfrak{A}})$, and $\hat{T}_1 := (\mathrm{Id} - \Phi^{-1}\hat{\bar{\mathfrak{A}}})$. We check the difference between the two iterates. From Lemmas 4.2 and 2.5, we know that T_2 is firmly nonexpansive, and thus by Definition 2.14

$$\|\varpi_{k+1} - \hat{\varpi}_{k+1}\|_\Phi^2 = \|T_2 T_1 \varpi_k - T_2 \hat{T}_1 \hat{\varpi}_k\|_\Phi^2 \leq \|T_1 \varpi_k - \hat{T}_1 \hat{\varpi}_k\|_\Phi^2.$$

By the definition of T_1 above, we have

$$\begin{aligned}\|T_1 \varpi_{k+1} - \hat{T}_1 \hat{\varpi}_{k+1}\|_\Phi^2 &= \|(\mathrm{Id} - \Phi^{-1}\bar{\mathfrak{A}})(\varpi_k) - (\mathrm{Id} - \Phi^{-1}\hat{\bar{\mathfrak{A}}})(\varpi_k)\|_\Phi^2 \\ &= \|\Phi^{-1}(\bar{\mathfrak{A}} - \hat{\bar{\mathfrak{A}}})(\varpi_k)\|_\Phi^2.\end{aligned}$$

Similar to the proof for Lemma 4.4, we change the matrix that induces the norm by utilizing the fact that $\|\Phi^{-1}\|_2 \leq \frac{1}{\delta}$, with δ chosen as in Proposition 6.1:

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

We proceed from the last line of the above by expanding the definitions for the operators $\hat{\mathfrak{A}}$ and $\hat{\bar{\mathfrak{A}}}$ and thus

$$\begin{aligned}& \|\varpi_{k+1} - \hat{\varpi}_{k+1}\|_\Phi^2 \\ \leq\ & \frac{1}{\delta}\|(\bar{\mathfrak{A}} - \hat{\bar{\mathfrak{A}}})(\varpi_k)\|_2^2 \\ =\ & \frac{1}{\delta}\|\mathrm{col}(H_w(\mathbf{u}) - H_{w,\hat{\Pi}}(\mathbf{u}), \mathbf{0}, \mathbf{0})\|_2^2 \\ =\ & \frac{1}{\delta}\left\|\mathfrak{E}\left[\left(\partial_{\mathbf{u}}\pi(\mathbf{u}, w) - \hat{\Pi}\right)^\top \partial_{\mathbf{y}} g(\pi(\mathbf{u}, w))^\top\right]\right\|_2^2 \\ \leq\ & \frac{E_{\max}}{\delta}\left\|\left(\partial_{\mathbf{u}}\pi(\mathbf{u}, w) - \hat{\Pi}\right)^\top \partial_{\mathbf{y}} g(\pi(\mathbf{u}, w))^\top\right\|_2^2,\end{aligned} \tag{6.15}$$

where the last two lines follow from the definition of the pseudogradient operators H_w and $H_{w,\hat{\Pi}}$ in (4.7) and (6.7) respectively, and noting the fact that the extraction operator $\mathfrak{E}$ is linear and bounded, with $E_{\max}$ being the upper bound defined in (B.2). Applying submultiplicativity to (6.15) we obtain

$$\|\varpi_{k+1} - \hat{\varpi}_{k+1}\|_\Phi^2 \leq \frac{E_{\max}}{\delta}\|\partial_{\mathbf{y}} g(\pi(\mathbf{u}, w))\|_2^2 \left\|\partial_{\mathbf{u}}\pi(\mathbf{u}, w) - \hat{\Pi}\right\|_2^2. \tag{6.16}$$

From here we can conclude that the distance between the iterates ϖ_{k+1} and $\hat{\varpi}_{k+1}$ is bounded if the distance between the true Jacobian and the estimated Jacobian is bounded. We next aim to show that the latter indeed is bounded. We begin by noting $\hat{\Pi}$ here is specific to one iteration of the algorithm, and the accuracy of the estimate varies quasi-stochastically over the course of the algorithm's run. Thus our approach is to find a worst-case error bound for the inaccuracy of $\hat{\Pi}$, and our algorithm will thus remain within a finite distance of iterates of the golden algorithm, upper-bounded by the worst case inaccuracy. We start from (6.2), which is used to construct each row of the Jacobian $\hat{\Pi}$ for any iteration of Algorithm 3. By (6.4), we know that the approximated row $\hat{\nabla}_{\mathbf{u}}^{\top}\pi_{ij}$ (defined in Algorithm 3) satisfies

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

where $\nabla_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w)$ is the true value of the row (extracted from the corresponding row of the true Jacobian $\partial_{\mathbf{u}}\pi(\mathbf{u}, w)$), the vector ϕ corresponds to the perturbation chosen for that estimate, and ϵ is the magnitude of the perturbation applied while constructing the estimate. Hence the error between a row of the estimate and that of the true Jacobian can be written as

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

By Assumption 7, ϕ is chosen from a periodic signal $\phi(t)$ periodic signal such that the average value of $\phi(t)\phi(t)^{\top}$ is I_n, which minimizes this error. We choose a worst case value of $\phi(t)$ to instead maximize the distance of $\phi(t)\phi(t)^{\top}$ from I_n, and denote that value $\bar{\phi}$. Then, for any iteration of Algorithm 3

$$\|\hat{\nabla}_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w) - \nabla_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w)\|_2 \leq \|(\bar{\phi}\bar{\phi}^{\top} - I_n)\|_2\|\nabla_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w)\|_2 + O(\epsilon^2). \tag{6.17}$$

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

Assumption 9. *Given an output mapping $\pi(\mathbf{u}, w)$, defined in* (4.1)*, 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 $i \in \mathcal{N}$*

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

Combining the bound (6.17) with Assumption 9, we obtain

$$\|\hat{\nabla}_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w) - \nabla_{\mathbf{u}}\pi_{ij}(\mathbf{u}, w)\|_2 \leq \sigma_{\max}(\bar{\phi}\bar{\phi}^{\top} - I_n)S_{\max} + O(\epsilon^2), \tag{6.18}$$

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 $\partial_{\mathbf{u}}\pi(\mathbf{u}, w)$. We now seek to relate this row bound to the overall matrix bound we developed earlier, to conclude this proof. To combine (6.16) and (6.18), 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 simply multiplying the right side of (6.18) by the number of rows (l), and thus

$$\|\varpi_{k+1} - \hat{\varpi}_{k+1}\|_\Phi^2 \leq \frac{E_{\max} l^2}{\delta} \|\partial_{\mathbf{y}} g(\pi(\mathbf{u}, w))\|_2^2 \left(\sigma_{\max}(\bar{\phi}\bar{\phi}^\top - I_n) S_{\max} + O(\epsilon^2)\right)^2 \tag{6.19}$$

Theorem 6.1. *Suppose that Assumptions 1, 3, 4, 6-9 are satisfied. 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 8. Take* $\delta > \frac{1}{2\beta}$. *Suppose that* τ_i, ν_i, σ_i *are chosen to satisfy* (4.25). *Then with Algorithm 3, the overall action profile* $\mathbf{u}$ *converges to an open ball* $\mathcal{B}_\kappa^n(\mathbf{u}^*)$ *where* $\mathbf{u}^*$ *is a point satisfying the vKKT conditions* (4.9) *(for some unknown value of the multipliers* λ^**), and the radius* κ *is defined by*

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

With that, we can conclude our convergence analysis. We showed that if we start with some point ϖ_k, our next iterate with Algorithm 3 will be within a finite bound of an iterate computed with access to the perfect information Jacobian. We also showed that Algorithm 2, when used with perfect knowledge of the Jacobian, will converge precisely to a vKKT point of the game. Combining those two facts let us conclude that Algorithm 3 converges within a bounded distance of a vKKT point of the game, and thus provides a set of candidate vGNEs.

6.3 Choice of Perturbation Signal

In this section, we illustrate one specific choice of perturbation signal that satisfies Assumption 7. Note that the signal is arbitrarily chosen, and *any* signal that satisfies the assumption will reach within a bounded error of a vKKT point of the problem. Studying the effects of varying this choice is an area of future research.

Proposition 6.2. *The perturbation signal*

$$\phi_i(t) = \sqrt{2}\sin(\omega_i t),\ i = 1,\ \ldots\ , n, \tag{6.20}$$

with $\omega_i \neq \omega_j \forall i \neq j$ *satisfies Assumption 7, given some* T *that is a common integer multiple of the sinusoids' periods.*

Proof. We simply evaluate the integral (6.5). Along the diagonal elements, we get:

$$\begin{aligned}
& \frac{2}{T}\int_t^{t+T} \sin^2(\omega_i \tau) d\tau \\
= \ & \frac{1}{T}\int_t^{t+T} 1 - \cos(2\omega_i \tau) d\tau \\
= \ & \frac{1}{T}\left[\tau\Big|_t^{t+T} - \frac{1}{2\omega_i}\sin(2\omega_i\tau)\Big|_t^{t+T}\right] \\
= \ & 1,
\end{aligned}$$

since the second term of the sum evaluates to 0 due to the signal being T-periodic. For the off-diagonal elements, we require the following sum/difference angle identities:

$$\begin{aligned}
\cos(\alpha+\beta) &= \cos\alpha\cos\beta - \sin\alpha\sin\beta \\
\cos(\alpha-\beta) &= \cos\alpha\cos\beta + \sin\alpha\sin\beta \\
\sin(\alpha+\beta) &= \sin\alpha\cos\beta + \cos\alpha\sin\beta \\
\sin(\alpha-\beta) &= \sin\alpha\cos\beta - \cos\alpha\sin\beta.
\end{aligned}$$

We then evaluate the integrals:

$$\begin{aligned}
& \frac{2}{T}\int_t^{t+T} \sin(\omega_i\tau)\sin(\omega_j\tau)d\tau \\
= \; & \frac{1}{T}\int_t^{t+T} \cos\left[(\omega_i-\omega_j)\tau\right] - \cos\left[(\omega_i+\omega_j)\tau\right] d\tau \\
= \; & \frac{\sin\left[(\omega_i-\omega_j)\tau\right]}{\omega_i-\omega_j}\Big|_t^{t+T} - \frac{\sin\left[(\omega_i+\omega_j)\tau\right]}{\omega_i+\omega_j}\Big|_t^{t+T} \\
= \; & \left(\frac{\sin(\omega_i\tau)\cos(\omega_j\tau) - \cos(\omega_i\tau)\sin(\omega_j\tau)}{\omega_i-\omega_j}\right)\Big|_t^{t+T} \\
& \qquad - \left(\frac{\sin(\omega_i\tau)\cos(\omega_j\tau) + \cos(\omega_i\tau)\sin(\omega_j\tau)}{\omega_i+\omega_j}\right)\Big|_t^{t+T} \\
= \; & 0,
\end{aligned}$$

because the sinusoids are all T-periodic. Thus the integral evaluates to I_n. □

Intuitively, sampling a single perturbation ϕ_k from the sinusoidal signal $\phi(t)$ is akin to choosing a pair of deterministic, varying (hence quasi-stochastic) points from the boundary of an n-dimensional ball of radius $\epsilon\sqrt{2}$ centred at the currently chosen control input $\mathbf{u}$. We then use that pair of points to linearly estimate the instantaneous Jacobian.

With that, we conclude this chapter. We introduced a framework that allows online, model-free optimization of our game-theoretic optimization problem. The key advantage of this technique is its ability to converge to a reasonable estimate of a vKKT point despite not having access to a nominal model of the system's input-output sensitivities. The major limitations of this framework are that it introduces more communication overhead between agents, it can require extensive tuning for the perturbation signal to get a good quality approximation, and the fact that it converges to a limit set rather than a limit point. In the next chapter, we will present simulation results that compare both algorithms to offline methods as well as to each other, showcasing the advantages and limitations of both.

Chapter 7

Simulations and Results

7.1 Academic Example

To validate the established theoretical framework and our presented algorithms, we first consider a simple 2-player example with quadratic costs on both decisions and outputs. Each player controls $u_i \in \mathbb{R}$, a scalar decision variable. The action set for the agents' decisions is defined as $\Omega = \Omega_1 \times \Omega_2$, with $\Omega_i = \{u \in \mathbb{R} : -5 \leq u \leq 5\}$ for $i \in \{1, 2\}$. The output mapping is defined by

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \pi_1(\mathbf{u}, w) \\ \pi_2(\mathbf{u}, w) \end{bmatrix} = \begin{bmatrix} w_1 u_1 + w_2^2 u_2 \\ w_2 u_1 + w_1^2 u_2 \end{bmatrix} \tag{7.1}$$

where $w = (w_1, w_2) \in [0, 1]^2$ is the unknown-but-bounded disturbance, which we assume has a nominal value $\bar{w} = (0.5, 0.5)$. The coupling constraints are an upper and lower bound on the total action allowed:

$$-1 \leq u_1 + u_2 \leq 1.$$

We rearrange the above constraints to obtain

$$\begin{aligned} u_1 + u_2 &\geq -1 \\ \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} &\geq -1, \end{aligned}$$

and

$$\begin{aligned} -u_1 - u_2 &\geq -1 \\ \begin{bmatrix} -1 & -1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} &\geq -1. \end{aligned}$$

Combining these into a single, affine inequality we obtain

$$\begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \geq \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

Thus we have sets $\mathcal{U}$ and $\mathcal{U}_i(\mathbf{u}_{-i})$ of the form defined in (3.1), with $A_i = [-1\ 1]^\top$ and $b_i = \frac{1}{2}[-1\ \ -1]^\top$ for $i \in \{1, 2\}$. Players 1 and 2 are respectively interested in solving the following optimization problems:

$$\begin{aligned} \min_{u_1}\ & \mathbf{u}^\top \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right] \mathbf{u} + \left[\begin{smallmatrix} 0 & 2 \end{smallmatrix}\right] \mathbf{u} + \mathbf{y}^\top \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right] \mathbf{y} + \left[\begin{smallmatrix} 0 & 1 \end{smallmatrix}\right] \mathbf{y} \\ & y_1 = w_1 u_1 + w_2^2 u_2 \\ & u_1 \in \mathcal{U}_1(u_2), \end{aligned} \tag{7.2}$$

$$\begin{aligned} \min_{u_2}\ & \mathbf{u}^\top \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right] \mathbf{u} + \left[\begin{smallmatrix} 3 & 0 \end{smallmatrix}\right] \mathbf{u} + \mathbf{y}^\top \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right] \mathbf{y} + \left[\begin{smallmatrix} 0 & 1 \end{smallmatrix}\right] \mathbf{y} \\ & y_2 = w_2 u_1 + w_1^2 u_2 \\ & u_2 \in \mathcal{U}_2(u_1). \end{aligned} \tag{7.3}$$

The Jacobian of π with respect to the control input is given by

$$\partial_{\mathbf{u}} \pi(\mathbf{u}, w) = \begin{bmatrix} w_1 & w_2^2 \\ w_2 & w_1^2 \end{bmatrix}, \tag{7.4}$$

and we select $\Pi = \partial_{\mathbf{u}} \pi(0, \bar{w})$ as our nominal Jacobian. Finding the pseudogradient $F(\mathbf{u})$ and the Jacobians $\partial_{\mathbf{u}} \pi(\mathbf{u}, w)$ and $\partial_{\mathbf{y}} g(\mathbf{y})$ for the above costs gives us the operator $H_w(\mathbf{u})$, which can be represented as

$$H_w(\mathbf{u}) = \begin{bmatrix} 2u_1 + u_2 + 2w_1 \pi_1(\mathbf{u}, w) + 2w_2 \pi_2(\mathbf{u}, w) + w_2 \\ u_1 + 2u_2 + 2w_2^2 \pi_1(\mathbf{u}, w) + 2w_1^2 \pi_2(\mathbf{u}, w) + w_1^2 \end{bmatrix},$$

with π_1 and π_2 as defined in (7.1). We substitute the values for w_1 and w_2 by using the nominal Jacobian $\Pi = \partial_{\mathbf{u}} \pi(0, \bar{w})$, which yields

$$H_{w,\Pi}(\mathbf{u}) = \begin{bmatrix} 2u_1 + u_2 + \pi_1(\mathbf{u}, w) + \pi_2(\mathbf{u}, w) + \frac{1}{2} \\ u_1 + 2u_2 + \frac{1}{2}\pi_1(\mathbf{u}, w) + \frac{1}{2}\pi_2(\mathbf{u}, w) + \frac{1}{4} \end{bmatrix}.$$

Differentiating this to get the Jacobian $\partial_{\mathbf{u}} H_w(\mathbf{u})$ we obtain

$$\partial_{\mathbf{u}} H_{w,\Pi}(\mathbf{u}) = \begin{bmatrix} 2 + \partial_{u_1}\pi_1(\mathbf{u}, w) + \partial_{u_1}\pi_2(\mathbf{u}, w) & 1 + \partial_{u_2}\pi_1(\mathbf{u}, w) + \partial_{u_2}\pi_2(\mathbf{u}, w) \\ 1 + \frac{1}{2}\partial_{u_1}\pi_1(\mathbf{u}, w) + \frac{1}{2}\partial_{u_1}\pi_2(\mathbf{u}, w) & 2 + \frac{1}{2}\partial_{u_2}\pi_1(\mathbf{u}, w) + \frac{1}{2}\partial_{u_2}\pi_2(\mathbf{u}, w) \end{bmatrix}$$

From here, we aim to parametrize the Jacobian using the linear fractional transformation, as outlined in Section 5.2. From the definition of the Jacobian (7.4) we know that

$$\partial_{\mathbf{u}} H_{w,\Pi}(\mathbf{u}) = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} w_1 + w_2 & w_2^2 + w_1^2 \\ \frac{1}{2}w_1 + \frac{1}{2}w_2 & \frac{1}{2}w_2^2 + \frac{1}{2}w_1^2 \end{bmatrix}.$$

The uncertainty for this Jacobian is contained in the terms $(w_1 + w_2)$ and $(w_1^2 + w_2^2)$, both of which are in the range $[0, 2]$. We can thus choose to parametrize the uncertainty in this problem with two variables:

$$\partial_{\mathbf{u}} H_{w,\Pi}(\mathbf{u}) = \begin{bmatrix} 2 + \delta_1 & 1 + \delta_2 \\ 1 + \frac{1}{2}\delta_1 & 2 + \frac{1}{2}\delta_2 \end{bmatrix}. \tag{7.5}$$

Note the introduction of conservatism in this uncertainty parametrization: given a fixed δ_1, the

true Jacobian has δ_2 constrained to a subset of the values that it can take in the LFT, whereas this parametrization lets it vary independently (i.e., if $w_1 = w_2 = 0.2$, we know that $\delta_1 = 0.4$ and $\delta_2 = 0.08$, but our conservative model admits all possible $(0.4, \delta_2 \in [0, 2])$). Thus the true Jacobian is contained within this LFT. We can define our parameter block with the set

$$\mathbf{\Delta} = \left\{ \Delta = \begin{bmatrix} \delta_1 & 0 \\ 0 & \delta_2 \end{bmatrix} : \delta_1,\ \delta_2 \in [0, 2] \right\}$$

We use the iterative technique outlined in Example 2, which we can verify by direct computation, to obtain the following LFT parametrization for the Jacobian (7.5):

$$M = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix},\ B = \begin{bmatrix} 1 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

With these matrices, we now have a Jacobian uncertainty set $\mathcal{J}^{\text{lft}}$, as defined in Definition 5.1. Following the logic for *block-structured uncertainty* in Section 5.2.1, with each block being a scalar (so $I_s = 1$) *unstructured monotone and Lipschitz uncertainty* $\mu = 0 \leq \delta_i \leq L = 2$, we can construct our convex cone:

$$\mathbf{\Theta} = \left\{ \Theta = \begin{bmatrix} 0 & 0 & 2\phi_1 & 0 \\ 0 & 0 & 0 & 2\phi_2 \\ 2\phi_1 & 0 & -2\phi_1 & 0 \\ 0 & 2\phi_2 & 0 & -2\phi_2 \end{bmatrix} : \phi_1, \phi_2 \geq 0 \right\}.$$

With that, we are equipped to use the matrix inequality outlined in Proposition 5.2. We now use the inequality to determine that the operator $H_{w,\Pi}(\mathbf{u})$ satisfies Assumption 5 with $\overline{\eta} = 0.2131$ and $\overline{\theta} = 7.5162$. Note that Assumptions 1 and 4 are satisfied by our choice of cost functions and output mapping, and Assumption 3 simply requires that the two players have one edge to communicate their Lagrange multipliers. Thus, the robust OA vGNE-seeking algorithm (Algorithm 2) is applicable. We present the results for the algorithm in Section 7.2.1, alongside comparisons to other algorithms.

7.2 Academic Example with Jacobian Estimation

In Section 7.1, we verified that 2 can be applied to an academic example. We now illustrate the Jacobian estimation framework, Algorithm 3 using the same cost functions, coupling constraints, and output mapping. We note that the most of the setup remains the same as in the previous section. The only change comes from the perturbation vector and the Jacobian approximation. We choose the following perturbation signal for the algorithm:

$$\phi(t) = \begin{bmatrix} \sqrt(2)\sin(10\pi t) \\ \sqrt(2)\sin(20\pi t) \end{bmatrix}. \tag{7.6}$$

We set the time-horizon of the simulation as $t \in [0, 6]$, and we sample 600 points evenly. This corresponds to a sampling period $T_s = \frac{1}{100}$, and a signal period of $T = \frac{1}{5}$, which corresponds to the

loose guidelines for sampling that we set in Remark 6. We set the disturbance as $w_1 = w_2 = 0.75$. For the sake of comparison, we simulate four different approaches:

1. A variation of Algorithm 2 where the Jacobian is known precisely, by simply subtituting the set-point $w_1 = w_2 = 0.75$ into (7.4).
2. The output is forecasted using a linearized output map (as described in Section 4.1.1 by computing the linear approximation of $\pi(\mathbf{u}, w)$. We assume that during this computation, players can directly measure the disturbance $w \in \mathbb{R}^2$ with an error equivalent to a zero-mean normal distribution with a variance of 0.0001.
3. Algorithm 2 with the nominal Jacobian and validation as described in Section 7.1.
4. Algorithm 3 with the setup as described in this section.

We simulate the four approaches for 600 iterations, 50 times with randomized initial conditions, and show our results in Figure 7.1.

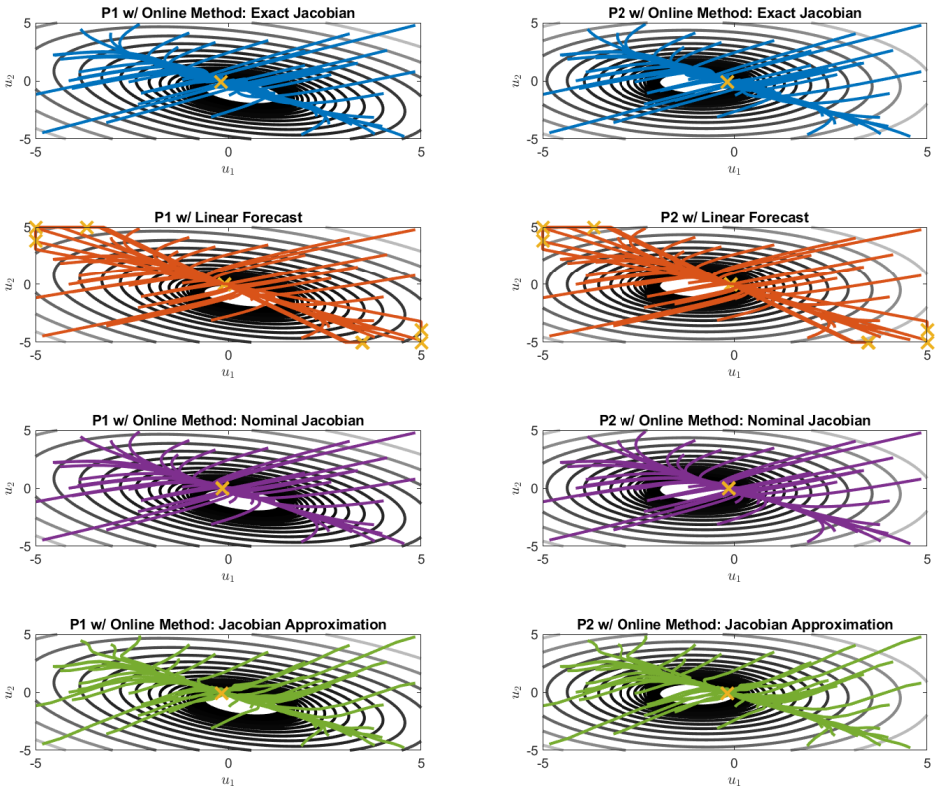

Figure 7.1: The 2-player game's decision trajectories over 600 iterations for 50 randomly generated initial conditions. The symbols and trajectories are explained in the following section.

7.2.1 Discussion of Simulation Results

Figure 7.1 is a comparison of the four different methods we used to compute the OA vGNE of the problem. All the plots show decision trajectories over the 600 iterations for each of the 50 randomly generated initial conditions. The two columns correspond to the two players, with each plot in a column being overlayed with a contour map of that player's cost function. The coloured curves mark the trajectories that the two players iterated through during the simulations, and a yellow X marks the final point of a trajectory. Intuitively, we can see that a vGNE lies somewhere around the intersection of both players' quadratic contours, as a deviation from such a point would correspond to a cost increase for one or both players.

Each row of the plot corresponds to one of the four algorithms we simulated with. The first row corresponds to a method with a perfect forecast for the Jacobian. The second row corresponds to one with a linear forecast; note that the linear forecast mispredicts the optimal trajectory and causes the decisions to diverge for some subset of the initial conditions, showcasing the need for an online, feedback-based method. The bottom two plots correspond to the online method with a nominal Jacobian (as computed in Section 7.1) and the Jacobian Estimation method (outlined in Section 7.2) respectively. Both methods robustly converge to a limiting point, though one would intuitively assume that the perfect information case corresponds to the best-case cost reduction. We thus note the costs corresponding to each optimum in Table 7.2.1. The numerical results match the

Method	(P_1,P_2) Actions	(P_1,P_2) Cost
Online, Exact Jacobian	$[-0.1886, -0.0755]^\top$	$[-0.1604, -0.5753]^\top$
Offline, Forecast Jacobian	Diverges	N/A
Online, Nominal Jacobian	$[-0.1509, 0.0000]^\top$	$[-0.0422, -0.4948]^\top$
Online, Estimated Jacobian	$[-0.1727, -0.0649]^\top$	$[-0.1494, -0.5378]^\top$

Table 7.1: Player actions and their corresponding costs, as calculated by the 4 different algorithms.

intuition presented. Perfect knowledge of the Jacobian provides the best-case cost reduction among the algorithms tested. The linear forecast model does not converge for some of the initializations. The nominal Jacobian has a worse cost than the model-free Jacobian does in this context.

7.3 2-Robot Game with Double Integrator Dynamics

In this section, we study the application that we used in Section 6.1 to motivate the model-free Jacobian framework. To recap, the game is a simple, 2-robot problem with no disturbances or coupling constraints. The two robots, $i \in \{1, 2\}$, each seek to approach a target position $\bar{r}_i \in \mathbb{R}^2$. In Section 6.1, we did not go into detail on the robot dynamics and controller, which we expand on here. Each robot obeys the following double-integrator dynamics:

$$\begin{aligned} \dot{r}_i &= v_i \\ \dot{v}_i &= \xi_i, \end{aligned} \tag{7.7}$$

where $r_i \in \mathbb{R}^2$ is the current position of the robot, and $\xi_i \in \mathbb{R}^2$ is the control input to the system. The control input $\xi_i(t)$ is decided by a PD controller with the following transfer function:

$$\frac{\Xi_i(s)}{E_i(s)} := C_i(s) = \begin{bmatrix} K_p + K_d s & 0 \\ 0 & K_p + K_d s \end{bmatrix}, \tag{7.8}$$

where $K_p = 1$ and $K_d = 2$ are the derivative and proportional gains respectively. The signal $e_i(t) = u_i(t) - r_i(t)$ is the error signal (which must be asymptotically stabilized), where $u_i(t)$ is a position control input and $r_i(t)$ is the state output. The functions $E_i(s)$, $U_i(s)$, and $R_i(s)$ are the Laplace transforms of these three signals, and $\Xi_i(s)$ is the Laplace transform of the plant input $\xi_i(t)$. Refer to Appendix D for a proof on this controller successfully tracking the constant position input.

Each robot has access to a sensor that tells it the squared distance between the two robots, biased by some disturbance $w_i \in \mathbb{R}$ (motivated, for example, by sensor measurement error or noise on the channel that the robots communicate their distances through):

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \pi_1(u_1, u_2, w_1) \\ \pi_2(u_1, u_2, w_2) \end{bmatrix} := \pi(\mathbf{u}, w) := \begin{bmatrix} \|u_1 - u_2\|^2 + w_1 \\ \|u_2 - u_1\|^2 + w_2 \end{bmatrix}. \tag{7.9}$$

Thus the Jacobian of the output mapping at steady-state can be precisely computed as

$$\partial_{\mathbf{u}} \pi(\mathbf{u}) = 2 \begin{bmatrix} u_1^\top - u_2^\top & u_2^\top - u_1^\top \\ u_1^\top - u_2^\top & u_2^\top - u_1^\top \end{bmatrix}. \tag{7.10}$$

Note that these equations are the same as in 6.1, used here separately for ease of referencing. We now develop the cost functions that the players are trying to minimize. To minimize the distance between the robot and its target $\bar{r}_i \in \mathbb{R}^2$, we have the cost function

$$f_i(u_i) := \frac{1}{2}\|u_i - \bar{r}_i\|^2. \tag{7.11}$$

We seek to enforce the boundary constraint $y_i \leq \sqrt{d}$. This constraint could be motivated, for example, by a communication goal between the robots, causing a critical failure if they move too far away from one another. We use a logarithmic barrier function to enforce this constraint:

$$g_i(y_i) := -\log\left(d - y_i\right). \tag{7.12}$$

Each player is thus trying to choose a position control input u_i to minimize the cost function $J_i(u_i, y_i) = f_i(u_i) + g_i(y_i)$. The gradient of this cost function is

$$\begin{aligned} \nabla_{u_i} J_i(\mathbf{u}, \mathbf{y}) &= \nabla_{u_i} f_i(u_i) + \frac{\partial \pi_i}{\partial u_i}(\mathbf{u})^\top \nabla_{y_i} g_i(y_i) + \frac{\partial \pi_{-i}}{\partial u_i}(\mathbf{u})^\top \nabla_{y_{-i}} g_i(y_i) \\ \nabla_{u_i} J_i(\mathbf{u}, y_i) &= \frac{1}{2}\nabla_{u_i}\|u_i - \bar{r}_i\|^2 - \frac{\partial \pi_i}{\partial u_i}(\mathbf{u})^\top \nabla_{y_i}\left[\log(d - y_i)\right] \\ \nabla_{u_i} J_i(\mathbf{u}, y_i) &= u_i - \bar{r}_i + \frac{\partial \pi_i}{\partial u_i}(\mathbf{u})^\top \frac{1}{d - y_i} \end{aligned} \tag{7.13}$$

As before, we use u_{-i} and y_{-i} to denote the control input and the sensor output for the "other" player (from player i's perspective). Note that while the cost function has no explicit coupling between players' actions, there is inherent coupling in the learning dynamics through the Jacobian. Thus a game-theoretic approach is appropriate for the problem, as it is not composed of two independent optimization problems.

We apply three approaches to solve the problem, for the sake of comparison:

1. Computing the gradient $\nabla_{u_i} J_i$ using the precisely known Jacobian (7.10), and applying Algorithm 2, replacing the Jacobian estimate with the precise one instead. We expect this algorithm to have the best performance, due to perfect information.

2. Calculating a nominal Jacobian Π to apply Algorithm 2, replacing the Jacobian in (7.13). To compute the nominal Jacobian, we simply draw the straight line between each of the initial and final points, and compute the average gradients corresponding to these points.

3. Computing the Jacobian online using the model-free framework outlined in Chapter 6, and applying Algorithm 3.

We simulate the three algorithms with disturbances uniformly distributed as $w_1 \in [0.1, 0.4]$ and $w_2 \in [0.1, 0.3]$. We initialize the robots to $u_1 = \text{col}(0.5, 0.75)$ and $u_2 = \text{col}(1, 3)$. We simulate twice, with different target positions: $\{\bar{r}_1 = \text{col}(2, 4),\ \bar{r}_2 = \text{col}(0, -1)\}$ (Figure 7.2), and $\{\bar{r}_1 = \text{col}(2, 4),\ \bar{r}_2 = \text{col}(4, -1)\}$ (Figure 7.3).

In the figures, the blue curves represent the trajectories each robot took. The golden 'O' represents their starting point, and the golden 'X' represents the final point in their trajectory. The green 'X' shows each robot's goal. The red circle denotes the constraint boundary, centred at the midpoint between the two robots' final positions. The dashed purple line denotes the "optimality line" between the two robots; the intuitive answer to the problem would be expected to be at the intersections of the line with the final constraint boundary, a circle of radius $\sqrt{d}$ centred at the midpoint of the optimality line. Note that the red circle is only the *final* constraint boundary; we independently verify that the robots obey the constraint for each time-step within the simulation.

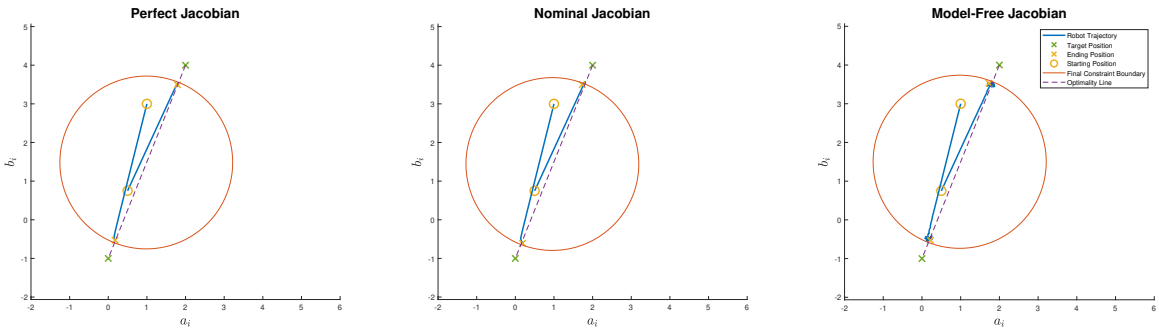

Figure 7.2: The results of the simulations with the three algorithms and the "ideal" target positions.

We see in Figure 7.2 that the three algorithms described above perform relatively similarly. From left to right, these correspond to the three approaches listed above (Algorithm 2 with a perfect Jacobian, Algorithm 2 with a nominal Jacobian, and the model-free Algorithm 3). The nominal Jacobian and the model-free approach both converge to a target in a close neighbourhood

of the optimality line, and they do so along a trajectory that roughly follows the perfect information trajectory. Observe that the perfect information trajectories are almost straight lines and thus this set of initial conditions is highly favourable for the nominal Jacobian's performance, since it is computed from averaging the gradients along the straight line paths to the goals. The noisy convergence near the end of the robots' trajectories for the model-free method is expected, as the Jacobian never converges to a "true" estimate, it perturbs every iteration (within bounded error of the true Jacobian), thus the robots' model-free final targets vary within a bounded set around the perfect final targets.

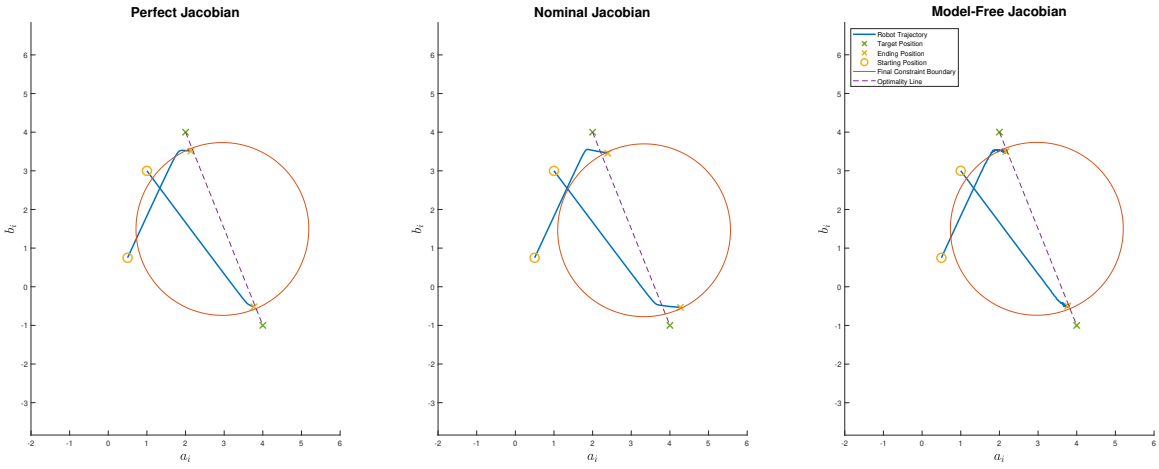

Figure 7.3: The results of the simulations with the three algorithms and unfavourable target positions.

Figure 7.3 illustrates the problems of the nominal approach in this setup. The perfect trajectory for robot 1 follows a straight line towards the goal, then sharply turns near the end to obey the constraint boundary as robot 2 approaches its respective goal and violates the boundary. At that turn, the error of the nominal Jacobian appears to compound, and robot 2 converges to a final position relatively far from the optimality line. The model-free Jacobian, on the other hand, converges to a similar bounded set around the perfect optimum.

This illustrates the drawbacks of the nominal Jacobian setup, as we outlined in Section 6.1. Without a priori knowledge of the path the robots are likely to follow, it is difficult to find a nominal Jacobian that is truly representative of the robots' cost functions near the constraint boundaries. The estimation framework requires no knowledge of the structure of the Jacobian, but comes with the drawback of not truly converging to a single optimum, instead converging to a bounded set in the neighbourhood of the expected optimum.

7.4 Application: Distribution Feeder

We now consider a practical application arising in control of renewable sources in a distribution feeder. We use the same setup as [1], with a power distribution feeder (Figure 7.4) whose details can be found in [7]. In this grid, large quantities of solar generation cause bus voltages to rise above acceptable levels; the goal is to limit over-voltage, while minimizing the system-wide power

curtailment in the photovoltaic (PV) systems.

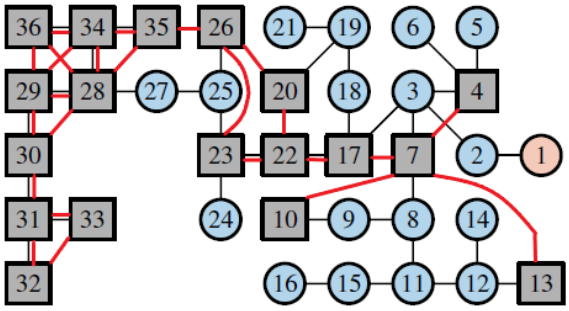

Figure 7.4: IEEE 37-node feeder [1]. Node 1 is the Point of Common Coupling (PCC). All other nodes are connected to a load and a voltage sensor. The square nodes are equipped with PV systems. The black lines denote electrical connections between nodes of the distribution feeder. The red lines denote the communication graph that nodes use to communicate multipliers.

Let $\mathcal{N} = \{4, 7, \ldots, 36\}$ be the set of all 18 nodes equipped with controllable PV systems (grey nodes in Figure 7.4). The control input for each PV system $i \in \mathcal{N}$ is is an active and reactive power injection set-point, denoted by $u_i = (p_i, q_i)$. Each of these are subject to the constraint $u_i \in \Omega_i$ where

$$\Omega_i := \{u_i = [p_i \; q_i]^\top \mid 0 \leq p_i \leq p_i^{\max}, q_i^2 + p_i^2 \leq (s_i^{\text{rated}})^2\}.$$

Here $p_i^{\max}$ is the available active power for each PV unit (dictated by solar irradiance), and s_i^{rated} is the rated apparent power of each PV inverter. The stacked decision vector is $\mathbf{u} = \text{col}(\{u_i\}_{i\in\mathcal{N}}) \in \Omega = \prod_{i\in\mathcal{N}} \Omega_i \subset \mathbb{R}^{18}$. Similarly, let $\mathbf{y} = \text{col}(\{y_i\}_{i\in\mathcal{N}})$ be the vector of measured voltage magnitudes at every PV-equipped node. Let $w \in \mathbb{R}^{70}$ be the collection of all uncontrollable loads and power injections (active and reactive) at all 35 nodes excluding the PCC. Each node's cost function is $f_i(u_i) = \|(u_{\text{ref}})_i - u_i\|^2$ where $u_{\text{ref}} = [p_i^{\max} \;\; 0]^\top$; this penalizes curtailment of the unit from its maximum real power production. The output $\mathbf{y} = \pi(\mathbf{u}, w)$ is dictated by the solution of the power flow equations for the distribution feeder [7]. The mapping is nonlinear and we assume that it is not known in full, and we only have access to the measurement $\mathbf{y}$ and a nominal Jacobian $\Pi_{\text{nom}} \in \mathbb{R}^{18\times 36}$ of the mapping π. We set the nominal Jacobian to be the Jacobian of the power flow equations of the feeder with a zero load profile and a voltage magnitude of 1 p.u at the PCC (refer to Figure 7.4) [1]. We define the safety constraints on each output to be the set $\mathcal{Y} = \prod_{i\in\mathcal{N}} \mathcal{Y}_i$, 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. We define the cost function $\frac{1}{2}\sum_{j\in\mathcal{N}} g_i(y_i) = [\max(0, \underline{y} - y_i, y_i - \bar{y})]^2$ for each node to penalize voltages outside the bus limits.

Finally, we define a set of coupling constraints differentiating this setup from the one in [1]. Consider the case where, perhaps due to contractual agreements, there is a hard upper limit on the total curtailment of PV power within the feeder. To model this, we define a global coupling constraint of the form $\sum_{i\in\mathcal{N}}(p_i^{\max} - p_i) \leq l$ where $l \in \mathbb{R}$ is the upper bound on the total curtailed real power. Note that $p_i = A_i u_i$ where $A_i = [1 \; 0]$, and each $p_i^{\max}$ is a known quantity at any given time. Thus the global constraint can be defined as $[A_1 \; \cdots \; A_N]\mathbf{u} \geq -l + \sum_{i\in\mathcal{N}} p_i^{\max}$. We can define $b_i = p_i^{\max} - \frac{l}{N}$. Thus we have the sets $\mathcal{U}$ and $\mathcal{U}_i(\mathbf{u}_{-i})$ as defined in (3.1).

With this setup, we are now ready to formulate the game-theoretic problem. At each time-step,

node $i \in \mathcal{N}$ plays the following constrained game:

$$\begin{aligned} \min_{u_i} \ & \|(u_{\text{ref}})_i - u_i\|^2 + \frac{1}{2}\sum_{j\in\mathcal{N}}[\max(0, \underline{y} - y_j, y_j - \bar{y})]^2 \\ \text{s.t. } & y_i = \pi_i(\mathbf{u}, w) \\ & u_i \in \mathcal{U}_i(\mathbf{u}_{-i}), \end{aligned} \tag{7.14}$$

to which we aim to apply our distributed OA vGNE seeking algorithm (Algorithm 2). In order to certify the convergence of Algorithm 2 for this problem, we parametrize it with a linear fractional transformation. Note that (7.14) is in the form described in Section 5.3: the class of problems of the same form as (5.13). Thus the pseudogradient is of the form in (5.20) with $\mathbf{Q} = I_{2N}$, $\alpha_i = 1$, and the rows of Π chosen correspondingly from the nominal Jacobian Π_{nom}. Our goal is to express the Jacobian in the form $\Pi_{\text{nom}} + \Delta_\pi$. Following the methodology in [1], we sample the Jacobian at $10,000$ randomly generated operating points, and compute the difference from the nominal Jacobian at each of these points to obtain a norm-bound γ as described in (5.18). We obtain a norm bound and multiply it by a tolerance factor of 1.1 to obtain $\gamma = 1.43$. We then use the recipes from Section 5.3 and the aforementioned parameters to verify the strong monotonicity and Lipschitz continuity of the pseudogradient. We show that Assumption 5 is satisfied with $\bar{\eta} = 0.44$ and $\bar{\theta} = 173$. It should be noted that the matrix inequalities developed in Section 5.1 are simply *sufficient* for convergence rather than necessary, and thus the estimates on the monotonicity and Lipschitz constants might be overly conservative. They simply provide convergence guarantees under the appropriately chosen (Lemma 4.3) step sizes, but the algorithm can have its step sizes tuned for better performance.

For the purpose of this simulation, the communication graph $\mathcal{G}_\lambda$ is defined as connecting any nodes in $\mathcal{N}$ that are adjacent to one another. The red lines on Figure 7.4 shows the communication graph for the problem. At each iteration of each time step, each node receives multipliers λ_j and auxiliary variables z_j from neighbouring nodes $j \in \mathcal{N}_{\mathcal{G}_\lambda}(i)$ and the relevant output measurements $y_j,\ j \in \mathcal{N}_{\mathcal{G}_f}(i)$, along with local voltage measurement y_i, and performs the update

$$u_{i,k+1} = P_{\Omega_i}\left[u_{i,k} - \tau_i\left(2(u_i - (u_{\text{ref}})_i) + \begin{bmatrix}\Pi_{i1}^\top & \dots & \Pi_{iN}^\top\end{bmatrix} s_{\underline{y},\bar{y}}(\mathbf{y}) - \begin{bmatrix}1\\0\end{bmatrix}\lambda_{i,k}\right)\right], \tag{7.15}$$

where $s_{\underline{y},\bar{y}}$ is the soft-thresholding function, defined as in (5.17). Each matrix $\Pi_{ij} \in \mathbb{R}^{1\times 2}$ approximately captures the sensitivity of local voltage changes with respect to player j's active/reactive power changes. The multipliers λ_i and the auxiliary variables z_i are updated as in Algorithm 2.

We simulate our distributed controller using ten hours of solar irradiance and load consumption data collected from Anatolia, CA, USA, with a granularity of one second. The maximum permissible curtailment is set to $l = 0.006$ p.u. As can be seen in Figure 7.5, voltages are maintained within safety limits and the total curtailed power is constrained. The constrained power curtailment shows the algorithm's ability to enforce coupling constraints between nodes despite lacking central knowledge of all nodes' constraints and their multipliers.

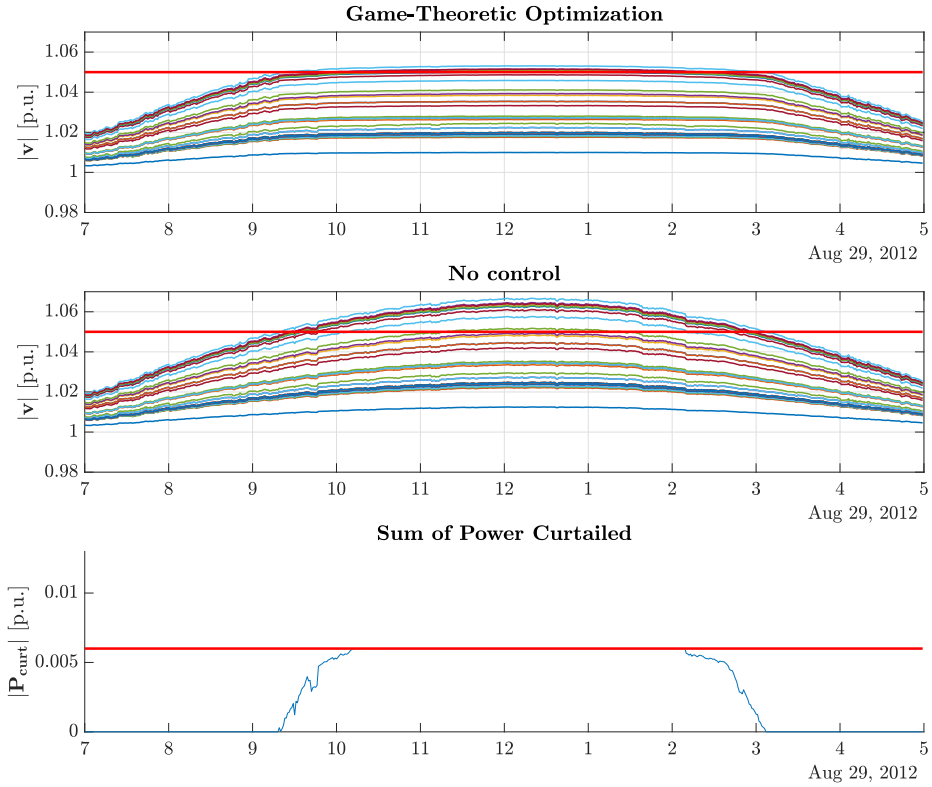

Figure 7.5: Comparison of the distributed algorithm vs. no control.

7.5 Application: Distribution Feeder without Global Information-Sharing

In the previous section we explored an application of the OA vGNE seeking algorithm to data sampled from a distribution feeder. Note that the cost function (7.14) requires knowledge of each other player's instantaneous output voltage. Further, the gradient step requires each player to have knowledge of their voltage levels' sensitivity to each other player's power injections. We next explore reducing the informational requirements on this problem formulation, and testing the robustness of the solution in this context. We assume that each node only has access to its own voltage level, rather than each other node's as it did prior. The game formulation thus becomes

$$\begin{aligned} \min_{u_i} \; & \|(u_{\text{ref}})_i - u_i\|^2 + \frac{3}{2}[\max(0, \underline{y} - y_i, y_i - \bar{y})]^2 \\ s.t. \; & y_i = \pi_i(\mathbf{u}, w) \\ & u_i \in \mathcal{U}_i(\mathbf{u}_{-i}). \end{aligned} \tag{7.16}$$

Note that we use the hyperparameter $\alpha_i = 3$, to increase each player's penalty for disobeying the constraint. This is meant to offset the fact that in (7.14) *each* player contributed to constraining the voltage levels. The matrix $\mathbf{Q}$ remains unchanged from before. However, note that when applying

the chain rule to the cost function as in (5.19), we obtain

$$\begin{aligned} g_i(\mathbf{y}) &= \frac{3}{2}\left[\max(0, \underline{y}_j - y_j, y_j - \bar{y}_j)\right]^2 \\ \nabla_{u_i} g_i(\pi(\mathbf{u}, w)) &= \sum_{k=1}^{N} \partial_{u_i} \pi_k(\mathbf{u}, w)^\top) \nabla_{y_k} g_i(\mathbf{y}) \\ &= 3\ \partial_{u_i} \pi_i(\mathbf{u}, w) s_{\underline{y}_i, \bar{y}_i}(y_i), \end{aligned}$$

because $\nabla_{y_k} g_i = 0$ for all $k \neq i$. Thus we can express the gradient in the form (5.20) with

$$\Pi_{ij} = \begin{cases} 3(\Pi_{\text{nom}})_{ii}, & i = j \\ \mathbb{0}, & i \neq j. \end{cases}$$

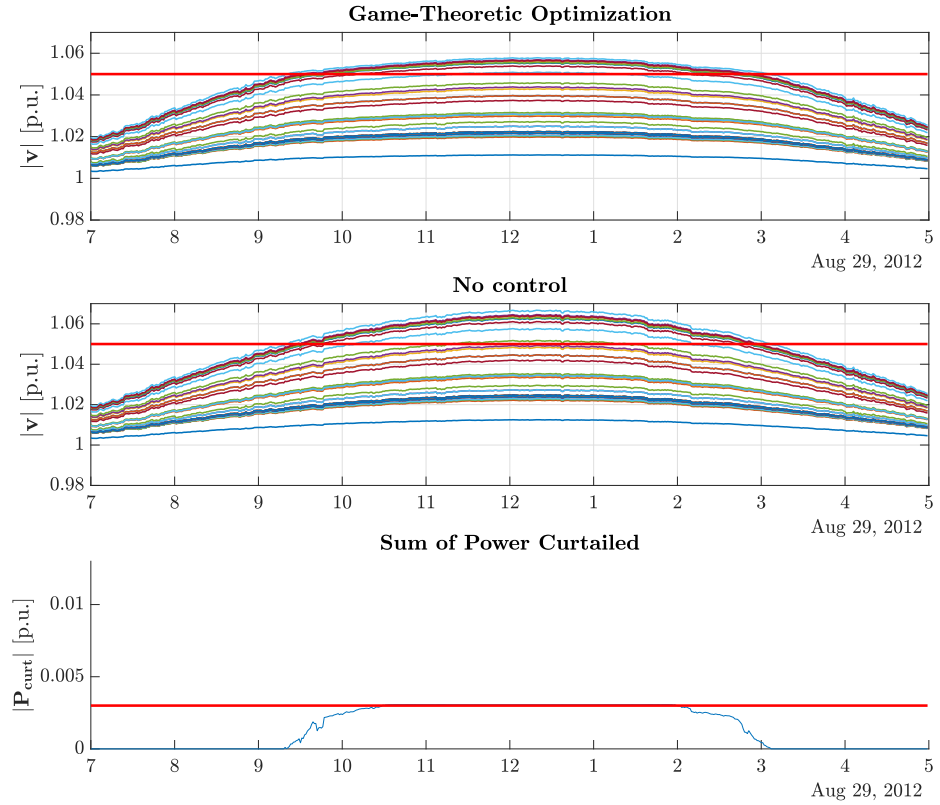

Figure 7.6: Comparison of the distributed algorithm vs. no control.

Figure 7.6 shows the results of this simulation. We set the upper limit to power curtailment $l = 0.003$ p.u. and show a run of the simulation over the same data as the previous section. We note that some of the nodes overshoot the safety limits by larger margins than in Figure 7.5, due to the lack of cross-sensitivity.

Chapter 8

Conclusion and Future Work

In this thesis we investigated the intersection of robust control and game theory, and its applications to various engineering problems. Our major contribution was the development of two algorithms that converge to candidate optimal points for game theoretic problems, while remaining robust to external disturbances and model uncertainty.

We outlined the first algorithm, the *online approximate vGNE seeking algorithm* in Chapter 4, and illustrated its convergence as a forward-backward operator splitting technique. We developed criteria to express the KKT points of games as a sum of monotone operators. This unification of robust online optimization and game-theoretic models can enable the implementation of scalable, decentralized optimization on large classes of problems in networks, power systems, smart cities, and other engineering applications.

As our forward-backward technique relies on verifying the strong monotonicity and Lipschitz continuity of uncertain operators, we focused on developing techniques for guaranteeing the fulfillment of these criteria in Chapter 5. We outlined the usefulness of the linear fraction transformation within our framework, and explained how uncertain operators can be *overbounded* by a larger set of uncertain operators, which makes the problem of verifying the above criteria much more tractable. We utilized this to develop a single matrix inequality which can be used to verify these criteria for the pseudogradients of games. Previous work in this area proves the usefulness of these techniques for gradients of strongly convex cost functions. Our main contribution in this regard is the extension of these techniques to *pseudogradients* of games, which make no assumption on the strong convexity of the underlying cost functions.

We then explored the limitations of a nominal Jacobian framework and noted the recent work in Simultaneous Perturbation Stochastic Approximation techniques. We noted that existing frameworks approximate the gradient as a whole. We developed a technique that allows the approximation of the Jacobian embedded within each agent's gradient, and proved its convergence using operator-theoretic techniques. This enabled the development of the *model-free game-theoretic algorithm* in Chapter 6. We provided convergence criteria and bounds, and explored the advantages and disadvantages of this approach compared to the nominal approach. In summary, the OA vGNE algorithm performs worse if the nominal Jacobian is too inaccurate, while the model-free framework has more overhead and does not converge to a final limiting point. The tradeoffs of these techniques can make them relevant in different applications.

8.1 Future Work

The primary motivation of this thesis was to implement distributed controllers in an information-light manner. This leads to natural extensions in the area of partial communication between agents. The interference graph $\mathcal{G}_f$ could be formed in such a way that agents can only communicate information with of the other agents whose actions and outputs they have a cost dependence on, rather than all of them. Prior work has been done in developing consensus dynamics to allow agents to estimate others' actions, similar to the setup used for the Lagrange multipliers in this thesis [20, 58]. Extending the estimation framework to output dynamics is a major area for future research. The use of more information-light techniques can allow for robust optimization of more complex classes of problems.

Another useful extension of the partial information framework would be one where one or more agents are *adversarial* in the communication graph. This can have applications in networks, security, etc. Training agents to come to a consensus while ignoring intentionally falsified information would also be a major improvement to the robustness of the framework, since it would add the ability for agents to reject classes of disturbances that affect the communication graph rather than directly impacting the output mapping as we studied within this thesis.

Another future avenue for research would be tuning the exploration signal from Chapter 6. For one, we defined the sampling of the signal relatively informally in Remark 6. A more rigorous set of techniques to tune quasi-stochastic signals is a possible future area to consider. Additionally, the assumptions we place on the signal's average work well for proving convergence conveniently, but it is possible that a more complex quasi-stochastic signal could be used to generate less noisy convergence or a tighter bound around the optimum that Algorithm 3 finds. It should be noted that quasi-stochastic techniques are a relatively recent innovation on stochastic approximation techniques [18, 27], leaving this an open area of research.

The focus of this research was in utilizing non-cooperative game theory to enable the use of distributed optimization. Enabling specific groups of agents within the larger network to play the game *cooperatively* could enable faster and more robust convergence. Consider, for example, the setup we used in the simulation in Section 7.5, where agents only had access to local voltage outputs, versus having global outputs in Section 7.4. Cooperative subgroups could define a sweet spot between these two extremes, where agents that are spatially close to one another could have full, cooperative communication, with far off groups either communicating along sparser channels, or relying on a partial information framework. This would enable the implementation of game-theoretic optimization techniques with a smaller communication overhead.

Appendix A

Maximal Monotonicity of Operators

Consider the operator $\mathfrak{B}$ defined in (3.6). We can express it as a sum of two operators:

$$\mathfrak{B}\left(\begin{bmatrix}\mathbf{u}\\ \lambda\end{bmatrix}\right) = \begin{bmatrix}\mathbb{0} & -A^\top\\ A & \mathbb{0}\end{bmatrix}\begin{bmatrix}\mathbf{u}\\ \lambda\end{bmatrix} + N_{\Omega\times\mathbb{R}^m_+}\left(\begin{bmatrix}\mathbf{u}\\ \lambda\end{bmatrix}\right).$$

The first term is a linear, single-valued operator with domain and range $\Omega \times \mathbb{R}^m_+$, thus it fulfills Minty's Theorem (Proposition 2.2), and is maximally monotone. The second term is the normal cone operator, which is maximally monotone as per Lemma 2.2.

By Lemma 2.7, the sum of a maximally monotone operator and a linear operator is maximally monotone, provided that the intersection of their domains is not the empty set. Hence $\mathfrak{B}$ is a maximally monotone operator.

Appendix B

Analysis of the Extraction Operator

In this section, we aim to show that the extraction operator $\mathfrak{E} : \mathbb{R}^{n\times N} \to \mathbb{R}^n$, defined in (4.8) is a bounded linear operator. We first show linearity. Consider some $M_1,\ M_2 \in \mathbb{R}^{n\times N}$ and $a,\ b \in \mathbb{R}$. Then

$$\begin{aligned}
& \mathfrak{E}(aM_1 + bM_2) \\
= & \sum_{k=1}^{N} \sum_{l\in\mathcal{N}_k} \left[e_l^\top \left(aM_1 + bM_2\right) e_k\right] e_l \\
= & \sum_{k=1}^{N} \sum_{l\in\mathcal{N}_k} \left[e_l^\top \left(aM_1\right) e_k + e_l^\top \left(bM_2\right) e_k\right] e_l \\
= & \sum_{k=1}^{N} \sum_{l\in\mathcal{N}_k} \left[e_l^\top \left(aM_1\right) e_k\right] e_l + \sum_{l\in\mathcal{N}_k} \left[e_l^\top \left(bM_2\right) e_k\right] e_l \\
= & \sum_{k=1}^{N} \sum_{l\in\mathcal{N}_k} \left[e_l^\top \left(aM_1\right) e_k\right] e_l + \sum_{k=1}^{N} \sum_{l\in\mathcal{N}_k} \left[e_l^\top \left(bM_2\right) e_k\right] e_l \\
= & \ a\sum_{k=1}^{N} \sum_{l\in\mathcal{N}_k} \left[e_l^\top M_1 e_k\right] e_l + b\sum_{k=1}^{N} \sum_{l\in\mathcal{N}_k} \left[e_l^\top M_2 e_k\right] e_l \\
= & \ a\mathfrak{E}(M_1) + b\mathfrak{E}(M_2).
\end{aligned}$$

Thus the operator $\mathfrak{E}$ is linear. We next show that the operator is continuous. From the operator's linearity, we start with

$$\mathfrak{E}(M_1) - \mathfrak{E}(M_2) = \mathfrak{E}(M_1 - M_2). \tag{B.1}$$

For the operator to be continuous, we need to show that

$$\lim_{M_1\to M_2} \|\mathfrak{E}(M_1) - \mathfrak{E}(M_2)\|_2 = 0 \ \forall M_1, M_2 \in \mathbb{R}^{n\times N},$$

which we can easily show by using the definition of the operator (4.8) in the right side of (B.1). Thus $\mathfrak{E}$ is a continuous operator.

A linear and continuous operator is a bounded operator, and thus $\mathfrak{E}$ is a bounded operator. We denote the operator norm of $\mathfrak{E}$ as $E_{\max}$, i.e.,

$$\|\mathfrak{E}(M)\| \leq E_{\max}\|M\|, \ \forall M \in \mathbb{R}^{n\times N}. \tag{B.2}$$

Appendix C

Matrix Inequality Proofs

C.1 Equivalence of Semidefinite Inequalities

In this section we aim to prove that the matrix inequality (5.1) is equivalent to the inequality

$$\|F(x) - F(y)\|_P^2 - (\rho + \theta)\langle F(x) - F(y), x - y\rangle_P + \rho\theta\|x - y\|_P^2 \leq 0, \tag{C.1}$$

for any $F : \mathbb{R}^n \to \mathbb{R}^n$ and $x,\ y \in \mathbb{R}^n$. We begin by expanding out (5.1):

$$2J^\top PJ - (\rho + \theta)\left[J^\top P + PJ\right] + 2\rho\theta P \preccurlyeq 0. \tag{C.2}$$

We first aim to show that (C.1) implies (C.2). We assume that (C.1) is true, and begin by considering some $y = x + \tau w$, where $\tau \in \mathbb{R}_+$ is a scalar magnitude and $w \in \mathbb{R}^n$ is a direction vector. We divide (C.1) by τ^2:

$$\frac{1}{\tau^2}\|F(x + \tau w) - F(x)\|_P^2 - \frac{\rho + \theta}{\tau^2}\langle F(x + \tau w) - F(x), \tau w\rangle_P + \frac{\rho\theta}{\tau^2}\|\tau w\|_P^2 \leq 0$$
$$\frac{1}{\tau^2}\langle F(x + \tau w) - F(x), F(x + \tau w) - F(x)\rangle_P - \frac{\rho + \theta}{\tau}\langle F(x + \tau w) - F(x), w\rangle_P + \rho\theta\langle w, w\rangle_P \leq 0.$$

Taking the limit as τ approaches 0 from above, we obtain

$$\lim_{\tau \to 0^+} \langle \frac{F(x + \tau w) - F(x)}{\tau}, \frac{F(x + \tau w) - F(x)}{\tau}\rangle_P - (\rho + \theta)\langle \frac{F(x + \tau w) - F(x)}{\tau}, w\rangle_P + \rho\theta\langle w, w\rangle_P \leq 0$$
$$\langle \partial F(x)w, \partial F(x)w\rangle_P - (\rho + \theta)\langle \partial F(x)w, w\rangle_P + \rho\theta\langle w, w\rangle_P \leq 0.$$

For simplicity of notation, we denote $J := \partial F(x)$. Thus, for all $x,\ w \in \mathbb{R}^n$, the following must be true:

$$w^\top J^\top PJw - (\rho + \theta)w^\top J^\top Pw + \rho\theta w^\top Pw \leq 0. \tag{C.3}$$

Since all the terms are scalars, we can rewrite the middle term:

$$\begin{aligned} & w^\top J^\top P w \\ = \; & \frac{1}{2}\left[w^\top J^\top P w + w^\top J^\top P w\right] \\ = \; & \frac{1}{2}\left[w^\top J^\top P w + (w^\top J^\top P w)^\top\right] \\ = \; & \frac{1}{2}\left[w^\top J^\top P w + w^\top P J w\right]. \end{aligned}$$

With that, we rewrite (C.3) and multiply it by 2 to obtain:

$$\begin{aligned} & w^\top \left[J^\top P J - \frac{\rho+\theta}{2}\left[J^\top P + P J\right] + \rho\theta P\right] w \leq 0 \\ & 2J^\top P J - (\rho+\theta)\left[J^\top P + P J\right] + 2\rho\theta P \preccurlyeq 0. \end{aligned}$$

Thus we conclude that (C.1) implies (C.2). We now prove the reverse direction. We start by assuming that (C.2) is true. We define a parametric line between two points $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$:

$$\gamma(t) = (1-t)y + tx, t \in [0,1].$$

Note that $\gamma(0) = y$ and $\gamma(1) = x$. The derivative of $\gamma(t)$ is

$$\dot{\gamma}(t) := \frac{d\gamma}{dt}(t) = x - y.$$

With this parametric definition, we can express our function F as follows:

$$\begin{aligned} F(x) - F(y) &= \int_0^1 \frac{dF}{dt}(\gamma(t))\, dt \\ &= \int_0^1 \partial F(\gamma(t))\dot{\gamma}(t) dt \\ &= \int_0^1 \partial F(\gamma(t))(x-y) dt. \end{aligned}$$

Taking the norm $\|\cdot\|_P$ on both sides, we get

$$\begin{aligned} \|F(x) - F(y)\|_P^2 &= \left\| \int_0^1 \partial F(\gamma(t))(x-y) dt \right\|_P^2 \\ &\leq \max_z \left\| \int_0^1 \partial F(z)(x-y) dt \right\|_P^2, \end{aligned}$$

where z is chosen from along the parametrized line, that is, $z = \gamma(\bar{t})$, for some $\bar{t} \in [0,1]$ that maximizes the norm on the right hand side. We denote the Jacobian corresponding to this maximal

z as $J := \partial F(z)$. Then

$$\begin{aligned} & \max_z \left\| \int_0^1 \partial F(z)(x-y)dt \right\|_P^2 \\ = & \left\| \int_0^1 J(x-y)dt \right\|_P^2 \\ = & \|J(x-y)\|_P^2 \\ = & (x-y)^\top J^\top P J(x-y). \end{aligned} \tag{C.4}$$

From (C.2), we know that

$$J^\top P J \preccurlyeq \frac{1}{2}\left[(\rho+\theta)\left[J^\top P + PJ\right] - 2\rho\theta P\right],$$

therefore, we can place an upper bound on (C.4):

$$\begin{aligned} \|F(x)-F(y)\|_P^2 &\leq (x-y)^\top J^\top P J(x-y) \\ &\leq \frac{1}{2}(x-y)^\top \left[(\rho+\theta)\left[J^\top P + PJ\right] - 2\rho\theta P\right](x-y) \\ &\leq \frac{\rho+\theta}{2}\left[(x-y)^\top \left[J^\top P + PJ\right](x-y)\right] - \rho\theta\|x-y\|_P^2. \end{aligned}$$

Rearranging this inequality, we can say

$$\frac{1}{2}(x-y)^\top \left[J^\top P + PJ\right](x-y) \geq \frac{1}{\rho+\theta}\left[\|F(x)-F(y)\|_P^2 + \rho\theta\|x-y\|_P^2\right]. \tag{C.5}$$

Note that the value on the right hand side is strictly greater than 0 for all $x \neq y$, hence it is true to say that there exists some $\mu \in \mathbb{R}$ such that

$$\frac{1}{2}(x-y)^\top \left[J^\top P + PJ\right](x-y) \geq \mu P.$$

By Proposition 8 in [59], which is a modification of Proposition 2.3.2 in [35], we can then say that F is a strongly monotone function. We can conclude that

$$\langle F(x)-F(y), x-y\rangle_P \geq \frac{1}{\rho+\theta}\left[\|F(x)-F(y)\|_P^2 + \rho\theta\|x-y\|_P^2\right],$$

which can then be rearranged to exactly obtain (C.1). Thus the two statements, (C.1) and (C.2), are equivalent.

C.2 Sector-Bounded Nonlinearity

Consider some function $F : \mathbb{R}^n \to \mathbb{R}^n$, and any two points $x,\ y \in \mathbb{R}^n$. Assume there exist $\rho,\ \theta > 0$, $P \succ 0$ such that

$$\left[\|F(x)-F(y)\|_P - \rho\|x-y\|_P\right]\left[\|F(x)-F(y)\|_P - \theta\|x-y\|_P\right] \leq 0. \tag{C.6}$$

This statement is equivalent to saying that of the two terms multiplied together, one must be positive and the other negative. Assume that the first term is positive. Assume, as we did in the proof for Proposition 5.1, that $\theta \geq \rho$. Then

$$\|F(x) - F(y)\|_P \leq \rho \|x - y\|_P,$$

necessarily implies

$$\|F(x) - F(y)\|_P \leq \theta \|x - y\|_P.$$

This would lead to both terms in (C.6) being negative, which contradicts the assumption we started with, that the inequality held true. We require that one be positive and the other be negative, which can then only happen if the second term of (C.6) is negative, and the first positive,and thus

$$\rho \|x - y\|_P \leq \|F(x) - F(y)\|_P \leq \theta \|x - y\|_P. \tag{C.7}$$

Thus (C.6) is equivalent to the inequality (C.7).

Appendix D

PD Controller Tracking

For a detailed reading on the fundamentals of Laplace Transforms and their application to feedback control, we refer the reader [60], specifically chapters 2 and 4. The double integrator dynamics from (7.7) can be represented in the form of a transfer function as

$$\frac{R_i(s)}{\Xi_i(s)} := G_i(s) = \begin{bmatrix} \frac{1}{s^2} & 0 \\ 0 & \frac{1}{s^2} \end{bmatrix}. \tag{D.1}$$

We seek to implement a controller $C_i(s)$ that chooses a control input $\Xi_i(s)$ such that the system asymptotically tracks a constant reference signal of the form $U_i(s) = \frac{1}{s}\text{col}(\bar{u}_{i,a}, \bar{u}_{i,b})$. Figure D.1 illustrates the block diagram of the system.

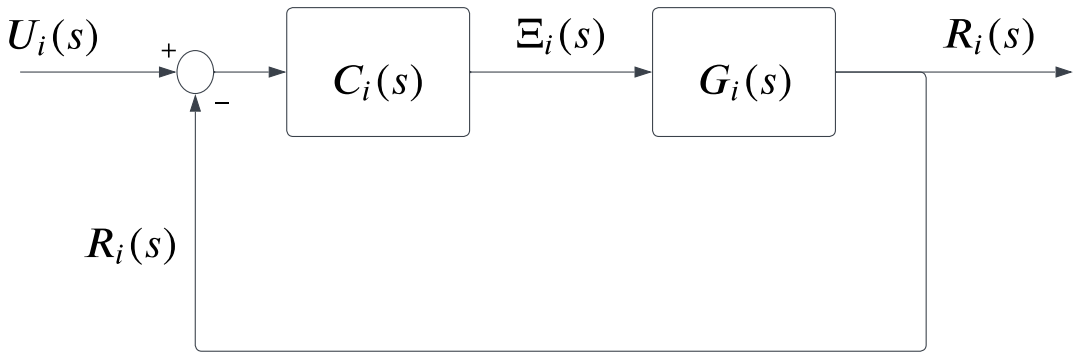

Figure D.1: Block diagram of the PD controller.

We consider a controller of the form

$$C_i(s) = \begin{bmatrix} K_p + K_d s & 0 \\ 0 & K_p + K_d s \end{bmatrix}, \tag{D.2}$$

that is, we apply a PD controller with derivative gain K_d and proportional gain K_p to each dimension

of the robot's state. We verify the values for $K_d,\ K_p$ for which the closed loop transfer matrix is asymptotically stable.

$$\begin{aligned}
R_i(s) &= G_i(s)\Xi_i(s) \\
R_i(s) &= G_i(s)C_i(s)\left[U_i(s) - R_i(s)\right] \\
\frac{R_i(s)}{U_i(s)} &= \left[1 + C_i(s)G_i(s)\right]^{-1} C_i(s)G_i(s) \\
\frac{R_i(s)}{U_i(s)} &= \begin{bmatrix} \frac{\frac{K_p}{s^2}+\frac{K_d}{s}}{1+\frac{K_p}{s^2}+\frac{K_d}{s}} & 0 \\ 0 & \frac{\frac{K_p}{s^2}+\frac{K_d}{s}}{1+\frac{K_p}{s^2}+\frac{K_d}{s}} \end{bmatrix} \\
\frac{R_i(s)}{U_i(s)} &= \begin{bmatrix} \frac{K_d s+K_p}{s^2+K_d s+K_p} & 0 \\ 0 & \frac{K_d s+K_p}{s^2+K_d s+K_p} \end{bmatrix}.
\end{aligned} \tag{D.3}$$

We want to choose K_d, K_p such that the poles of the transfer function are in the open left-half plane. We can verify by direct calculation that the choice of $K_p = 1,\ K_d = 2$ satisfies this. Applying the quadratic formula to the denominator, we get

$$\begin{aligned}
& \frac{-K_d \pm \sqrt{K_d^2 - 4K_p}}{2} \\
= \ & -1.
\end{aligned}$$

Since the poles are asymptotically stable, we can apply the final value theorem to obtain the steady-state value of the control system. We assume that the input is a constant, causal reference signal of the form $u_i(t) = \mathrm{col}(\bar{u}_{i,a}, \bar{u}_{i,b})$.

$$\begin{aligned}
\lim_{t\to\infty} r_i(t) &= \lim_{s\to 0} sR_i(s) \\
&= \lim_{s\to 0} s \begin{bmatrix} \frac{K_d s+K_p}{s^2+K_d s+K_p} & 0 \\ 0 & \frac{K_d s+K_p}{s^2+K_d s+K_p} \end{bmatrix} \frac{1}{s} \begin{bmatrix} \bar{u}_{i,a} \\ \bar{u}_{i,b} \end{bmatrix} \\
&= \begin{bmatrix} \bar{u}_{i,a} \\ \bar{u}_{i,b} \end{bmatrix}.
\end{aligned} \tag{D.4}$$

Thus the control system asymptotically tracks a constant reference signal.

Bibliography

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

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

[3] 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.

[4] 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.

[5] S. Low and D. Lapsley, "Optimization flow control. i. basic algorithm and convergence," *IEEE Trans. Networking*, vol. 7, no. 6, pp. 861–874, 1999.

[6] Y. Tang, K. Dvijotham, and S. Low, "Real-time optimal power flow," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2963–2973, 2017.

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

[8] S. Bolognani, R. Carli, G. Cavraro, and S. Zampieri, "Distributed reactive power feedback control for voltage regulation and loss minimization," *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 966–981, Apr. 2015.

[9] 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.

[10] M. Vaquero and J. Cortés, "Distributed augmentation-regularization for robust online convex optimization," *IFAC-PapersOnLine*, vol. 51, no. 23, pp. 230–235, 2018, ISSN: 2405-8963.

[11] J. S. Shamma, "Game theory, learning, and control systems," *National Science Review*, vol. 7, no. 7, pp. 1118–1119, Nov. 2019. [Online]. Available: https://doi.org/10.1093/nsr/nwz163.

[12] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444537669000161.

[13] 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. [Online]. Available: https://arxiv.org/abs/2103.13988.

[14] A. R. Romano and L. Pavel, "Dynamic NE seeking for multi-integrator networked agents with disturbance rejection," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 129–139, Mar. 2020.

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

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

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

[18] 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. [Online]. Available: `https://arxiv.org/abs/1909.13132`.

[19] P. Yi and L. Pavel, "An operator splitting approach for distributed generalized Nash equilibria computation," *Automatica*, vol. 102, pp. 111–121, 2019, ISSN: 0005-1098.

[20] L. Pavel, "Distributed GNE seeking under partial-decision information over networks via a doubly-augmented operator splitting approach," *IEEE Trans. Autom. Control*, vol. 65, no. 4, pp. 1584–1597, Apr. 2020.

[21] F. Facchinei and C. Kanzow, "Generalized Nash equilibrium problems," *Annals of Operations Research*, vol. 175, pp. 177–211, 2010.

[22] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono, *Optimal rates for zero-order convex optimization: The power of two function evaluations*, 2013.

[23] Y. Nesterov and V. Spokoiny, *Random gradient-free minimization of convex functions*, 2017.

[24] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951. DOI: `10.1214/aoms/1177729586`. [Online]. Available: `https://doi.org/10.1214/aoms/1177729586`.

[25] "A one-measurement form of simultaneous perturbation stochastic approximation," *Automatica*, vol. 33, no. 1, pp. 109–112, 1997, ISSN: 0005-1098. DOI: `https://doi.org/10.1016/S0005-1098(96)00149-5`.

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

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

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

[29] C. Scherer and S. Weiland, *Linear matrix inequalities in control*, 2015.

[30] L. Hogben. Routledge, 2013.

[31] G. Strand, *Linear Algebra and its Applications*, 4th. Brooks Cole, 2005.

[32] M. Maggiore, *Foundations of Nonlinear Control Theory*. 2019.

[33] A. Nagurney and D. Zhang, *Projected Dynamical Systems and Variational Inequalities with Applications*. Springer, 1996.

[34] J.-B. Hiriart-Urruty and C. Lemaréchal, *Fundamentals of Convex Analysis*. Springer, 2001.

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

[36] K. J. Devlin, *Fundamentals of contemporary set theory*. Springer-Verlag, 1979.

[37] F. H. Clarke, "Generalized gradients and applications," *Trans. of the AMS*, vol. 205, 1975.

[38] A. Baíllo and J. E. Chacón, "Chapter 1 - statistical outline of animal home ranges: An application of set estimation," in *Data Science: Theory and Applications*, ser. Handbook of Statistics, A. S. Srinivasa Rao and C. Rao, Eds., vol. 44, Elsevier, 2021, pp. 3–37.

[39] M. D. Voisei, *Maximal monotone normal cones in locally convex spaces*, 2019.

[40] R. T. Rockafellar, "On the maximal monotonicity of subdifferential mappings.," *Pacific Journal of Mathematics*, vol. 33, pp. 209–216, 1970.

[41] J. M. Borwein and L. Yao, "Maximality of the Sum of a Maximally Monotone Linear Relation and a Maximally Monotone Operator," *Set-Valued and Variational Analysis*, vol. 25, pp. 603–616, 2013.

[42] C. Godsil and G. F. Royle, *Algebraic Graph Theory*, ser. Graduate Texts in Mathematics Book 207. Springer, 2001.

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

[44] N. M. M. de Abreu, "Old and new results on algebraic connectivity of graphs," *Linear Algebra and its Applications*, vol. 423, no. 1, pp. 53–73, 2007.

[45] "The laplacian spectrum of a graph," *Computers & Mathematics with Applications*, vol. 48, no. 5, pp. 715–724, 2004.

[46] J. W. Simpson-Porco, "Analysis and synthesis of low-gain integral controllers for nonlinear systems," *IEEE Trans. Autom. Control*, vol. 66, no. 9, pp. 4148–4159, Sep. 2021.

[47] G. E. Dullerud and F. Paganini, "A course in robust control theory," 2000.

[48] M. Zhu and E. Frazzioli, "Distributed robust adpative equilibrium computation for generalized convex games," *Automatica*, vol. 63, pp. 82–91, 2016.

[49] 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.

[50] M. J. Kearns, M. L. Littman, and S. P. Singh, "Graphical models for game theory," *CoRR*, vol. abs/1301.2281, 2013. arXiv: `1301.2281`.

[51] L. J. Ratliff, S. A. Burden, and S. S. Sastry, "On the characterization of local Nash equilibria in continuous games," *IEEE Trans. Autom. Control*, vol. 61, no. 8, pp. 2301–2307, Aug. 2016.

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

[53] C. Yu, M., van der Schaar, and A. H. Sayed, "Distributed learning for stochastic generalized Nash equilibrium problems," *IEEE Trans. Signal Proc.*, vol. 65, no. 15, pp. 3893–3908, 2017.

[54] S. Grammatico, "Dynamic control of agents playing aggregate games with coupling constraints," *IEEE Trans. Autom. Control*, vol. 62, no. 9, pp. 4537–4548, 2017.

[55] A. Nagurne, *Network Economics: A Variational Inequality Approach*. Springer, 1993.

[56] A. Auslender and M. Teboulle, "Lagrangian duality and related multiplier methods for variational inequality problems," *SIAM Journal on Optimization*, vol. 10, no. 4, pp. 1097–1115, 2000.

[57] A. Megretski and A. Rantzer, "System analysis via integral quadratic constraints," *IEEE Trans. Autom. Control*, vol. 42, no. 6, pp. 819–830, 1997.

[58] 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.

[59] ——, *On the exact convergence to nash equilibrium in hypomonotone regimes under full and partial-information*, 2021.

[60] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 12th. Pearson, 2017.