Connected Quadratic Programs for Autonomy

Forrest Laine

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-182
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-182.html

August 12, 2021

Acknowledgement

To my advisor, Claire, thank you for all of your support over the years. You took a chance when accepting me graduate school, and have since let me run with my ideas, both the good and the bad. I surely would not have been able to get where I am today without your guidance along the way.

To my industry mentors, Jur van den Berg and Sachin Patil, thank you for welcoming me to work with you on trucks and cars over the years. The practical experience I gained working with you has been invaluable to my research and growth.

Finally, to Annie, thank you for supporting and being patient with me throughout this entire journey. You keep me grounded and balanced, and are a constant light in my life. I love you dearly.

Connected Quadratic Programs for Autonomy

by

Forrest J Laine

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering — Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Claire Tomlin, Chair
Professor S. Shankar Sastry
Professor Francesco Borrelli

Summer 2021

Connected Quadratic Programs for Autonomy

Abstract

Connected Quadratic Programs for Autonomy

by

Forrest J Laine

Doctor of Philosophy in Engineering — Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Claire Tomlin, Chair

This dissertation describes the work I have done on the subject of computing equilibrium solutions of connected quadratic programs (QPs), particularly for the connected programs arising in the context of autonomous system design. Reasoning about the interaction between multiple mathematical programs enables the analysis and solution to a wealth of problems inaccessible by standard optimization formulations.

One of the most commonly arising examples of connected optimization problems is the bilevel program, comprised of an outer-level and inner-level program. The outer-level problem is subject to a constraint that a subset of decision variables solve the inner-level problem. Another class of connected optimization problems are those of Nash equilibrium problems, in which optimization problems are solved simultaneously.

Beyond these and some other simple organizations of problem, general interactions among optimization problems are not well studied, despite the potential for modeling many interesting real-world problems. Presumably this is because with generalization in problem organization comes increased complexity, both from an analysis and computation perspective.

To address this, it is claimed in this work that it is possible to compute equilibrium solutions to a broad range of connected optimization problems, assuming those problems take the form of convex quadratic programs. This claim is based off a result that equilibrium solutions to a collection of QPs with piecewise linear constraints can be represented as a piecewise linear mapping. Recursive application of this result leads to the results for generally nested equilibrium problems. Theoretical results and computational approaches for this class of problem are developed. The efficacy of the presented methodologies is demonstrated on various problems faced by autonomous vehicles.

To Dad and Mom

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgments

To my advisor, Claire, thank you for all of your support over the years. You took a chance when accepting me graduate school, and have since let me run with my ideas, both the good and the bad. I surely would not have been able to get where I am today without your guidance along the way.

To my industry mentors, Jur van den Berg and Sachin Patil, thank you for welcoming me to work with you on trucks and cars over the years. The practical experience I gained working with you has been invaluable to my research and growth.

Finally, to Annie, thank you for supporting and being patient with me throughout this entire journey. You keep me grounded and balanced, and are a constant light in my life. I love you dearly.

Chapter 1

Introduction

Mathematical programming provides an extremely powerful tool for solving real-world problems arising in numerous contexts. The study of standard-form optimization problems allows for the development of numerical methods and analytic tools which are broadly applicable to any problem which can be modeled in that standard form. In a sense, mathematical programming can be thought of as the study of problem templates. If any instantiation of a problem fits the template, then the methodologies and analysis developed for the template can be applied to that problem.

Often times engineers are trained (intentionally or not) to think about problems in the context of these mathematical programming templates. The most common template is that of convex optimization problems, for which a wealth of strong analysis is available, as well as professional-grade solution techniques. Non-convex programs capture a broader class of problem, but the added flexibility comes at the cost of increased complexity. Nevertheless, extensive research and development into these (and other) standard formats has lead to the availability of very mature solution approaches.

While the framing of common problems as mathematical programs is often a very fruitful practice, there are many real-world problems which are fundamentally *not* optimization problems. The standard form of optimization problem, stated as the minimization (or maximization) of some function, subject to feasible set constraints, cannot capture the intended decision to be made in many applications. Game theory was introduced for this very reason — when there are multiple agents each acting rationally according to their own best interest, the resultant behavior cannot be characterized as the solution to a single optimization problem, but rather the simultaneous solution to a collection of optimization problems.

Under the perspective that mathematical games are simply collections of optimization problems which interact in a particular manner, then games are a strict generalization of optimization problems. In this view, the game-theoretic template is one that is applicable to a very broad class of problems. Even so, the particular structure of the games for which most templates are designed seems relatively limited.

Classic game formulations, like Nash or Stackelberg equilibrium problems, exhibit a very particular type of interaction among the multiple optimization problems making up the game.

Nash games consider lateral connections of problems, meaning all optimization problems are treated equally, whereas Stackelberg formulations consider vertical connections, where one optimization problem gets preference over the other. When understanding these two primary classes of games as simply two different ways of organizing optimization problems, however, it doesn't take much to imagine more elaborate ways to connect problems.

With the freedom to chain, nest and group optimization problems in all sorts of configurations, the possible modeling capacity of games seems endless. For example, entire graphs of connected optimization problem could be envisioned. The only question, is whether formulating problems as complicated organizations of optimization problems is useful at all. Is the complexity of an elaborate connection pattern such that it makes any meaningful analysis impossible? Are computational methods for finding solutions out of reach?

These questions have been the main driver for the work presented in this dissertation. In what follows, I argue that a broad class of organized connected optimization problems can be analyzed, and methods for computing solutions to such problems can be generated. Namely, when each of the connected optimization problems takes the form of a quadratic program, some appealing properties emerge which enable arbitrary nesting of equilibrium problems to become possible. I will present theoretical results establishing this claim, and then will apply those results to a handful of real-world problems.

1.1 Outline

In chapter 2, I will develop theory regarding the class of organized quadratic programs that will be considered in this work. This class of problem is specifically that of Equilibrium Problems with Nested Equilibrium Constraints (EPNECs). In chapter 3 I will then present a proposed method for computing solutions to these EPNECs, which will be based on the theory developed in chapter 2. In chapter 4 I will focus on an important subclass of $EPNECs$, which are the class of Generalized Feedback Nash Equilibrium problems, which can be used to model interesting phenomena like interactive driving. In chapter 5 I will extend that work to discuss methods for making game-theoretic motion planning more practical for autonomous vehicles. Finally in chapter 6 I will propose a framework for solving Problems of Ordered Preferences, which are problems with multiple objectives in a prioritized hierarchy. Finally, I will conclude with some remarks on proposed future work in this area.

Chapter 2

Theory

In this chapter, the theory of connected quadratic programs is developed, leading up to the concept of equilibrium problems with nested equilibrium constraints, which is the most general form of connected QP that will be considered in this dissertation. Development will proceed by successively generalizing problem formulations, from parametric QPs, to parametric equilibrium problems, to parametric equilibrium problems with equilibrium constraints, to finally, equilibrium problems with nested equilibrium constraints. Along the way, various theorems on the existence of solutions and the properties of those solutions will be given.

2.1 Parametric Quadratic Programs

The fundamental building block of most problems considered in this work are parametric quadratic programs. These programs take the following form:

$$
\begin{aligned}
QP(y) := \underset{x \in \mathbb{R}^n}{\arg\min} \quad & x^\intercal(\frac{1}{2}Qx + Ry + q) && \text{(2.1a)}\\
\text{subject to} \quad & Ax + By + c \geq 0, && \text{(2.1b)}\\
& Dx + Ey + f = 0 && \text{(2.1c)}
\end{aligned}
$$

Here (2.1) is a parametric quadratic program, since it takes as input the parameter $y \in \mathbb{R}^m$. Let p_i denote the number of inequality constraints and p_e the number of equality constraints, such that $A \in \mathbb{R}^{p_i \times n}$, $D \in \mathbb{R}^{p_e \times n}$, etc.

Theorem 2.1.1 (Necessary and Sufficient Conditions for Optimality)**.** *Assume that for every $y \in \mathbb{R}^m$, the following hold:*

- *There exist at least some $x \in \mathbb{R}^n$ such that $Ax + By + c \geq 0$ and $Dx + Ey + f = 0$ (feasibility).*

- *$Z^\intercal QZ \succ 0$, where Z is a nullspace basis for the matrix D (strong convexity).*

Then, there exists a unique $x^* \in \mathbb{R}^n$ such that $QP(y) = x^*$, and for which there exist some $\lambda^* \in \mathbb{R}^{p_e}$ and $\mu^* \in \mathbb{R}^{p_i}$ satisfying the following:

$$\begin{aligned} Qx^* - A^\intercal \mu^* - D^\intercal \lambda^* + Ry + q &= 0 && (2.2a)\\ Ax^* + By + c &\geq 0 && (2.2b)\\ Dx^* + Ey + f &= 0 && (2.2c)\\ \mu^* &\geq 0 && (2.2d)\\ \mu^{*\intercal}(Ax^* + By + c) &= 0. && (2.2e) \end{aligned}$$

Proof. See [192], Theorem 12.6. □

At some y, x^*, the active, inactive, degenerate, strictly-active, and strictly-inactive index sets are defined as the following:

$$\begin{aligned} \mathcal{I}_\mathrm{a} &:= \{\, j \mid (Ax^* + By + c)_j \;= 0\}, && (2.3a)\\ \mathcal{I}_\mathrm{i} &:= \{\, j \mid \qquad\qquad\qquad \lambda_j^* = 0\}, && (2.3b)\\ \mathcal{I}_\mathrm{d} &:= \mathcal{I}_\mathrm{a} \cap \mathcal{I}_\mathrm{i}, && (2.3c)\\ \mathcal{I}_\mathrm{sa} &:= \mathcal{I}_\mathrm{a} \setminus \mathcal{I}_\mathrm{d}, && (2.3d)\\ \mathcal{I}_\mathrm{si} &:= \mathcal{I}_\mathrm{i} \setminus \mathcal{I}_\mathrm{d}. && (2.3e) \end{aligned}$$

When the set $\mathcal{I}_\mathrm{d} \equiv \emptyset$, the pair y, x^* are described as satisfying *strict complementarity.* Using the definition of the above index sets, the constraint coefficients A, B, and c, and multipliers μ are decomposed:

$$\begin{array}{llll} A_{sa} := (A)_{\mathcal{I}_\mathrm{sa}} & B_{sa} := (B)_{\mathcal{I}_\mathrm{sa}} & c_{sa} := (c)_{\mathcal{I}_\mathrm{sa}} & \mu_{sa} := (\mu)_{\mathcal{I}_\mathrm{sa}} \\ A_{si} := (A)_{\mathcal{I}_\mathrm{si}} & B_{si} := (B)_{\mathcal{I}_\mathrm{si}} & c_{si} := (c)_{\mathcal{I}_\mathrm{si}} & \mu_{si} := (\mu)_{\mathcal{I}_\mathrm{si}} \\ A_{d} := (A)_{\mathcal{I}_\mathrm{d}} & B_{d} := (B)_{\mathcal{I}_\mathrm{d}} & c_{d} := (c)_{\mathcal{I}_\mathrm{d}} & \mu_{d} := (\mu)_{\mathcal{I}_\mathrm{d}} \end{array} \qquad (2.4)$$

The notation used in Eq. (2.4) is used to indicate, for example, that A_{sa} is the matrix formed by stacking the rows of the matrix A specified by the index set $\mathcal{I}_\mathrm{sa}$.

Theorem 2.1.2 (Piecewise Linearity of QP)**.** *Under the assumptions listed in Theorem 2.1.1, the mapping $QP(y) \to x^*$ is piecewise linear. Furthermore, for any $y \in \mathbb{R}^m$, the directional derivative of $QP(y)$ in the direction of some vector $\delta y \in \mathbb{R}^m$, denoted as $\mathbf{D}QP(y)[\delta y]$ is given as the solution to the following program:*

$$
\begin{aligned}
\mathbf{D}QP(y)[\delta y] = \quad & \underset{\delta x \in \mathbb{R}^n}{\text{arg min}} && \delta x^\intercal(\frac{1}{2}Q\delta x + R\delta y) && (2.5a)\\
& \text{subject to} && D\delta x + E\delta y = 0, && (2.5b)\\
& && A_{sa}\delta x + B_{sa}\delta y = 0, && (2.5c)\\
& && A_d\delta x + B_d\delta y \geq 0 && (2.5d)
\end{aligned}
$$

Proof of Theorem 2.1.2. By the assumptions and result of Theorem 2.1.1, there always exists a unique δx^* which solves (2.5), and there additionally exist corresponding multipliers $\delta\lambda^*$, $\delta\mu^*_{sa}$, and $\delta\mu^*_d$ such that

$$Q\delta x^* - A_{sa}^\intercal\delta\mu^*_{sa} - D^\intercal\delta\lambda^* - A_d{}^\intercal\delta\mu^*_d + R\delta y = 0 \tag{2.6a}$$
$$A_s\delta x^* + B_s\delta y = 0 \tag{2.6b}$$
$$D\delta x^* + E\delta y = 0 \tag{2.6c}$$
$$A_d\delta x^* + B_d\delta y \geq 0 \tag{2.6d}$$
$$\delta\mu^*_d \geq 0 \tag{2.6e}$$
$$\delta\mu_d^{*\intercal}(A_d\delta x^* + B\delta y) = 0. \tag{2.6f}$$

It follows that for all δy and small enough α, the point $x^* + \alpha\delta x$ is the unique solution to $QP(y+\alpha\delta y)$. To see this, let $\delta\mu_{si} := 0$, and $\delta\mu$ be the reconstruction of $\delta\mu_d$, $\delta\mu_{sa}$, and $\delta\mu_{si}$ in the original index ordering. It suffices to show that $x^* + \alpha\delta x$, $\lambda^* + \alpha\delta\lambda$, and $\mu^* + \alpha\delta\mu$ satisfy the necessary and sufficient conditions for optimality of $QP(y+\alpha\delta y)$ listed in Theorem 2.1.1.

First note that from Eqs. (2.2) and (2.6),

$$Q(x^* + \alpha\delta x^*) - A_{sa}^\intercal(\mu^*_{sa} + \alpha\delta\mu^*_{sa}) - D^\intercal(\lambda^* + \alpha\delta\lambda^*) - A_d{}^\intercal\delta\mu^*_d + R(y + \alpha\delta y) = 0, \tag{2.7a}$$
$$D(x^* + \alpha\delta x^*) + E(y + \alpha\delta y) + f = 0, \tag{2.7b}$$
$$A_{sa}(x^* + \alpha\delta x^*) + B_{sa}(y + \alpha\delta y) + c_s = 0, \tag{2.7c}$$
$$A_d(x^* + \alpha\delta x^*) + B_d(y + \alpha\delta y) + c_d \geq 0, \tag{2.7d}$$
$$\mu^*_d + \alpha\delta\mu^*_d \geq 0. \tag{2.7e}$$

Furthermore, by the definition of the set $\mathcal{I}_d$, both $A_d x^* + B_d y + c_d = 0$ and $\mu^*_d = 0$, so

$$\begin{aligned}(\mu^*_d + \alpha\delta\mu_d)^\intercal(A_d(x^* + \alpha\delta x^*) + B_d(y + \alpha\delta y) + c_d) &= \\ (\alpha\delta\mu_d)^\intercal(A_d(\alpha\delta x^*) + B_d(\alpha\delta y)) &= 0.\end{aligned} \tag{2.8}$$

Therefore, it only remains to show that

$$A_{si}(x^* + \alpha\delta x^*) + B_{si}(y + \alpha\delta y) + c_s \geq 0, \tag{2.9}$$

since $\mu^*_{si} + \alpha\delta\mu_{si} = 0$. From the definition of the index set $\mathcal{I}_{si}$, it is known that

$$A_{si}x^* + B_{si}y + c_s > 0. \tag{2.10}$$

Therefore Eq. (2.9) must be satisfied for all $\alpha \in [0, \alpha_{max}]$, where

$$\alpha_{max} := \min_{1 \leq i \leq |\mathcal{I}_{si}|} \frac{(A_{si}x^* + B_{si}y + c_s)_i}{\max(0, -(A_{si}\delta x^* + B_{si}\delta y)_i)}. \tag{2.11}$$

Provided that δx^* is finite, α_{max} is strictly positive for vectors δy. But by the existence and uniqueness of δx^*, that must be true.

Therefore for any direction δy, it is concluded that $QP(y + \alpha\delta y) = x^* + \alpha\delta x^*$ for all $\alpha \in [0, \alpha_{max}]$. This implies the piecewise-linearity of QP and concludes the proof. □

The implications of Theorem 2.1.2 are twofold. First, for any $x^* = QP(y)$ such that strict complementarity holds, the set $\mathcal{I}_d = \emptyset$, and the conditions Eq. (2.6) are strictly linear. This implies that δ_x^* is a constant linear function of δy for all directions δy, and hence QP is differentiable at y.

When strict complementarity does not hold, determining the directional derivative of $QP(y)$ in some direction δy requires solving an inequality constrained quadratic program, however the number of inequality constraints to be resolved is limited to the size of the set $\mathcal{I}_d$, which is potentially much smaller than the number of inequality constraints in $QP(y)$ itself.

LCP Formulation

The quadratic program Program 2.1 can be equivalently represented as a linear complementarity problem (LCP), as seen in the following. Program 2.1 can be rewritten as

$$\begin{aligned} QP(y) := \ & \underset{x' \in \mathbb{R}^{n'}}{\text{arg min}} && x'^{\intercal}(\frac{1}{2}Q'x' + R'y + q') && \text{(2.12a)} \\ & \text{subject to} && A'x' + By + c \geq 0, && \text{(2.12b)} \\ & && x' \geq 0. && \text{(2.12c)} \end{aligned}$$

Note that it may be that the dimension $n' \neq n$, and similarly the terms Q', R', q', and A' may be different than their counterparts in (2.1). Any equality constraints can be absorbed into the inequality constraints (2.12b) by including both the constraint and the negation of the constraint as inequalities. The existence of a transformation between (2.1) and (2.12) can be seen by first introducing variables $x_+ \in \mathbb{R}^n$ and $x_- \in \mathbb{R}^n$, such that $n' = 2n$, and

$$\begin{aligned} & x = x_+ - x_-, && x' := \begin{bmatrix} x_+^\intercal & x_-^\intercal \end{bmatrix}^\intercal, && \\ & Q' := \begin{bmatrix} Q & \epsilon I - Q \\ \epsilon I - Q & Q \end{bmatrix}, && R' := \begin{bmatrix} R \\ -R \end{bmatrix}, && q' := \begin{bmatrix} q \\ -q \end{bmatrix}, \\ & A' := \begin{bmatrix} A & -A \end{bmatrix}. && && \end{aligned} \tag{2.13}$$

Substituting these values into (2.12) results in the objective

$$\frac{1}{2}(x_+ - x_-)^\intercal Q(x_+ - x_-) + (x_+ - x_-)^\intercal(Ry + q) + \epsilon x_+^\intercal x_-, \tag{2.14}$$

which is the objective in (2.1), with the addition of the term $\epsilon x_+^\intercal x_-$. Notice that for any unique value x^* solving (2.1), there are infinite possibilities of $x_+, x_- \in \mathbb{R}_+^n$ such that $x^* = x_+ - x_-$. Therefore the term $\epsilon x_+^\intercal x_-$ is introduced into the objective to enforce the condition that for each element of x_+ and x_-, only one can be non-zero, i.e. $(x_+)_i \geq 0 \perp (x_-)_i \geq 0$. Since both x_+ and x_- are restricted to be positive, the optimal value of the added term $\epsilon x_+^\intercal x_-$ is 0, and is attainable for every choice of $x = x_+ - x_-$. Therefore the addition of this term does not change the optimal value of the objective.

The value $\epsilon > 0$ is included such as to guarantee that the matrix Q' is positive definite if the matrix Q is positive definite. The matrix Q' can be seen to be

$$Q' := \bar{Q} + E, \quad \bar{Q} := \begin{bmatrix} Q & -Q \\ -Q & Q \end{bmatrix}, \quad E := \begin{bmatrix} & \epsilon I \\ \epsilon I & \end{bmatrix}. \tag{2.15}$$

Since the matrices $\bar{Q}$ and E commute and are therefore simultaneously diagonalizable, the eigenvalues of the symmetric matrix Q' are the sum of the eigenvalues of $\bar{Q}$ and E. From this it is easy to show that the spectrum of Q' is

$$\sigma(Q') := \{2\sigma(Q) - \epsilon\} \cup \{\epsilon\}. \tag{2.16}$$

Therefore when Q is positive definite, for small enough ϵ, Q' is also positive definite. When Q is positive semi-definite, Q' is also positive semi-definite for $\epsilon = 0$.

It is possible that transformations other than Eq. (2.13) can bring QPs of the form 2.1 to the form 2.12. However the guaranteed existence of this transformation serves to show that general quadratic programs can be expressed in the form 2.12 without loss of generality, and while preserving the properties of the quadratic objective.

Expressing the first-order necessary conditions for Program 2.12 (analogous to those in Eq. (2.2)), gives rise to the following conditions:

$$\begin{bmatrix} Q' & -A'^\intercal \\ A' & \end{bmatrix} \begin{bmatrix} x' \\ \lambda' \end{bmatrix} + \begin{bmatrix} R' \\ B \end{bmatrix} y + \begin{bmatrix} q' \\ c \end{bmatrix} \geq 0 \perp \begin{bmatrix} x' \\ \lambda' \end{bmatrix} \geq 0. \tag{2.17}$$

The condition Eq. (2.17) takes the form of a linear complementarity problem. Therefore, when the first-order necessary conditions are also sufficient for optimality (i.e. under a convexity assumption), solving the QP (2.1) is equivalent to solving the LCP (2.17). Methods for solving LCPs can be applied to solving QPs. As will be seen in the following sections, more general problem formulations can also be cast as LCPs, implying that some of the same techniques used for solving and analyzing QPs (via LCPs) can be used to solve and analyze those more general problem formulations.

2.2 Parametric Equilibrium Problems

On the road to generalizing QPs to groups of QPs, the first natural extension to introduce is the (Generalized Nash) equilibrium problem. An equilibrium problem, for the purposes of this dissertation, is a collection of N QPs, each of which are to be solved simultaneously. In particular, for each $i \in \{1, ..., N\}$, a parametric QP (using form of (2.12)) is defined as

$$
\begin{aligned}
QP_i(y, x_{-i}) := \quad & \underset{x_i \in \mathbb{R}^{n_i}}{\arg\min} && x^\intercal(\frac{1}{2}Q_i x + R_i y + q_i) && \text{(2.18a)}\\
& \text{subject to} && A_i x + B_i y + c_i \geq 0, && \text{(2.18b)}\\
& && x_i \geq 0. && \text{(2.18c)}
\end{aligned}
$$

Here, $x := [x_1^\intercal \ \dots \ x_N^\intercal]^\intercal$, $x_{-i} := [x_1^\intercal \ \dots \ x_{i-1}^\intercal \ x_{i+1}^\intercal \ \dots \ x_N^\intercal]^\intercal$, $n = \sum_{i=1}^N n_i$, and as before, the parameter $y \in \mathbb{R}^m$.

An equilibrium solution to the collection of problems (2.18) is a vector x^* belonging to the following set:

$$
EQP(y) := \left\{ x^* \in \mathbb{R}^n \mid x_i^* \in QP_i(y, x_{-i}^*), \quad \forall i \in \{1, ..., N\} \right\}. \tag{2.19}
$$

When the programs (2.18) are strongly convex, the corresponding sets $QP_i(y, x_{-i})$, $i \in \{1, ..., N\}$ are singleton sets for every choice of y, x_{-i}. Even when this is the case, it is still possible that the set $EQP(y)$ contains multiple elements for any given choice of the parameter y.

In general, the set of equilibrium solutions is a set-valued mapping $EQP : \mathbb{R}^m \rightrightarrows \mathbb{R}^n$. In sections to follow, we will be interested in how the set $EQP(y)$ responds to changes in the parameter y, and under which conditions the set $EQP(y)$ contains a single unique element for each y. To analyze these questions, we equivalently cast the EQP problem as a LCP (as was done in the preceding section).

Assuming each of the QPs (2.18) are convex in their decision variable, the set $EQP(y)$ can be equivalently represented as the set of vectors x^* such that there exist multipliers λ^* satisfying the concatenated first-order necessary conditions of optimality of all N QPs:

$$
EQP(y) := \left\{ \begin{array}{ll} x^* \in \mathbb{R}^n \mid \exists \lambda^* : (Q_i)_{(i,:)} x^* + (R_i)_{(i,:)} y + (q_i)_{(i)} - (A_i)^\intercal_{(:,i)} \lambda_i^* \geq 0 & \perp x_i \geq 0 \\ \qquad\qquad\qquad\quad A_i x^* + B_i y + c_i \geq 0 & \perp \lambda_i^* \geq 0, \\ \qquad\qquad\qquad\quad \forall i \in \{1, ..., N\} & \end{array} \right\} \tag{2.20}
$$

With a slight abuse in notation, the shorthand $(Q_i)_{(i,:)}$, for example, is used to represent the $\sum_{j=1}^{i-1} n_i + 1$ through $\sum_{j=1}^{i} n_i$ rows of Q_i. From this, it can be seen that the conditions (2.20) can be compactly represented as the following parametric LCP:

$$EQP(y) := \left\{ x^* \in \mathbb{R}^n \mid \exists \lambda^* : \begin{bmatrix} \mathbf{Q} & -\bar{\mathbf{A}}^\intercal \\ \mathbf{A} & \end{bmatrix} \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} + \begin{bmatrix} \mathbf{R} \\ \mathbf{B} \end{bmatrix} y + \begin{bmatrix} \mathbf{q} \\ \mathbf{c} \end{bmatrix} \geq 0 \perp \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} \geq 0 \right\}, \tag{2.21}$$

where the block matrix terms introduced are defined as

$$\begin{aligned} \mathbf{Q} &:= \begin{bmatrix} (Q_1)_{(1,:)} \\ \vdots \\ (Q_N)_{(N,:)} \end{bmatrix}, \quad \mathbf{A} := \begin{bmatrix} A_1 \\ \vdots \\ A_N \end{bmatrix}, \quad \bar{\mathbf{A}} := \begin{bmatrix} (A_1)_{(:,1)} & & \\ & \ddots & \\ & & (A_N)_{(:,N)} \end{bmatrix}, \\ \mathbf{R} &:= \begin{bmatrix} (R_1)_{(1,:)} \\ \vdots \\ (R_N)_{(N,:)} \end{bmatrix}, \quad \mathbf{B} := \begin{bmatrix} B_1 \\ \vdots \\ B_N \end{bmatrix}, \quad \mathbf{q} := \begin{bmatrix} (q_1)_{(1)} \\ \vdots \\ (q_N)_{(N)} \end{bmatrix}, \quad \mathbf{c} := \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix}. \end{aligned} \tag{2.22}$$

Using the LCP form of $EQP(y)$, we can establish results about the non-degeneracy of the set for each value y — in other words, we can establish the existence of solutions to the parametric equilibrium problem.

Existence of Solutions

The following theorem makes use of the concept of copositivity of a matrix. A matrix M is said to be copositive, if $x \geq 0 \implies x^\intercal M x \geq 0$. The matrix M is said to be strictly copositive if $x \geq 0, x \neq 0 \implies x^\intercal M x > 0$. For ease of notation, denote the matrix $\mathbf{M} := \begin{bmatrix} \mathbf{Q} & -\bar{\mathbf{A}}^\intercal \\ \mathbf{A} & \end{bmatrix}$.

Theorem 2.2.1. *Let $\mathbf{Q}$ be strictly copositive. Let the positive cone formed by the columns of $(A_i)_{(:,-i)}$ be a subset of the non-negative orthant, and let the intersection of the positive cone formed by the columns of $(A_i)_{(:,i)}$ with the negative orthant be empty. Then the set $EQP(y)$ is non-empty for every choice of $y \in \mathbb{R}^m$, meaning an equilibrium solution exists.*

Proof of Theorem 2.2.1. Under the stated assumption, it can be seen that the matrix $\mathbf{M}$ is copositive.

$$\begin{bmatrix} x \\ \lambda \end{bmatrix}^\intercal \begin{bmatrix} \mathbf{Q} & -\bar{\mathbf{A}}^\intercal \\ \mathbf{A} & \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = x^\intercal \mathbf{Q} x + \lambda^\intercal \check{\mathbf{A}} x, \tag{2.23}$$

where

$$\check{\mathbf{A}} := \mathbf{A} - \bar{\mathbf{A}}. \tag{2.24}$$

When both $x \geq 0$ and $\lambda \geq 0$, $\check{\mathbf{A}}x \geq 0$ and $x^\intercal \mathbf{Q} x \geq 0$, therefore the right-hand side of Eq. (2.23) must be non-negative, implying that $\mathbf{M}$ is indeed copositive.

Additionally, whenever

$$\begin{bmatrix} x \\ \lambda \end{bmatrix}^\intercal \mathbf{M} \begin{bmatrix} x \\ \lambda \end{bmatrix} = 0, \quad \mathbf{M} \begin{bmatrix} x \\ \lambda \end{bmatrix} \geq 0, \quad \text{and} \quad \begin{bmatrix} x \\ \lambda \end{bmatrix} \geq 0, \tag{2.25}$$

it must be that

$$\begin{bmatrix} x \\ \lambda \end{bmatrix}^\intercal \left(\begin{bmatrix} \mathbf{R} \\ \mathbf{B} \end{bmatrix} y + \begin{bmatrix} \mathbf{q} \\ \mathbf{c} \end{bmatrix} \right) \geq 0. \tag{2.26}$$

To see this, notice that by the strict copositivity of $\mathbf{Q}$, the first condition in Eq. (2.25) implies that $x = 0$. Furthermore, by the second condition in Eq. (2.25), $\bar{\mathbf{A}}^\intercal \lambda \leq 0$. However, since $\lambda \geq 0$ by the third condition, the value $\bar{\mathbf{A}}^\intercal \lambda$ lies in the positive cone formed by the columns of $\bar{\mathbf{A}}^\intercal$, which by our assumption does not intersect the negative orthant. Therefore it must be that $\lambda = 0$, implying Eq. (2.26).

Finally, by Theorem 3.8.6 in [48], the LCP in Eq. (2.21) must have a solution for all values of $y \in \mathbb{R}^m$. □

The main restriction of Theorem 2.2.1 is the non-negativity of $\check{\mathbf{A}}$. This requirement will be violated, for example, if problems of the form (2.18)Note that this requirement is of course satisfied when $\check{\mathbf{A}} = 0$, which is the case when the constraints (2.18b) are independent of x_{-i}. Such problems arise in the classic Nash equilibrium problems (as opposed to the generalized Nash equilibrium problem of consideration here).

Sometimes this restriction is too stringent though, and it is therefore fruitful to leverage alternate existence theorems to guarantee solutions exist for $EQP(y)$, namely using a fixed-point argument.

Theorem 2.2.2. *For each $i \in \{1, ..., N\}$, let the objective (2.18a) appearing in QP_i 2.18 be convex with respect to x_i. Furthermore, let the constraint set defined by Eqs.* (2.18b) *and* (2.18c) *be non-empty and bounded for every value of $y \in \mathbb{R}^m, x_{-i} \in \mathbb{R}^{n-n_i}$. Then there exists an equilibrium solution to the problem $EQP(y)$ for every y.*

Proof of Theorem 2.2.2. This follows from direct application of Theorem 3.1 in [66]. □

The requirement in Theorem 2.2.2 that the constraint set Eqs. (2.18b) and (2.18c) be non-empty and bounded for every value of $y \in \mathbb{R}^m, x_{-i} \in \mathbb{R}^{n-n_i}$ may in some cases be more restrictive than the requirement that the positive cone of $\check{\mathbf{A}}$ is a subset of the non-negative orthant, appearing in Theorem 2.2.1. Therefore, these two existence theorems provide alternative conditions which are sufficient for the existence of solutions to the equilibrium problem at hand, and therefore both useful.

Multiplicity and Differentiability of Solutions

Throughout this dissertation, it will be of importance to rely on the locally unique solution of the EQP (and other related problems) at various points y. Therefore in this section, some results on the multiplicity of solutions are presented. When solutions are indeed locally unique, it will be seen that the the local solution is represented by a piecewise linear function in the input y.

The concept of an R_0 matrix is used in the following theorem. A matrix M is an R_0 matrix, if for every vector $x \geq 0$, if $Mx \geq 0$ and $x^\intercal M x = 0$, it must be that $x = 0$.

Theorem 2.2.3. *Let the assumptions made in Theorem 2.2.1 hold. Then if the matrix* $\mathbf{M}$ *additionally has no negative principal minors (i.e.* $\mathbf{M}$ *is a* P_0 *matrix), then the number of solution to* $EQP(y)$ *is finite, and each solution is locally unique.*

Proof of Theorem 2.2.3. By Theorem 2.2.1, the set $EQP(y)$ is non-empty for all y, implying $\mathbf{M}$ is a Q matrix. Therefore, by Theorem 3.9.22 in [48], it follows that $\mathbf{M}$ is also an R_0 matrix (see reference), establishing the local uniqueness of solutions. □

From this result, we can further establish that the set of solutions to this problem for small changes of y in some direction d can be represented by a set of linear functions of y. In order to establish this result, it is useful to return to an alternate, equivalent formulation of Program 2.18:

$$\widetilde{QP}_i(y, x_{-i}) := \underset{x_i \in \mathbb{R}^{n_i}}{\arg\min} \quad x^\intercal(\frac{1}{2}\tilde{Q}_i x + \tilde{R}_i y + \tilde{q}_i) \tag{2.27a}$$

$$\text{subject to} \quad \tilde{A}_i x + \tilde{B}_i y + \tilde{c}_i \geq 0 \tag{2.27b}$$

Here, the bound constraints $x_i \geq 0$ appearing in Eq. (2.18c) are dropped, which can always be done without loss of generality by simply absorbing them into the constraint (2.27b). To distinguish this form of the QPs of consideration from those in (2.18), the tilde notation is used. When each of the programs (2.27) are convex, the first-order necessary conditions of optimality no longer give rise to a linear complementarity problem, but rather a linear mixed complementarity problem (MCP). These conditions give rise to an alternate definition of the set $EQP(y)$:

$$EQP(y) := \left\{ \begin{array}{c} x^* \in \mathbb{R}^n \mid \exists \lambda^* : (\tilde{Q}_i)_{(i,:)} x^* + (\tilde{R}_i)_{(i,:)} y + (\tilde{q}_i)_{(i)} - (\tilde{A}_i)^\intercal_{(:,i)} \lambda_i^* = 0 \\ \tilde{A}_i x^* + \tilde{B}_i y + \tilde{c}_i \geq 0 \perp \lambda_i^* \geq 0, \\ \forall i \in \{1, ..., N\} \end{array} \right\} \tag{2.28}$$

Similar to how was done in the preceding section, for some solution $x^* \in EQP(y)$, with associated multiplier λ^* define the following index sets, and associated coefficient terms:

$$\begin{aligned} \mathcal{I}^i_{\text{a}} &:= \{\, j \mid (\tilde{A}_i x^* + \tilde{B}_i y + \tilde{c}_i)_j = 0\}, \\ \mathcal{I}^i_{\text{i}} &:= \{\, j \mid (\lambda_i^*)_j = 0\}, \\ \mathcal{I}^i_{\text{d}} &:= \mathcal{I}^i_{\text{a}} \cap \mathcal{I}^i_{\text{i}}, \\ \mathcal{I}^i_{\text{sa}} &:= \mathcal{I}^i_{\text{a}} \setminus \mathcal{I}^i_{\text{d}}, \\ \mathcal{I}^i_{\text{si}} &:= \mathcal{I}^i_{\text{i}} \setminus \mathcal{I}^i_{\text{d}}. \end{aligned} \tag{2.29}$$

$$\begin{array}{cccc} (\tilde{A}_i)_{sa} := (\tilde{A}_i)_{\mathcal{I}^i_{\text{sa}}} & (\tilde{B}_i)_{sa} := (\tilde{B}_i)_{\mathcal{I}_{\text{sa}}} & (\tilde{c}_i)_{sa} := (\tilde{c}_i)_{\mathcal{I}^i_{\text{sa}}} & (\lambda_i)_{sa} := (\lambda_i)_{\mathcal{I}^i_{\text{sa}}} \\ (\tilde{A}_i)_{si} := (\tilde{A}_i)_{\mathcal{I}^i_{\text{si}}} & (\tilde{B}_i)_{si} := (\tilde{B}_i)_{\mathcal{I}^i_{\text{si}}} & (\tilde{c}_i)_{si} := (\tilde{c}_i)_{\mathcal{I}^i_{\text{si}}} & (\lambda_i)_{si} := (\lambda_i)_{\mathcal{I}^i_{\text{si}}} \\ (\tilde{A}_i)_{d} := (\tilde{A}_i)_{\mathcal{I}^i_{\text{d}}} & (\tilde{B}_i)_{d} := (\tilde{B}_i)_{\mathcal{I}^i_{\text{d}}} & (\tilde{c}_i)_{d} := (\tilde{c}_i)_{\mathcal{I}^i_{\text{d}}} & (\lambda_i)_{d} := (\lambda_i)_{\mathcal{I}^i_{\text{d}}} \end{array} \tag{2.30}$$

Theorem 2.2.4. *Let the assumptions made in Theorem 2.2.3 hold. Let $x^* \in EQP(y)$ with associated multiplier vector λ^*. Define $\mathcal{N}(x^*) \subset \mathbb{R}^n$ to be some local neighborhood around x^*. Then for small enough ϵ, the set of solutions to the set $EQP(y+\epsilon\delta y) \cap \mathcal{N}(x^*)$ is the set of $x^* + \epsilon\delta x$, where δx is defined to be any solution to the equilibrium problem defined by the following collection of quadratic programs:*

$$\mathbf{D}QP_i(y,x^*)[\delta y, \delta x_{-i}] := \underset{\delta x_i \in \mathbb{R}^{n_i}}{\arg\min} \quad \delta x^\intercal(\frac{1}{2}\tilde{Q}_i\delta x + \tilde{R}_i\delta y) \tag{2.31a}$$

$$\text{subject to} \quad (\tilde{A}_i)_{sa}\delta x + (\tilde{B}_i)_{sa}\delta y = 0, \tag{2.31b}$$

$$(\tilde{A}_i)_d\delta x + (\tilde{B}_i)_d\delta y \geq 0 \tag{2.31c}$$

$$\mathbf{D}EQP(y,x^*)[\delta y] := \left\{\delta x^* \in \mathbb{R}^n \mid \delta x_i^* \in \mathbf{D}QP_i(y,x^*)[\delta y, \delta x_{i-1}^*], \quad \forall i \in \{1,...,N\}\right\}. \tag{2.32}$$

Proof of Theorem 2.2.4. Expressing the first-order necessary conditions for the collection of convex QPs (2.31) allows us to express the set $\mathbf{D}EQP(y,x^*)[\delta y]$ as the following:

$$\mathbf{D}EQP(y,x^*)[\delta y] := \left\{ \begin{array}{ll} \delta x^* \in \mathbb{R}^n \mid \exists\,((\delta\lambda)_{sa}^*, (\delta\lambda)_d^*) : & (\tilde{Q}_i)_{(i,:)}\delta x^* + (\tilde{R}_i)_{(i,:)}\delta y - \\ & ((\tilde{A}_i)_d)_{(:,i)}^\intercal(\delta\lambda_i)_d^* - \\ & ((\tilde{A}_i)_{sa})_{(:,i)}^\intercal(\delta\lambda_i)_{sa}^* = 0, \\ & (\tilde{A}_i)_{sa}\delta x^* + (\tilde{B}_i)_{sa}\delta y = 0, \\ & (\tilde{A}_i)_d\delta x^* + (\tilde{B}_i)_d\delta y \geq 0 \perp (\delta\lambda_i)_d^* \geq 0, \\ & \forall i \in \{1,...,N\} \end{array} \right\} \tag{2.33}$$

Define $(\delta\lambda)_{si}^* := 0$, and let $\delta\lambda^*$ be the reconstruction of $(\delta\lambda)_{sa}^*, (\delta\lambda)_{si}^*,$ and $(\delta\lambda)_d^*$, in the original index ordering. Following the technique used in Theorem 2.1.2, it follows directly that for small enough ϵ, the set of solutions $EQP(y^* + \epsilon\delta y)$ is represented by the set of $x^* + \epsilon\delta x^*$, $\lambda^* + \epsilon\delta\lambda^*$, where $\delta x^* \in \mathbf{D}EQP(y,x^*)[\delta y]$ and $\delta\lambda^*$ are associated multipliers. In particular, define

$$\epsilon_{max} := \underset{\epsilon > 0}{\arg\max} \quad \epsilon \tag{2.34a}$$

$$\text{subject to} \quad (\tilde{A}_i)_{si}(x^* + \epsilon\delta x^*) + (\tilde{B}_i)_{si}(y + \epsilon\delta y) + (\tilde{c}_i)_{si} \geq 0, \quad \forall i, \tag{2.34b}$$

$$(\lambda_i)_{sa} + \epsilon(\delta\lambda_i)_{sa}^* \geq 0, \quad \forall i. \tag{2.34c}$$

Then for all $0 \leq \epsilon \leq \epsilon_{max}$, $x^* + \epsilon\delta x^*$ is a solution to $EQP(y + \epsilon\delta y)$.

□

The assumptions made in Theorem 2.2.4 are sufficient conditions for the solution set of $EQP(y^* + \epsilon\delta y) \cap \mathcal{N}(x^*)$ to be represented by a collection of affine functions in the change δy. If the equilibrium problem defined by (2.31) contains multiple solutions (the number of solutions must be finite), then there exist multiple valid local affine representations of solutions to the set $EQP(y+\epsilon\delta y)$ emanating from the point x^*. For example, if δx_a^* and δx_b^* are two solutions to (2.31), then $x^*+\epsilon\delta x_a^*$ and $x^*+\epsilon\delta x_b^*$ are both solutions to $EQP(y+\epsilon\delta y)$. This implies that the point (y, x^*) is a multifurcation point in the graph $\mathcal{G}(EQP)$.

Theorem 2.2.5. *Let $EQP(y)$ have a finite and non-zero number of solutions for every $y \in \mathbb{R}^m$. For some particular y^*, let $x^* \in EQP(y^*)$. Then there exists a piecewise-linear mapping $\bar{x} = K(\bar{y})$, such that $\bar{x}$ is a solution to $EQP(\bar{y})$ for all $\bar{y} \in \mathbb{R}^m$, and with $x^* = K(y^*)$.*

Proof. Consider x^* as a solution in $EQP(y)$, and any direction $\delta y \in \mathbb{R}^m$. By the assumptions in Theorem 2.2.3, because $EQP(y)$ is non-empty for all y, $\mathbf{D}EQP(y, x^*)[\delta y]$ is non-empty for all choice of δy as well. Furthermore, because $EQP(y)$ contains a finite number of elements for all y, so must $\mathbf{D}EQP(y, x^*)[\delta y]$.

Let $((\delta\lambda)_{sa}^*, (\delta\lambda)_d^*)$ be the multipliers associated with some solution δx^* for the direction δy. Define for this solution, the index sets

$$\begin{aligned} \mathcal{J}_{\mathrm{a}}^i &:= \{\ j \mid \left((\tilde{A}_i)_d\delta x^* + (\tilde{B}_i)_d\delta y\right)_j = 0\} \\ \mathcal{J}_{\mathrm{i}}^i &:= \{\ j \mid \left((\tilde{A}_i)_d\delta x^* + (\tilde{B}_i)_d\delta y\right)_j > 0\}, \end{aligned} \tag{2.35}$$

Consider the system of equations formed from Eq. (2.33) by replacing each row j of the complementarity condition with $\left((\tilde{A}_i)_d\delta x^* + (\tilde{B}_i)_d\delta y\right)_j = 0$ if $j \in \mathcal{J}_{\mathrm{a}}^i$ and the constraint is linearly independent from all other "active" constraints, else replacing it with $((\delta\lambda_i)_d)_j = 0$ if $j \in \mathcal{J}_{\mathrm{i}}^i$. This is now an equality-constrained system, which by the property that $\mathbf{D}EQP(y, x^*)[\delta y]$ contains a finite number of elements, must be non-singular. This system takes the form

$$\bar{\mathbf{M}}\begin{bmatrix}\delta x^*\\ \delta\lambda^*\end{bmatrix} + \bar{\mathbf{N}}\delta y = 0, \tag{2.36}$$

which by the non-singularity of the system, implies linear relationships

$$\delta x^* = K\delta y,\ \ \delta\lambda^* = L\delta y. \tag{2.37}$$

If $(L_i\delta y)_d = (\delta\lambda_i)_d$, then these linear relationships must hold for all direction δy satisfying

$$\begin{aligned} ((L_i\delta y)_d)_j \geq 0,&\ \ j \in \mathcal{J}_{\mathrm{a}}^i \\ \left((\tilde{A}_i)_dK + (\tilde{B}_i)_d)\delta y\right)_j \geq 0,&\ \ j \in \mathcal{J}_{\mathrm{i}}^i \end{aligned} \tag{2.38}$$

By Theorem 2.2.4, this implies that for small enough ϵ, $x^* + \epsilon K\delta y$ is a solution to $EQP(y + \epsilon\delta y)$ for all δy satisfying the conditions (2.38). From (2.34), the exact region for which this linear mapping is valid can be established.

$x^* + K\delta y$ is a solution to $EQP(y+\delta y)$ for all $\delta y \in \mathcal{R}$, defined as

$$\mathcal{R} := \left\{ \delta y \in \mathbb{R}^m \; : \; \forall i, \; \begin{bmatrix} ((L_i \delta y)_d)_j \geq 0, \; j \in \mathcal{J}^i_{\mathrm{a}} \\ \Big((\tilde{A}_i)_d K + (\tilde{B}_i)_d)\delta y \Big)_j \geq 0, \; j \in \mathcal{J}^i_{\mathrm{i}} \\ (\tilde{A}_i)_{si}(K\delta y + x^*) + (\tilde{B}_i)_{si}(y+\delta y) + (\tilde{c}_i)_{si} \geq 0 \\ ((L\delta y + \lambda^*)_i)_{sa} \geq 0 \end{bmatrix} \right\} \tag{2.39}$$

It is useful to represent the region defined by $\{y+\delta y, \; \delta y \in \mathcal{R}\}$ compactly as $\{y \in \mathbb{R}^m : Dy + d \geq 0\}$, as will be seen in later sections.

Thus far it has been established that for any direction δy, there exists an affine mapping relating solutions to $EQP(y+\delta y)$ to $y+\delta y$, which passes through x^*, which is valid for some polyhedral region containing the origin. Because the existence of this piecewise-affine mapping must exist for every direction δy at every solution pair (y, x^*), this implies a globally existent piecewise linear mapping from points $\bar{y} \in \mathbb{R}^m$ to solutions $EQP(\bar{y})$. □

The conditions provided in this section are sufficient to ensure the existence of equilibrium solutions to the equilibrium problem, the finite cardinality of solutions, and a piecewise linear relationship between parameter and solutions. All of the conditions listed are not necessary for these properties to hold. In general, each of these desirable properties may hold for other equilibrium problems, even if they do not satisfy the sufficient conditions outlined.

2.3 Equilibrium Problems with Equilibrium Constraints

We now consider a generalization of the equilibrium problem defined in the previous section, as we move one step closer to introducing the general connected quadratic programs that are the primary focus of this dissertation.

Let $EQP(y)$ denote the solution to a parametric equilibrium problem defined as in (2.18), (2.19). We denote an equilibrium problem with equilibrium constraints to be an equilibrium solution of the following N_y quadratic programs, each which are subject to shared equilibrium constraints involving the variable x:

$$QPEC_i(z, y_{-i}) := \underset{y_i \in \mathbb{R}^{m_i}, x \in \mathbb{R}^n}{\arg\min} \quad \begin{bmatrix} y_i \\ x \end{bmatrix}^\intercal (\frac{1}{2} Q_{y,i} \begin{bmatrix} y_i \\ x \end{bmatrix} + R_{y,i} \begin{bmatrix} y_{-i} \\ z \end{bmatrix} + q_{y,i}) \tag{2.40a}$$

$$\text{subject to} \quad A_{y,i} \begin{bmatrix} y_i \\ x \end{bmatrix} + B_{y,i} \begin{bmatrix} y_{-i} \\ z \end{bmatrix} + c_{y,i} \geq 0, \tag{2.40b}$$

$$x \in EQP(y). \tag{2.40c}$$

Here it is implied that $\sum_{i=1}^{N_y} m_i = m$. Each of the above programs take as parameter some value $z \in \mathbb{R}^l$, as well as the decision variables y_{-i}. The problem of finding an equilibrium for this set of parametric programs is called an equilibrium problem with equilibrium constraints, or an EPEC:

$$EPEC(z) := \left\{x^* \in \mathbb{R}^n, y^* \in \mathbb{R}^m | (y_i^*, x^*) \in QPEC_i(z, y_{-i}^*), \forall i \in \{1, ..., N_y\}\right\} \tag{2.41}$$

Here $EPEC$ can be seen to be a set-valued mapping $EPEC : \mathbb{R}^l \rightrightarrows \mathbb{R}^{n+m}$.

Theorem 2.3.1. *Assume that the EQP appearing in Eq.* (2.40c) *has a finite and non-zero number of isolated solutions for every value of y, and therefore the result of Theorem 2.2.5 applies. Then for some* $z \in \mathbb{R}^l$, *if* (y^*, x^*) *are solutions to the equilibrium problem with equilibrium constraints, i.e.* $(y^*, x^*) \in EPEC(z)$, *then for every portion of the piecewise-linear mapping(s) defined by* $\{y = Kx + k,\ Dy + d \geq 0\}$ *and satisfying* $x^* = Ky^* + k,\ Dy^* + d \geq 0$ *as given in Theorem 2.2.5,* (y^*, x^*) *is also a solutions to the equilibrium problem defined by the following programs:*

$$\begin{aligned}
QPEC_i^{(K,k,D,d)}(z, y_{-i}) := \quad & \underset{y_i \in \mathbb{R}^{m_i}, x \in \mathbb{R}^n}{\arg\min} \quad \begin{bmatrix} y_i \\ x \end{bmatrix}^\intercal \left(\frac{1}{2} Q_{y,i} \begin{bmatrix} y_i \\ x \end{bmatrix} + R_{y,i} \begin{bmatrix} y_{-i} \\ z \end{bmatrix} + q_{y,i}\right) && (2.42a) \\
& \text{subject to} \quad A_{y,i} \begin{bmatrix} y_i \\ x \end{bmatrix} + B_{y,i} \begin{bmatrix} y_{-i} \\ z \end{bmatrix} + c_{y,i} \geq 0, && (2.42b) \\
& Dy + d \geq 0, && (2.42c) \\
& x - (Ky + k) = 0. && (2.42d)
\end{aligned}$$

$$EPEC^{(K,k,D,d)}(z) := \left\{(x, y) \mid (y_i, x) \in QPEC_i^{(K,k,D,d)}(z, y_{-i}),\ \forall i \in \{1, ..., N_y\}\right\} \tag{2.43}$$

In other words, $(x^*, y^*) \in EPEC^{(K,k,D,d)}(z)$ *for all such* (K, k, D, d).

Proof of Theorem 2.3.1. This proof follows directly from the fact that locally, the feasible domain for the programs (2.40) is the union of the feasible domains for the programs (2.42). Therefore if there exist two pieces of the piecewise linear representation of the set $EQP(y)$, denoted by (K_1, k_1, D_1, d_1) and (K_2, k_2, D_2, d_2), such that they are neighboring at (x^*, y^*), i.e. $D_1 y^* + d_1 \geq 0$, $D_2 y^* + d_2 \geq 0$, and $K_1 y^* + k_1 = K_2 y^* + k_2 = x^*$. Let $(x^*, y^*) \in EPEC^{(K_1,k_1,D_1,d_1)}(z)$, but $(x^*, y^*) \notin EPEC^{(K_2,k_2,D_2,d_2)}(z)$. This immediately implies (x^*, y^*) can't be an equilibrium point of $EPEC(z)$, since there must exist a point (x, y) in the domain $D_2 y + d_2 \geq 0, \quad x = K_2 y + k_2$ which unilaterally decreases the objective value for one of the programs (2.40 (by definition of equilibrium), and therefore a point (x, y) in the domain $x \in EQP(y)$ which decreases the objective value for one of the programs (2.42).

On the contrary, if there exist no points in any of the independent domains which unilaterally decreases the objective value for one of the programs (2.40, then there cannot exist a point in the union of domains which unilaterally decreases the objective value for one of the programs (2.42). This establishes the result. □

Theorem 2.3.2. *Let the matrices $Q_{y,i}$ appearing in Eq.* (2.42a) *be positive semi-definite for all $i \in \{1, .., N_y\}$. Furthermore, for all i, and for the polyhedral region $Dy + d \geq 0$ appearing in Eq.* (2.42c)*, let the following region be a bounded and non-empty set for every $y_{-i} \in \mathbb{R}^{m-m_i}$, $z \in \mathbb{R}^l$:*

$$\mathcal{C}_i(y_{-i}, z) := \left\{ y_i \in \mathbb{R}^{m_i} : \begin{bmatrix} A_{y,i} \begin{bmatrix} y_i \\ Ky + k \end{bmatrix} + B_{y,i} \begin{bmatrix} y_{-i} \\ z \end{bmatrix} + c_{y,i} \geq 0, \\ Dy + d \geq 0 \end{bmatrix} \right\} \tag{2.44}$$

Then, there exists an equilibrium solution to the EPEC defined in terms of the quadratic programs (2.42), i.e. the set $EPEC^{(K,k,D,d)}(z)$ (2.43) is non-empty.

Proof of Theorem 2.3.2. For each program (2.42), substitute x using the constraint $x-(Ky+k) = 0$. We can then re-express each program as the following:

$$\begin{aligned} QPEC_i^{(K,k,D,d)}(z, y_{-i}) := \quad & \underset{y_i \in \mathbb{R}^{m_i}}{\arg\min} \quad y_i^\intercal \left(\frac{1}{2}\tilde{Q}_{y,i} y_i + \tilde{R}_{y,i} \begin{bmatrix} y_{-i} \\ z \end{bmatrix} + \tilde{q}_{y,i}\right) && \text{(2.45a)} \\ & \text{subject to} \quad \tilde{A}_{y,i} y_i + \tilde{B}_{y,i} \begin{bmatrix} y_{-i} \\ z \end{bmatrix} + \tilde{c}_{y,i} \geq 0, && \text{(2.45b)} \\ & \qquad\qquad Dy + d \geq 0. && \text{(2.45c)} \end{aligned}$$

Here, the new coefficients are given as

$$\begin{aligned} \tilde{Q}_{y,i} &:= \begin{bmatrix} I \\ K_i \end{bmatrix}^\intercal Q_{y,i} \begin{bmatrix} I \\ K_i \end{bmatrix}, & \tilde{R}_{y,i} &:= \begin{bmatrix} I \\ K_i \end{bmatrix}^\intercal \left(Q_{y,i} \begin{bmatrix} 0 & 0 \\ K_{-i} & 0 \end{bmatrix} + R_{y,i} \right) \\ \tilde{q}_{y,i} &:= \begin{bmatrix} I \\ K_i \end{bmatrix}^\intercal (Q_{y,i} \begin{bmatrix} 0 \\ k \end{bmatrix} + q_{y,i}), & \tilde{A}_{y,i} &:= A_{y,i} \begin{bmatrix} I \\ K_i \end{bmatrix}, \\ \tilde{B}_{y,i} &:= A_{y,i} \begin{bmatrix} 0 & 0 \\ K_{-i} & 0 \end{bmatrix} + B_{y,i}, & \tilde{c}_{y,i} &:= A_{y,i} \begin{bmatrix} 0 \\ k \end{bmatrix} + c_{y,i}. \end{aligned} \tag{2.46}$$

Here K_i refers to the $\sum_{j=1}^{i-1} m_j + 1$ through $\sum_{j=1}^{i} m_j$ columns of K, and K_{-i} refers to all other columns. It can be seen that since $Q_{y,i}$ is positive semi-definite by assumption, and the matrix $\begin{bmatrix} I & K_i^\intercal \end{bmatrix}^\intercal$ is full rank, $\tilde{Q}_{y,i}$ is positive semi-definite, and the program is convex in the variable y_i.

The constraints (2.45b and 2.45c) give rise to the set $\mathcal{C}_i(y_{-i}, z)$ in (2.44), which is non-empty and bounded for all y_{-i} and z by assumption. Therefore by Theorem 3.1 in [66], there must exist an equilibrium point for this collection of problems. □

Before introducing and proving another key theorem, it is useful to describe the set of solutions to the equilibrium problem $EPEC^{K,k,D,d}(z)$ in terms of the first-order necessary conditions for the individual component programs. This gives:

$$EPEC^{(K,k,D,d)}(z) := \left\{ \begin{aligned} &(x^*, y^*) \mid \exists \lambda^*, \gamma^* : (\tilde{Q}_{y,i})_{(i,:)} y_i^* + (\tilde{R}_{y,i})_{(i,:)} \begin{bmatrix} y_{-i}^* \\ z \end{bmatrix} + \\ &\qquad\qquad\qquad (\tilde{q}_{y,i})_{(i)} - (\tilde{A}_{y,i})^\intercal_{(:,i)} \lambda_i^* - D^\intercal_{(:,i)} \gamma^* = 0, \\ &\qquad\qquad\qquad \tilde{A}_{y,i} y_i^* + \tilde{B}_{y,i} \begin{bmatrix} y_{-i} \\ z \end{bmatrix} + \tilde{c}_{y,i} \geq 0 \perp \lambda_i^* \geq 0, \\ &\qquad\qquad\qquad \forall i \in \{1, ..., N_y\}, \\ &\qquad\qquad\qquad Dy + d \geq 0 \perp \gamma^* \geq 0 \end{aligned} \right\} \tag{2.47}$$

Notice that since all $QPEC_i^{(K,k,D,d})$ share the constraints $Dy + d \geq 0$, a shared multiplier is used in the first-order necessary conditions of optimality.

Consider some solution (x^*, y^*) to $EPEC^{K,k,D,d}(z)$, with associated multipliers λ^* and γ^*. As in previous sections, define the index sets and coefficients

$$\begin{aligned} \mathcal{I}_\text{a}^i &:= \{ j \mid (\tilde{A}_i x^* + \tilde{B}_i y + \tilde{c}_i)_j = 0\}, & \mathcal{I}_\text{a}^D &:= \{ j \mid (Dy + d = 0\}, \\ \mathcal{I}_\text{i}^i &:= \{ j \mid (\lambda_i^*)_j = 0\}, & \mathcal{I}_\text{i}^D &:= \{ j \mid (\gamma^*)_j = 0\}, \\ \mathcal{I}_\text{d}^i &:= \mathcal{I}_\text{a}^i \cap \mathcal{I}_\text{i}^i, & \mathcal{I}_\text{d}^D &:= \mathcal{I}_\text{a}^D \cap \mathcal{I}_\text{i}^D, \\ \mathcal{I}_\text{sa}^i &:= \mathcal{I}_\text{a}^i \setminus \mathcal{I}_\text{d}^i, & \mathcal{I}_\text{sa}^D &:= \mathcal{I}_\text{a}^D \setminus \mathcal{I}_\text{d}^D, \\ \mathcal{I}_\text{si}^i &:= \mathcal{I}_\text{i}^i \setminus \mathcal{I}_\text{d}^i. & \mathcal{I}_\text{si}^D &:= \mathcal{I}_\text{i}^D \setminus \mathcal{I}_\text{d}^D. \end{aligned} \tag{2.48}$$

$$\begin{aligned} &(\tilde{A}_{y,i})_{sa} := (\tilde{A}_{y,i})_{\mathcal{I}_\text{sa}^i} && (\tilde{B}_{y,i})_{sa} := (\tilde{B}_{y,i})_{\mathcal{I}_\text{sa}} && (\tilde{c}_{y,i})_{sa} := (\tilde{c}_{y,i})_{\mathcal{I}_\text{sa}^i} && (\lambda_i)_{sa} := (\lambda_i)_{\mathcal{I}_\text{sa}^i} \\ &(\tilde{A}_{y,i})_{si} := (\tilde{A}_{y,i})_{\mathcal{I}_\text{si}^i} && (\tilde{B}_{y,i})_{si} := (\tilde{B}_{y,i})_{\mathcal{I}_\text{si}^i} && (\tilde{c}_{y,i})_{si} := (\tilde{c}_{y,i})_{\mathcal{I}_\text{si}^i} && (\lambda_i)_{si} := (\lambda_i)_{\mathcal{I}_\text{si}^i} \\ &(\tilde{A}_{y,i})_{d} := (\tilde{A}_{y,i})_{\mathcal{I}_\text{d}^i} && (\tilde{B}_{y,i})_{d} := (\tilde{B}_{y,i})_{\mathcal{I}_\text{d}^i} && (\tilde{c}_{y,i})_{d} := (\tilde{c}_{y,i})_{\mathcal{I}_\text{d}^i} && (\lambda_i)_{d} := (\lambda_i)_{\mathcal{I}_\text{d}^i} \\ &(D)_{sa} := (D)_{\mathcal{I}_\text{sa}^D} && && (d)_{sa} := (d)_{\mathcal{I}_\text{sa}^D} && (\gamma)_{sa} := (\gamma)_{\mathcal{I}_\text{sa}^D} \\ &(D)_{si} := (D)_{\mathcal{I}_\text{si}^D} && && (d)_{si} := (d)_{\mathcal{I}_\text{si}^D} && (\gamma)_{si} := (\gamma)_{\mathcal{I}_\text{si}^D} \\ &(D)_{d} := (D)_{\mathcal{I}_\text{d}^D} && && (d)_{d} := (d)_{\mathcal{I}_\text{d}^D} && (\gamma)_{d} := (\gamma)_{\mathcal{I}_\text{d}^D} \end{aligned} \tag{2.49}$$

Using this notation, we can introduce the following result on the local piecewise-linearity of solutions to $EPEC(z)$.

Theorem 2.3.3. *Let the EPEC (2.41) have an isolated equilibrium solution for some $z^* \in \mathbb{R}^l$, denoted by (y^*, x^*). Then there exists a piecewise linear mapping $(y, x) = K(z)$ such that (y, x) are solutions to $EPEC(z)$ for all z in some local region containing z^*.*

Proof of Theorem 2.3.3. Denote the set of all non-trivial pieces of the piecewise linear mappings relating the set of local to $EQP(y)$ in the vicinity of x^* by

$$\mathcal{K}(z^*, y^*, x^*) := \{(K, k, D, d) \mid \forall y : Dy + d \geq 0, (Ky + k) \in EQP(y)\}, \tag{2.50}$$

where the coefficients for each piece K, k, D, d are those defined in Theorem 2.2.5.

For each $(K, k, D, d) \in \mathcal{K}(z^*, y^*, x^*)$, let $\mathbf{D}EPEC^{(K,k,D,d)}(z^*, y^*, x^*)[\delta z]$ be defined as the set of equilibrium solutions to the following collection of problems:

$$\mathbf{D}QPEC_i^{(K,k,D,d)}(z^*, y^*, x^*)\begin{bmatrix}\delta z\\ \delta y_{-i}\end{bmatrix} := \underset{\delta y_i \in \mathbb{R}^{m_i}}{\arg\min} \quad \delta y_i^\intercal\left(\frac{1}{2}\tilde{Q}_{y,i}\delta y_i + \tilde{R}_{y,i}\begin{bmatrix}\delta y_{-i}\\ \delta z\end{bmatrix}\right) \tag{2.51a}$$

$$\text{subject to} \quad (\tilde{A}_{y,i})_d\delta y_i + (\tilde{B}_{y,i})_d\begin{bmatrix}\delta y_{-i}\\ \delta z\end{bmatrix} \geq 0, \tag{2.51b}$$

$$(\tilde{A}_{y,i})_{sa}\delta y_i + (\tilde{B}_{y,i})_{sa}\begin{bmatrix}\delta y_{-i}\\ \delta z\end{bmatrix} = 0, \tag{2.51c}$$

$$(D)_d\delta y \geq 0, \tag{2.51d}$$

$$(D)_{sa}\delta y = 0. \tag{2.51e}$$

$$\mathbf{D}EPEC^{(K,k,D,d)}(z^*, y^*, x^*)[\delta z] := \left\{\delta y \ : \ \begin{bmatrix}\delta y_i \in \mathbf{D}QPEC_i^{(K,k,D,d)}(z^*, y^*, x^*)[\delta z, \delta y_{-i}]\\ i \in \{1, ..., N_y\}\end{bmatrix}\right\} \tag{2.52}$$

It is easy to verify (as was done in earlier sections) that for sufficiently small ϵ, the pair $(y^* + \epsilon\delta y, K(y^* + \epsilon\delta y) + k)$ is a solution to $EPEC_i^{(K,k,D,d)}(z + \epsilon\delta z)$, for any

$$\delta y \in \mathbf{D}EPEC^{(K,k,D,d)}(z^*, y^*, x^*)[\delta z].$$

Following a similar process as in Theorem 2.2.5, this result can be extended to show that there exists a linear mapping defined by some coefficients K', k', D', d', such that

$$\forall z \ : \ D'z + d' \geq 0, \quad (K'z + k') \in EPEC^{(K,k,D,d)}(z). \tag{2.53}$$

While the results so far establish the local piecewise-linearity of the equilibrium problems $EPEC^{(K,k,D,d)}$, the directional derivative associated with each piece $(K, k, D, d) \in \mathcal{K}(z^*, y^*, x^*)$ may not agree. The effect of this is that changes in some direction δz may cause the equilibrium point in one piece to leave the boundary of its defining region, while in other pieces the equilibrium point remains at the boundary of the bordering region. the local equilibrium point of the original $EPEC$ moves as well.

Specifically, by Theorem 2.3.1, a piece of the piecewise linear mapping (K', k', D', d') for $EPEC^{(K,k,D,d)}(z)$ as in (2.53) is a piece of the piecewise linear mapping for $EPEC(z)$ if either of the following conditions hold:

- for all other $(\tilde{K}, \tilde{k}, \tilde{D}, \tilde{d}) \in \mathcal{K}(z^*, y^*, x^*)$, if there exists a piece $(\tilde{K}', \tilde{k}', \tilde{D}', \tilde{d}')$ of the piecewise-linear representation of the local solution to $EPEC^{(\tilde{K},\tilde{k},\tilde{D},\tilde{d})}$ as in (2.53), and $\tilde{D}'z + \tilde{d}' \geq 0$, then $\tilde{K}'z + \tilde{k}' = K'z + k'$.
- $D(K'z + k') + d > 0$.

The conditions above establish sufficient conditions for the existence of a linear mapping of z which constitutes solutions to $EPEC(z)$ for all z in some polyhedral region containing z^*, but dot guarantee that the union of all such polyhedral regions contains an open ball around z^*. This implies that for some changes in the parameter z, the set $EPEC$ may fail to have a solution. This is contrast to the results in section 2.2, in which the piecewise linear mapping was shown to be globally existent for all choice of parameters y. Nevertheless, whenever the above conditions do hold, they establish a local region around z^* for which the local linear representation is indeed a solution to $EPEC(z)$, establishing the result of this theorem. □

In this section, some results have been developed for solutions to equilibrium problems with equilibrium constraints. In particular, it was established that when the equilibrium constraints can be represented locally as a piecewise linear mapping, then whenever an equilibrium point to the $EPEC$ is isolated, and solutions exist for neighboring values of z, then those solutions can be represented (locally) by a piecewise linear function of z. This fact will enable the development of a generalization presented in the next section.

2.4 Equilibrium Problems with Nested Equilibrium Constraints

With the results established in previous sections, more general formulations of connected quadratic programs can be introduced. In particular, layers of nested equilibrium problems with equilibrium constraints will be considered. These organizations of programs effectively form *towers* of QPs, made up of vertically nested layers of equilibrium problems. Each layer in the vertical stack is made up of an equilibrium problem, with each quadratic program in the layer subject to a shared constraint that the nested sub-tower of QPs satisfy an equilibrium. The organization we consider is formally defined recursively in terms of parametric groups of nested equilibrium problems.

The term *parametric equilibrium problem with nested equilibrium constraints* (EPNEC) is used to define this organization, and is formalized in the recursive definitions below.

Definition 1 (QPNEC)**.** A parametric quadratic program with nested equilibrium constraints (QPNEC) at layer index l for player index i is a set-valued mapping $QPNEC_{l,i} : \mathbb{R}^{m_{l,i}} \rightrightarrows \mathbb{R}^{n_{l,i}+n_{l+1}}$ defined as:

$$QPNEC_{l,i}(y_{l,i}) := \underset{\bar{x}_{l,i} := \begin{bmatrix} x_{l,0}{}^\intercal & x_{l,i}{}^\intercal \end{bmatrix}^\intercal, \; \substack{x_{l,0} \in \mathbb{R}^{n_{l+1}} \\ x_{l,i} \in \mathbb{R}^{n_{l,i}}}}{\arg\min} \quad \bar{x}_{l,i}^\intercal (\frac{1}{2} Q_{l,i} \bar{x}_{l,i} + R_{l,i} y_{l,i} + q_{l,i}) \tag{2.54a}$$

$$\text{subject to} \quad A_{l,i}\bar{x}_{l,i} + B_{l,i} y_{l,i} + c_{l,i} \geq 0, \tag{2.54b}$$

$$EPNEC_{l+1}(\begin{bmatrix} y_{l,i}{}^\intercal & x_{l,i}{}^\intercal \end{bmatrix}^\intercal) \ni x_{l,0} \tag{2.54c}$$

Here, all variables, cost, and constraint coefficients are labeled by the tuple (l, i) to indicate that in a connected organization of quadratic programs, each $QPNEC$ is in general, different. In the case that the constraint (2.54c) (defined below) is omitted, it is assumed that the dimension of $x_{l,0}$ is 0, and therefore the $QPNEC$ at level l for player i is equivalent to a QP.

Definition 2 (EPNEC)**.** A parametric equilibrium problem with nested equilibrium constraints at level l is a set-valued mapping $EPNEC_l : \mathbb{R}^{m_l} \rightrightarrows \mathbb{R}^{n_l}$ defined as:

$$EPNEC_l(y_l) := \left\{ \begin{bmatrix} x_{l,0} \\ x_{l,1} \\ \vdots \\ x_{l,N_l} \end{bmatrix} \in \mathbb{R}^{n_l} \;\middle|\; \begin{bmatrix} x_{l,0} \\ x_{l,i} \end{bmatrix} \in QPNEC_{l,i}\left(\begin{bmatrix} y_l \\ x_{l,-i} \end{bmatrix} \right), \forall i \in \{1, ..., N_l\} \right\} \tag{2.55}$$

Above, N_l is the number of players at level l of the organization, and $x_{l,-i}$ is defined as:

$$x_{l,-i} := \begin{bmatrix} x_{l,1}{}^\intercal & \dots & x_{l,i-1}{}^\intercal & x_{l,i+1}{}^\intercal & \dots & x_{l,N_l}{}^\intercal \end{bmatrix} \tag{2.56}$$

Similarly let

$$x_{1:N_l} := \begin{bmatrix} x_{l,1}{}^\intercal & \dots & x_{l,N_l}{}^\intercal \end{bmatrix}^\intercal . \tag{2.57}$$

Notice that the parameter $y_{l,i}$ being passed to $QPNEC_{l,i}$ is implicitly defined as $y_{l,i} := \begin{bmatrix} y_l{}^\intercal & x_{l,-i}{}^\intercal \end{bmatrix}^\intercal$. Then, the stacking of variables $\begin{bmatrix} y_{l,i}{}^\intercal & x_{l,i}{}^\intercal \end{bmatrix}^\intercal$ appearing in constraint (2.54c) (via a slight abuse of notation) should be interpreted as $\begin{bmatrix} y_l{}^\intercal & x_{1:N_l}{}^\intercal \end{bmatrix}^\intercal$, which by the definition of $ECQP_{l+1}$ is equivalent to y_{l+1}.

Therefore, the dimension of y_{l+1} is given by $m_{l+1} = m_l + \sum_{i=1}^{N_l} n_{l,i}$, and hence m_l is the dimension of all variables appearing in levels strictly before l. Conversely, it is seen that $n_l = n_{l+1} + \sum_{i=1}^{N_l} n_{l,i}$, and is the dimension of all variables appearing in levels at or after l.

l	Input to $EPNEC_l$
1	$y_1 = []$
2	$y_2 = \begin{bmatrix} x_{1,1} & x_{1,2} \end{bmatrix}$
3	$y_3 = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{2,1} & x_{2,2} & x_{2,3} \end{bmatrix}$
4	$y_4 = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{2,1} & x_{2,2} & x_{2,3} & x_{3,1} \end{bmatrix}$
5	$y_5 = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{2,1} & x_{2,2} & x_{2,3} & x_{3,1} & x_{4,1} & x_{4,2} & x_{4,3} \end{bmatrix}$
l	Output of $EPNEC_l$
1	$x_1 = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{2,1} & x_{2,2} & x_{2,3} & x_{3,1} & x_{4,1} & x_{4,2} & x_{4,3} \end{bmatrix}$
2	$x_2 = \begin{bmatrix} x_{2,1} & x_{2,2} & x_{2,3} & x_{3,1} & x_{4,1} & x_{4,2} & x_{4,3} \end{bmatrix}$
3	$x_3 = \begin{bmatrix} x_{3,1} & x_{4,1} & x_{4,2} & x_{4,3} \end{bmatrix}$
4	$x_4 = \begin{bmatrix} x_{4,1} & x_{4,2} & x_{4,3} \end{bmatrix}$
5	$x_5 = []$

Table 2.1: Variable scopes for the $EPNECs$ appearing in Fig. 2.1.

The variable $x_{l,0}$ appearing in $EPNEC_l$ has dimension n_{l+1}, and is a shared variable among all N_l players at layer l. It represents all variables being determined at level $l+1$ and after. This variable is constrained by the shared constraint $x_{l,0} \in EPNEC_{l+1}(y_{l+1})$, appearing in $QPNEC_{l,i}$ for all $i \in \{1, ..., N_l\}$. It is through these variables that the nesting of the connected quadratic programs is made explicit. This shared constraint (2.54c) enables the optimization for player $i \in \{1, ..., N_l\}$ at layer l to reason about the reaction of players in successive layers to changes in player i's decision variables.

In Fig. 2.1, an example organization of connected quadratic programs is outlined. In this example there are 4 nested layers of $EPNECs$, with an empty $EPNEC$ at the $5th$ level for completeness. Each of the four $EPNECs$ considered are comprised of a group of between one and three $QPNECs$. Variable scopes for the various $EPNECs$ and $QPNECs$ in the organization are listed in Tables 2.1 and 2.2.

Note that the equilibrium problems with equilibrium constraints considered in section 2.3 are simply $EPNECs$ with two layers. The formulation here generalizes those problems to arbitrary number of layers.

The form of connected quadratic programs considered here, which can be described by

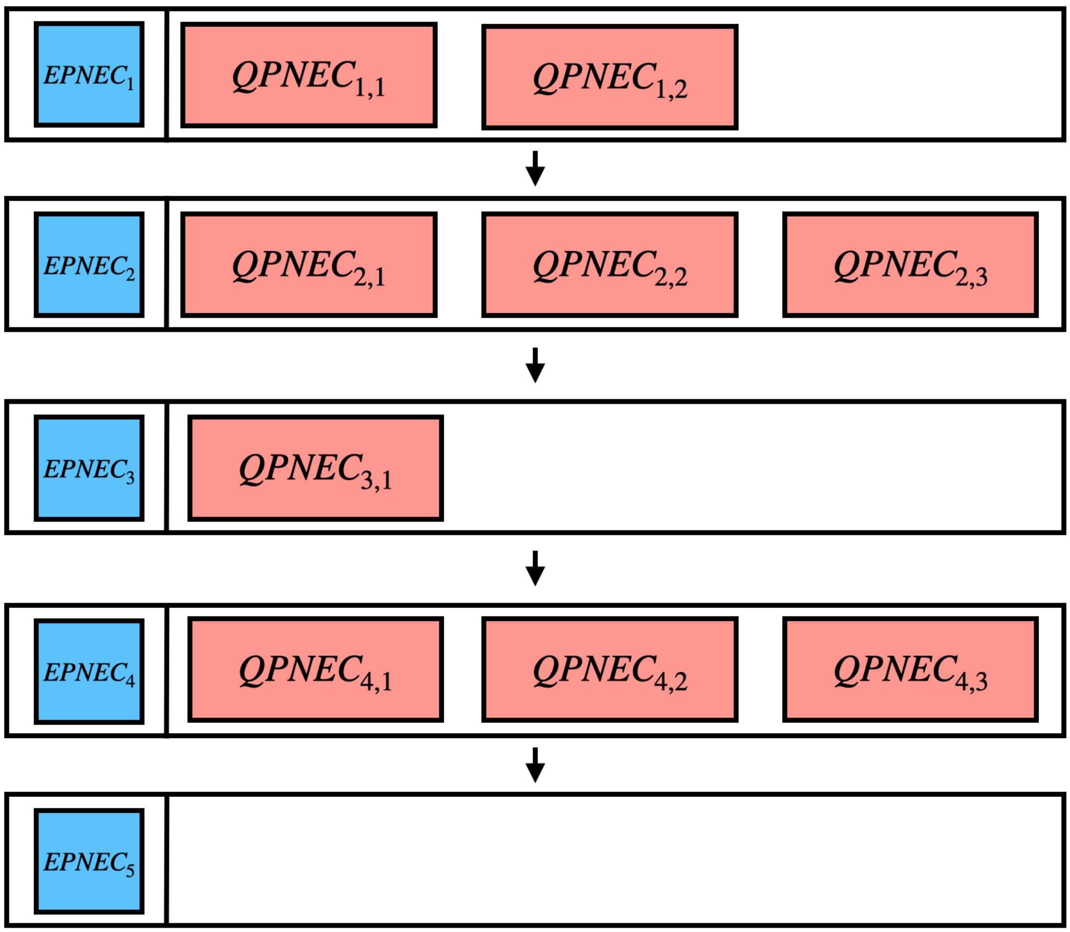


Figure 2.1: An example organization of a connected quadratic program that can be described in the $EPNEC$ framework. Decision variable scopes for the problems at each layer are given in Tables 2.1 and 2.2.

l	i	Input to $QPNEC_{l,i}$	Output of $QPNEC_{l,i}$
1	1	$y_{1,1} = \begin{bmatrix} x_{1,2} \end{bmatrix}$	$\bar{x}_{1,1} = \begin{bmatrix} x_{1,1} & x_2 \end{bmatrix}$
1	2	$y_{1,2} = \begin{bmatrix} x_{1,1} \end{bmatrix}$	$\bar{x}_{2,1} = \begin{bmatrix} x_{2,1} & x_2 \end{bmatrix}$
2	1	$y_{2,1} = \begin{bmatrix} x_1 & x_{2,2} & x_{2,3} \end{bmatrix}$	$\bar{x}_{2,1} = \begin{bmatrix} x_{2,1} & x_3 \end{bmatrix}$
2	2	$y_{2,2} = \begin{bmatrix} x_1 & x_{2,1} & x_{2,3} \end{bmatrix}$	$\bar{x}_{2,2} = \begin{bmatrix} x_{2,2} & x_3 \end{bmatrix}$
2	3	$y_{2,3} = \begin{bmatrix} x_1 & x_{2,1} & x_{2,2} \end{bmatrix}$	$\bar{x}_{2,3} = \begin{bmatrix} x_{2,3} & x_3 \end{bmatrix}$
$\vdots$	$\vdots$		

Table 2.2: Variable scopes for some of the $QPNECs$ appearing in example in Fig. 2.1.

$EPNECs$, constitute a very general class of problems. It will be shown in later sections that a variety of important and interesting real-world problems can be cast in this framework. Hence, this framework is sufficiently general to encapsulate a broad swath of problems, but is structured enough to be analyzed and enable the development of general solution techniques. This is particularly important, since instead of designing customized solution methods which are specialized for the particular organization of quadratic programs that arise in any given problem, the same method designed for general $EPNECs$ can be applied to that problem. The study of such a computational method is the focus of the next chapter, which will make use of the following theorem.

Theorem 2.4.1. *Consider an $EPNEC_l$, at some layer $1 \leq l \leq L$ in a stack of nested $EPNECs$. Then if solutions to $x_{l,0} \in EPNEC_{l+1}(y_{l+1})$ can be locally represented as a piecewise linear mapping, and $x^*_{l-1,0}$ is an isolated solution to $EPNEC_l(y^*_l)$. Then there exists a piecewise linear representation of local solutions to $EPNEC_l(y_l)$ for all y_l in a polyhedral region containing y^*_l.*

Proof. This proof follows from direct application of Theorem 2.3.3, since if the problem $EPNEC_{l+1}(y_{l+1})$ admits a piecewise linear representation, the problem $EPNEC_l(y_l)$ is indistinguishable from the problem (2.41). $\square$

The result of Theorem 2.4.1 implies an inductive relationship between the piecewise linearity of solutions at various layers in an $EPNEC$. Assuming an isolated solution exists to the $EPNEC$ at some layer $l + 1$, the piecewise linear representation of solutions can be used to search for an isolated equilibrium point to the $EPNEC$ at layer l. Assuming this can be found, and is an isolated point, a piecewise linear representation of local solutions can be generated for this problem, which can in turn be used to search for solutions to the $EPNEC$

at layer $l-1$, and so on. This core idea will be the foundation for the computational method developed in the next chapter for computing solutions to these collections of connected quadratic programs.

Chapter 3

General Computation

In this chapter, a method for computing solutions to the L-level $EPNECs$ defined in section 2.4 is described, which builds on Theorem 2.4.1. We first present the proposed algorithm in whole, and then discuss various properties of it in the section to follow. In later chapters, example problems are described and some solutions of those problems are reported.

The basic premise of the presented method is the following. An initial iterate is found which is simultaneously feasible for the constraints (2.54b) at every $l \in \{1, ..., L\}$, $i \in \{1, ..., N_l\}$. This can be done by solving a simple feasibility problem. If there is no such point satisfying all of these constraints, then of course no equilibrium solution can exist.

Once the initial iterate is found, the decision variables y_L are held fixed, while a solution is found to $EPNEC_L(y_L)$. This equilibrium problem at level L is equivalent to the equilibrium formulation described in section 2.2. Once a solution to the problem at level L is found, a piecewise linear representation is formed for the local solutions to that level L problem, which is then used to replace the equilibrium constraints appearing in the equilibrium problem at level $L-1$.

The decision variables x_{L-1} are now released (such that only y_{L-1} is fixed), and the equilibrium problem at level $L-1$ is solved using the piecewise linear representation of the sub-level equilibrium constraints. Inspired by the results of Theorem 2.3.1, a method for traversing pieces of the piecewise linear representation is used to find the solution of the level $L-1$ equilibrium problem. Once a solution is found, a piecewise linear representation of the local solution set is found, and this representation is used to replace the equilibrium constraints appearing in the equilibrium problem $EPNEC_{L-2}$.

This process is repeated until, ideally, a solution is found for $EPNEC_1$, implying an equilibrium solution for the entire set of connected quadratic programs has been found. If at any point, a non-isolated solution is found to the $EPNEC_l$ for some level $2 \leq l \leq L$, then the algorithm terminates with failure, as a suitable representation for the set of local solutions is unable to be found. Furthermore, if an equilibrium is failed to be found at any level l, then failure is also returned.

3.1 Equilibrium Problems with Piecewise Linear Constraints

Since much of the method outlined above relies on the solution of equilibrium problems with shared piecewise linear constraints, a specialized subroutine is developed for that problem. Given the development of that subroutine, the method for computing L-level $EPNECs$ can be established.

Before stating the subroutine for solving equilibrium problems with shared piecewise linear equality constraints, the problem formulation is stated with specificity.

Let $\mathbf{K} : \mathbb{R}^m \to \mathbb{R}^n$ be some piecewise linear mapping. Specifically, let $\mathbf{K}$ be defined by N_r regions, each region $i \in \{1, ..., N_r\}$ defined as

$$\mathcal{R}_i := \{y \in \mathbb{R}^m : D_i y + d_i \geq 0\}. \tag{3.1}$$

Let any two regions only overlap on their borders. This is indicated by stating that the interior of the intersection of any two regions $\mathcal{R}_i$ and $\mathcal{R}_j$ for any $i \neq j$ be empty:

$$\forall\ (i, j) \in \{1, ..., N_r\},\ i \neq j\ :\ (\mathcal{R}_i \cap \mathcal{R}_j)^{\mathrm{o}} = \emptyset \tag{3.2}$$

Assume further that the regions are all connected, i.e. there exists some simply-connected region $\mathcal{D} \subset \mathbb{R}^m$, and that

$$\left(\cup_{i=1}^{N_r} \mathcal{R}_i\right) = \mathcal{D}. \tag{3.3}$$

Finally the function $\mathbf{K}$ can be defined:

$$\mathbf{K}(y) := \Big\{ K_i y + k_i \text{ if } y_i \in \mathcal{R}_i \tag{3.4}$$

Notice that the definition of $\mathbf{K}$ implies that the mapping is continuous over the domain $\mathcal{D}$.

With this definition of the form of piecewise-linear mappings we are concerned with, the equilibrium problem of interest can be defined. We denote this an equilibrium problem with piecewise linear constraints, or $EPPWLC$. This is an equilibrium problem among multiple quadratic programs with piecewise linear constraints, or $QPPWLC$s.

$$QPPWLC_i(z, y_{-i}) := \underset{y_i \in \mathbb{R}^{m_i}, x \in \mathbb{R}^n}{\arg\min} \quad \begin{bmatrix} y_i \\ x \end{bmatrix}^{\intercal} \left(\frac{1}{2} Q_{y,i} \begin{bmatrix} y_i \\ x \end{bmatrix} + R_{y,i} \begin{bmatrix} y_{-i} \\ z \end{bmatrix} + q_{y,i}\right) \tag{3.5a}$$

$$\text{subject to} \quad A_{y,i} \begin{bmatrix} y_i \\ x \end{bmatrix} + B_{y,i} \begin{bmatrix} y_{-i} \\ z \end{bmatrix} + c_{y,i} \geq 0, \tag{3.5b}$$

$$x - \mathbf{K}(y) = 0. \tag{3.5c}$$

$$EPPWLC(z) := \left\{ x^* \in \mathbb{R}^n, y^* \in \mathbb{R}^m | (y_i^*, x^*) \in QPPWLC_i(z, y_{-i}^*), \forall i \in \{1, ..., N_y\} \right\} \tag{3.6}$$

Algorithm 1 Equilibrium Problems with Shared Piecewise Linear Equality Constraints

---

Require:

- Piecewise linear function $\mathbf{K}$, defined in (3.4)
- Parameter value z
- Collection of N_y $QPPWLC$s, defined in (3.5)
- Positive semi-definite coefficients $Q_{y,i}$, for all $i \in \{1, ..., N_y\}$
- Positive definite submatrix $(Q_{y,i})_{1:m_i,1:m_i}$, for all $i \in \{1, ..., N_y\}$

for $i = 1..N_r$ **do**
  $(x, y) \leftarrow EPEC^{(K_i,k_i,D_i,d_i)}(z)$ (Using LCP Solve)
  if failure **then**
    continue
  end if
  if $D_i y + d_i > 0$ **then return** (x, y)
  else
    for $j = 1..N_r$ **do**
      if $D_j y + d_j \geq 0$ and $(x, y) \notin EPEC^{(K_j,k_j,D_j,d_j)}(z)$ (Using LCP check) **then**
        break
      end if
    end for
    return (x, y)
  end if
end for
return $\emptyset$

The first algorithm presented for solving $EPPWLC(z)$ is a simple enumeration-based approach. This approach is conceptually simple, but has some drawbacks which are addressed in algorithm 3.

The premise of Algorithm 1 is to simply enumerate the solutions to the equilibrium problems formed when restricted to each piece of the piecewise linear mapping $\mathbf{K}$, and check to see if they satisfy the requirements to be equilibrium points for (3.6) as stated in Theorem 2.3.1.

The method for solving the $EPEC$ associated with each piece is to use a method for solving linear complementarity problems, such as Lemke's method [48]. In order to convert the $EPEC$ into the form of a linear complementarity problem, first the equality constraints $x = K_i y + k_i$ can be used to eliminate the variable x from the programs as was done in Theorem 2.3.2. At this point the programs are converted to bound-constrained problems via the technique introduced in section 2.1 for converting general QPs into bound-constrained QPs. Finally, expressing the joint first-order necessary conditions for these resultant problems (analogous to (2.47)) result in a LCP which can be solved using known methods.

In Algorithm 1, it is also assumed that the solution to some $EPEC^{K_i,k_i,D_i,d_i}$ may not exist or can not be found. The method is robust to such situations, in that it can still attempt to find an equilibrium point for the problem at large.

The obvious drawback of using Algorithm 1 to solve $EPPWLC$ problems, is that it requires being able to enumerate every piece to the piecewise linear mapping $\mathbf{K}$, not to mention then solving the N_r corresponding LCPs. As we will see, in many cases the pieces comprising $\mathbf{K}$ are not known in advance, but rather are generated on demand. Specifically, when y leaves a region $\mathcal{R}_i$, a new piece $x = K_j y + k_j$ is formed, and the region $\mathcal{R}_j$ for which the piece is a valid representation of $\mathbf{K}$ is computed.

These difficulties inspire an alternative approach to solving $EPPWLC$ problems, which is outlined in Algorithm 2. The basic idea of this approach is instead to search locally for equilibrium solutions, transitioning from piece to neighboring piece as needed. The challenges in such an approach are that if a particular piece does not warrant an equilibrium solution, establishing which neighboring region to transition to requires care. Furthermore, it can be difficult to find an initial feasible solution which satisfies both the inequality constraints appearing in each QP, as well as the piecewise linear equality constraint. For this reason, Algorithm 2 must have a higher rate of failure than Algorithm 1, yet nevertheless the loss in robustness made in exchange for practicality is often worth it.

There are three subroutines introduced in Algorithm 2 that have not yet been discussed. These are the routines FindFeasible, LCPCheck and EnumerateNeighbors. The method FindFeasible is intended to locate an initial piece of the piecewise linear mapping $\mathbf{K}$ for which there is a joint-feasible solution, meaning it simultaneously satisfies the conditions (3.5b, 3.5c) for all $i \in \{1, ..., N_y\}$. In general, this is a non-convex problem and can warrant difficulties depending on how the piecewise linear mapping $\mathbf{K}$ is represented. Presumably any arbitrary piece of the mapping could be used to initialize the search for an equilibrium point, but then it is unclear how to proceed when no solution exists to the problem $EPEC^{(K,k,D,d)}$. It may even be that after finding a jointly-feasible solution, that an equilibrium still does not exist. Nevertheless, initializing in a piece in which at least the feasible search region is non-empty gives a strong guess which yields good performance empirically.

The method proposed for finding a feasible solution is given in Algorithm 3. The core idea is to move along the piecewise linear mapping until a point satisfying the constraints (3.5b) is found.

Now both Algorithm 2 and Algorithm 3 rely on the method EnumerateNeighbors as part of their computation. The concept for finding neighbors of a piece is based on determining which borders of the region $Dy+d$ the point lies on, i.e. for which indices $(Dy+d)_i = 0$, and evaluating the form of the piece(s) sharing those boundaries. This process will in general depend on the representation of the mapping $\mathbf{K}$. When the piecewise linear mapping is representing the relationship between local solutions to an equilibrium problem and the input parameter, this will correspond to choosing a different representation of the active constraints as outlined in Theorem 2.2.5.

The remaining subroutine appearing in Algorithm 2 (and in Algorithm 1) is the method for verifying whether some point (x, y) is a solution to an equilibrium problem $EPEC^{(K,k,D,d)}(z)$.

Algorithm 2 Equilibrium Problems with Shared Piecewise Linear Equality Constraints

Require:

- Piecewise linear function $\mathbf{K}$, defined in (3.4)
- Parameter value z
- Collection of N_y $QPPWLC$s, defined in (3.5)
- Positive semi-definite coefficients $Q_{y,i}$, for all $i \in \{1, ..., N_y\}$
- Positive definite submatrix $(Q_{y,i})_{1:m_i,1:m_i}$, for all $i \in \{1, ..., N_y\}$
- Initialization point $(\hat{x}, \hat{y})$ and piece of mapping $\hat{K}, \hat{k}, \hat{D}, \hat{d}$ such that $\hat{D}\hat{y} + \hat{d} \geq 0$.
- `max_iters` (maximum number of pieces to search before declaring failure)

```
(success, K^0, k^0, D^0, d^0) = FindFeasible(x̂, ŷ, K̂, k̂, D̂, d̂)
if not success then return ∅
end if
j ← 0
while j < max_iters do
    (success, x^j, y^j) ← EPEC^(K^j,k^j,D^j,d^j)(z) (using LCP solve)
    if success then
        if D^j y^j + d^j > 0 then
            return (x^j, y^j)
        else
            N ← EnumerateNeighbors(x^j, y^j, K^j, k^j, D^j, d^j)
            for (K', k', D', d') ∈ N do
                if (x^j, y^j) ∉ EPEC^(K',k',D',d')(z) (using LCP check) then
                    (K^{j+1}, k^{j+1}, D^{j+1}, d^{j+1}) ← (K', k', D', d')
                    break
                end if
            end for
            return (x^j, y^j)
        end if
    else
        success, K^{j+1}, k^{j+1}, D^{j+1}, d^{j+1} ← FindFeasible()
        if not success then return ∅
        end if
    end if
    j ← j + 1
end while
return ∅
```

Algorithm 3 FindFeasible

Require:

- Piecewise linear function $\mathbf{K}$, defined in (3.4)
- Constraints: $Ax + By + c \geq 0$
- (Optional) Initial point (x, y)
- (Optional) Initial piece (K, k, D, d)

```
if (x, y) not provided then
    if (K, k, D, d) not provided then
        Choose (x, y) satisfying Ax + By + c ≥ 0
        if (y ∉ Domain(K)) then
            return failure,null,null,null,null
        end if
        (K, k, D, d) ← FindLocalPiece(K,y)
    else
        Choose (x, y) satisfying Dy + d ≥ 0, x = Ky + k
    end if
else
    if (K, k, D, d) not provided then
        if (y ∉ Domain(K)) then
            return failure,null,null,null,null
        end if
    else
        Assert that Dy + d ≥ 0, x = Ky + k
    end if
end if
while True do
    Solve min_{x^1,y^1,x^1,y^1} || [x^1 − x^2 ; y^1 − y^2] ||_2 s.t. {Dy^1 + d ≥ 0, x^1 = Ky^1 + k, Ax^2 + By^2 + c ≥ 0}.
    (x, y) ← (x^1, y^1)
    if x^1 == x^2 and y^1 == y^2 then return success,K, k, D, d
    else
        N =EnumerateNeighbors(x, y, K, k, D, d)
        if N == ∅ then return failure,null,null,null,null
        end if
        (K, k, D, d) ← ChooseFrom(N)
    end if
end while
```

where the solve step is

$$\min_{x^1,y^1,x^1,y^1} \left\| \begin{matrix} x^1 - x^2 \\ y^1 - y^2 \end{matrix} \right\|_2 \text{ s.t. } \{Dy^1 + d \geq 0,\ x^1 = Ky^1 + k,\ Ax^2 + By^2 + c \geq 0\}.$$

One way to attempt this is to find a solution to $EPEC^{(K,k,D,d)}(z)$ and check if that point matches (x, y). However, when multiple solutions exist, any given method for computing solutions may return a different solution than (x, y). In such a case, it cannot be said whether or not (x, y) is indeed a solution to $EPEC^{(K,k,D,d)}(z)$. Therefore we propose a simplified technique for checking whether (x, y) is indeed a solution to $EPEC^{(K,k,D,d)}(z)$, which involves fixing (x, y), and attempting to find multipliers which satisfy the conditions (2.47). When the set of active constraints are linearly independent, this is as simple as solving a linear system of equations to solve for the constraint multipliers, and determining whether or not they are non-negative.

3.2 Equilibrium Problems with Nested Equilibrium Constraints

Given the methods so far described, we are ready to present the method for computing solutions to equilibrium problems with nested equilibrium constraints.

Algorithm 4 Equilibrium Problems with Nested Equilibrium Constraints

Require:

- L-level $EPNEC$, defined by the equilibrium layers (2.55) and the $QPNECs$ (2.54). item Positive semi-definite coefficients $Q_{l,i}$, for all $l \in \{1, .., L\}$, $i \in \{1, ..., N_l\}$
- Positive definite submatrix $(Q_{l,i})_{n_{l,i}:,n_{l,i}:}$, for all $l \in \{1, .., L\}$, $i \in \{1, ..., N_l\}$
- Initialization x_1 (x_1 is the output variables of the 1st layer of the $EPNEC$, and therefore comprises all decision variables in the $EPNEC$).

Decompose $x_1 := \begin{bmatrix} y_L \\ x_L \end{bmatrix}$
$x_L \leftarrow EPNEC_L(y_L)$ using LCP solve (as in section 2.2)
if failure, or non-isolated solution **then**
  return failure
end if
for $l = (L-1)..1$ **do**
  $\mathbf{K}_{l+1} \leftarrow$piecewise linear representation of $x_{l+1} = EPNEC_{l+1}(y_{l+1})$
  $EPPWLC_l \leftarrow$ representation of $EPNEC_l(y_l)$ using $\mathbf{K}_{l+1}$
  $x_l \leftarrow EPPWLC_l(y_l)$ using Algorithm 2.
  if no solution, or non-isolated solution **then return** failure
  end if
end for
return x_1

Algorithm 4 follows the basic idea put forth in the beginning of this section. For the very reasons outlined in chapter 2, it is very difficult to guarantee the success of this method.

Nevertheless, by leveraging the local piecewise linearity of each equilibrium sub-problem, equilibrium points for large-scale $EPNECs$ can often be found using this approach.

It is important to point out a possible alternate approach for solving $EPNEC$ problems. Presumably, necessary conditions for equilibrium solutions could be inscribed in terms of primal and dual variables for the entire problem. This would involve starting at layer L, and representing the solution to the equilibrium problem at that level as a complementarity problem. This formulation involves not only the primal variables at layer L, but also the dual variables. Then, solutions to the equilibrium problem at layer $L-1$ are formed, now in terms of necessary conditions of equilibrity for the resultant equilibrium problem with equilibrium constraints. Now, additional dual variables are introduced to handle the constrained values of the sub-level dual variables. Presuming such conditions can be succinctly represented, they could be passed in as a constraint into the equilibrium problem at layer $L-2$, and so forth. It is seen that the total number of decision variables quickly grows very large with the number of layers L. Furthermore, the form of the constraints becomes increasingly complicated to represent. It is not hard to see why such an approach is not practical.

So, despite lacking guarantees of the ability to find a solution if one exists, the method presented in Algorithm 4 is still very appealing for its ability to find solutions to previously unsolvable problems. In the sections to follow, various interesting problems are cast as $EPNECs$, and some results of applying Algorithm 4 to those problems are presented.

Chapter 4

Feedback Nash Equilibrium Problems

In this chapter, the concept of a generalized Feedback Nash equilibrium (GFNE) is introduced, as it serves as an interesting example of an $EPNEC$, and in many ways was my inspiration for studying connected optimization problems in general. The concept of a GFNE will be formalized, and then some techniques for implementing the approach in Algorithm 4 for this class of problem will be discussed. Finally some examples of solutions will be presented. Much of this chapter is taken from [151], which is co-authored with David Fridovich-Keil, Chih-Yuan Chiu, and Claire Tomlin.

4.1 Introduction: Dynamic Games

As discussed in earlier chapters, connected optimization problems, and $EPNEC$s in particular, arise frequently in the context of game-theoretic problems. Recently there has been a growing interest in the application of game-theoretic concepts to applications in automated systems, as explored in [150, 263, 52, 47, 89, 86, 51, 78]. Indeed, numerous problems arising in these domains can be modeled as games, and particularly as dynamic or repeated games. Particularly in the context of discrete-time dynamic games, in which players have the ability to influence the state of the game over a finite set of game *stages*, these games can be formulated in the $EPNEC$ framework. Associated with dynamic/repeated game solutions are game trajectories, which capture the evolution of the *continuous* game state and player inputs over the sequence of stages.

Solutions to dynamic games have been studied extensively, as in [19]. Such studies are largely complementary to the study of extended mathematical programming, such as the study of general equilibrium problems or nested optimization problems. Within the theory of dynamic games, the concept of an information pattern is used to represent the information each player has access to at the point of making decision. Each of these patterns results in a fundamentally different solution. The perspective taken here is that information patterns correspond to connection organizations in the context of connected optimization problems.

Perhaps the simplest information pattern is the *open-loop* pattern, in which the repeated nature of the game is ignored. In this pattern the stages of the game are combined into a single *static* game, and the entire trajectory is chosen at once to satisfy a Nash, Stackelberg, or other type of equilibrium. When constraints on the state variables are imposed upon the players, a generalized equilibrium must be considered, meaning the constraints imposed on each player depend on the decision variables of other players. Formulating games with an open-loop information pattern has many advantages, as the resultant static game often then admits known methods for analysis and computing solutions, as presented, for example, in [70, 71]. However, by ignoring the dynamic nature of these games, the expressiveness of the resultant solutions are significantly limited. Intelligent game play in repeated games often involves observing the evolving game state and reacting accordingly.

Reactive game-play can emerge when associating a *closed-loop* information pattern to the game of interest. Effectively, games with this information pattern are such that the players choose control *policies* which define the control input as a function of the game state at that stage. When those policies at each game stage are chosen to constitute an equilibrium for the dynamic subgame played over the subsequent game stages, the resultant solution is called a feedback equilibrium. This type of solution is capable of capturing strategies for a player which anticipate and account for the reaction of other players. Some advantages of this type of solution are explored in the autonomous driving context in, e.g., [150].

While feedback equilibrium solutions are often desirable over their open-loop counterparts, for all but simple cases, there do not exist well-developed numerical routines for computing them. The unconstrained Linear-Quadratic (LQ) setting is perhaps the simplest case, and methods for computing feedback equilibria for these games are well known, as presented in [19]. The extension to the computation of a feedback Nash equilibrium for a class of inequality-constrained LQ games is introduced in [212, 213], although restrictive assumptions are made on the form of the dynamics, constraints and cost terms of the game. Numerous other approaches have considered the computation of feedback equilibria under various special cases, such as those in [248, 249, 138], among others. Methods for computing feedback Nash equilibria have been recently developed in the unconstrained, nonlinear case using a value-iteration based approach [109], and an iterative LQ game approach [89]. Nevertheless, to the best of our knowledge, no methods exist for computing feedback equilibria in games with constraints appearing on both the state and input dimensions, both in the general LQ and nonlinear settings.

Since many emerging applications of dynamic games involve inequality constraints on the game states and inputs, we have pursued the development of a robust and efficient method for computing feedback equilibria in this setting. The result of that work is the topic of this chapter. From the view that information patterns are associated with connection patterns of optimization problems, a closed-loop information pattern can be cast as a $EPNEC$. Therefore, the analysis and methods introduced in preceding chapters are used to address to this type of problem.

The outline of the chapter is the following. In Section 4.2 we introduce the concept of a Generalized Feedback Nash Equilibrium (GFNE), which formally defines the feedback concept in the constrained setting. This formulation does not yet make any assumptions on the objective or constraint forms presented to each player. We discuss pitfalls with a parameterized approach to encoding GFNE problems as a means to motivate and introduce a non-parametric alternative. We show that a GFNE can be cast in the framework of an $EPNEC$, with some special properties. We then develop necessary and sufficient conditions on game trajectories to satisfy a GFNE using this non-parametric formulation. Challenges associated with the computation of arbitrary GFNE are highlighted, and a close approximation is introduced which is amenable to efficient computation. Finally, numerical methods for the computation of such approximate solutions are developed in detail, for the equality-constrained LQ setting (Section 4.3), inequality-constrained LQ setting (Section 4.4), and ultimately, the general nonlinear setting (Section 4.5). We demonstrate our method on an application to autonomous driving in Section 4.6 and conclude the chapter in Section 4.7.

4.2 Formulation

We focus our attention to the class of N-player discrete-time, deterministic, infinite, general-sum dynamic games of discrete stage-length T. Let $\mathbf{N}$ denote the set $\{1, ..., N\}$, and similarly $\mathbf{T}$ the set $\{1, ..., T\}$. We also make use of the sets $\mathbf{T}^+ := \mathbf{T} \cup \{T+1\}$, $\mathbf{T_t} := \{t, ..., T\}$, and $\mathbf{T_t^+} := \{t, ..., T+1\}$. The game state at each discrete time-step t is represented by $x_t \in \mathcal{X} = \mathbb{R}^n$. The game is assumed to start at stage $t = 1$, from a pre-specified initial state $\hat{x}_1$. Throughout this paper we refer to subgames starting at stage t, which refers to the game played over a portion of the original game, on the stages $\{t, ..., T\}$.

The evolution of the game state is described by the dynamic equation:

$$x_{t+1} = f_t(x_t, u_t^1, ..., u_t^N), \quad t \in \mathbf{T}, \tag{4.1}$$

where $u_t^i \in \mathcal{U}_t^i = \mathbb{R}^{m_t^i}$ are the control variables chosen by players $i \in \mathbf{N}$ at time t. Let $m_t := \sum_{i=1}^N m_t^i$, and $m_t^{-i} := m_t - m_t^i$.

To simplify the notation in definitions and derivations, we make use of the following shorthand to refer to various sets of state and control variables:

Notation reference:

$$\begin{aligned}
x &:= (x_1, x_2, ..., x_{T+1}), \\
u^i &:= (u_1^i, u_2^i, ..., u_T^i), \\
u &:= (u^1, ..., u^N), \\
u_t &:= (u_t^1, ..., u_t^N), \\
u_t^{-i} &:= (u_t^1, ..., u_t^{i-1}, u_t^{i+1}, ..., u_t^N), \\
(u_t^i, u_t^{-i}) &:= (u_t^1, ..., u_t^N) = u_t
\end{aligned}$$

Each player in the game is associated with time-separable cost-functionals:

$$L^i(x, u^1, ..., u^N) := \sum_{t=1}^{T} l_t^i(x_t, u_t) + l_{T+1}^i(x_{T+1}) \tag{4.2}$$

Furthermore, each player is assigned stage-wise, non-dynamic, equality and inequality constraints.

$$0 = h_t^i(x_t, u_t),\ t \in \mathbf{T} \qquad\qquad 0 = h_{T+1}^i(x_{T+1}) \tag{4.3a}$$
$$0 \leq g_t^i(x_t, u_t)\ t \in \mathbf{T}, \qquad\qquad 0 \leq g_{T+1}^i(x_{T+1}) \tag{4.3b}$$

Let the dimension of the constraints h_t^i and g_t^i, for all $t \in \mathbf{T}^+$ and $i \in \mathbf{N}$, be denoted as $a_t^i \geq 0$ and $b_t^i \geq 0$, respectively. Define $V_t^i : \mathcal{X} \to (\mathbb{R} \cup \infty)$ as the *Value-function* for player $i \in \mathbf{N}$ at stage $t \in \mathbf{T}^+$, and $Z_t^i : \mathcal{X} \times \mathcal{U}_t^i \times ... \times \mathcal{U}_t^N \to (\mathbb{R} \cup \infty)$ the *Control-Value-function* for player $i \in \mathbf{N}$ at time $t \in \mathbf{T}$.

A Generalized Feedback Nash Equilibrium is defined in terms of measurable maps $\pi_t^i : \mathcal{X} \to \mathcal{U}_t^i$, for $i \in \mathbf{N}$, $t \in \mathbf{T}$, which we refer to as feedback policies or strategies. The feedback policies, Value-functions, and Control-Value-functions are together defined according to the following recursive relationships Eq. (4.4)-Eq. (4.7):

$$V_{T+1}^i(x_{T+1}) := \begin{cases} l_{T+1}^i(x_{T+1}), & \begin{array}{l} 0 = h_{T+1}^i(x_{T+1}) \\ 0 \leq g_{T+1}^i(x_{T+1}) \end{array} \\ \infty, & \text{else} \end{cases} \tag{4.4}$$

Given V_{t+1}^i for some $t \in \mathbf{T}$ and $i \in \mathbf{N}$, we define Z_t^i by

$$Z_t^i(x_t, u_t^1, ..., u_t^N) := \begin{cases} l_t^i(x_t, u_t) + V_{t+1}^i(f_t(x_t, u_t)), & \begin{array}{l} 0 = h_t^i(x_t, u_t) \\ 0 \leq g_t^i(x_t, u_t) \end{array} \\ \infty, & \text{else} \end{cases} \tag{4.5}$$

For a particular state x_t at stage t, the feedback policies π_t are defined to return a local Nash equilibrium solution for the static game defined in terms of the N Control-Value-functions evaluated at x_t (one for each player).

$$\begin{aligned} &\tilde{u}_t = \pi_t(x_t) \implies \\ &\qquad Z_t^1(x_t, \tilde{u}_t^1, ..., \tilde{u}_t^N) \leq Z_t^1(x_t, u_t^1, \tilde{u}_t^2, ..., \tilde{u}_t^N), \qquad \forall u_t^1 \in \mathcal{N}(\tilde{u}_t^1), \\ &\qquad \vdots \\ &\qquad Z_t^N(x_t, \tilde{u}_t^1, ..., \tilde{u}_t^N) \leq Z_t^N(x_t, \tilde{u}_t^1, ..., \tilde{u}_t^{N-1}, u_t^N), \quad \forall u_t^N \in \mathcal{N}(\tilde{u}_t^N), \end{aligned} \tag{4.6}$$

The set $\mathcal{N}(\tilde{u}_t^i)$ is some neighborhood around $\tilde{u}_t^i$. There may exist multiple, potentially non-isolated, local Nash equilibria. For the purposes considered here, we require only that for any state x_t, the policies evaluate to one arbitrarily chosen, yet particular, local equilibrium. A more stringent definition for the policies π_t could require that the inequalities in Eq. (4.6) hold over the entire sets $\mathcal{U}_t^i$. In any case, the Value-functions for stages $t \in \mathbf{T}$ are defined as

$$V_t^i(x_t) := Z_t^i(x_t, \pi_t^1(x_t), ..., \pi_t^N(x_t)). \tag{4.7}$$

Definition 3 (GFNE)**.** A Local Generalized Feedback Nash Equilibrium is defined by a set of policies π_t^i, $t \in \mathbf{T}$, $i \in \mathbf{N}$ defined in Eq. (4.6), such that the value of $V_1^i(\hat{x}_1)$, defined in Eq. (4.7), is finite for all $i \in \mathbf{N}$. Note that in this chapter we refer to local GFNE whenever we write GFNE.

Consider a collection of policies constituting a GFNE. Let the corresponding *equilibrium trajectory* be denoted by x_t^*, u_t^* such that

$$\begin{aligned} x_1^* &:= \hat{x}_1, & \\ u_t^{i*} &:= \pi_t^i(x_t^*), & t \in \mathbf{T}, \\ x_{t+1}^* &:= f_t(x_t^*, u_t^{1*}, ..., u_t^{N*}), & t \in \mathbf{T}. \end{aligned} \tag{4.8}$$

Parametric Formulation

To encode a GFNE problem, one approach would be to use a parametric representation of the policies π_t^i, and reinterpret the decisions made by players as the parameters of these policies. That is, we could restrict each player to choose from policies $\pi_t^i \equiv \pi_{t,\theta_t^i}^i$ at each time t, parameterized by a real vector θ_t^i of arbitrary finite dimension. For clarity we also define $\theta^i := (\theta_1^i, ..., \theta_T^i)$ and $\theta_t := (\theta_t^1, ..., \theta_t^N)$. A GFNE problem could then be expressed as the following set of coupled optimization problems in which each player i minimizes over policy parameters θ^i, and the trajectory x, u (including the controls of other players, u^{-i}):

$$\min_{\theta^i, x, u} \; L^i(x, u) \tag{4.9a}$$
$$s.t. \quad 0 = x_{t+1} - f_t(x_t, u_t), \quad t \in \mathbf{T} \tag{4.9b}$$
$$0 = u_t - \pi_{t,\theta_t}(x_t), \quad t \in \mathbf{T} \tag{4.9c}$$
$$0 = h_t^i(x_t, u_t), \quad t \in \mathbf{T} \tag{4.9d}$$
$$0 \le g_t^i(x_t, u_t), \quad t \in \mathbf{T} \tag{4.9e}$$
$$0 = h_{T+1}^i(x_{T+1}) \tag{4.9f}$$
$$0 \le g_{T+1}^i(x_{T+1}) \tag{4.9g}$$

Because the parameterized control policies are treated as constraints, and the controls u^{-i} are treated as decision variables, the reaction of other players to the decisions of player i are explicitly accounted for in the optimization Eq. (4.9). Solutions to this encoding of the game could be found by finding a Generalized Nash Equilibrium for the set of N optimization problems (one corresponding to each player), using a method such as those described in [70].

Although this formulation is theoretically equivalent to a Generalized Nash Equilibrium problem (as are trajectory games with an open-loop information pattern), there are several important issues associated with it. Specifically, even if the parameterization of policies is a simple stage-varying affine map, player i has $m^i \times (n+1) \times T$ degrees of freedom, many more than the underlying control dimension m^i. Furthermore, this over-parameterization leads to an ambiguity in the choice of policy—for any single state x_t and control u_t, there exist an infinite number of linear maps which relate the two. In general, any collection of policy parameters which lead to the same equilibrium trajectory (i.e., equivalent *representations* of one another, [19, Definition 5.12]), are indistinguishable in Eq. (4.9). Therefore, without additional regularization, the optimization is under-specified, and this leads to ill-conditioning of the problem.

Perhaps the most important problem that arises with the formulation Eq. (4.9) is that despite relationships between the control variables u_t and state x_t being accounted for via the policies π_{t,θ_t}, the gradient of this policy is not necessarily meaningful. Specifically, if the dynamics Eq. (4.9b) and policies Eq. (4.9c) are substituted into the cost functional Eq. (4.9a), and the gradient is taken with respect to the parameters θ_t^i, the chain rule relates the effect of the parameters on the cost through the policy gradients $\nabla_x \pi_{t,\theta_t}$. However, the over-representation of the policies π_{t,θ_t} implies that the gradient need not correspond to the true gradient of the subgame-optimal policy. To illustrate this, consider a simple example, describing the relationship between a scalar state x_t, and player 1's scalar control u_t^1. Let $\pi_{t,\theta_t}(x_t) = \theta_1 \cdot x_t + \theta_2$. If at the state $x_t = 2$, the subgame-optimal value for u_t^1 is 1, then one possible parameterization is $\theta_1 = 0.5$, and $\theta_2 = 0$. However, if at the state $x_t = 2 + \epsilon$, the subgame-optimal value for u_t^1 is $1 - \epsilon$, then this implies that the original estimate of $\nabla_x \pi_{t,\theta_t} = 0.5$ is unrelated to the observed gradient of the subgame-optimal policy, which is -1.

Non-parametric Formulation

As discussed above, while it is conceivable to express GFNE problems by use of parameterized policies, the resulting formulation leads to significant numerical and theoretical challenges. The remainder of this work is devoted to an "implicit," non-parametric encoding of the GFNE problem, which does not suffer the same problems associated with the parametric formulation. We begin the presentation of this approach by first noting that the GFNE problem can be expressed precisely as a *EPNEC*, as depicted in Fig. 4.1. Using this interpretation, we can lean on the same ideas presented in chapters 2 and 3, and express the policies at each stage t in terms of the subgame starting at that stage.

Theorem 4.2.1. *The policies defined in Eq.* (4.6) *can equivalently be expressed in terms of the nested Generalized Nash Equilibrium Problems Eq.* (4.10).

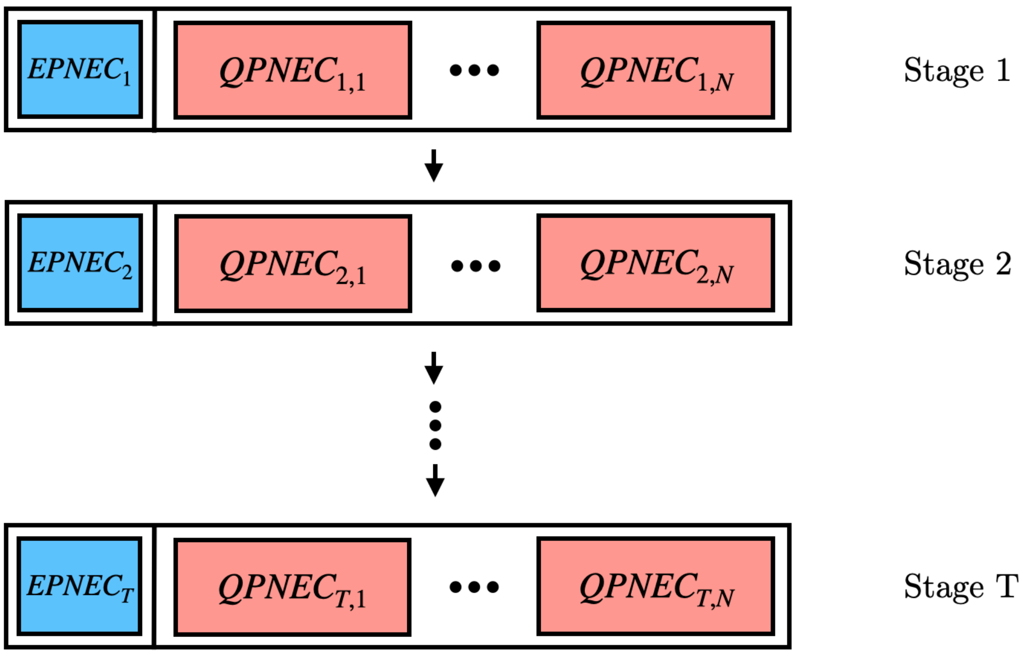


Figure 4.1: The $EPNEC$ interpretation of GFNE problems. Decision variable scopes for the problems at each layer are given in the Table 4.1.

l	Input to $EPNEC_l$
1	x_1
2	x_2
$\vdots$	$\vdots$
T	x_T
l	Output of $EPNEC_l$
1	$\begin{bmatrix} u_1 & x_2 & \dots & u_T & x_{T+1} \end{bmatrix}$
1	$\begin{bmatrix} u_2 & x_3 & \dots & u_T & x_{T+1} \end{bmatrix}$
$\vdots$	$\vdots$
T	$\begin{bmatrix} u_T & x_{T+1} \end{bmatrix}$

Table 4.1: Variable scopes for the $EPNECs$ representing the GFNE appearing in Fig. 4.1

$$\pi_t^i(x_t) := \tilde{u}_t^i \in \arg_{u_t^i} \min_{\substack{u_{t:T}^i, \\ \tilde{u}_{t+1:T}^{-i}, \\ x_{t+1:T+1}}} \sum_{s=t}^{T} l_s^i(x_s, u_s^i, \tilde{u}_s^{-i}) + l_{T+1}^i(x_{T+1}) \tag{4.10a}$$

$$\text{s.t.} \quad 0 = \tilde{u}_s^{-i} - \pi_s^{-i}(x_s), \qquad s \in \mathbf{T_{t+1}} \tag{4.10b}$$

$$0 = x_{s+1} - f_s(x_s, u_s^i, \tilde{u}_s^{-i}), \qquad s \in \mathbf{T_t} \tag{4.10c}$$

$$0 = h_s^i(x_s, u_s^i, \tilde{u}_s^{-i}), \qquad s \in \mathbf{T_t} \tag{4.10d}$$

$$0 \leq g_s^i(x_s, u_s^i, \tilde{u}_s^{-i}), \qquad s \in \mathbf{T_t} \tag{4.10e}$$

$$0 = h_{T+1}^i(x_{T+1}) \tag{4.10f}$$

$$0 \leq g_{T+1}^i(x_{T+1}) \tag{4.10g}$$

Here $\pi_t^{-i}(x_t) := (\pi_t^1(x_t), ..., \pi_t^{i-1}(x_t), \pi_t^{i+1}(x_t), ..., \pi_t^N(x_t))$, and the notation $\arg_a \min_{a,b}$ is used to indicate that the minimum is taken over a and b, but only the value of a at the minimum is returned. Furthermore, the value of the minimization appearing in Eq. (4.10) is considered infinite for any combination of optimization variables violating the constraints. Note that the set $\mathbf{T}_{T+1}$ is empty, so for stage $t = T$, the constraint Eq. (4.10b) vanishes.

Proof. Starting with stage T, and substituting x_{T+1} using the dynamics Eq. (4.10c), and moving constraints Eqs. (4.10d) to (4.10g) into the objective by means of infinite-valued indicator functions, observe that the objective of the minimization is equivalent to Z_T^i Eq. (4.5) as claimed.

Now, for some other stage $t \in \mathbf{T}$, assuming $\pi_t^i(x_{t+1})$ can be expressed by Eq. (4.10), it can be shown that $\pi_t^i(x_t)$ must also be equivalently expressed by Eq. (4.10). Eq. (4.10) can be re-written as the following:

$$
\begin{aligned}
\pi^i_t(x_t) := \tilde{u}^i_t \in \underset{\substack{u^i_t\\ \tilde{u}^{-i}_{t+1},\\ x_{t+1}}}{\underset{u^i_t}{\operatorname{argmin}}} \Bigg\{ & l^i_t(x_t, u^i_t, \tilde{u}^{-i}_t) + \min_{\substack{u^i_{t+1:T},\\ \tilde{u}^{-i}_{t+2:T},\\ x_{t+2:T+1}}} \sum_{s=t+1}^{T} l^i_s(x_s, u^i_s, \tilde{u}^{-i}_s) + l^i_{T+1}(x_{T+1}) \Bigg\} && \text{(4.11a)}\\
& \text{s.t.} \quad 0 = \tilde{u}^{-i}_s - \pi^{-i}_s(x_s), \qquad s \in \mathbf{T_{t+2}} && \text{(4.11b)}\\
& \qquad\quad 0 = x_{s+1} - f_s(x_s, u^i_s, \tilde{u}^{-i}_s), \; s \in \mathbf{T_{t+1}} && \text{(4.11c)}\\
& \qquad\quad 0 = h^i_s(x_s, u^i_s, \tilde{u}^{-i}_s), \qquad s \in \mathbf{T_{t+1}} && \text{(4.11d)}\\
& \qquad\quad 0 \le g^i_s(x_s, u^i_s, \tilde{u}^{-i}_s), \qquad s \in \mathbf{T_{t+1}} && \text{(4.11e)}\\
& \qquad\quad 0 = h^i_{T+1}(x_{T+1}) && \text{(4.11f)}\\
& \qquad\quad 0 \le g^i_{T+1}(x_{T+1}) && \text{(4.11g)}\\
\text{s.t.} \quad & 0 = \tilde{u}^{-i}_{t+1} - \pi^{-i}_{t+1}(x_{t+1}) && \text{(4.11h)}\\
& 0 = x_{t+1} - f(x_t, u^i_t, \tilde{u}^{-i}_t) && \text{(4.11i)}\\
& 0 = h^i_t(x_t, u^i_t, \tilde{u}^{-i}_t) && \text{(4.11j)}\\
& 0 \le g^i_t(x_t, u^i_t, \tilde{u}^{-i}_t) && \text{(4.11k)}
\end{aligned}
$$

The nested minimum appearing in Eq. (4.11) is exactly that appearing in Eq. (4.10) for stage $t+1$ (ignoring Eq. (4.11b) if $t+1 = T$). Because the controls $\tilde{u}^{-i}_{t+1}$ (for $t+1 \le T$) are constrained by the policies $\pi^{-i}_{t+1}(x_{t+1})$, the value of this nested minimization must equal the value function $V_{t+1}(x_{t+1})$ as defined in Eq. (4.7) for any minimizer u^i_{t+1}, regardless of whether or not the minimizer corresponds to the particular one corresponding to $\pi^i_{t+1}(x_{t+1})$. By substituting x_{t+1} using the constraint Eq. (4.11i), and using infinite-valued indicator functions to move Eqs. (4.11j) and (4.11k) into the objective of Eq. (4.11), we see that the objective of the minimization is equivalent to Eq. (4.5) for stage t. Thus, the alternate definition of $\pi^i_t(x_t)$ in Eq. (4.10) is equivalent to that in Eq. (4.6) for all stages t. □

Here we have defined the GFNE policies in terms of the *nested* equilibrium problems with equilibrium constraints Eq. (4.10). These equilibrium constraints arise in these problems because the constraints Eq. (4.10b) are defined in terms of equilibrium problems. Critically though, the set of players in all inner-level equilibrium problems are exactly those in the outer-level problems, allowing for the removal of the redundant constraint that $u^i_s = \pi^i_s(x_s)$, $s \in \mathbf{T_{t+1}}$ from player i's problem statement Eq. (4.10), as demonstrated in Theorem 4.2.1. When the necessary conditions of all players are concatenated, the constraints $u^i_s = \pi^i_s(x_s)$, $s \in \mathbf{T_{t+1}}$ become redundant for *all* i, as we show in Theorem 4.2.2. This fact will allow for a compact representation of necessary conditions associated with solutions of a GFNE, and ultimately algorithms for finding such solutions.

Theorem 4.2.2 (Necessary Conditions). *For some stage $t \in \mathbf{T}$, consider any set of policies π^i_s, $s \in \mathbf{T_t}$, $i \in \mathbf{N}$, as defined in Eq.* (4.10)*. Let the state $\hat{x}_t$ be such that a solution exists to equilibrium problem Eq.* (4.10) *at stage t. Denote the resultant sub-game solution trajectory* *by* $\{x_s^*;\ s \in \mathbf{T_t^+},\ x_t^* = \hat{x}_t\}$, $\{u_s^*;\ s \in \mathbf{T_t}\}$. *Furthermore, assume the policies* $\pi_s(x_s)$ *are differentiable at the point* x_s^* *for* $s \in \mathbf{T_{t+1}}$, *and a standard constraint qualification such as the linear independence constraint qualification holds for the optimization problem appearing in Eq.* (4.10), *for each* $i \in \mathbf{N}$. *Then there exist multipliers* $\{\lambda_s^i \in \mathbb{R}^n;\ s \in \mathbf{T_t},\ i \in \mathbf{N}\}$, $\{\mu_s^i \in \mathbb{R}^{a_s^i};\ s \in \mathbf{T_t^+},\ i \in \mathbf{N}\}$, $\{\gamma_s^i \in \mathbb{R}^{b_s^i};\ s \in \mathbf{T_t^+},\ i \in \mathbf{N}\}$, *and* $\{\psi_s^i \in \mathbb{R}^{m^{-i}};\ s \in \mathbf{T_{t+1}},\ i \in \mathbf{N}\}$ *which satisfy:*

$$0 = \nabla_{u_s^i}\left[l_s^i + f_s^\intercal \lambda_s^i - h_s^{i\intercal}\mu_s^i - g_s^{i\intercal}\gamma_s^i\right]_{x_s^*,u_s^*}, \quad i \in \mathbf{N},\ s \in \mathbf{T_t} \tag{4.12a}$$

$$0 = \nabla_{x_s}\left[l_s^i - \lambda_{s-1}^i + f_s^\intercal \lambda_s^i - h_s^{i\intercal}\mu_s^i - g_s^{i\intercal}\gamma_s^i + \pi_s^{-i\intercal}\psi_s^i\right]_{x_s^*,u_s^*}, \quad i \in \mathbf{N},\ s \in \mathbf{T_{t+1}}, \tag{4.12b}$$

$$0 = \nabla_{u_s^{-i}}\left[l_s^i + f_s^\intercal \lambda_s^i - h_s^{i\intercal}\mu_s^i - g_s^{i\intercal}\gamma_s^i - \psi_s^i\right]_{x_s^*,u_s^*}, \quad i \in \mathbf{N},\ s \in \mathbf{T_{t+1}}, \tag{4.12c}$$

$$0 = \nabla_{x_{T+1}}\left[l_{T+1}^i - \lambda_T^i - h_{T+1}^{i\intercal}\mu_{T+1}^i - g_{T+1}^{i\intercal}\gamma_{T+1}^i\right]_{x_{T+1}^*}, \quad i \in \mathbf{N}, \tag{4.12d}$$

$$0 = x_{s+1}^* - f_s(x_s^*, u_s^*), \quad s \in \mathbf{T_t}, \tag{4.12e}$$

$$0 = h_s^i(x_s^*, u_s^*), \quad i \in \mathbf{N},\ s \in \mathbf{T_t}, \tag{4.12f}$$

$$0 \le g_s^i(x_s^*, u_s^*) \perp \gamma_s^i \ge 0, \quad i \in \mathbf{N},\ s \in \mathbf{T_t}, \tag{4.12g}$$

$$0 = h_{T+1}^i(x_{T+1}^*), \quad i \in \mathbf{N}, \tag{4.12h}$$

$$0 \le g_{T+1}^i(x_{T+1}^*) \perp \gamma_{T+1}^i \ge 0, \quad i \in \mathbf{N}. \tag{4.12i}$$

Furthermore, let $L_t(z_t, x_t^*) = 0$ *denote the entire set of conditions Eq.* (4.12) *formed by treating active inequalities Eqs.* (4.12g) *and* (4.12i) *as equalities, and ignoring the inactive inequalities. Here,* z_t *is the set of all variables appearing in Eq.* (4.12) *other than* x_t^* and *all multipliers* γ_s^i *associated with inactive inequality constraints. If strict complementarity holds, and the matrix* $\nabla_{z_t} L_t$ *is non-singular, then the policy* $\pi_t(x_t)$ *is also differentiable at the point* $\hat{x}_t$, *and* $\nabla_x \pi_t(x_t) := -[[\nabla_{z_t} L_t]^{-1} \nabla_{x_t} L_t]_{u_t}$.

The notation $[\cdot]_{u_t}$ implies that if u_t appears in the j_1 through j_2 indices of z_t, then the j_1 through j_2 rows of the matrix argument are selected. The notation in the first equation of Eq. (4.12) is used to indicate that the gradients of the functions $l_s^i(x_s, u_s)$, $f(x_s, u_s)$, $h_s^i(x_s, u_s)$, and $g_s^i(x_s, u_s)$ are evaluated at x_s^* and u_s^*. The symbol $\perp$ is used to indicate complementarity of the left- and right-hand-side conditions. For example, if the k-th element of $g_s^i(x_s^*, u_s^*) > 0$ then the k-th element of γ_s^i must be 0, and vice-versa. As before, for the final stage $t = T$, the set of conditions Eqs. (4.12b) and (4.12c) is empty, as the set $\mathbf{T_{T+1}} = \emptyset$.

Proof. By the assumption that the optimization problems Eq. (4.10) satisfy a standard constraint qualification, the Lagrange Multiplier theorem states that there must exist multipliers associated with each player $i \in \mathbf{N}$'s optimization problem at stage t, such that both the conditions in Eq. (4.12) (evaluated for the particular $i \in \mathbf{N}$) and the constraints $u_s^{-i*} = \pi_s^{-i}(x_s)$

hold. Concatenating these conditions for all of the N players gives rise to the conditions Eq. (4.12), with the addition of the constraints Eq. (4.10b) for each $i \in \mathbf{N}$. Since there must exist multipliers satisfying those conditions, the conditions Eq. (4.12) must also be satisfied, as Eq. (4.12) are formed by simply removing the constraints $u_s^* = \pi_s(x_s)$.

After removing all inactive inequality constraints and associated multipliers from Eq. (4.12), it is straightforward to verify that the dimension of the system $L_t(z_t, x_t^*)$ and z_t are the same, and therefore that $\nabla_{z_t} L_t$ is a square matrix. If z_t^* is comprised of $x_{t+1:T+1}^*$, $u_{t:T}^*$ and multipliers satisfying Eq. (4.12) such that $L_t(z_t^*, x_t^*) = 0$, then assuming this matrix is non-singular, the Implicit Function Theorem states that there must exist a unique function $\Pi_t(x_t)$ and open set $\mathcal{X}_t^* \subset \mathcal{X}$ containing x_t^* such that $L_t(\Pi_t(x_t), x_t)) = 0$ for all $x_t \in \mathcal{X}_t^*$. By the uniqueness of this function, we must have that for all $x_t \in \mathcal{X}_t^*$, $[\Pi_t(x_t)]_{u_t} = \pi_t(x_t)$, where $[\Pi_t(x_t)]_{u_t}$ selects the components of the function Π_t corresponding to the subset of z_t containing u_t. Therefore, for all $x_t \in \mathcal{X}_t^*$, $\nabla_{x_t} \pi_t(x_t) = \nabla_{x_t}([\Pi_t(x_t)]_{u_t}) = -[[\nabla_{z_t} L_t]^{-1} \nabla_{x_t} L_t]_{u_t}$. □

Observe that the conditions $L_t(z_t, x_t) = 0$ contain as a subset the conditions $L_s(z_s, x_s) = 0$, $s \in \mathbf{T_{t+1}}$. If the matrices $\nabla_{x_s} L_s(z_s, x_s)$, $s \in \mathbf{T_{t+1}}$ are also non-singular, then in some neighborhood of z_t^*, the constraints $u_s^* = \pi_s(x_s^*)$, $s \in \mathbf{T_{t+1}}$ are equivalent to $L_s(z_s^*, x_s^*) = 0$, motivating the removal of the constraints Eq. (4.10b) from the conditions Eq. (4.12).

For games and corresponding solutions satisfying the stated assumptions, Theorem 4.2.2 suggests a method for computing those solutions. Evaluating the conditions Eq. (4.12) only requires the evaluation of the policy gradients $\nabla_{x_s} \pi_s(x_s)$, and not the policies themselves. This is important for computational reasons, since while in general an explicit representation of the policies is unavailable, it is possible to evaluate the policy gradients.

The procedure we propose for computing GFNE trajectories is to find a solution to the conditions Eq. (4.12), which can then be checked against a sufficiency condition to ensure that the solution indeed constitutes a GFNE.

Theorem 4.2.3 (Sufficient Conditions)**.** *Consider any set of states $\{x_s^*,\ s \in \mathbf{T_t^+}\}$ and controls $\{u_s^*,\ s \in \mathbf{T_t}\}$, which together with multipliers $\{\lambda_s^i \in \mathbb{R}^n;\ s \in \mathbf{T_t},\ i \in \mathbf{N}\}$, $\{\mu_s^i \in \mathbb{R}^{a_s^i};\ s \in \mathbf{T_t^+},\ i \in \mathbf{N}\}$, $\{\gamma_s^i \in \mathbb{R}^{b_s^i};\ s \in \mathbf{T_t^+},\ i \in \mathbf{N}\}$, and $\{\psi_s^i \in \mathbb{R}^{m^{-i}};\ s \in \mathbf{T_{t+1}},\ i \in \mathbf{N}\}$ satisfy the conditions Eq. (4.12), with the matrix $\nabla_{z_s} L_s$ non-singular for all $s \in \mathbf{T}$. If additionally, for all $i \in \mathbf{N}$,*

$$\sum_{s=t}^{T} \begin{bmatrix} d_{x,s} \\ d_{u,s} \end{bmatrix}^\intercal \nabla^2 l_s^i(x_s^*, u_s^*) \begin{bmatrix} d_{x,s} \\ d_{u,s} \end{bmatrix} + d_{x,T+1}^\intercal \nabla^2 l_{T+1}^i(x_{T+1}^*) d_{x,T+1} > 0, \tag{4.13a}$$

$$\forall \{d_{x,s}, d_{u,s}\} \text{ s.t. } 0 = d_{u_s} - \nabla_x \pi_s(x_s^*) d_{x,s}, \qquad s \in \mathbf{T_t}, \tag{4.13b}$$

$$0 = d_{x,s+1} - \nabla_x f_s(x_s^*, u_s^*) d_{x,s} - \nabla_u f(x_s^*, u_s^*) d_{u_s}, \quad s \in \mathbf{T_t}, \tag{4.13c}$$

then the trajectory x_s^*, u_s^* constitutes a GFNE trajectory.

Proof. The set of $d_{x,s}$, $d_{u,s}$ satisfying Eqs. (4.13b) and (4.13c) is a super-set of the critical constraint cone [192, Eq. 12.53], therefore the trajectory x_s^*, u_s^* must constitute a true local minimum of the problems Eq. (4.10) for stage t and each player $i \in \mathbf{N}$, as stated in [192, Theorem 12.6]. □

The sufficiency condition outlined in Theorem 4.2.3 is stricter than necessary since it ignores other active constraints which reduce the volume of the critical constraint cone, and could be relaxed by considering the linearization of all active constraints.

Although Theorems 4.2.2 and 4.2.3 together outline a procedure for computing GFNE trajectories, there remain some difficulties which must be addressed if such a procedure is to be practical. It is important to note that while the conditions Eq. (4.12) do not require the evaluation of the policies $\pi_s(x_s), s \in \mathbf{T_{t+1}}$ explicitly, evaluating the policy gradients $\nabla_x \pi_s(x_s)$ is required, and their evaluation involves inverting a matrix which depends on second-order derivatives of all policies $\pi_r(x_r)$, $r \in \mathbf{T_{s+1}}$. Furthermore, evaluating second-order derivatives of any policy requires evaluating third-order derivatives of all subsequent policies, and so forth. The effect of this is that T^{th}-order derivatives of dynamic and constraint functions appearing at the late stages of the game must be computed to evaluate the conditions Eq. (4.12) when $t = 1$. While technically possible, this requirement is impractical for many games. We therefore introduce a reasonable approximation to the computation of policy gradients $\nabla_x \pi_t(x_t)$ which do not require the evaluation of any higher-order derivatives of policies $\pi_s(x_s)$, $s \in \mathbf{T_{t+1}}$.

Definition 4 (Policy Quasi-Gradients)**.** We approximate $\nabla_x \pi_t(x_t)$ by what is termed the policy quasi-gradient, K_t, defined implicitly by the following conditions:

$$
\begin{aligned}
0 &= \nabla_{u_s^i}\left[l_s^i + f_s^\intercal \lambda_s^i - h_s^{i\intercal}\mu_s^i - g_s^{i\intercal}\gamma_s^i\right]_{x_s^*,u_s^*}, && i \in \mathbf{N},\ s \in \mathbf{T_t} \\
0 &= \nabla_{x_s}\left[l_s^i - \lambda_{s-1}^i + f_s^\intercal \lambda_s^i - h_s^{i\intercal}\mu_s^i - g_s^{i\intercal}\gamma_s^i\right]_{x_s^*,u_s^*} + K_s^{-i\intercal}\psi_s^i, && i \in \mathbf{N},\ s \in \mathbf{T_{t+1}}, \\
0 &= \nabla_{u_s^{-i}}\left[l_s^i + f_s^\intercal \lambda_s^i - h_s^{i\intercal}\mu_s^i - g_s^{i\intercal}\gamma_s^i - \psi_s^i\right]_{x_s^*,u_s^*}, && i \in \mathbf{N},\ s \in \mathbf{T_{t+1}}, \\
0 &= \nabla_{x_{T+1}}\left[l_{T+1}^i - \lambda_T^i - h_{T+1}^{i\intercal}\mu_{T+1}^i - g_{T+1}^{i\intercal}\gamma_{T+1}^i\right]_{x_{T+1}^*}, && i \in \mathbf{N}, \\
0 &= x_{s+1}^* - f_s(x_s^*, u_s^*), && s \in \mathbf{T_t}, \\
0 &= h_s^i(x_s^*, u_s^*), && i \in \mathbf{N},\ s \in \mathbf{T_t}, \\
0 &\le g_s^i(x_s^*, u_s^*) \perp \gamma_s^i \ge 0, && i \in \mathbf{N},\ s \in \mathbf{T_t}, \\
0 &= h_{T+1}^i(x_{T+1}^*), && i \in \mathbf{N}, \\
0 &\le g_{T+1}^i(x_{T+1}^*) \perp \gamma_{T+1}^i \ge 0, && i \in \mathbf{N}.
\end{aligned}
\tag{4.14}
$$

The conditions Eq. (4.14) are nearly the same as Eq. (4.12), although the actual policy gradients $\nabla_x \pi_s(x_s)$ have been replaced with the quasi-gradients K_s. Letting $\hat{L}_t(z_t, x_t^*)$

represent the concatenation of active conditions in Eq. (4.14) (analogous to $L_t(z_t, x_t^*)$ in Theorem 4.2.2), then

$$K_t := -[[\nabla_{z_t} \hat{L}_t]^{-1} \nabla_{x_t} \hat{L}_t]_{u_t}, \tag{4.15}$$

where $K_s,\ s \in \mathbf{T_{t+1}}$ are treated as constants ($\nabla_{z_t} K_s = 0, s \in \mathbf{T_{t+1}}$). If the matrix $[\nabla_{z_t} \hat{L}_t]$ is singular at some (z_t, x_t), we say that the policy quasi-gradient does not exist at that point.

The conditions Eq. (4.14) can be evaluated without the need for computing any third- or higher-order derivatives of any constraint or objective terms of the game, and can also be evaluated efficiently, as we will show. Solutions satisfying the conditions Eq. (4.14) will not satisfy the conditions Eq. (4.12) in general. Rather, the solutions to Eq. (4.14) will be distinct from solutions to Eq. (4.12), and therefore we introduce the notion of a Generalized Feedback Quasi-Nash Equilibrium (GFQNE) to characterize these solutions. Empirical results indicate that GFQNE solutions closely approximate GFNE solutions.

Definition 5 (GFQNE)**.** Let $\{x_s^*;\ s \in \mathbf{T_t^+},\ x_t^* = \hat{x}_t\}$, $\{u_s^*;\ s \in \mathbf{T_t}\}$, $\{\lambda_s^i \in \mathbb{R}^n;\ s \in \mathbf{T_t},\ i \in \mathbf{N}\}$, $\{\mu_s^i \in \mathbb{R}^{a_s^i};\ s \in \mathbf{T_t^+},\ i \in \mathbf{N}\}$, $\{\gamma_s^i \in \mathbb{R}^{b_s^i};\ s \in \mathbf{T_t^+},\ i \in \mathbf{N}\}$, and $\{\psi_s^i \in \mathbb{R}^{m^{-i}};\ s \in \mathbf{T_{t+1}},\ i \in \mathbf{N}\}$ be such that the conditions Eq. (4.14) are satisfied. Furthermore, let

$$\begin{aligned}
&\sum_{s=t}^{T} \begin{bmatrix} d_{x,s} \\ d_{u,s} \end{bmatrix}^\intercal \nabla^2 l_s^i(x_s^*, u_s^*) \begin{bmatrix} d_{x,s} \\ d_{u,s} \end{bmatrix} + d_{x,T+1}^\intercal \nabla^2 l_{T+1}^i(x_{T+1}^*) d_{x,T+1} > 0, \\
&\forall \{d_{x,s}, d_{u,s}\}\ s.t.\ \ 0 = d_{u_s} - K_s d_{x,s}, \qquad\qquad\qquad\qquad\qquad\qquad\quad s \in \mathbf{T_t}, \\
&\qquad\qquad\qquad\qquad\ \ 0 = d_{x,s+1} - \nabla_x f_s(x_s^*, u_s^*) d_{x,s} - \nabla_u f(x_s^*, u_s^*) d_{u_s}, \ \ s \in \mathbf{T_t},
\end{aligned} \tag{4.16}$$

then we say that the trajectory x_s^*, u_s^* constitutes a Generalized Feedback Quasi-Nash Equilibrium (GFQNE) trajectory. Note that the condition Eq. (4.16) differs from the condition in Theorem 4.2.3 in the definition of the critical cone.

If all cost functionals Eq. (4.2) in the game are quadratic, and all dynamic Eq. (4.1) and non-dynamic Eqs. (4.3a) to (4.3b) constraints are linear, then the policy quasi-gradients are equivalent to the policy gradients, and all GFQNE are therefore GFNE.

In the general setting, the policy quasi-gradients do not exactly match the policy gradients, which potentially introduces a different type of computational difficulty. Using Newton-type methods to solve for solutions to Eq. (4.14), we will ideally be able to evaluate $[\nabla_{z_1} \hat{L}_1]^{-1}$ exactly, without treating $K_s,\ s \in \mathbf{T_2}$ as constants (since indeed, they depend on z_1). If we are unwilling or unable to compute derivatives of sub-game policy quasi-gradients, we will be forced to use a quasi-Newton method at best. Because we are working in the game setting, and the matrix $\nabla_{z_t} \hat{L}_t$ will, in general, be asymmetric, it is difficult to provide guarantees that such a quasi-Newton method will converge. Nevertheless, we find that in practice, such an approach does in fact converge and is useful in interesting settings.

So far we have also made an important, limiting assumption, which is that the matrices $\nabla_{z_t} L_t$ and $\nabla_{z_t} \hat{L}_t$ are non-singular for all $t \in \mathbf{T}$. For many common forms of constraints

Eq. (4.3a), Eq. (4.3b), this assumption cannot hold. This will occur, for example, when there is a terminal constraint on the entire game state, such as $h^i_{T+1}(x_{T+1}) := x_{T+1}$. If $m^i_t < n$, then the matrices $\nabla_{z_t}\hat{L}_t$ necessarily must be singular. Since many games involve constraints of this form, we handle them in the following way.

If at any stage t, the matrix $\nabla_{z_t}\hat{L}_t$ is found to be singular, and the game is otherwise well-posed[1], then this is likely due to an over-constrained sub-game. In this situation, we can combine the stage t with the preceding stage $t-1$, and define new combined-stage dynamics and constraint functions accordingly. For example, assume at stage t the matrix $\nabla_{z_t}\hat{L}_t$ is singular. We then define $\hat{u}_{t-1} := [u_{t-1}^\intercal\ u_t^\intercal]^\intercal$, $\hat{\mathcal{U}}^i_{t-1} := \mathcal{U}^i_{t-1} \times \mathcal{U}^i_t$, and the updated dynamic, constraint, and stage-wise cost functionals as

$$\hat{f}_{t-1}(x_{t-1}, \hat{u}_{t-1}) := f_t(f_{t-1}(x_{t-1}, u_{t-1}), u_t), \tag{4.17a}$$

$$\hat{g}^i_{t-1}(x_{t-1}, \hat{u}_{t-1}) := [g^i_{t-1}(x_{t-1}, u_{t-1})^\intercal\ g^i_t(f(x_{t-1}, u_{t-1}), u_t)^\intercal]^\intercal, \tag{4.17b}$$

$$\hat{h}^i_{t-1}(x_{t-1}, \hat{u}_{t-1}) := [h^i_{t-1}(x_{t-1}, u_{t-1})^\intercal\ h^i_t(f(x_{t-1}, u_{t-1}), u_t)^\intercal]^\intercal, \tag{4.17c}$$

$$\hat{l}^i_{t-1}(x_{t-1}, \hat{u}_{t-1}) := l^i_{t-1}(x_{t-1}, u_{t-1}) + l^i_t(f(x_{t-1}, u_{t-1}), u_t). \tag{4.17d}$$

In this procedure, we effectively reduce the number of stages of the game by 1, but the dimension of all controls input to the game and the cost and constraints imposed upon each player are unchanged.

Throughout the remainder of this chapter, we will assume that game stages are combined as necessary to ensure the subgame policy quasi-gradients are well-defined, and the game horizon T, dynamics, constraints, and cost-functionals all reflect any such modifications. In what follows we focus on the derivation of numerical methods for computing Generalized Feedback Quasi-Nash Equilibria. We begin our presentation by considering a special-case, which will serve as a building block for more general methods.

4.3 Equality-Constrained LQ Games

We consider the case in which the dynamics equation describing the game evolution Eq. (4.1) is linear in its arguments, the cost-functionals Eq. (4.2) for each player are quadratic functions of the state and control variables, and each player is subject only to linear equality constraints.

In particular, let

$$x_{t+1} = A_t x_t + B^1_t u^1_t + ... + B^N_t u^N_t + c_t, \quad t \in \mathbf{T}, \tag{4.18}$$

[1]For example, the quadratic cost functionals of every player have sufficient curvature in the tangent cone of the game.

for (time-varying) matrices $A_t \in \mathbb{R}^{n\times n}$, $B_t^i \in \mathbb{R}^{m_t^i \times n}$, and vectors $c_t \in \mathbb{R}^n$. Associated with the dynamic constraints are multipliers λ_t^i for each player $i \in \mathbf{N}$. Let

$$\begin{aligned} B_t &:= \begin{bmatrix} B_t^1 & \dots & B_t^N \end{bmatrix}, \quad \hat{B}_t := \begin{bmatrix} B_t^1 & & \\ & \ddots & \\ & & B_t^N \end{bmatrix}, \quad \lambda_t := \begin{bmatrix} \lambda_t^1 \\ \vdots \\ \lambda_t^N \end{bmatrix}, \\ \tilde{B}_t &:= \begin{bmatrix} B_t^2 & \dots & B_t^N & & & \\ & & & \ddots & & \\ & & & & B_t^1 & \dots & B_t^{N-1} \end{bmatrix}. \end{aligned} \tag{4.19}$$

In this setting, the cost functionals for each player can be expressed as:

$$\begin{aligned} l_t^i(x_t, u_t) &:= \frac{1}{2}\left(\begin{bmatrix} x_t \\ u_t \end{bmatrix}^\intercal \begin{bmatrix} Q_t^i & S_t^{i\intercal} \\ S_t^i & R_t^i \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + 2 \begin{bmatrix} x_t \\ u_t \end{bmatrix}^\intercal \begin{bmatrix} q_t^i \\ r_t^i \end{bmatrix} \right), \\ l_{T+1}^i(x_{T+1}) &:= \frac{1}{2}\left(x_{T+1}^\intercal Q_{T+1}^i x_{T+1} + 2 x_{T+1}^\intercal q_{T+1}^i \right), \end{aligned} \tag{4.20}$$

for (time-varying) matrices $Q_t^i \in \mathbb{R}^{n\times n}$, $S_t^i \in \mathbb{R}^{m_t \times n}$, $R_t^i \in \mathbb{R}^{m_t \times m}$, and vectors $q_t^i \in \mathbb{R}^n$, $r_t^i \in \mathbb{R}^{m_t}$. For notational purposes, let the terms R_t^i, S_t^i, and r_t^i be comprised of sub-matrices, $R_t^{i,j,k} \in \mathbb{R}^{m_t^j \times m_t^k}$, $S_t^{i,j} \in \mathbb{R}^{m_t^j \times n}$, and sub-vectors $r_t^{i,j} \in \mathbb{R}^{m_t^j}$, for $j, k \in \mathbf{N}$:

$$R_t^i := \begin{bmatrix} R_t^{i,1,1} & \dots & R_t^{i,1,N} \\ \vdots & \ddots & \vdots \\ R_t^{i,N,1} & \dots & R_t^{i,N,N} \end{bmatrix}, \quad S_t^i := \begin{bmatrix} S_t^{i,1} \\ \vdots \\ S_t^{i,N} \end{bmatrix}, \quad r_t^i := \begin{bmatrix} r_t^{i,1} \\ \vdots \\ r_t^{i,N} \end{bmatrix} \tag{4.21}$$

We additionally make use of the following matrix terms for brevity, which combine components from the cost functionals of all players:

$$
\begin{aligned}
R_t &:= \begin{bmatrix} R_t^{1,1,1} & \dots & R_t^{1,1,N} \\ R_t^{2,2,1} & \dots & R_t^{2,2,N} \\ \vdots & \ddots & \vdots \\ R_t^{N,N,1} & \dots & R_t^{N,N,N} \end{bmatrix}, \quad S_{x_t} := \begin{bmatrix} S_t^{1,1} \\ S_t^{2,2} \\ \vdots \\ S_t^{N,N} \end{bmatrix}, \quad r_t := \begin{bmatrix} r_t^{1,1} \\ r_t^{2,2} \\ \vdots \\ r_t^{N,N} \end{bmatrix}, \\
Q_t &:= \begin{bmatrix} Q_t^1 \\ \vdots \\ Q_t^N \end{bmatrix}, \quad S_{u_t} := \begin{bmatrix} S_t^{1\intercal} \\ \vdots \\ S_t^{N\intercal} \end{bmatrix}, \quad q_t := \begin{bmatrix} q_t^1 \\ \vdots \\ q_t^N \end{bmatrix}, \\
\tilde{R}_t &:= \begin{bmatrix} \begin{matrix} R_t^{1,2,1} & \dots & R_t^{1,2,N} \\ \vdots & \ddots & \vdots \\ R_t^{1,N,1} & \dots & R_t^{1,N,N} \end{matrix} & & \\ & \ddots & \\ & & \begin{matrix} R_t^{N,1,1} & \dots & R_t^{N,1,N} \\ \vdots & \ddots & \vdots \\ R_t^{N,N-1,1} & \dots & R_t^{N,N-1,N} \end{matrix} \end{bmatrix}, \tilde{S}_{x_t} := \begin{bmatrix} S_t^{1,2} \\ \vdots \\ S_t^{1,N} \\ S_t^{2,1} \\ S_t^{2,3} \\ \vdots \\ S_t^{2,N} \\ \vdots \\ S_t^{N,N-1} \end{bmatrix}
\end{aligned}
\tag{4.22}
$$

We impose the regularity assumptions

$$
R_t^{i,i,i} \succ 0, \quad R_t^i \succeq 0, \quad Q_t^i \succeq 0 \tag{4.23}
$$

to ensure that the objective of each player is strictly convex. These conditions are sufficient for any solution to the conditions Eq. (4.12) to constitute a GFNE, as stated in Theorem 4.2.3, but not necessary.

The constraints imposed upon each player take the form

$$
\begin{aligned}
0 &= H_{x_t}^i x_t + H_{u_t^1}^i u_t^1 + \dots + H_{u_t^N}^i u_t^N + h_t^i, \quad t \in \mathbf{T} \\
0 &= H_{x_{T+1}}^i x_{T+1} + h_{T+1}^i,
\end{aligned}
\tag{4.24}
$$

for matrices $H_{x_t}^i \in \mathbb{R}^{a_t^i \times n}$, $H_{u_t^j}^i \in \mathbb{R}^{a_t^i \times m_t^j}$, and vectors $h_t^i \in \mathbb{R}^{a_t^i}$, where a_t^i is the dimension of the equality constraint imposed on player i at stage $t \in \mathbf{T}^+$. As in Section 4.2, we associate multipliers $\mu_t^i \in \mathbb{R}^{a_t^i}$, $t \in \mathbf{T}^+$ with these constraints for each player $i \in \mathbf{N}$. Let

$$\begin{aligned}
H_{u_t} &:= \begin{bmatrix} H^1_{u_t^1} & \dots & H^1_{u_t^N} \\ \vdots & \ddots & \vdots \\ H^N_{u_t^1} & \dots & H^N_{u_t^N} \end{bmatrix}, & \hat{H}_{u_t} &:= \begin{bmatrix} H^1_{u_t^1} & & \\ & \ddots & \\ & & H^N_{u_t^N} \end{bmatrix}, \\
H_{x_t} &:= \begin{bmatrix} H^1_{x_t} \\ \vdots \\ H^N_{x_t} \end{bmatrix}, & \hat{H}_{x_t} &:= \begin{bmatrix} H^1_{x_t} & & \\ & \ddots & \\ & & H^N_{x_t} \end{bmatrix}, \\
h_t^\intercal &:= \begin{bmatrix} (h_t^1)^\intercal & \dots & (h_t^N)^\intercal \end{bmatrix}, & \mu_t^\intercal &:= \begin{bmatrix} (\mu_t^1)^\intercal & \dots & (\mu_t^N)^\intercal \end{bmatrix}, \\
\tilde{H}_{u_t} &:= \begin{bmatrix} H^1_{u_t^2} & \dots & H^1_{u_t^N} & & & \\ & & & \ddots & & \\ & & & & H^N_{u_t^1} & \dots & H^N_{u_t^{N-1}} \end{bmatrix}
\end{aligned} \tag{4.25}$$

Due to the linearity of all dynamic and non-dynamic constraint functions appearing in the game, and the quadratic cost functionals, the solutions of the conditions Eq. (4.12) and Eq. (4.14) will be identical, as stated in Section 4.2. Therefore we will use the terms K_t and $\nabla_{x_t}\pi_t(x_t)$ interchangeably in this section.

Using the above-defined dynamic, constraint and cost terms, we are able to proceed with development of numerical methods for computing GFNE solutions to this game. Instead of taking a dynamic programming perspective as is, for example, taken in the classic derivation of Feedback Nash Equilibria for unconstrained LQ games in [19, Chapter 6], we derive our method using what we refer to as a *dynamic matrix factorization*. The primary idea behind this derivation is simply that the computation used to evaluate K_{t+1} can be reused to compute K_t efficiently.

To start, note that the conditions Eq. (4.14) for stage $t = T$ can be expressed in terms of the following matrix system:

$$\begin{bmatrix} R_T & -\hat{H}^\intercal_{u_T} & \hat{B}^\intercal_T & & & \\ H_{u_T} & & & & & \\ -B_T & & & I_n & & \\ & & -I_{N*n} & Q_{T+1} & \hat{H}^\intercal_{x_{T+1}} \\ & & & H_{x_{T+1}} & \end{bmatrix} \begin{bmatrix} u_T \\ \mu_T \\ \lambda_T \\ x_{T+1} \\ \mu_{T+1} \end{bmatrix} + \begin{bmatrix} S_{x_t} \\ H_{x_T} \\ -A_T \\ 0 \\ 0 \end{bmatrix} x_T + \begin{bmatrix} r_t \\ h_T \\ -c_T \\ q_{T+1} \\ h_{T+1} \end{bmatrix} = 0 \tag{4.26}$$

In Eq. (4.26), the matrices $I_\square$ denote the $\square \times \square$-dimensional identity matrix. Letting the system in Eq. (4.26) be denoted in shorthand as

$$M_T z_T + N_T x_T + n_T = 0, \tag{4.27}$$

where $z_T := [u_T^\intercal\ \mu_T^\intercal\ \lambda_T^\intercal\ x_{T+1}^\intercal\ \mu_{T+1}^\intercal]^\intercal$, we have that

$$
\begin{aligned}
\pi(x_T) &:= K_T x_T + k_T, \\
\lambda_T &:= K_{\lambda_T} x_T + k_{\lambda_T}, \\
\mu_T &:= K_{\mu_T} x_T + k_{\mu_T}, \\
\mu_{T+1} &:= K_{\mu_{T+1}} x_T + k_{\mu_{T+1}}, \\
K_T &:= -[M_T^{-1} N_T]_{u_T}, \quad && k_T := -[M_T^{-1} n_T]_{u_T}, \\
K_{\lambda_T} &:= -[M_T^{-1} N_T]_{\lambda_T}, \quad && k_{\lambda_T} := -[M_T^{-1} n_T]_{\lambda_T}, \\
K_{\mu_T} &:= -[M_T^{-1} N_T]_{\mu_T}, \quad && k_{\mu_T} := -[M_T^{-1} n_T]_{\mu_T}, \\
K_{\mu_{T+1}} &:= -[M_T^{-1} N_T]_{\mu_{T+1}}, \quad && k_{\mu_{T+1}} := -[M_T^{-1} n_T]_{\mu_{T+1}},
\end{aligned}
\tag{4.28}
$$

For any stage t, we also make use of the matrix

$$
\Pi_t^\intercal := \begin{bmatrix}
K_t^{2\intercal} & \cdots & K_t^{N\intercal} & & & & & & & & \\
& & & K_t^{1\intercal} & K_t^{3\intercal} & \cdots & K_t^{N\intercal} & & & & \\
& & & & & & & \ddots & & & \\
& & & & & & & & K_t^{1\intercal} & \cdots & K_t^{(N-1)\intercal}
\end{bmatrix}^\intercal
\tag{4.29}
$$

Notice that if we denote the conditions Eq. (4.14) for some stage $t+1$ as

$$
M_{t+1} z_{t+1} + N_{t+1} x_{t+1} + n_{t+1} = 0, \tag{4.30}
$$

as we did for $t+1 = T$, then we have also that the conditions Eq. (4.14) for stage t can be denoted as

$$
M_t z_t + N_t x_t + n_t = 0, \tag{4.31}
$$

where $z_t := [u_t^\intercal\ \mu_t^\intercal\ \lambda_t^\intercal\ \psi_{t+1}^\intercal\ x_{t+1}^\intercal\ z_{t+1}^\intercal]^\intercal$, and the matrices M_t, N_t, and n_t are defined as follows:

$$
\begin{aligned}
M_t &:= \begin{bmatrix} D_t^1 & D_t^2 \\ \begin{bmatrix} 0 & N_{t+1} \end{bmatrix} & M_{t+1} \end{bmatrix} \\
D_t^1 &:= \begin{bmatrix}
R_t & -\hat{H}_{u_t}^\intercal & \hat{B}_t^\intercal & & \\
H_{u_t} & & & & \\
-B_t & & & & I_n \\
& & -I_{N*n} & \Pi_t^\intercal & Q_{t+1} \\
& & & -I_{(N-1)*m_{t+1}} & \tilde{S}_{x_{t+1}}
\end{bmatrix}, \\
D_t^2 &:= \begin{bmatrix}
& & & & \\
& & & & \\
& & & & \\
S_{u_{t+1}} & -\hat{H}_{x_{t+1}}^\intercal & \hat{A}_{t+1}^\intercal & & \\
\tilde{R}_{t+1} & -\tilde{H}_{u_{t+1}}^\intercal & \tilde{B}_{t+1}^\intercal & &
\end{bmatrix}, \\
N_t^\intercal &:= \begin{bmatrix} S_{x_t}^\intercal & H_{x_t}^\intercal & -A_t^\intercal & \end{bmatrix}, \quad n_t^\intercal := \begin{bmatrix} r_t^\intercal & h_t^\intercal & -c_t^\intercal & 0 & 0 & n_{t+1}^\intercal \end{bmatrix}.
\end{aligned}
\tag{4.32}
$$

From this form, we have as before, that

$$
\begin{aligned}
\pi(x_t) &:= K_t x_t + k_t, \\
\lambda_t &:= K_{\lambda_t} x_t + k_{\lambda_t}, \\
\mu_t &:= K_{\mu_t} x_t + k_{\mu_t}, \\
\psi_{t+1} &:= K_{\psi_{t+1}} x_t + k_{\psi_{t+1}}, \\
K_t &:= -[M_t^{-1} N_t]_{u_t}, \quad k_t := -[M_t^{-1} n_t]_{u_t}, \\
K_{\lambda_t} &:= -[M_t^{-1} N_t]_{\lambda_t}, \quad k_{\lambda_t} := -[M_t^{-1} n_t]_{\lambda_t}, \\
K_{\psi_{t+1}} &:= -[M_t^{-1} N_t]_{\psi_{t+1}}, \quad k_{\psi_{t+1}} := -[M_t^{-1} n_t]_{\psi_{t+1}}, \\
K_{\mu_t} &:= -[M_t^{-1} N_t]_{\mu_t}, \quad k_{\mu_t} := -[M_t^{-1} n_t]_{\mu_t},
\end{aligned}
\tag{4.33}
$$

The advantage of expressing our system in the form Eq. (4.32) is that the computation performed to solve K_{t+1} and k_{t+1} can be reused to solve K_t and k_t. Using the block form of M_t defined in Eq. (4.32), and presuming that M_{t+1} is non-singular, we have from [165]:

$$
\begin{aligned}
[M_t^{-1}[N_t \; n_t]]_{u_t,\mu_t,\lambda_t,\psi_t,x_{t+1}} &= \begin{bmatrix} P_t^1 & P_t^2 \end{bmatrix} \begin{bmatrix} N_t & n_t \end{bmatrix}, \\
&= \begin{bmatrix} P_t^1 N_t, & P_t^1 \begin{bmatrix} r_t \\ h_t \\ -c_t \\ \\ \end{bmatrix} + P_t^2 n_{t+1} \end{bmatrix}
\end{aligned}
\tag{4.34}
$$

where the matrices P_t^1 and P_t^2 are defined as

$$
\begin{aligned}
P_t^1 &:= \left(D_t^1 - D_t^2 M_{t+1}^{-1}[0 \; N_{t+1}]\right)^{-1} \\
P_t^2 &:= -P_t^1 D_t^2 M_{t+1}^{-1}
\end{aligned}
\tag{4.35}
$$

Substituting in the form of the matrices in Eq. (4.32), we have that

$$
\begin{aligned}
P_t^1 &:= \begin{bmatrix} R_t & -\hat{H}_{u_t}^\intercal & \hat{B}_t^\intercal & & \\ H_{u_t} & & & & \\ -B_t & & & & I_n \\ & & -I_{N*n} & \Pi_t^\intercal & \hat{P}_t^{1,a} \\ & & & -I_{(N-1)*m_{t+1}} & \hat{P}_t^{1,b} \end{bmatrix}^{-1}, \\
\hat{P}_t^{1,a} &:= Q_{t+1} - S_{u_{t+1}} K_{t+1} + \hat{H}_{x_{t+1}}^\intercal K_{\mu_{t+1}} - \hat{A}_{t+1}^\intercal K_{\lambda_{t+1}}, \\
\hat{P}_t^{1,b} &:= \tilde{S}_{x_{t+1}} - \tilde{R}_{t+1} K_{t+1} + \tilde{H}_{u_{t+1}}^\intercal K_{\mu_{t+1}} - \tilde{B}_{t+1}^\intercal K_{\lambda_{t+1}}, \\
P_t^2 n_{t+1} &:= -P_t^1 \begin{bmatrix} \\ \\ \\ S_{u_{t+1}} k_{t+1} - \hat{H}_{x_{t+1}}^\intercal k_{\mu_{t+1}} + \hat{A}_{t+1}^\intercal k_{\mu_{t+1}} \\ \tilde{R}_{t+1} k_{t+1} - \tilde{H}_{u_{t+1}}^\intercal k_{\mu_{t+1}} + \tilde{B}_{t+1}^\intercal k_{\lambda_{t+1}} \end{bmatrix}.
\end{aligned}
\tag{4.36}
$$

From the above, it can be seen that the entire matrix inverse M_t^{-1} does not need to be computed for any stage t (other than the terminal stage T), and the factorization presented here allows computation of the entire GFNE trajectory and associated multipliers very efficiently. In particular, the overall computational complexity of solving this system is $O(T \cdot ((N+1) \cdot (n+m))^2 \cdot (n+1))$ time due to the dominating cost of at each stage solving the system of equations of the form $P_t^1 W_t$, where $(P_t^1)^{-1}$ is a square matrix of width no greater than $(N+1) \cdot (n+m)$, and W_t is some matrix with $n+1$ columns.

After computing the terms Eq. (4.33) for all stages t using the procedure above, the resultant GFNE trajectory and associated multipliers can be extracted:

$$
\begin{aligned}
x_1^* &:= \hat{x}_1, & \\
u_s^* &:= K_s x_s^* + k_s, & s \in \mathbf{T}, \\
\mu_s &:= K_{\mu_s} x_s^* + k_{\mu_s}, & s \in \mathbf{T}, \\
\lambda_s &:= K_{\lambda_s} x_s^* + k_{\lambda_s}, & s \in \mathbf{T}, \\
\psi_s &:= K_{\psi_s} x_{s-1}^* + k_{\psi_s}, & s \in \mathbf{T_2}, \\
x_{s+1}^* &:= A_s x_s^* + B_s u_s^* + c_s, & s \in \mathbf{T}, \\
\mu_{T+1} &:= K_{\mu_{T+1}} x_T + k_{\mu_{T+1}}. &
\end{aligned}
\tag{4.37}
$$

4.4 Inequality-Constrained LQ Games

We now extend the basic results presented in Section 4.3 on the computation of GFNE for equality-constrained LQ games, to the computation of GFNE for inequality-constrained LQ games. The approach we take here is that of an active-set method, analogous to active-set methods for quadratic programming (see, e.g. [192]). This method is simply a particular implementation of the approach presented in Algorithm 4, tailored for the structure of objective and constraint functions appearing in these feedback Nash equilibrium games. Furthermore, instead of using an LCP solver to solve each local equilibrium problem subject to a piece of piecewise linear constraint, the proposed active-set method enables the decision variables associated with multiple levels to change even if the decision variables at deeper levels of the $EPNEC$ have not yet satisfied their equilibrium conditions.

Consider a dynamic game among N players over T stages, with linear dynamics described by Eq. (4.18), and quadratic cost functionals Eq. (4.20). Assume that each player is also subject to linear equality constraints of the form Eq. (4.24), along with linear inequality constraints of the form

$$
\begin{aligned}
&0 \leq G^i_{x_t} x_t + G^i_{u_t^1} u_t^1 + ... + G^i_{u_t^N} u_t^N + g_t^i, \quad t \in \mathbf{T} \\
&0 \leq G^i_{x_{T+1}} x_{T+1} + g^i_{T+1},
\end{aligned}
\tag{4.38}
$$

for matrices $G^i_{x_t} \in \mathbb{R}^{b_t^i \times n}$, $G^i_{u_t^j} \in \mathbb{R}^{b_t^i \times m_t^j}$, and vectors $g_t^i \in \mathbb{R}^{b_t^i}$, where b_t^i is the dimension of the inequality constraint imposed on player i at stage $t \in \mathbf{T}^+$. As in Section 4.2, we associate multipliers $\gamma_t^i \in \mathbb{R}^{b_t^i}$, $t \in \mathbf{T}^+$ with these constraints for each player $i \in \mathbf{N}$. Assume that a solution to the system Eq. (4.12) exists for this game at stage $t = 1$, and that strict complementarity holds for the conditions Eq. (4.12) for the subgame starting at every $t \in \mathbf{T}$ along any solution.

The method we present for computing a GFNE of this game is an adaptation of Algorithm 16.3 in [192] to the current setting. Under the strict complementarity assumption (which ensures differentiability of the policies π_t along the solution), we have that at any GFNE solution, some subset of the constraints Eq. (4.38) associated with strictly positive multipliers hold with equality at the solution. If the set of active constraints along some solution were known in advance, we could consider all active constraints as equality constraints, ignore all inactive constraints, and solve for the resultant equality-constrained game using the method presented in Section 4.3. In general the set of active constraints along a solution is obviously unknown in advance. The active-set method we propose accounts for this by iteratively solving for the unique GFNE solution for different guesses of the active constraint set, and uses dual variable information to update the guess of the active set. In the remainder of this section we describe the proposed method. The presentation of this section is based off of section 16.5 in [192], with necessary modifications made to account for the multiplayer feedback setting considered here.

The method begins with a feasible initialization for the game (defined by the linear dynamics Eq. (4.18), equality constraints Eq. (4.24), and inequality constraints Eq. (4.38), and the quadratic cost functionals Eq. (4.20)). We denote the set of all primal variables associated with the game at the kth iteration of the method by $\mathbf{X}_k := [x_{(1:T+1),k}, u_{(1:T),k}]$. Also associated with the kth iteration of the algorithm is the working set $\mathcal{W}_k$ which denotes the set of constraints which are treated with equality at the kth iteration. Note that the working set $\mathcal{W}_k$ always contains all of the equality constraints Eq. (4.24). The working set $\mathcal{W}_1$ is taken to be a subset of the constraints active along the initialization $\mathbf{X}_1$.

Given an iterate $\mathbf{X}_k$ and working set $\mathcal{W}_k$, a step $\mathbf{P}_k := [p_{x_{1:T+1}}, p_{u_{1:T}}]$ is found which moves $\mathbf{X}_k$ to the GFNE associated with the working set of equality constraints in $\mathcal{W}_k$. Specifically, the problem to be solved at each iteration is the GFNE problem for the equality-constrained LQ game defined by the stage-wise cost functionals for each player

$$
\begin{aligned}
l_{t,k}^i(p_{x_t}, p_{u_t}) &:= \frac{1}{2}\left(\begin{bmatrix} p_{x_t} \\ p_{u_t} \end{bmatrix}^\intercal \begin{bmatrix} Q_t^i & S_t^{i\intercal} \\ S_t^i & R_t^i \end{bmatrix} \begin{bmatrix} p_{x_t} \\ p_{u_t} \end{bmatrix} + 2 \begin{bmatrix} p_{x_t} \\ p_{u_t} \end{bmatrix}^\intercal \begin{bmatrix} q_{t,k}^i \\ r_{t,k}^i \end{bmatrix} \right), && t \in \mathbf{T} \\
q_{t,k}^i &:= Q_t^i x_{t,k} + S_t^{i\intercal} u_{t,k} + q_t^i, && t \in \mathbf{T}, \\
r_{t,k}^i &:= S_t^i x_{t,k} + R_t^i u_{t,k} + r_t^i, && t \in \mathbf{T}, \\
l_{T+1,k}^i(p_{x_{T+1}}) &:= \frac{1}{2}\left(p_{x_{T+1}}^\intercal Q_{T+1}^i p_{x_{T+1}} + 2 p_{x_{T+1}}^\intercal q_{T+1,k}^i \right), \\
q_{T+1,k}^i &:= Q_{T+1}^i x_{T+1,k} + q_{T+1}^i,
\end{aligned} \tag{4.39}
$$

the dynamics

$$
p_{x_{t+1}} = A_t p_{x_t} + B_t^1 p_{u_t^1} + ... + B_t^N p_{u_t^N}, \quad t \in \mathbf{T}, \tag{4.40}
$$

and the linear equality constraints

$$
\begin{aligned}
0 &= H^i_{(x_t,k)} p_{x_t} + H^i_{(u^1_t,k)} p_{u^1_t} + \dots + H^i_{(u^N_t,k)} p_{u^N_t}, \quad t \in \mathbf{T} \\
0 &= H^i_{(x_{T+1},k)} p_{x_{T+1}}.
\end{aligned} \tag{4.41}
$$

Above, the matrices $H^i_{(x_t,k)}$ and $H^i_{(u^j_t,k)}$ are defined to be the set of active equality constraint coefficients corresponding to $\mathcal{W}_k$:

$$
H^i_{(x_t,k)} := \begin{bmatrix} H^i_{x_t} \\ \hline \vdots \\ \{G^{i,j}_{x_t}\}_{(t,i,j)\in\mathcal{W}_k\cap\mathcal{I}} \\ \vdots \end{bmatrix}, H^i_{(u_t,k)} := \begin{bmatrix} H^i_{u_t} \\ \hline \vdots \\ \{G^{i,j}_{u_t}\}_{(t,i,j)\in\mathcal{W}_k\cap\mathcal{I}} \\ \vdots \end{bmatrix}. \tag{4.42}
$$

The set $\mathcal{W}_k \cap \mathcal{I}$ is the index set of all active inequality constraints, and $G^{i,j}_{x_t}$ is the jth row of the matrix $G^i_{x_t}$.

Associated with the constraints Eq. (4.41) are multipliers $\mu^i_{(t,k)}$, defined as

$$
\mu^i_{(t,k)} := \begin{bmatrix} \mu^i_t \\ \hline \vdots \\ \{\gamma^{i,j}_t\}_{(t,i,j)\in\mathcal{W}_k\cap\mathcal{I}} \\ \vdots \end{bmatrix} \tag{4.43}
$$

where $\gamma^{i,j}_t$ is the jth element of γ^i_t.

After solving for $\mathbf{P}_k$, the GFNE of the resultant equality-constrained LQ game, $\mathbf{X}_k+\mathbf{P}_k$ is the GFNE for the equality-constrained LQ game defined by Eq. (4.18), Eq. (4.20), Eq. (4.24), and the active constraints Eq. (4.38) in $\mathcal{W}_k\cap\mathcal{I}$. However, it may be that $\mathbf{X}_k+\mathbf{P}_k$ is infeasible with respect to the entire set of inequality constraints Eq. (4.38). Therefore we instead find the point $\mathbf{X}_k + \beta_k\mathbf{P}_k$, where

$$
\begin{aligned}
\beta_k := \max_{\beta\in[0,1]} \beta \\
\text{s.t. } \mathbf{X}_k + \beta\mathbf{P}_k \text{ feasible w.r.t. Eq. (4.38).}
\end{aligned} \tag{4.44}
$$

The optimization in Eq. (4.44) is a linear program, and β_k can be computed exactly and efficiently. When $\beta_k < 1$, it implies that there is an inequality constraint not considered in the working set which must be accounted for. When this is the case, the iterate $\mathcal{X}_{k+1}$ is updated to the point $\mathbf{X}_k + \beta_k\mathbf{P}_k$, and the working set is updated to include the blocking constraint. If instead, $\beta_k = 1$, then the point $\mathbf{X}_k + \mathbf{P}_k$ both is a GFNE solution for the working set and is feasible with respect to all equality and inequality constraints. All that is left to check is whether the constraints $\mu^{i,j}_{(t,k)} > 0$ for all $j > a^i_t$, meaning the complementarity conditions in Eq. (4.12) are satisfied, and therefore a solution satisfying the entire set of conditions Eq. (4.12) for the inequality-constrained problem has been found. If some multiplier associated with the inequality constraints in the working set $\mathcal{W}_k$ is negative[2], then the corresponding constraint is to be dropped from the working set at the next iteration. Unlike the the convex quadratic programming setting, where forming $\mathcal{W}_{k+1}$ by dropping a constraint associated with a negative multiplier in the set $\mathcal{W}_k \cap \mathcal{I}$ and setting $\mathbf{X}_{k+1} = \mathbf{X}_k + \mathbf{P}_k$, the update $\mathbf{P}_{k+1}$ does not necessarily move away from the dropped constraint boundary. In such situations, the procedure fails to make progress, since the value of β_{k+1} in Eq. (4.44) will be 0. In practice these conditions can be treated by dropping a different constraint (also associated with a negative multiplier) from the working set. If no other constraints are associated with negative multipliers, a GFNE does not exist in the vicinity of the iterate, and failure is declared. The full procedure is stated in Algorithm 5.

4.5 Nonlinear Games

So far the focus of this dissertation has been on connected quadratic programs. In this section we propose a method for extending some of the previous results to the nonlinear setting, in which each connected optimization problem is a nonlinear optimization problem.

In particular, we outline a procedure for finding a solution to the conditions Eq. (4.14) for games defined by the dynamics Eq. (4.1), cost functionals Eq. (4.2), and constraints Eqs. (4.3a) and (4.3b). We assume that all functions appearing in the conditions Eq. (4.14) are continuously twice differentiable, with the exception of the implicitly defined policies. The procedure leverages the method for computing solutions to inequality-constrained LQ games presented in Section 4.4 and Algorithm 5, and generally is inspired by Sequential Quadratic Programming methods for non-convex numerical optimization (see e.g. [192], Chapter 18).

The foundation of this approach is in the observation that a Newton-style method can be used to finding a solutions to the conditions Eq. (4.14), where each iteration involves solving for a GFNE for the locally approximate LQ game formed around the current iterate. Considering first the case in which the game does not include any inequality constraints, computing a search direction using Newton's method on the conditions Eq. (4.14) at $t = 1$ can be seen to be equivalent to computing a search direction by solving an LQ approximation of the game. In the inequality-constrained case, search directions can be found by solving an inequality-constrained LQ approximation, analogous to the method in [192].

In particular, we propose an iterative method for finding solutions to the conditions Eq. (4.14). Note that throughout this section, iterations are again indexed by $k = 1, 2, 3, ...$, as they were in the method for computing inequality-constrained LQ games in Section 4.4. Here, let the current iterate of the primal and dual game variables at iteration k be denoted as

$$\begin{aligned} \mathbf{X}_k &:= [x_{(1:T+1),k}, u_{(1:T),k}], \\ \mathbf{\Lambda}_k &:= [\lambda_{(1:T),k}, \mu_{(1:T+1),k}, \gamma_{(1:T+1),k}, \psi_{(2:T),k}] \end{aligned} \tag{4.45}$$

[2]Note that the multipliers cannot be zero by the strict complementarity assumption made.

Algorithm 5 Active Set Inequality Constrained LQ Game GFNE Solver

Start with $\mathbf{X}_1$ feasible w.r.t. Eq. (4.18), Eq. (4.24), Eq. (4.38)
Initialize $\mathcal{W}_1$ to be subset of active inequality constraints along $\mathbf{X}_1$
for k=1,2,3,... **do**
    Solve equality-constrained LQ GFNE defined by Eq. (4.39), Eq. (4.40), Eq. (4.41), and denote the solution as $\mathbf{P}_k$
    if $\mathbf{P}_k == \mathbf{0}$ **then**
        Extract multipliers $\mu^i_{(t,k)}$, ψ^i_t, λ^i_t, using Eq. (4.37)
        if $\mu^{i,j}_{(t,k)} > 0, \ \forall j > a^i_t$ (Inequality constraint multipliers) **then return** $\mathbf{X}_k$,
        else
            $(t_m, i_m, j_m) \leftarrow \underset{(t,i,j)\in\mathcal{W}_k\cap\mathcal{I}}{\operatorname{argmin}} \mu^{i,j}_{(t,k)}$
            $\mathbf{X}_{k+1} \leftarrow \mathbf{X}_k$
            $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{(t_m, i_m, j_m)\}$
        end if
    else
        Find largest $\beta_k \in [0, 1]$ such that $\mathbf{X}_k + \beta_k\mathbf{P}_k$ is feasible w.r.t. Eq. (4.38)
        $\mathbf{X}_{k+1} \leftarrow \mathbf{X}_k + \beta_k\mathbf{P}_k$
        if $\beta_k < 1$ **then**
            $(t_b, i_b, j_b) \leftarrow$ index of blocking inequality constraint not already in $\mathcal{W}_k \cap \mathcal{I}$
            if $(t_b, i_b, j_b) == (t_m, i_m, j_m)$ (Blocking index is previously dropped index) **then**
                Choose other blocking constraint or declare failure
            else
                $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup (t_b, i_b, j_b)$
            end if
        else
            $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$
        end if
    end if
end for

Assume some initialization of all variables $\mathbf{X}_1$ and $\mathbf{\Lambda}_1$. Then for each iteration k, a search direction $\mathbf{P}_k$ is found by solving the inequality-constrained LQ game formed in the following way.

Let the dynamics Eq. (4.18) for the approximate game at the kth iteration be defined by the terms

$$\begin{aligned} A_{t,k} &:= \nabla_x f_t(x_{t,k}, u_{t,k}), \\ B^i_{t,k} &:= \nabla_{u^i} f_t(x_{t,k}, u_{t,k}), \\ c_{t,k} &:= f_t(x_{t,k}, u_{t,k}) - x_{t+1,k}. \end{aligned} \tag{4.46}$$

Similarly, let the equality and inequality constraint terms in Eqs. (4.24) and (4.38) be defined as

$$\begin{aligned} H^i_{x_t,k} &:= \nabla_x h^i_t(x_{t,k}, u_{t,k}), & G^i_{x_t,k} &:= \nabla_x g^i_t(x_{t,k}, u_{t,k}), \\ H^i_{u^i_t,k} &:= \nabla_{u^i} h^i_t(x_{t,k}, u_{t,k}), & G^i_{u^i_t,k} &:= \nabla_{u^i} g^i_t(x_{t,k}, u_{t,k}), \\ h^i_{t,k} &:= h^i_t(x_{t,k}, u_{t,k}), & g^i_{t,k} &:= g^i_t(x_{t,k}, u_{t,k}). \end{aligned} \tag{4.47}$$

Finally, let the cost functional coefficients in Eq. (4.20) for stages $t \in \mathbf{T}$ be defined as

$$\begin{aligned} Q^i_{t,k} &:= \nabla^2_{x,x} l^i_t(x_{t,k}, u_{t,k}) + (\nabla^2_{x,x} f_t)^\intercal \lambda^i_{t,k} - (\nabla^2_{x,x} h^i_t)^\intercal \mu^i_{t,k} - (\nabla^2_{x,x} g^i_t)^\intercal \gamma^i_{t,k}, \\ S^i_{t,k} &:= \nabla^2_{u,x} l^i_t(x_{t,k}, u_{t,k}) + (\nabla^2_{u,x} f_t)^\intercal \lambda^i_{t,k} - (\nabla^2_{u,x} h^i_t)^\intercal \mu^i_{t,k} - (\nabla^2_{u,x} g^i_t)^\intercal \gamma^i_{t,k}, \\ R^i_{t,k} &:= \nabla^2_{u,u} l^i_t(x_{t,k}, u_{t,k}) + (\nabla^2_{u,u} f_t)^\intercal \lambda^i_{t,k} - (\nabla^2_{u,u} h^i_t)^\intercal \mu^i_{t,k} - (\nabla^2_{u,u} g^i_t)^\intercal \gamma^i_{t,k}, \\ q^i_{t,k} &:= \nabla_x l^i_t(x_{t,k}, u_{t,k}), \\ r^i_{t,k} &:= \nabla_u l^i_t(x_{t,k}, u_{t,k}). \end{aligned} \tag{4.48}$$

The solution to this inequality-constrained LQ game at each iteration k yields the search direction $\mathbf{P}_k$ and multipliers $\bar{\mathbf{\Lambda}}_{k+1}$. To ensure progress towards a solution of the conditions Eq. (4.14), a line-search procedure is invoked. We seek a parameter $\alpha_k \in [0, 1]$ such that the iterates

$$\begin{aligned} \mathbf{X}_{k+1} &:= \mathbf{X}_k + \alpha_k \mathbf{P}_k, \\ \mathbf{\Lambda}_{k+1} &:= \mathbf{\Lambda}_k + \alpha_k (\bar{\mathbf{\Lambda}}_{k+1} - \mathbf{\Lambda}_k) \end{aligned} \tag{4.49}$$

make maximal progress towards satisfying the conditions Eq. (4.14), with respect to an appropriate merit function. Because the game setting involves multiple players, a decrease in the objective of all players' objectives cannot be guaranteed. Therefore the merit function we consider is simply the residual squared norm of the conditions Eq. (4.14). In particular, define the merit function to search over as the following:

$$
\begin{aligned}
\mathbf{M}(\mathbf{X},\mathbf{\Lambda}) := &\sum_{i\in\mathbf{N}}\sum_{s\in\mathbf{T}} \left\|\nabla_{u_s^i}\left[l_s^i + f_s^\intercal\lambda_s^i - h_s^{i\intercal}\mu_s^i - g_s^{i\intercal}\gamma_s^i\right]\right\|_2^2 + \\
&\sum_{i\in\mathbf{N}}\sum_{s\in\mathbf{T_2}} \left\|\nabla_{x_s}\left[l_s^i - \lambda_{s-1}^i + f_s^\intercal\lambda_s^i - h_s^{i\intercal}\mu_s^i - g_s^{i\intercal}\gamma_s^i + K_s^{-i\intercal}\psi_s^i\right]\right\|_2^2 + \\
&\sum_{i\in\mathbf{N}}\sum_{s\in\mathbf{T_2}} \left\|\nabla_{u_s^{-i}}\left[l_s^i + f_s^\intercal\lambda_s^i - h_s^{i\intercal}\mu_s^i - g_s^{i\intercal}\gamma_s^i - \psi_s^i\right]\right\|_2^2 + \\
&\sum_{i\in\mathbf{N}} \left\|\nabla_{x_{T+1}}\left[l_{T+1}^i - \lambda_T^i - h_{T+1}^{i\intercal}\mu_{T+1}^i - g_{T+1}^{i\intercal}\gamma_{T+1}^i\right]\right\|_2^2 + \\
&\sum_{s\in\mathbf{T}} \left\|x_{s+1} - f_s(x_s,u_s)\right\|_2^2 + \\
&\sum_{i\in\mathbf{N}}\Big(\sum_{s\in\mathbf{T}} \left\|h_s^i(x_s,u_s)\right\|_2^2 + \left\|h_{T+1}^i(x_{T+1})\right\|_2^2\Big) + \\
&\sum_{i\in\mathbf{N}}\Big(\sum_{s\in\mathbf{T}} \left\|\min(g_s^i(x_s,u_s),0)\right\|_2^2 + \left\|\min(g_{T+1}^i(x_{T+1}),0)\right\|_2^2\Big) + \\
&\sum_{i\in\mathbf{N}}\sum_{s\in\mathbf{T}^+} \left\|\min(\mu_s^i,0)\right\|_2^2 + |(g_s^i)^\intercal\mu_s^i|.
\end{aligned}
\tag{4.50}
$$

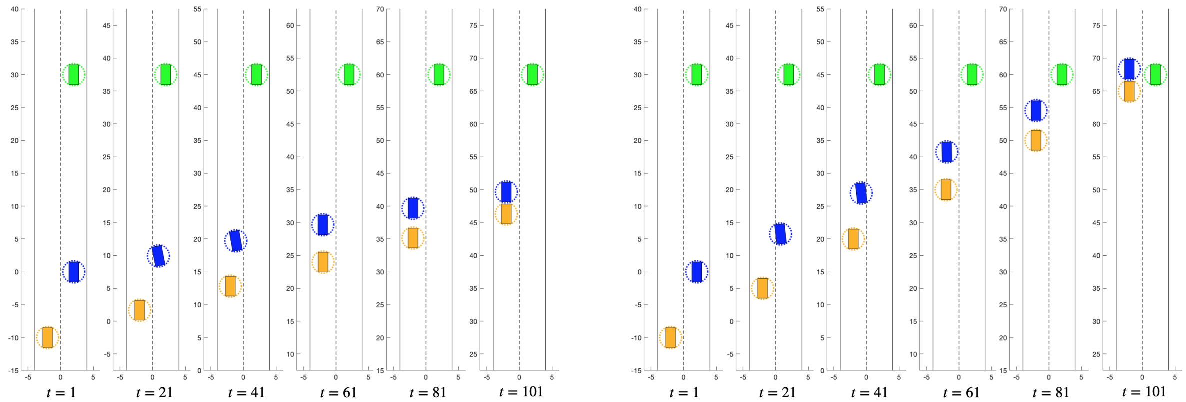
Recall that $\mathbf{T}_2 := \{2, ..., T\}$. Then α_k is defined as:

$$
\alpha_k := \min_{\alpha\in[0,1]} \mathbf{M}(\mathbf{X}_k + \alpha\mathbf{P}_k, \mathbf{\Lambda}_k + \alpha(\bar{\mathbf{\Lambda}}_{k+1} - \mathbf{\Lambda}_k)). \tag{4.51}
$$

Algorithm 6 GFQNE Solver for Nonlinear Games

Set convergence tolerance $\epsilon > 0$
Start with initial $\mathbf{X}_1$, $\mathbf{\Lambda}_1$
for k=1,2,3,... **do**
    Solve inequality-constrained LQ GFNE defined by Eq. (4.46), Eq. (4.47), Eq. (4.48), and denote the solution and corresponding multipliers as $\mathbf{P}_k$, $\bar{\mathbf{\Lambda}}_{k+1}$
    Find α_k according to Eq. (4.51) and Eq. (4.50) via backtracking line-search
    $\mathbf{X}_{k+1} \leftarrow \mathbf{X}_k + \alpha_k\mathbf{P}_k$
    $\mathbf{\Lambda}_{k+1} \leftarrow \mathbf{\Lambda}_k + \alpha_k(\bar{\mathbf{\Lambda}}_{k+1} - \mathbf{\Lambda}_k)$
    if $\mathbf{M}(\mathbf{X}_{k+1}, \mathbf{\Lambda}_{k+1}) < \epsilon$ **then**
        Return $\mathbf{X}_{k+1}, \mathbf{\Lambda}_{k+1}$ and break
    end if
end for

The choice of merit function need not be as defined in Eq. (4.50). Any positive-valued function which evaluates to 0 if and only if the arguments constitute a solution to Eq. (4.14) is



(a) $\sigma_{polite} = 0$ (b) $\sigma_{polite} = 5$

Figure 4.2: Snapshots of the resultant GFQNE solutions found to the example in Section 4.6, when excluding (a) and including (b) the politeness term in player 1's objective.

acceptable. Note that in the line-search procedure Eq. (4.51) using any such merit function, it is necessary in general to evaluate the policy quasi-gradients K_t at the candidate point $\mathbf{X}_k + \alpha\mathbf{P}_k$, $\mathbf{\Lambda}_k + \alpha(\bar{\mathbf{\Lambda}}_{k+1} - \mathbf{\Lambda}_k))$ for general α. However, recall that these terms are implicitly defined, and in general require solving the approximate LQ game defined at the candidate point to evaluate them. This makes the evaluation of Eq. (4.51) very expensive. Therefore, in practice, we find it acceptable in most cases to replace the policy quasi-gradients K_t appearing in Eq. (4.50) corresponding to the candidate point, with the quasi-gradients corresponding to the point $\mathbf{X}_k, \mathbf{\Lambda}_k$, which are evaluated in the computation of the search direction $\mathbf{P}_k$.

Even with the reuse of the policy quasi-gradients, the minimization in Eq. (4.51) cannot be carried out exactly. In practice, a backtracking line-search satisfying a sufficient decrease condition is used instead.

The complete algorithm for computing solutions to Eq. (4.14) for nonlinear games is given in Algorithm 6.

4.6 Example

We now demonstrate the methodologies so far presented on a practical example.

Consider a game describing a driving scenario involving an autonomous vehicle and two other vehicles on a freeway. Here $N = 3$, and let $T = 100$ denote the number of discrete time-points in the trajectory game. Let the game dynamics be defined as the concatenation of the independent dynamics of each vehicle in the game. We assume that each vehicle

Major Iteration	Minor Iteration	Working Set Indices	Comment	α_k	$\mathbf{M}(\mathbf{X}_{k+1}, \mathbf{\Lambda}_{k+1})$
1	F1	{}			
	F2	{(101,2)}			
	F3	{(101,1),(101,2)}			
	1	{(101,1),(101,2)}			
	2	{(101,2)}			
	3	{(101,1),(101,2)}	Cycle	1	96.25
2	F1	{(101,2)}			
	F2	{(101,1),(101,2)}			
	1	{(101,1),(101,2)}			
	2	{(101,2)}			
	3	{(101,1),(101,2)}	Cycle	1	55.65
3	F1	{(101,2)}			
	1	{(101,2)}	Solution	1	0.221
4	F1	{(101,2)}			
	1	{(101,2)}	Solution	1	2.3e-5
Total Time	Total LQ Solves	Solve Time	Function Eval Time		
5.14	17	0.97	4.17		

Table 4.2: Algorithm iterate information when using Algorithm 6 (Major Iterations) and Algorithm 5 (Minor Iterations) to compute a GFQNE to the example in Section 4.6, when $\sigma_{polite} = 5$ (Fig. 4.2b). Here $\mathbf{M}(\mathbf{X}_{k+1}, \mathbf{\Lambda}_{k+1})$ is the merit function value after performing a line search in the direction of $\mathbf{P}_k$ in Algorithm 6. In each major iteration of this solve, $\alpha_k = 1$, meaning no backtracking was necessary in the line search procedure. Here the minor iterations labeled "F□" indicate equality-constrained LQ solves used in the search of a feasible initial solution to the inequality-constrained LQ game associated with the major iteration. The detection of a cycle indicates that the removal of a constraint associated with a negative multiplier did not move the iterate associated with Algorithm 5 away from the dropped constraint boundary (Line 19). In these cases the iterate is accepted and the algorithm continues with the next major iteration.

follows simple unicycle dynamics. Specifically, for each $i \in \mathbf{N}$, let

$$x^i_{t+1} = \begin{bmatrix} x^{i,1}_{t+1} \\ x^{i,2}_{t+1} \\ x^{i,3}_{t+1} \\ x^{i,4}_{t+1} \end{bmatrix} = f^i_t(x^i_t, u^i_t) = \begin{bmatrix} x^{i,1}_t + \Delta \cdot x^{i,3}_t \cos(x^{i,4}_t) \\ x^{i,2}_t + \Delta \cdot x^{i,3}_t \sin(x^{i,4}_t) \\ x^{i,3}_t + \Delta \cdot u^{i,1}_t \\ x^{i,4}_t + \Delta \cdot u^{i,2}_t \end{bmatrix}. \tag{4.52}$$

Here Δ represents some small sampling time. The interpretation of the vehicles states and controls are the following: the first dimension is the longitudinal (along-lane) position, the second is the lateral (across-lane) position. The third dimension is the speed in the direction of the vehicle heading, and the fourth dimension is the vehicle heading in radians. The first control dimension is the vehicle acceleration, and the second dimension is the angular velocity.

The dynamics for the entire game state are then given as

$$x_{t+1} := \begin{bmatrix} f^1_t(x^1_t, u^1_t) \\ f^2_t(x^2_t, u^2_t) \\ f^3_t(x^3_t, u^3_t) \end{bmatrix} = f_t(x_t, u_t), \ t \in \mathbf{T}. \tag{4.53}$$

The goal of player 2 is to minimize its acceleration, angular velocity, and deviation from desired speed, while staying in-lane and avoiding collision with player 1. Similarly, the goal of player 3 is to minimize its acceleration, angular velocity, and deviation from desired speed, while staying in-lane.

The goal of player 1 is to complete a lateral lane change while minimizing its own acceleration, angular velocity, and deviation from its desired speed, and avoiding collision with player 3. Player 1 also attempts to minimize the objective of player 2 in addition to its own objective.

A depiction of this game is given in Fig. 4.2. The functions describing the objective and constraints of each player are the following:

$$\begin{aligned}
0 &= h^1_{T+1}(x_{T+1}) := x^{1,2}_{T+1} + 2, \\
0 &= h^2_{T+1}(x_{T+1}) := x^{2,2}_{T+1} + 2, \\
0 &= h^3_{T+1}(x_{T+1}) := x^{3,2}_{T+1} - 2, \\
0 &\le g^1_t(x_t) := \left\| \begin{matrix} x^{1,1}_t - x^{3,1}_t \\ x^{1,2}_t - x^{3,2}_t \end{matrix} \right\|_2 - d_{min}, \quad t \in \mathbf{T}^+, \\
0 &\le g^2_t(x_t) := \left\| \begin{matrix} x^{2,1}_t - x^{1,1}_t \\ x^{2,2}_t - x^{1,2}_t \end{matrix} \right\|_2 - d_{min}, \quad t \in \mathbf{T}^+,
\end{aligned} \tag{4.54}$$

$$
\begin{aligned}
L^1(x,u) &= \Big(\sum_{t=1}^{T} \sigma_1 \left\|u_t^1\right\|_2^2 + \sigma_2(x_t^{1,2}+2)^2 + \sigma_3(x_t^{1,3} - v_{goal}^1)^2\Big) + \sigma_{polite} L^2(x,u), \\
L^2(x,u) &= \sum_{t=1}^{T} \sigma_1 \left\|u_t^2\right\|_2^2 + \sigma_2(x_t^{2,2}+2)^2 + \sigma_3(x_t^{2,3} - v_{goal}^2)^2, \\
L^3(x,u) &= \sum_{t=1}^{T} \sigma_1 \left\|u_t^3\right\|_2^2 + \sigma_2(x_t^{3,2}-2)^2 + \sigma_3(x_t^{3,3} - v_{goal}^3)^2,
\end{aligned}
\tag{4.55}
$$

The initial state $\hat{x}_1$ is defined to be $\hat{x}_1 := [0,2,1,0,-10,-2,1.5,0,30,2,0.75,0]^\intercal$. The lateral center of the left lane is -2, and the lateral center of the right lane is 2. The constant d_{min} is the minimum separation distance from vehicle centers needed to avoid collision, which in this example is 3.3. The parameters $\sigma_{\{1,2,3\}}$ scale the relative weights between objective terms, and are $\sigma_1 = 10$, $\sigma_2 = 0.2$, and $\sigma_3 = 10$. The desired speeds of the three players are $v_{goal}^1 = 1$, $v_{goal}^2 = 1.5$, and $v_{goal}^3 = 0.75$. The term σ_{polite} is the politeness coefficient and weights how much player 1 cares about interfering with player 2's objective. We consider two variants, with $\sigma_{polite} = 0$ and $\sigma_{polite} = 5$. This example is similar to games explored in [150]. Visualizations of the GFQNE solutions for the two variants are given in Fig. 4.2, and computation details for the $\sigma_{polite} = 5$ case are given in Table 4.2 (details for the case $\sigma_{polite} = 0$ are omitted for brevity).

4.7 Conclusion

In this chapter, we have presented a non-parametric, implicit policy formulation for generalized feedback Nash equilibrium problems. We developed efficient solution methods for the equality-constrained Linear-Quadratic (LQ) case, the inequality-constrained LQ case, and the general nonlinear case. To the best of our knowledge, these constitute the first solution methods for finding Feedback Nash Equilibria both LQ and nonlinear games with general constraints. Dynamic games have numerous applications; we demonstrate the utility of our method in a trajectory planning setting for a lane-changing autonomous vehicle.

Future work should consider other solution methods for the general game which build upon our solution to the equality-constrained LQ game. In particular, penalty methods and interior point methods may also be competitive. Furthermore, the results presented should be extended to cases in which strict complementarity does not hold. Necessary conditions based on sub-differentials could be used in such cases. It is also important to further develop a deeper theoretical understanding of the "policy quasi-gradient" approximation and its implications on convergence to local solutions. Finally, a high-performance, optimized implementation of our method will facilitate its use in practical applications by other researchers and practitioners.

Chapter 5

Uncertainty in Game-Theoretic Motion Planning

In this chapter we consider some issues that arise from attempting to use game-theoretic motion planning techniques for autonomous driving prediction and planning, and leverage some benefits of the GFNE solutions (preseted in the previous chapter) to overcome them. In particular, some properties of the GFNE solution is used to account for the uncertainty present in assuming the underlying cost and constraint structure that other game players are optimizing for. The results in this section build on the non-quadratic objective, non-linear constraint formulation of the GFNE solutions, and therefore stray from the primary focus of this dissertation, but are included as an application (via nonlinear extension) of connected quadratic programs.

The contents of this chapter are primarily taken from [150], which is co-authored with David Fridovich-Keil, Chih-Yuan Chiu, and Claire Tomlin.

5.1 Introduction

Trajectory planning for autonomous agents often proceeds in a model-predictive control fashion, where trajectories are frequently re-planned as new information about the environment is collected. When operating in the presence of other agents, as in autonomous driving, motion plans must reason about predicted trajectories of those other agents. When those predictions are assumed fixed, the resultant motion plans of the "ego" vehicle are incapable of reasoning about the reactions of the other agents to the ego's own decisions.

Game-theoretic motion planners instead model the interaction with other agents directly, by handling planning and prediction jointly. That is, the intentions of all agents in the scene are encoded as optimization problems that they are each trying to solve, and an equilibrium solution for the collection of optimization problems is found. This equilibrium consists of a set of interacting trajectories of all the agents, which can be used as predictions for non-ego agents, and as a motion plan for the ego agent.

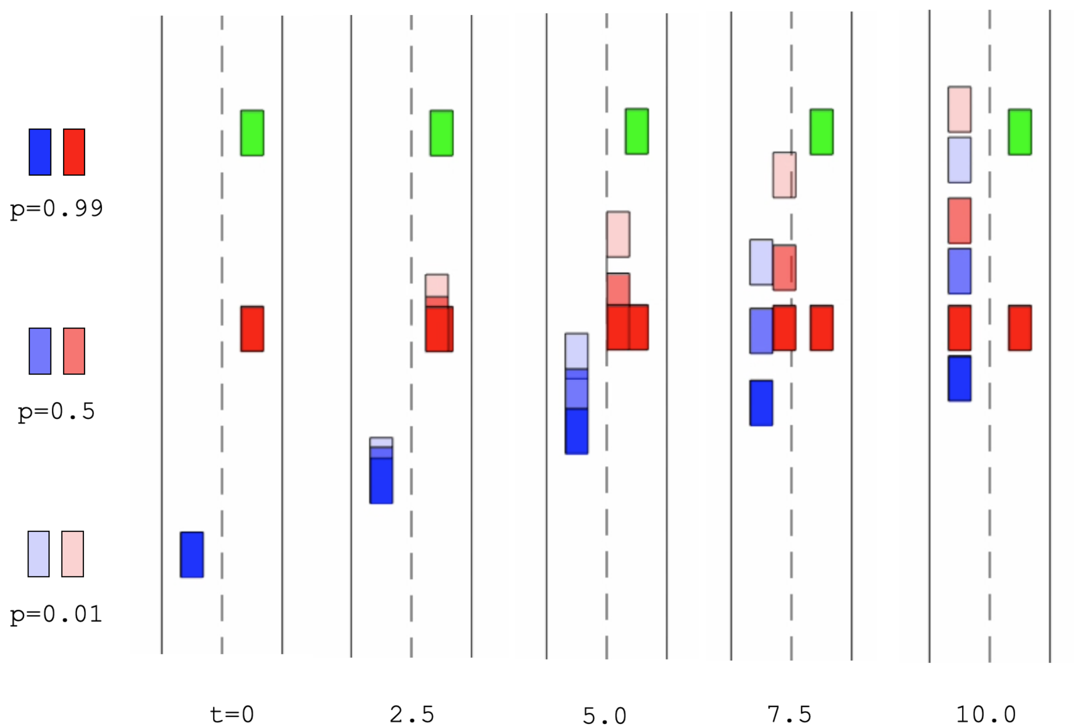


Figure 5.1: The fast-moving ego vehicle (blue) believes that the slow-moving (red) agent will change lanes with probability p. Our method constructs and solves a dynamic game involving both lane-changing and non-lane-changing versions of the red agent. Depicted here are three different solutions to this game, corresponding to different probabilities associated to the two hypotheses. When certain that the red agent will change lanes ($p \approx 1$), the ego agent takes full responsibility for avoiding collision with the lane-changing version of the red agent, allowing it to change lanes unimpeded. When certain the red agent will stay in-lane ($p \approx 0$), the lane-changing agent is required to take full responsibility for collision-avoidance with ego, which allows ego to pass at its original speed. Probabilities in the range $0 < p < 1$ result in trajectories which qualitatively interpolate between these two behaviors. Note that the non-lane-changing version of the red agent is unaffected by the actions of the ego agent for all values of p in this case.

A serious drawback of game-theoretic planning is that it typically assumes that the intentions of other agents are known to the ego, and that the agents act rationally with respect to those intentions. In this chapter, we introduce a novel way of accounting for the uncertainty an autonomous system has regarding the intentions of other agents in game-theoretic planning contexts.

In particular, our approach assumes the ego agent maintains a probability distribution over a discrete set of hypotheses regarding the intentions of other agents. These hypotheses model uncertainties about intentions such as potential lane changes, nominal speeds, aversion to acceleration, etc. These hypotheses could, for example, be generated by a predictor trained from data. For the purposes of this work, we are not concerned with how these hypotheses are generated, but rather how the system should process them. Our proposed approach is to construct distinct optimization problems for each agent, which when optimized over, result in trajectories which are qualitatively representative of the different hypotheses associated with that agent. Replicas of each agent (one for each hypothesis) are created, and are assigned to the corresponding optimization problem. We introduce a way of combining the replicated versions of other agents into a dynamic game, which allows the ego agent to reason about interactions with the other agents based on the probabilities of their hypothesized actions in a principled manner. This is the main contribution of our work.

The remainder of this chapter is outlined as follows. We detail comparisons of our method to prior approaches in Section 5.2. In Section 5.3, we review concepts related to game-theoretic planning needed to formalize our method, which is then introduced in Section 5.4. In Section 5.5, we present empirical results of our method on various realistic driving scenarios, and finally, we make concluding remarks in Section 5.6.

5.2 Related Work

Motion planning for autonomous vehicles is a well-researched field, and many algorithms and formulations exist. The method we present builds directly on some recent works which explore specialized game-theoretic planning algorithms and their use for autonomous systems, so we primarily discuss those works here.

There are multiple equilibrium concepts which can be considered for trajectory games, and therefore game-theoretic motion planning. The primary types of equilibria most often considered are Open-Loop Nash Equilibria, Feedback Nash Equilibria, and Open-Loop Stackelberg Equilibria. For detailed description of these concepts, see, for example, [18]. Each of these types of equilibria further offer a *generalized* variant, which are considered when constraints are shared among players in the game, such as collision-avoidance constraints. Different methods in the literature have considered the use of each of these equilibrium concepts for the use of game-theoretic motion planning.

The works [47, 52, 51] present algorithms for computing (generalized) Open-Loop Nash equilibria in trajectory games. Considering this type of equilibrium is equivalent to treating the trajectory game as a static Nash game. Hence, players have equal precedence in determining their trajectories, and the players choose their entire trajectories at once. Generalized static Nash games are discussed in [71]. A study of Newton-type methods for solving generalized Nash Equilibrium problems is given in [70].

The drawback of the Open-Loop Nash equilibrium formulation is that players cannot directly reason about the reaction other agents will have to their own actions. To overcome this challenge, Open-Loop Stackelberg equilibria can be found, which assign an ordered precedence to the players in the game. Players with higher precedence can reason directly about how players with lower precedence will reason about their trajectories. The works [236] and [265] leverage this framework (by means of sensitivity analysis) to accomplish this in the context of drone racing. In [103] a Stackelberg formulation is also used. In [160], both Open-Loop Stackelberg and Nash Equilibria are considered. The authors in [128] use a generalized Stackelberg formulation for use in tree-search planning methods.

Similar to Stackelberg equilibria, Feedback Nash equilibria can capture inter-agent reactions, by treating trajectory games as repeated games. These games are played over a sequence of turns, corresponding to discrete time points in the trajectory. While very expressive, these equilibria are generally more complicated to compute exactly. Nevertheless, [89] presents a method which efficiently computes approximate Feedback Nash equilibria. The authors in [227] also present a method to solve for approximate Feedback Nash Equilibria of trajectory games, although in the context of unconstrained belief-space games. In [151], a method for computing approximate Generalized Feedback Nash Equilibria is introduced, allowing for the computation of Feedback equilibria for games with shared constraints.

While many of the above-mentioned works assume the objective and constraint functions of each player are known, some methods also reason about uncertainty in the underlying game. In [227], uncertainty caused by partial and noisy observations is considered, but the objective and constraints of the players are known. [103] consider different possible driver social models, but these profiles are proposed by a decision making module and are assumed fixed in the inner trajectory optimization. In [128], many possible discrete actions of other agents are considered in the tree search process, but require a discretized space.

There are also a vast number of non game-theoretic works which have considered planning under uncertainty regarding the behavior of other agents in an environment. For example, regarding autonomous vehicle navigation, [274] describes planning in the context of uncertain (yet static) predictions for other agents in the scene. [10] similarly present a real-time method for chance-constrained collision avoidance problems, while [104] uses optimization-based path planning to establish probabilistic collision avoidance guarantees for autonomous vehicles. Meanwhile, [222, 223, 204, 286] describe methods for human activity prediction, in which safety guarantees are established for autonomous vehicles by collecting information to infer the intent of the human agents in the environment. Distributionally-robust optimal control methods such as [256, 191] relax the common Gaussian uncertainty assumption in chance-constrained optimization. These works are just a sample of the many in the literature, but because we are focused on a game-theoretic context, we do not discuss them further.

To the best of our knowledge, the method we present is the only approach capable of efficiently handling uncertainty in the intentions of non-ego agents, in a continuous state and action, game-theoretic context.

5.3 Preliminaries

The method we present assumes that there is no ordering in the precedence of the players in the game, and therefore considers Nash equilibrium formulations of the trajectory games encountered by an autonomous vehicle. In order to properly account for the uncertainty the ego agent has regarding the intention of other players in the game, we rely on a Generalized Feedback Nash equilibrium formulation. Before introducing our method of accounting for uncertainty in the game, we review this concept of equilibrium.

Consider a trajectory game comprised of N agents acting in a discrete-time, continuous-space environment, for which the global state at some time-step t can be represented by the variable $x_t \in \mathcal{X}$, where $\mathcal{X}$ defines the state-space for the environment (typically $\mathbb{R}^n$ for n-dimensional state-spaces). Often, the state-space $\mathcal{X}$ is the product space of the state-spaces for individual agents, i.e. $\mathcal{X} := \mathcal{X}^1 \times ... \times \mathcal{X}^N$. When this is the case, $x_t = (x_t^1, \dots, x_t^N)$. The agents influence the state of the environment by applying private control variables, denoted as $u_t^i \in \mathcal{U}^i$ for agent $i \in \{1, \dots, N\}$ at time-step t. We denote the dimension of $\mathcal{U}^i$ by m^i (both $\mathcal{U}^i$ and m^i may differ across agents), and use the shorthand $u_t := (u_t^1, \dots, u_t^N)$ and $u_t^{-i} := (u_t^1, \dots, u_t^{i-1}, u_t^{i+1}, \dots, u_t^N)$. The state evolution of the system over discrete time-steps is given by the dynamic update

$$x_{t+1} = f(x_t, u_t^1, \dots, u_t^N) = f(x_t, u_t) \tag{5.1}$$

We consider finite-horizon games of discrete time-step length T, and assume without loss of generality that games start at $t = 0$ from a known state $\hat{x}_0$. The objective and constraints imposed on the actions of the agents in the game are also assumed known. Uncertainty in the initial state of the game, as well as in the objective and constraints imposed on other agents, is accounted for in Section 5.4.

The task of computing a generalized Feedback Nash Equilibrium [151] for the game of consideration is that of computing a generalized Nash equilibrium [70] of the following collection of optimization problems for each player i, for $s = 0$:

$$
\begin{aligned}
V_s^i(\hat{x}_s) := \min_{\substack{x_s,\dots,x_T \\ u_s^i,u_{s+1},\dots,u_{T-1}}} \quad & \sum_{t=s}^{T-1} l^i(x_t,u_t) + l_T^i(x_T) && \text{(5.2a)}\\
\text{s.t.} \quad & x_s - \hat{x}_s = 0, && \text{(5.2b)}\\
& x_{t+1} - f(x_t,u_t) = 0, \quad s \le t \le T-1 && \text{(5.2c)}\\
& h_t^i(x_t,u_t^i) = 0, \quad s \le t \le T-1 && \text{(5.2d)}\\
& g_t^i(x_t,u_t^i) \ge 0, \quad s \le t \le T-1 && \text{(5.2e)}\\
& h_T^i(x_T) = 0 && \text{(5.2f)}\\
& g_T^i(x_T) \ge 0 && \text{(5.2g)}\\
& u_t^{-i} - \pi_t^{-i}(x_t) = 0, \quad s+1 \le t \le T-1 && \text{(5.2h)}
\end{aligned}
$$

The cost functions $l^i(\cdot,\cdot)$ and $l_T^i(\cdot)$ are assumed to be continuous and twice-differentiable, and in general may depend on the control variables of other agents, u_t^{-i}.

The equality and inequality constraints imposed on each agent are unique to that particular agent in general, but the dynamic constraints (and initial condition) are common to all agents. The dimension of the constraints h_t^i and g_t^i may vary at every time-step t, including potentially having dimension 0 (representing no constraint). We assume these constraint functions are twice-differentiable in their arguments. In the context of autonomous driving, these terms often encode constraints such as collision avoidance or road boundary constraints.

Here, both the shared state variables and the control variables for agent i are treated as decision variables in the optimization, as are the control variables of other players u_t^{-i} for time $t > s$, where they are constrained by a feedback policy $\pi_t^{-i}(x_t)$. The polices $\pi_t^{-i}(x_t)$ are defined implicitly to yield the controls u_t^{-i} which form a Generalized Nash Equilibrium for the set of problems $\{V_t^1(x_t), ..., V_t^N(x_t)\}$.

Note that the computation of a generalized Feedback Nash equilibrium problem is actually a series of T nested equilibrium problems, due to the definition of the policy constraints (5.2h).

The nested information pattern arising in the problems (5.2) is a fundamental aspect of the method for accounting for uncertainty of the intentions of agents in the game presented in Section 5.4. The nested structure allows for optimizing over cost-functions and constraints directly on the control and state variables of other players in the game, which is otherwise impossible in Open-Loop Nash equilibria. The required costs or constraints are also possible in Stackelberg equilibria, although those formulations require assigning an order of precedence to the players in the game, and pose the same computational challenges as Feedback equilibria do. For a in-depth presentation of generalized Feedback Nash equilibria, see [151].

5.4 Methods

We now present a method which accounts for common forms of uncertainty arising in game-theoretic planning frameworks.

The foundation of our method is based on the observation that interaction among agents in a driving scenario can be attributed to some kind of collision-avoidance constraint. The objective and other constraints imposed upon agents in the scene can often be expressed solely as functions of the private state and control variables for independent agents. The responsibility of collision-avoidance between two agents can be assigned in multiple ways, resulting in qualitatively different behaviors. These constraints can be symmetrical, meaning in a pair of agents, both are responsible, or asymmetrical, meaning only one of the two agents bears more responsibility. Asymmetrical situations are useful from a modeling perspective for situations such as when one agent approaches another from behind on the highway, and the rear agent is responsible for avoiding collision.

Always assigning collision-avoidance responsibility to the ego agent is perhaps the safest option, although this can result in overly-conservative behavior, such as being unable to merge into dense traffic. However, assigning collision-avoidance responsibility to other agents in the scene is risky, since assuming other agents will take responsibility can result in dangerous driving behavior from the ego agent if this assumption is incorrect.

Our method uses asymmetrical constraints to account for uncertainty the ego vehicle has about the other agents in the scene. In order to do this, we introduce a way to interpolate constraint responsibility continuously between two agents.

Given multiple hypotheses regarding the intentions of each non-ego agent in the scene, we propose introducing a copy of the corresponding agent for each hypothesis. These replicas are endowed with objective and constraint terms of the form Eq. (5.2), which reflects the hypothesized intentions. We associate a probability of occurrence with each hypothesis replica. Then, for hypotheses with low probability, we shift collision-avoidance constraint responsibility to the replica agent, allowing the ego to ignore the replica. For hypotheses with high probability, we shift constraint responsibility to the ego vehicle, allowing the replica to ignore the ego vehicle. A continuum of behaviors in between these two extremes is achieved for intermediate probabilities.

Interpolating Responsibility of Collision-Avoidance

The interpolation between collision-avoidance constraint responsibility is made possible by the properties of the Feedback Nash equilibrium introduced in Section 5.3.

Consider first the simple case of two agents in a trajectory game, one of them being the ego agent. Let the other be an agent with a hypothesized objective and set of constraints. Denote these two agents as P^1 (ego) and P^2. We denote the *independent* objective function of both players to take the form in Eq. (5.2a), but only depend on the controls and state variables associated with their self:

$$L^1(x_0, u_0, ..., x_T) := \sum_{t=0}^{T-1} l^1(x_t^1, u_t^1) + l_T^1(x_T^1). \tag{5.3a}$$

$$L^2(x_0, u_0, ..., x_T) := \sum_{t=0}^{T-1} l^2(x_t^2, u_t^2) + l_T^2(x_T^2). \tag{5.3b}$$

We add a copy of the objective of P^2 to the objective of P^1, weighted by the odds (corresponding to probability p) of this hypothesis for P^2. Specifically, the complete objective function for the ego agent is

$$\begin{aligned}\bar{L}^1(x_0, u_0, ..., x_T; p) :=& \\ & L^1(x_0, u_0, ..., x_T) + \frac{p}{1-p} L^2(x_0, u_0, ..., x_T).\end{aligned} \tag{5.4}$$

We refer to the added term in Eq. (5.4) as a "politeness" term.

Because we have introduced the politeness term in the objective of P^1, we assign only P^2 the collision-avoidance constraints between P^1 and P^2. The other constraints imposed on P^1 in Eqs. (5.2d) to (5.2g) depend only on its own state and control variables. In other words, we only explicitly require P^2 to account for the collision-avoidance constraints between P^1 and P^2.

Although P^1 does not account for the collision-avoidance constraints itself, P^1 can take effective ownership of the constraints for large values of p. When $p \to 1$, the right-hand side of Eq. (5.4) dominates P^1's independent objective. This places a very large penalty on sub-optimal values of P^2's objective. This can effectively be viewed as imposing a constraint on P^1 that P^2's objective term is minimized with respect to the decision variables of P^1.

In this limiting case, P^1 does its best to ensure that P^2 does not have to incur any unnecessary cost to optimize its objective or satisfy its constraints, including the collision-avoidance constraints it is responsible for satisfying. Therefore, P^1 prioritizes staying clear of P^2's desired trajectory, so that P^2 doesn't have to exert effort to avoid P^1. In the other limit, when $p \to 0$, the right-hand side of Eq. (5.4) vanishes, and P^1 ignores all notions of collision-avoidance, leaving P^2 to be responsible. For intermediate values of p, the responsibility of collision avoidance is shared between the two agents.

It is here that the dependence of our method on a nested equilibrium concept such as the Feedback Nash equilibrium introduced in Section 5.3 is made clear. The loss function L^1 includes terms that are functions only of the state and control variables of P^2, which would be ignored in, for example, an Open-Loop Nash equilibrium.

Handling Unknown Intentions

Given the ability to interpolate between the collision-avoidance constraint ownership between agents, we propose using the probability of existence associated with multiple hypothesized agents as the interpolating factor appearing in Eq. (5.4).

In particular, assume again that the ego-vehicle is indexed as agent 1, and all other agents in the scene correspond to indices 2 through N. For each of the other agents, there are K^i hypotheses for the potential intentions of each agent i. Again, assume there is a known categorical belief distribution Q^i over these K^i hypotheses. Q^i is comprised of probabilities $\{p^{i,1}, ..., p^{i,K^i}\}$, where $0 < p^{i,k} < 1$ and $\sum_{k=1}^{K^i} p^{i,k} = 1$.

We construct a trajectory game from the set of hypotheses in the following way. For each hypothesis k regarding agent i, we form a replica agent, denoted $P^{i,k}$. Let the independent state of agent $P^{i,k}$ at time-step t be denoted by $x_t^{i,k}$, and the corresponding control variable by $u_t^{i,k}$. The total number of agents (including replicas) is now given by $\hat{N} = 1 + \sum_{i=2}^{N} K^i$. Let x_t and u_t denote the vector of states and controls for all $\hat{N}$ agents, i.e. $x_t := (x_t^1, x_t^{2,1}, \ldots, x_t^{2,K^2}, \ldots, x_t^{N,K^N})$, and u_t is defined analogously.

Each replica agent $P^{i,k}$ is assigned an objective $L^{i,k}$ of the form in Eq. (5.3), which only depends on the states $x_t^{i,k}$ and controls $u_t^{i,k}$. The constraints associated with each agent include collision-avoidance constraints with the ego agent.

The complete objective for the ego agent is then given by

$$\begin{aligned}\bar{L}^1(x_0, u_0, ..., x_T) := {} & L^1(x_0, u_0, ..., x_T) \\ & + \sum_{i=2}^{N} \sum_{k=1}^{K^i} \frac{p^{i,k}}{1 - p^{i,k}} L^{i,k}(x_0, u_0, ..., x_T),\end{aligned} \tag{5.5}$$

and is subject only to dynamic and constraints only on its own state and control variables.

When considering the objectives and constraints of the ego agent and all replica agents in a generalized Feedback Nash equilibrium, the same notions of constraint ownership described in Section 5.4 apply. If any given hypothesis associated with a particular agent has high probability, the politeness term associated with that hypothesis will force the ego agent to take constraint ownership.

In general, non-ego agents can also interact with each other, meaning the hypotheses regarding those agents are coupled. In such a setting, replicas of each agent should be made for every global hypothesis, and collision avoidance responsibility can be assumed to be shared among all non-ego agents associated with the same global hypothesis. In most situations, however, we find that it is not necessary to consider the interaction of non-ego agents, as demonstrated in Section 5.5.

5.5 Results

We demonstrate our proposed approach on several different traffic scenario examples. In each, the ego agent maintains multiple hypotheses about one or more other agents in the scene. By varying its belief about which version of the various agents will realize, the ego generates a spectrum of maneuvers, all of which are game-theoretic equilibria of the game posed in Section 5.4. For simplicity, we presume that all agents follow a linear driving model with independent lateral and longitudinal accelerations. Nothing in our method precludes the use of more expressive dynamics models, however.

Passing Slow-Moving Traffic

Consider a common situation occurring on highways or roads with two or more lanes in each direction, in which there is slow-moving traffic in one lane, and the ego vehicle alone occupies the other lane. This is the situation depicted in Fig. 5.1. Although there are no vehicles in front of the ego vehicle forcing it to slow down, it is unsafe to travel at high speeds past slow-moving traffic. Our method allows a natural way to achieve safe driving behavior that is reflective of the probability that one of the slow-moving vehicles will turn into the lane of the ego vehicle.

Specifically, the two hypotheses considered in this example correspond to P^2 lane changing vs. staying to the right. In terms of the Dynamic Feedback Game, we represent these hypotheses with the terminal constraints on the lateral position of the red agent. Associated with these two hypotheses is the distribution

$$Q := \{p, 1-p\}, \tag{5.6}$$

where p represents the probability of the lane-change hypothesis. The independent objectives for all agents are the sums of quadratic costs on their private control inputs (accelerations) and quadratic costs on desired speed. The ego agent additionally minimizes the odds-weighted independent objective of the two red agent replicas, as described in Section 5.4. Because the non-lane-changing version of the red agent does not interact with the ego agent, the independent objective corresponding to that hypothesis can be ignored.

As demonstrated in Fig. 5.1, varying the probability p naturally produces a range of behaviors of the ego agent, ranging from slowing down in full-anticipation of a lane change, to speeding by, as if the ego vehicle is ignoring the possibility of a lane change.

Lane Change: Coupled Predictions

Consider a similar situation, except now the ego vehicle is on the right and attempting to change lanes, and it is uncertain about the speed of an agent approaching in the left lane. This situation is depicted in Fig. 5.2. The ego agent again considers two hypotheses regarding the approaching agent, associated with different speeds, with p denoting the probability of that the red agent approaches at the higher speed. Unlike the example in Section 5.5, the hypothesis associated with a lower speed of the approaching agent still requires the ego vehicle to apply some otherwise-non-optimal acceleration. As in the previous example, interpolating p from 0 to 1 results in behaviors ranging from respecting the slower-moving hypothesis to respecting the faster-moving hypothesis.

For intermediate values of p, although the odds factor for the slower-moving hypothesis does not vanish, the independent objective of the corresponding replica is not affected by

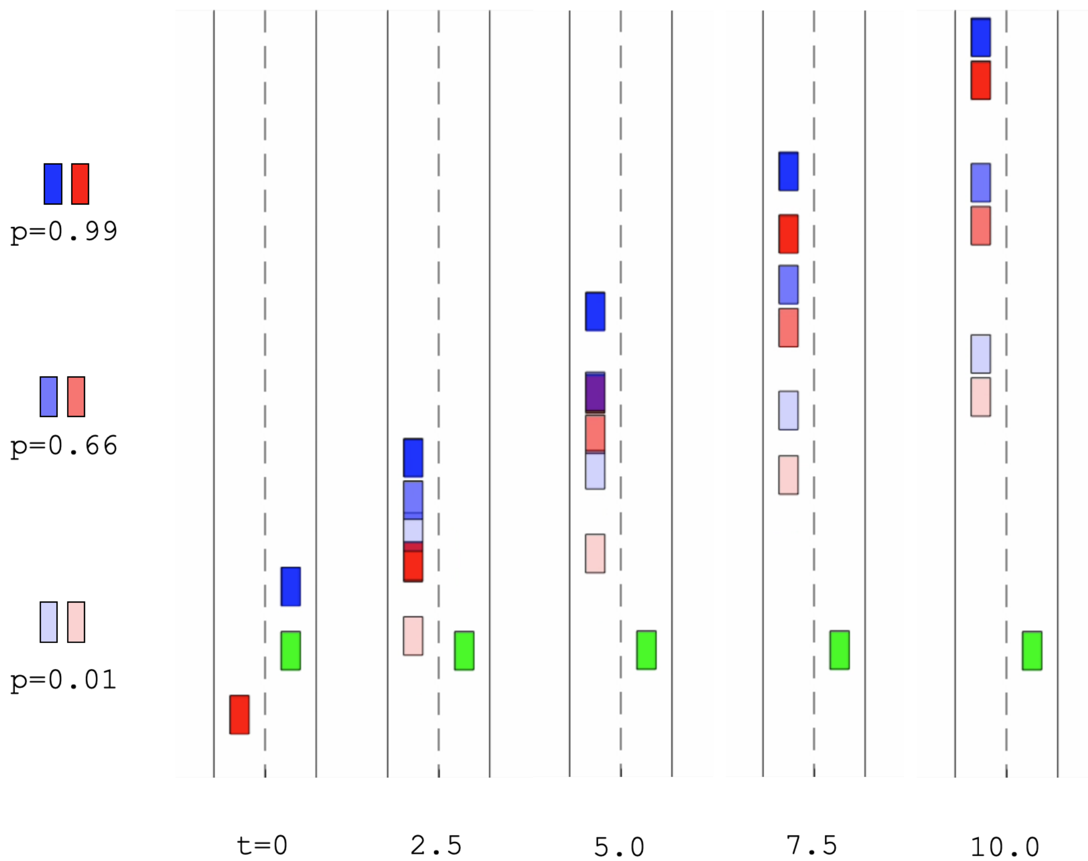


Figure 5.2: The ego (blue) agent is attempting to change lanes in front of a fast-approaching vehicle (red) in the target lane. The ego agent maintains two hypotheses about the speed of the red agent. A belief probability p is placed on the hypothesis that the red-vehicle is traveling very fast, as opposed to moderately fast. Varying the probability p results in a spectrum of behaviors for both agents.

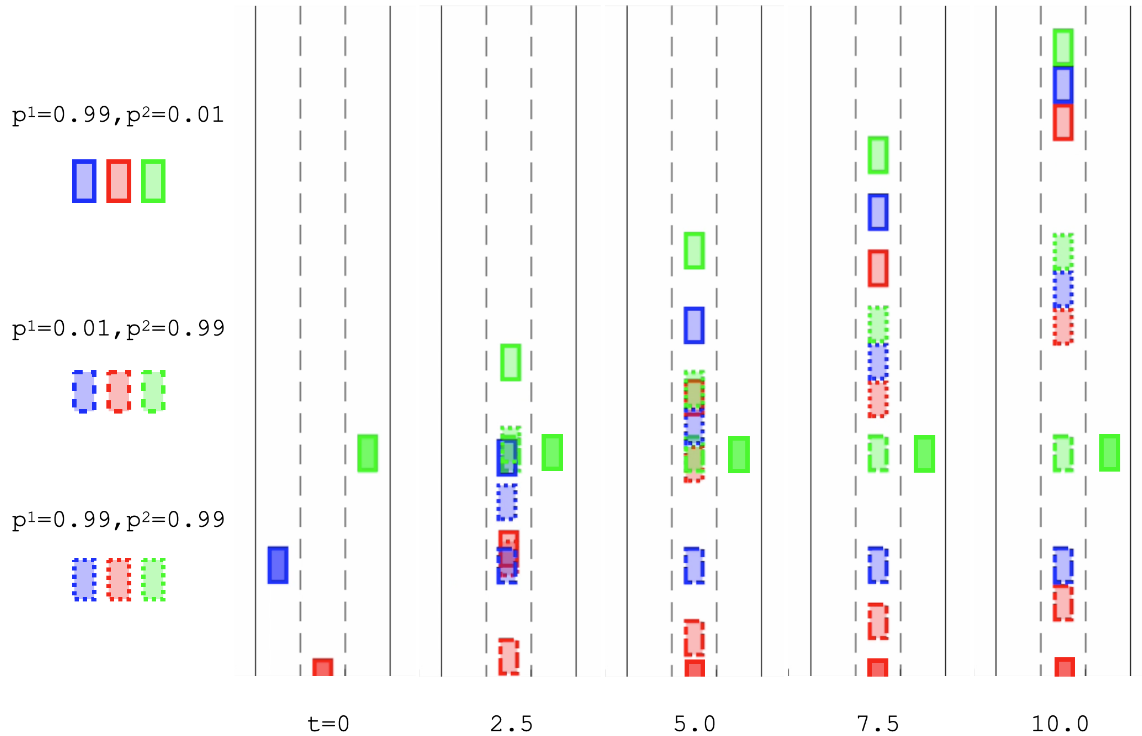


Figure 5.3: The ego (blue) agent is attempting to change lanes in front of the red agent. The ego agent maintains two hypotheses about the speed of the red agent—(1) that it is traveling fast ($p^1 \approx 1$) and (2) that it is traveling at the same speed as ego ($p^1 \approx 0$). The ego agent additionally maintains two hypotheses about the green agent—(1) that it may also change lanes into the middle lane ($p^2 \approx 1$), or (2) may not ($p^2 \approx 0$). In both hypotheses, the green agent is assumed to travel at the same speed as ego. By varying the belief associated with these independent hypotheses, various behaviors emerge.

the ego agent's decision variables. This is because since the ego agent is also considering the fast-moving hypothesis, it does not obstruct the agent in the slower-moving hypothesis. The result of this is that the equilibria associated with intermediate values of p will not linearly interpolate between the behaviors associated with $p \approx 0$ and $p \approx 1$. This is seen in Fig. 5.2, in which $p = 0.66$ is roughly associated with a linear interpolation between the two other behaviors, as opposed to $p = 0.5$ as one might expect.

Double Lane-Change: Multiple Independent Hypotheses

This situation demonstrates the ability of our method to handle hypotheses associated with multiple agents in the scene. As depicted in Fig. 5.3, the ego agent is attempting to lane-change in front of the red agent. There is another (green) vehicle which may also change lanes into the target lane. The ego agent maintains two hypotheses about the speed of the red agent, one being that it is traveling at the same speed as ego, and the other that it is traveling much faster. The ego agent also maintains two hypotheses regarding the green agent, one in which it changes lane and one in which it doesn't. We associate a probability p^1 with the hypothesis that the red agent is traveling fast, and p^2 with the hypothesis that the green agent will change lanes.

The behaviors generated for various values of probabilities p^1 and p^2 are shown in Fig. 5.3. As in previous examples, when the probability of a hypothesis is high, the ego agent takes ownership of collision-avoidance with the corresponding replica, and when the probability is low, the hypothesis is effectively ignored. An interesting aspect of this example is the case in which both $p^1 \to 1$ and $p^2 \to 1$. In this case, the ego agent is conflicted between traveling fast to be polite with respect to the approaching red agent, and traveling slow to allow the green agent to also change lanes. The compromise is to travel at a moderate speed, trading off the preference of both agents. From a practical perspective, this can be problematic. Even if the ego agent is certain that the red agent is moving fast, taking ownership of collision-avoidance with respect to the green agent should take precedence over avoiding collision with the red agent. This behavior can be achieved by simply requiring $p^1 < p^2$ by a sufficient margin. In this perspective, the values p^1 and p^2 would not be interpreted directly as probability of hypothesis, but simply as "politeness" parameters.

Computation

The method we used to compute solutions in the above examples is presented in [151]. In particular, we implemented a method analogous to Sequential Quadratic Programming to jointly solve the first-order necessary conditions corresponding to the game. Linear-Quadratic (LQ) problems formed at each major iteration are solved using an Active-Set (AS) approach. The equality-constrained LQ games to be solved in each minor iteration of the AS approach are solved using a method analogous to [149], adapted to the game setting.

Run-time was not a major concern for the purposes of this work. For ease of prototyping, we implemented our method in MATLAB®. All examples were solved with 50 discrete knot points, with solve times listed in Table 5.1. We strongly believe that the method presented in this work is amenable to real-time computation ($(0.5s)$), if implemented efficiently. For example, proper initializations of the solver can avoid unnecessary iterations, especially when using an active-set method as we do here. Exploiting linearity can further avoid unnecessary computations of gradients or Hessians, and utilizing parallelization to compute problem data could result in large savings. Parallelization could also be utilized in LQ solves as demonstrated in [147]. Finally, implementation in a strongly typed, compiled language such as C++ or Julia would result in major speedups as well.

Example	p	Total (s)	Data (s)	LQ Iters
A	p =0.01	2.15	0.77	7
A	p =0.50	1.99	0.73	7
A	p =0.99	3.12	2.03	11
B	p =0.01	2.26	0.73	8
B	p =0.66	19.08	5.24	70
B	p =0.99	2.35	0.71	9
C	p^1 =0.99, p^2 =0.01	1.92	0.70	7
C	p^1 =0.01, p^2 =0.99	3.43	1.04	13
C	p^1 =0.99, p^2 =0.99	30.76	6.62	113

Table 5.1: Solve times and iterations for all examples. Example IDs are according to subsections of Section 5.5. The information under "Total" refers to total solve time. "Data" refers to the portion of total time spent evaluating problem data such as function gradients and Hessians. "LQ Iters" refer to the number of solves of equality-constrained LQ-games in minor iterations of the solver. Both instances with exceptionally long solve-times had very-large numbers of active-constraints, necessitating many iterations due to poor initializations.

5.6 Conclusion

This chapter accounts for categorical uncertainty in the intentions of other players by forming replicas of those players, and assigning collision-avoidance responsibility according to the probability associated to each replica. We show several examples of its expressive ability to handle uncertain intentions in the context of autonomous driving.

Future work will focus on a real-time implementation of our method, and will investigate the application of this method to receding horizon, model-predictive control settings. It may also be important to study our method in more depth for contexts where interactions among non-ego agents cannot be ignored. Finally, methods to estimate agents' constraints should be investigated. Preliminary work in the context of optimal control is ongoing, e.g., [228, 258, 44], but should be extended to the game setting.

Chapter 6

Problems of Ordered Preference

In this chapter, problems of ordered preference (POPs) are introduced, and it is demonstrated that these problems are naturally cast as connected quadratic programs, and in the framework of $EPNECs$. The particular structure of the resultant connected programs are studied. In the general case, the problems of interest will not satisfy some of the requirements assumed in Chapter 2, motivating further development so the methodologies presented there can be extended to this important problem class. However, a special case is presented which does admit solutions, namely the over-constrained dynamic programming problem.

6.1 Motivation

Many engineering disciplines involve the art of crafting optimization problems, such that when solved, a particular desired result is returned. One such example is that of motion planning engineers working on robotic or autonomous driving projects. Motion planning is naturally cast as an optimization problem: find a trajectory which minimizes energy expenditure, subject to satisfying collision avoidance constraints, obeying the speed limit, etc.

While it might be simple in theory to conjure up a reasonable set of cost and constraint functions for this purpose, practitioners often encounter difficult decisions when making these modeling choices. For an autonomous vehicle, should trajectories be absolutely constrained to obey the speed limit? What about staying within the lane lines? While in nominal situations, these restrictions should be treated as constraints, in some cases these rules should be broken. If collision is otherwise unavoidable, a planned trajectory should absolutely exit the lane boundaries.

A sensible way of encoding this type of desired behavior might be the following:

1. Minimize collision violations.
2. With remaining degrees of freedom, minimize road boundary violations.
3. With remaining degrees of freedom, minimize speed limit violations.

4. With remaining degrees of freedom, minimize discomfort.

This type of *ordered preference* of objectives is not possible in a standard optimization framework. There, every term would have to be considered either a constraint or be lumped into the cost function. If a system is expected to violate one of these "constraints" at some time or another, an engineer will likely prefer to absorb them into the cost function to avoid infeasible problem instances. This imposes the difficult challenge of choosing weightings which trade off the various penalized constraint terms such that the desired result is achieved in a variety of situations.

Alternatively, one could formulate this problem of ordered preferences as a connected quadratic program, and enable the automatic generation of trajectories which obey those preferences. In general, connected quadratic programs can be used to solve any problem which involves a quadratic objective and considers an ordered preference on violations of linear constraints (equality or inequality), as shown in the sections to follow.

6.2 General Formulation

Consider a problem which is to optimize over some decision variable $x \in \mathbb{R}^n$, according to the following ordered preferences:

$$\begin{array}{ll} 1. & \min\limits_x \ \min(0, ({a_m}^\intercal x + b_m))^2 \\ 2. & \min\limits_x \ \min(0, ({a_{m-1}}^\intercal x + b_{m-1}))^2 \\ \vdots & \vdots \\ m. & \min\limits_x \ \min(0, ({a_1}^\intercal x + b_1))^2 \\ m+1. & \min\limits_x \ x^\intercal(\frac{1}{2}Qx + q) \end{array} \tag{6.1}$$

Here, the first m preferences (with 1 being top priority) are to minimize the violation of the m inequalities given by the term

$$Ax + b \geq 0. \tag{6.2}$$

In other words, ${a_i}^\intercal$ corresponds to the ith row of the matrix A, and the last inequality in (6.2) is the top priority. The lowest priority, which is only considered if the first m violations can be avoided, is to minimize the quadratic objective.

It is straightforward to generalize this framework to allow for other types or orders of preferences, such as violations of equality constraints, groups of terms with equal preference, or weakly convex cost terms. However, for simplicity, only the ordering in (6.1) is considered here.

To see how this problem can be viewed as connected quadratic programs, helper variables $c \in \mathbb{R}^m$ and $v \in \mathbb{R}^m$ are introduced. Let $c_{l:m} := \begin{bmatrix} c_l & \dots & c_m \end{bmatrix}^\intercal$, and define $v_{l:m}$ analogously. The following connected QPs define the problem:

$$
\begin{aligned}
OPQP_m^a := \quad & \underset{x, c_m, v_m}{\text{arg min}} && v_m^2 && (6.3a)\\
& \text{subject to} && x, c_m, v_m \in OPQP_m^b && (6.3b)
\end{aligned}
$$

$$
\begin{aligned}
OPQP_m^b := \quad & \underset{x, c_m, v_m}{\text{arg min}} && (v_m - c_m)^2 && (6.4a)\\
& \text{subject to} && v_m \leq 0, && (6.4b)\\
& && a_m^\intercal x + b_m - c_m = 0 && (6.4c)
\end{aligned}
$$

The set of minimizers of $OPQP_m^a$ are all x such that $a_m^\intercal x + b_m \geq 0$. Hence these two problems encode, in an elaborate way, the first preference in (6.1). The remaining ordered preferences are encoded through the remainder of the connected QPs, defined for $1 \leq l \leq m-1$:

$$
\begin{aligned}
OPQP_l^a := \quad & \underset{x, c_{l:m}, v_{l:m}}{\text{arg min}} && v_l^2 && (6.5a)\\
& \text{subject to} && x, c_{l:m}, v_{l:m} \in OPQP_l^b && (6.5b)
\end{aligned}
$$

$$
\begin{aligned}
OPQP_l^b := \quad & \underset{x, c_{l:m}, v_{l:m}}{\text{arg min}} && (v_l - c_l)^2 && (6.6a)\\
& \text{subject to} && v_l \leq 0, && (6.6b)\\
& && a_l^\intercal x + b_l - c_l = 0, && (6.6c)\\
& && x, c_{l+1:m}, v_{l+1:m} \in OPQP_{l+1}^a && (6.6d)
\end{aligned}
$$

At each level $1 \leq l \leq m$, the set of minimizers of $OPQP_l^a$ are all x such that the preferences 1 through $m+1-l$ are optimized, in order. The lowest priority preference, the quadratic objective, is minimized in the final connected QP:

$$
\begin{aligned}
OPQP_0 := \quad & \underset{x, c_{1:m}, v_{1:m}}{\text{arg min}} && x^\intercal(\frac{1}{2}Qx + q) && (6.7a)\\
& \text{subject to} && x, c_{1:m}, v_{1:m} \in OPQP_1^a && (6.7b)
\end{aligned}
$$

Jointly solving the set of QPs given by (6.3), (6.4), (6.5), (6.6), and (6.7) can be accomplished by finding a solution to the equilibrium problem with nested equilibrium constraints as formalized in Section 2.4. In the framework of connected optimization problems, there are $2m+1$ levels of $EPNECs$, each comprised of a single $QPNEC$, corresponding to exactly

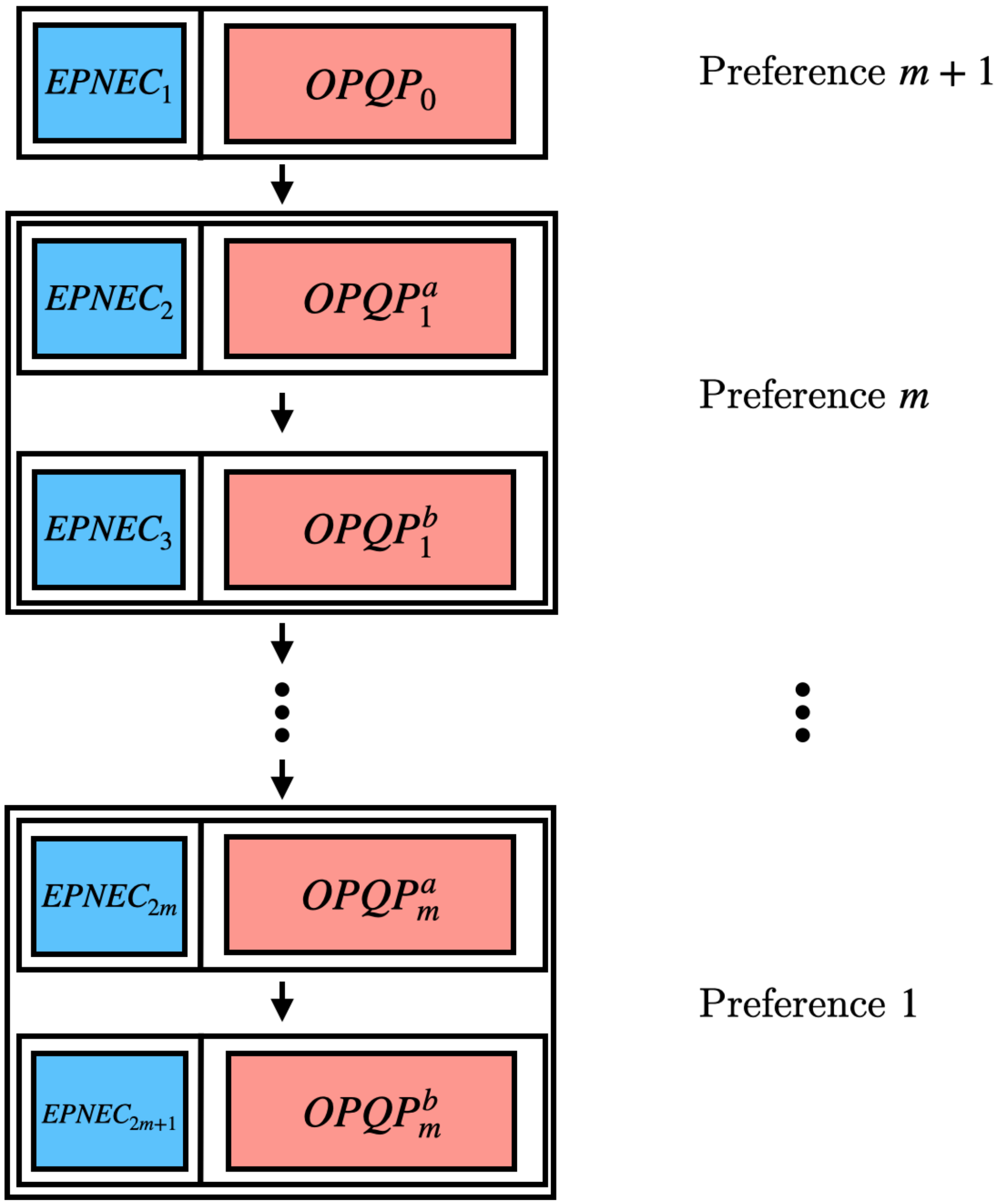


Figure 6.1: Organization of connected QPs arising in problems of ordered preference. Here, the $QPNECs$ depicted in red are named by the specific $OPQP$ they correspond to. In this case, wrapping each layer of this organization as an $EPNEC$ is not necessary, and only done to relate this particular organization to the general organization introduced in Chapter 1. Variable scopes for this organization are given in Table 6.1.

l	Input to $ECQP_l$
1	$y_1 = []$
2	$y_2 = []$
3	$y_3 = []$
$\vdots$	$\vdots$
2m	$y_4 = []$
2m+1	$y_5 = []$

l	Output of $ECQP_l$
1	$x_1 = \begin{bmatrix} x & v_{1:m} & c_{1:m} \end{bmatrix}$
2	$x_2 = \begin{bmatrix} x & v_{1:m} & c_{1:m} \end{bmatrix}$
3	$x_3 = \begin{bmatrix} x & v_{1:m} & c_{1:m} \end{bmatrix}$
4	$x_4 = \begin{bmatrix} x & v_{2:m} & c_{2:m} \end{bmatrix}$
5	$x_5 = \begin{bmatrix} x & v_{2:m} & c_{2:m} \end{bmatrix}$
$\vdots$	$\vdots$
2m	$x_{2m} = \begin{bmatrix} x & v_m & c_m \end{bmatrix}$
2m+1	$x_{2m+1} = \begin{bmatrix} x & v_m & c_m \end{bmatrix}$

Table 6.1: Variable scopes for the $EPNECs$ appearing in the general problem of ordered preference presented in Section 6.2. Here, each layer is independent of all preceding layers, in that there are no parametric inputs. However, each layer can, in general, affect the entire vector x, in effect, imposing constraints which the preceding layers must satisfy.

one of the $2m + 1$ QPs. A diagram of the organization characteristic of these problems of ordered preference is shown in Fig. 6.1.

Unfortunately, the properties of the $EPNEC$ arising in this context violate some of the key assumptions made on problems which enable computation via, for example, Algorithm 4. Namely, for a given parameter value y_l to any $EPNEC_l(y_l)$, any given solution will in general not be isolated. This prohibits generating a piecewise linear representation of local solutions to $EPNEC_l$. On the contrary, the set of solutions form some convex subset of $\mathbb{R}^{n_l}$ for every choice of $y_l \in \mathbb{R}^{m_l}$. An explicit representation of this convex region can be generated for any solution (y_l^*, x_l^*). For any local region of the resultant solution set, this set of solutions can be represented by

$$\mathcal{R} := \{x \in \mathbb{R}^n, y \in \mathbb{R}^m \ : \ Dy + d \geq 0, \ Hx + Gy + g = 0\}, \tag{6.8}$$

with $\operatorname{rank}(H) \leq n$, as opposed to the regions assumed in Chapters 2 and 3,

$$\mathcal{R} := \{x, y \ : \ Dy + d \geq 0, \ x = Ky + k\}. \tag{6.9}$$

The result of this is that the shared variable x is under-specified within this region, and it is unclear which of the various programs collectively optimizing over x own the unconstrained degrees of freedom.

Although the general formulation of these problems of ordered preference remain inaccessible by the methodologies presented in this dissertation, they serve to motivate future work to extend those methodologies to be able to handle such types of problem. More on this topic is discussed in the following chapter.

6.3 Example: Constrained Dynamic Programming

Here, our focus is turned to a special case of a problem with ordered preference, which arises in the context of constrained dynamic programming for optimal control, wherein at each stage of the control problem, a potentially over-constrained optimization problem is encountered. The controlling agent is required to prioritize residual constraint satisfaction, if possible. Only with any remaining degrees of freedom does the agent optimize for its objective. Clearly this problem takes the form of a problem of ordered preference. The context in which this problem arises, and methods for computing solutions are given in what follows. This section is taken from [149], which is co-authored with Claire Tomlin.

Introduction

Due to its mathematical elegance and wide-ranging usefulness, the Linear Quadratic Regulator has become perhaps the most widely studied problem in the field of control theory. Referring to both continuous and discrete-time systems, the LQR problem is that of finding an infinite or finite-length control sequence for a linear dynamical system that is optimal with respect to a quadratic cost function. Either as a stand-alone means for computing trajectories and controllers for linear systems, or as a method for solving successive approximate trajectories for nonlinear systems, it shows up in one way or another in the computation of nearly all finite-length trajectory optimization problems.

Because of the importance of trajectory optimization in controlling robotic systems, and because of the prevalence of the LQR problem in those optimizations, devoting time to highly efficient methods capable of solving LQR-type problems is an important endeavor. The focus of this section is on a particular instance of the discrete-time, finite-horizon variant of the LQR problem, which is subject to linear equality constraints. These constrained problems are important in their own right, and arise in relatively common situations.

As an example, imagine we want to plan a trajectory that minimizes the amount of energy needed to get a robot to some desired configuration. If the dynamics of the robot can be modeled as a linear system, this problem takes the form of linear-equality-constrained LQR. We can also imagine constraints appearing at multiple stages in the trajectory and having varying dimensions. Perhaps we require that the center of mass of the robot is constrained to not move in the first half of the trajectory. Again, this type of constraints typically take the form of linear equality constraints.

Of course, many robots are many robots have nonlinear dynamics. But even when planning constrained trajectories for such systems, iterative solution methods such as Sequential Quadratic Programming make successive local approximations of the trajectory optimization problem which result in a series of constrained LQR problems to be solved. We will discuss this relationship in more detail in a later section.

In order to motivate the need for better solutions to this problem, first note that this type of problem are quadratic programs (QPs). Since the dynamic constraints are linear, and all auxiliary constraints we consider are also linear, these problems result in QPs just as unconstrained LQR problems are QPs [32]. Under standard assumptions, the constrained problems are also strictly convex and each has a unique solution. Unlike unconstrained LQR, however, the presence of additional constraints cause some computational difficulties.

From a pure optimization standpoint, all of the approaches to solving convex QPs can be applied to the constrained LQR problem without problem. However, using general methods in a naive way fails to exploit the unique structure of the optimal control problem, and suffers a computational complexity which grows cubicly with the time horizon being considered in the control problem (trajectory length). Due to the sparsity of the problem data in the time domain, the KKT conditions of optimality for optimal control problems have a banded nature, and linear algebra packages designed for such systems can be used to solve the problem in a linear complexity with respect to the trajectory length [269]. However, these approaches result in what we will call *open-loop* trajectories, producing only numerical values of the state and control vectors making up the trajectory.

It is well-known that the unconstrained LQR problem offers a solution based on dynamic programming which is sometimes referred to as the discrete-time Riccati recursion. This method can also solve unconstrained LQR problems in linear time complexity while *also* providing an affine relationship between the state and control variables. This relationship provides a feedback policy which can be used in control, and offers many advantages over the open-loop variants.

It is because we would like to derive these policies for the constrained case that the aforementioned computational difficulties show up. The presence of auxiliary constraints have made it so that up until now, a completely general method for the equality-constrained LQR problem that is analogous in computation time *and* solution type (feedback vs. open-loop) to the Riccati recursion method has not been developed. It is important to note that many approaches have been developed, but as we will discuss in Section 6.3, they all have important limitations. The key difficulty of this problem is due to the fact that linear constraints (of dimension exceeding that of the control input) can not always be thought of as time-separable. This means that the choice of control at a particular time-point may not always be able to satisfy a constraint appearing at that time-point (for arbitrary values of the corresponding state at that time). We will see that this complication requires satisfying portions of such a constraint at points in time before it actually appears, making dynamic programming solutions non-trivial. This is why existing methods either make restrictive assumptions on the dimension of constraints, or require a higher order of computational complexity to compute solutions than one might expect.

If the problem to be solved does not satisfy the restricting assumptions made by those methods, solution approaches are therefore limited to QP solvers and only offer open-loop trajectories, or suffer cubic time-complexity with respect to the trajectory length if control policies are desired. Given this context, we can now state the contribution of this work:

We present a method for computing constraint-aware feedback control policies for discrete-time, time-varying, linear-dynamical systems which are optimal with respect to a quadratic cost function and subject to auxiliary linear equality constraints. This is done by handling the constraints in a novel way such that a dynamic programming solution can be formulated. We make no assumptions about the dimension of the constraints, with effective handling of over-constrained or redundantly constrained problems.

In section 6.3 we discuss in more detail the existing methods which have addressed the same problem and the corresponding limitations of those works. In section 6.3 we formally define the problem and present our method. In section 6.3 we discuss computational complexity, and present an alternative approach to solving the problem. We also demonstrate some of the advantages of the control policies derived from our method when compared to the open-loop solutions, and discuss applicability to SQP methods.

Prior Work

Consideration of the constrained linear-quadratic optimal control problem extends back to the early days in the field of control. Many authors have presented methods for constraining control systems to a time-invariant linear subspace. The author in [122] studied this issue for continuous systems under the name subspace stabilization. In the works [107] and [279] the same problem is addressed by designing pole-assignment controllers. More recently, [207] utilizes a very similar method to generate a time-varying controller for tracking existing trajectories. This method is also derived in continuous-time, and hence requires the constraint dimension to be constant.

The authors in [133] developed a more comprehensive method for computing optimal control policies for discrete-time, time-varying objective functions, but only consider a single time-invariant constraint of constant dimension. In [199] a method is presented for solving continuous- and discrete-time LQR problems with fixed terminal states. This method is able to reason about a constraint only appearing at the end of the trajectory, but does not account for additional constraints appearing at other times.

Perhaps the most general method for computing linearly constrained LQR control policies is presented in [231]. However, this method suffers a computational complexity which scales cubicly in the worst-case, i.e. when many constraints which have dimension exceeding the control dimension are present. As a part of the method presented in [273], a technique for satisfying linear constraints at arbitrary times in the trajectory is presented, but that method assumes that the constraint dimension does not exceed that of the control. Most recently, [94] presents a method for solving problems with time-varying constraints, but still requires that the relative degree of these constraints does not exceed 1. This is a slightly less restrictive condition than requiring the dimension of the constraints be less than that of the control, but still limits the applicability of this method.

As mentioned above, the problem can also be solved using numerical linear algebra techniques, as discussed for example in [268] and particularly for the optimal control problem in [269]. Again, these methods are very general and efficient but fail to produce the desired feedback control policies.

The method we present combines the desirable properties of all these methods into one. The contribution of this method is that it is capable of generating optimal feedback control policies for general, discrete-time, linearly-constrained LQR problems while maintaining a linear computational complexity with respect to control horizon. To the best of our knowledge, the approach we present is the only method in existence that is capable of this.

Problem and Method

The method we present here is a means of deriving optimal feedback control policies for the following problem:

$$\min_{x_0,u_0,...,u_{T-1},x_T} cost_T(x_T) + \sum_{t=0}^{T-1} cost_t(x_t, u_t) \tag{6.10a}$$

$$\text{s.t.} \quad dynamics_t(x_{t+1}, x_t, u_t) = 0 \ \forall t \in \{0, ..., T-1\} \tag{6.10b}$$

$$x_0 = x_{init} \tag{6.10c}$$

$$constraint_t(x_t, u_t) = 0, \ \forall t \in \{0, ..., T-1\} \tag{6.10d}$$

$$constraint_T(x_T) = 0 \tag{6.10e}$$

Where $x_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}^m$, and the functions

$$\begin{aligned} &cost_t : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} && cost_T : \mathbb{R}^n \to \mathbb{R} \\ &constraint_t : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{l_t} && constraint_T : \mathbb{R}^n \to \mathbb{R}^{l_T} \\ &dynamics_t : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n && \end{aligned}$$

are defined as:

$$cost_t(x, u) = \frac{1}{2} \begin{pmatrix} 1 \\ x \\ u \end{pmatrix}^\intercal \begin{pmatrix} 0 & q_{x1_t}^\intercal & q_{u1_t}^\intercal \\ q_{x1_t} & Q_{xx_t} & Q_{ux_t}^\intercal \\ q_{u1_t} & Q_{ux_t} & Q_{uu_t} \end{pmatrix} \begin{pmatrix} 1 \\ x \\ u \end{pmatrix} \tag{6.11}$$

$$cost_T(x) = \frac{1}{2} \begin{pmatrix} 1 \\ x \end{pmatrix}^\intercal \begin{pmatrix} 0 & q_{x1_T}^\intercal \\ q_{x1_T} & Q_{xx_T} \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} \tag{6.12}$$

$$dynamics_t(x_{t+1}, x_t, u_t) = x_{t+1} - (F_{x_t} x_t + F_{u_t} u_t + f_{1_t}) \tag{6.13}$$

$$constraint_t(x_t, u_t) = G_{x_t} x_t + G_{u_t} u_t + g_{1_t} \tag{6.14}$$

$$constraint_T(x_T) = G_{x_T} x_T + g_{1_T}, \tag{6.15}$$

where l_t (for $0 \leq t < T$) and l_T are the dimensions of the constraints at the corresponding times.

In the above expressions, and in the rest of this chapter, coefficients are assumed to have dimension such that the expression makes sense. We assume for now that the coefficient matrix Q_{uu_t} of the quadratic functions $cost_t$ is positive-definite, and that $Q_{xx_t} - Q_{ux_t} Q_{uu_t}^{-1} Q_{ux_t}^\intercal$ is positive semi-definite. This assumption is possible to relax, and we will discuss this below.

Constrained LQR

The method for computing the constrained control policies will follow a dynamic programming approach. Starting from the end of the trajectory and working towards the beginning, a given control input u_t will be chosen such that for any value of the resulting state x_t, the control will satisfy all constraints imposed at time t, as well as any constraints remaining to be satisfied in the remainder of the trajectory, if possible.

If the constraint is unable to be satisfied by the control for arbitrary states, the control will minimize the sum of squared residuals of the constraints. This has the effect of eliminating r dimensions of the constraint, where r is the rank of the constraint coefficient multiplying u_t. For a trajectory to satisfy the constraint in this case, the state x_t must therefore be such that the constraint residuals will be zero. This can be enforced by passing on a residual linear constraint to the choice of control at the preceding time, u_{t-1} (and controls preceding that, if necessary). If there are degrees of freedom in the control input that do not affect the constraint, the portion of the control lying in the null-space of the constraint will be chosen such as to minimize the cost in the remainder of the trajectory. Following this procedure will result in solutions to problem (6.10), when one exists.

To formalize this procedure, we introduce a time-varying quadratic function, $cost\_to\_go_t : \mathbb{R}^n \to \mathbb{R}$, representing the minimum possible cost remaining in the trajectory from stage t onward as a function of state. Additionally, we introduce a linear function $constraint\_to\_go_t : \mathbb{R}^n \to \mathbb{R}^{p_t}$, which defines, through a constraint on x_t, the subspace of admissible states such that the control u_t will be able to satisfy the constraints in the remainder of the trajectory. Here p_t is the dimension of constraints needed to enforce this condition. These functions are defined as follows:

$$cost\_to\_go_t(x) = \frac{1}{2}\begin{pmatrix}1\\x\end{pmatrix}^\intercal \begin{pmatrix}0 & v_{x1_t}^\intercal\\ v_{x1_t} & V_{xx_t}\end{pmatrix}\begin{pmatrix}1\\x\end{pmatrix} \tag{6.16}$$

$$constraint\_to\_go_t(x) = H_{x_t}x + h_{1_t}. \tag{6.17}$$

We initialize these terms at time T:

$$\begin{aligned} V_{xx_T} &= Q_{xx_T} & v_{x1_T} &= q_{x1_T} \\ H_{x_T} &= G_{x_T} & h_{1_T} &= g_{1_T}. \end{aligned} \tag{6.18}$$

Note that in the value function (6.16) we do not include any constant terms (which would appear in the top-left block of (6.16)). This is because the calculations we will derive do not depend on them, and so we omit them for clarity.

Given the above definitions, starting at $T-1$ and working backwards to 0, we solve the following optimization problem for each time t:

$$u_t^*(x_t) = \arg\min_{u_t}\ cost_t(x_t, u_t) + cost\_to\_go_{t+1}(x_{t+1}) \tag{6.19a}$$

$$\text{s.t.}\quad 0 = dynamics_t(x_{t+1}, x_t, u_t) \tag{6.19b}$$

$$u_t \in \arg\min_u \|\begin{bmatrix} constraint_t(x_t, u) \\ constraint\_to\_go_{t+1}(x_{t+1}) \end{bmatrix}\|_2 \tag{6.19c}$$

To see how a solution to this problem can be found, we first simplify it by using the form of (6.19b) to eliminate x_{t+1}, and plug in coefficients:

$$u_t^*(x_t) = \arg\min_{u_t} \frac{1}{2}\begin{pmatrix}1\\x_t\\u_t\end{pmatrix}^\intercal \begin{pmatrix}0 & m_{x1_t}^\intercal & m_{u1_t}^\intercal \\ m_{x1_t} & M_{xx_t} & M_{ux_t}^\intercal \\ m_{u1_t} & M_{ux_t} & M_{uu_t}\end{pmatrix}\begin{pmatrix}1\\x_t\\u_t\end{pmatrix} \tag{6.20a}$$

$$\text{s.t.}\quad u_t \in \arg\min_u \|N_{x_t}x_t + N_{u_t}u + n_{1_t}\|_2. \tag{6.20b}$$

Where the above terms are defined as:

$$\begin{aligned} m_{x1_t} &= q_{x1_t} + F_{x_t}^\intercal v_{x1_{t+1}} & m_{u1_t} &= q_{u1_t} + F_{u_t}^\intercal v_{x1_{t+1}} \\ M_{xx_t} &= Q_{xx_t} + F_{x_t}^\intercal V_{xx_{t+1}} F_{x_t} & M_{uu_t} &= Q_{uu_t} + F_{u_t}^\intercal V_{xx_{t+1}} F_{u_t} \\ M_{ux_t} &= Q_{ux_t} + F_{u_t}^\intercal V_{xx_{t+1}} F_{x_t} & N_{x_t} &= \begin{pmatrix} G_{x_t} \\ H_{x_{t+1}}F_{x_t}\end{pmatrix} \\ N_{u_t} &= \begin{pmatrix} G_{u_t} \\ H_{x_{t+1}}F_{u_t}\end{pmatrix} & n_{1_t} &= \begin{pmatrix} g_{1_t} \\ H_{x_{t+1}}f_{1_t} + h_{1_{t+1}}\end{pmatrix}. \end{aligned} \tag{6.21}$$

Given this form, we can again re-write the problem as an unconstrained optimization problem, which admits a closed-form solution to $u_t^*(x_t)$:

$$\begin{aligned} y_t^*, w_t^* = \arg\min_{y_t, w_t} \frac{1}{2}\|N_{x_t}x_t + N_{u_t}P_{y_t}y_t + n_{1_t}\|_2 + \\ \frac{1}{2}\begin{pmatrix} 1 \\ x_t \\ Z_{w_t}w_t \end{pmatrix}^\intercal \begin{pmatrix} 0 & m_{x1_t}^\intercal & m_{u1_t}^\intercal \\ m_{x1_t} & M_{xx_t} & M_{ux_t}^\intercal \\ m_{u1_t} & M_{ux_t} & M_{uu_t} \end{pmatrix} \begin{pmatrix} 1 \\ x_t \\ Z_{w_t}w_t \end{pmatrix} \end{aligned} \tag{6.22a}$$

$$u_t^* = P_{y_t}y_t^* + Z_{w_t}w_t^* \tag{6.22b}$$

Here, Z_{w_t} is chosen such that the columns form an orthonormal basis for the null space of N_{u_t}, and P_{y_t} is chosen such that its columns form a orthonormal basis for the range space of $N_{u_t}^\intercal$. Hence N_{u_t} and P_{y_t} are also orthogonal and their columns together span $\mathbb{R}^m$ [34]. We can interpret y_t as the constrained dimensions of the control, and w_t as the free dimensions of the control. One simple way of computing P_{y_t} and Z_{w_t} is to compute the singular-value decomposition of N_{u_t}, and take P_{y_t} to be the first r and Z_{w_t} the last $(m-r)$ columns of the "V" matrix from the SVD of N_{u_t} ($N_{u_t} = USV^\intercal$), where r is the rank of N_{u_t}.

The solution to (6.22) can now easily be expressed:

$$y_t^* = -(N_{u_t}P_{y_t})^\dagger(N_{x_t}x_t + n_{1_t}) \tag{6.23}$$

$$w_t^* = -(Z_{w_t}^\intercal M_{uu_t}Z_{w_t})^{-1}Z_{w_t}^\intercal(M_{ux_t}x_t + m_{u1_t}). \tag{6.24}$$

The symbol † in (6.23) indicates the pseudo-inverse. Note that the pseudo-inverse can always be computed efficiently in this usage[1]. In the case that P_{y_t} is a zero matrix (i.e. $\text{rank}(N_{u_t}) = 0$), then $Z_{w_t} = I_m$ (Identity matrix $\in \mathbb{R}^{m\times m}$), and y_t has dimension 0. Correspondingly, when the nullity of Z_{w_t} is 0, we have $P_{y_t} = I_m$ and w_t has dimension 0. Therefore, in these cases, we ignore the update that is of size 0, i.e. (6.23) or (6.24). With this in mind, and combining terms, we can express the control u_t in closed-form as an affine function of the state x_t:

$$u_t^* = K_{x_t}x_t + k_{1_t} \tag{6.25}$$

$$K_{x_t} = -\big(P_{y_t}(N_{u_t}P_{y_t})^\dagger N_{x_t} + Z_{w_t}(Z_{w_t}^\intercal M_{uu_t}Z_{w_t})^{-1}Z_{w_t}^\intercal M_{ux_t}\big) \tag{6.26}$$

$$k_{1_t} = -\big(P_{y_t}(N_{u_t}P_{y_t})^\dagger n_{1_t} + Z_{w_t}(Z_{w_t}^\intercal M_{uu_t}Z_{w_t})^{-1}Z_{w_t}^\intercal m_{u1_t}\big) \tag{6.27}$$

Since the control is a function of the state, we can also express the value of the constraint residual (6.19c) as a function of the state. We define the function *constraint_to_go*$_t$ to be this constraint residual. We substitute (6.25, 6.26, 6.27) into (6.20b) to obtain:

$$\begin{aligned} &\mathit{constraint\_to\_go}_t(x_t) = \\ &\qquad N_{x_t}x_t - N_{u_t}P_{y_t}(N_{u_t}P_{y_t})^\dagger(N_{x_t}x_t + n_{1_t}) + n_{1_t} \end{aligned} \tag{6.28}$$

[1] $N_{u_t}P_{y_t}$ will always be full column rank, so $(N_{u_t}P_{y_t})^\dagger = ((N_{u_t}P_{y_t})^\intercal(N_{u_t}P_{y_t}))^{-1}(N_{u_t}P_{y_t})^\intercal$.

This results in the update for the terms H_{x_t} and h_{1_t}:

$$H_{x_t} = (I - N_{u_t}P_{y_t}(N_{u_t}P_{y_t})^\dagger)N_{x_t} \tag{6.29}$$

$$h_{1_t} = (I - N_{u_t}P_{y_t}(N_{u_t}P_{y_t})^\dagger)n_{1_t}. \tag{6.30}$$

Here I is the identity matrix having the same leading dimension as N_{x_t}. By observing these updates, we see that the terms in (6.28) are computed by projecting $N_{x_t}x_t + n_{1_t}$ into the kernel of $(N_{u_t}P_{y_t})^\intercal$. Hence, the residual constraint will lie in a subspace of dimension no larger than the nullity of $(N_{u_t}P_{y_t})^\intercal$. We can therefore remove redundant constraints by removing linearly-dependent rows of the matrix $\begin{bmatrix} h_{1_t} & H_{x_t} \end{bmatrix}$, in order to maintain a minimal representation and keep computations small. In general, we will be able to remove r constraints, where r is the rank of N_{u_t}. This can be done by multiplying $\begin{bmatrix} h_{1_t} & H_{x_t} \end{bmatrix}$ by $U^\intercal$ and deleting the last r rows of the resulting matrix, where U comes from the SVD $USV^\intercal = \begin{bmatrix} h_{1_t} & H_{x_t} \end{bmatrix}$.

Note that if h_{1_t} is not in the range space of H_{x_t} at any time, then there exists no x_t that can satisfy the constraints, and we have detected that the trajectory optimization problem (6.10) is infeasible. Otherwise, by enforcing that x_t satisfy the constraint $H_{x_t}x_t + h_{1_t} = 0$, the control u_t^* will be such that all remaining constraints in the trajectory are satisfied.

We also plug the expression for the control into the objective function of our optimization problem (6.20a) to obtain an update on our *cost_to_go* function as a function of the state (again, omitting constant terms):

$$\mathit{cost\_to\_go}_t(x_t) = \frac{1}{2}\begin{pmatrix} 1 \\ x_t \\ u_t^* \end{pmatrix}^\intercal \begin{pmatrix} 0 & m_{x1_t}^\intercal & m_{u1_t}^\intercal \\ m_{x1_t} & M_{xx_t} & M_{ux_t}^\intercal \\ m_{u1_t} & M_{ux_t} & M_{uu_t} \end{pmatrix} \begin{pmatrix} 1 \\ x_t \\ u_t^* \end{pmatrix} \tag{6.31}$$

$$= \frac{1}{2}\begin{pmatrix} 1 \\ x_t \end{pmatrix}^\intercal \begin{pmatrix} 0 & v_{x1_t}^\intercal \\ v_{x1_t} & V_{xx_t} \end{pmatrix} \begin{pmatrix} 1 \\ x_t \end{pmatrix}, \tag{6.32}$$

where terms are defined as

$$V_{xx_t} = M_{xx_t} + 2M_{ux_t}^\intercal K_{x_t} + K_{x_t}^\intercal M_{uu_t} K_{x_t} \tag{6.33}$$

$$v_{x1_t} = m_{x1_t} + K_{x_t}^\intercal m_{u1_t} + (M_{ux_t}^\intercal + K_{x_t}^\intercal M_{uu_t})k_{1_t}. \tag{6.34}$$

We have now presented updates for the terms V_{xx_t}, v_{x1_t}, H_{x_t}, and h_{1_t}, and computed control policy terms K_{x_t} and k_{1_t} in the process. Assuming the initial state x_{init} satisfies $\mathit{constraint\_to\_go}_0(x_{init}) = 0$, then the sequence of control policies $\{K_{x_t}, k_{1_t}\}_{t\in\{0,\ldots,T-1\}}$ will produce, by construction, a sequence of states and controls that are feasible and optimal for our original problem (6.10).

Analysis

In the preceding section, we have presented a method for computing control policies for the equality-constrained LQR problem (6.10). In this section we will analyze the method and the resulting policies by evaluating the computational complexity of the method and by relating the policies to those that are produced in standard, unconstrained LQR.

Table 6.2: Comparing computation times of constrained and unconstrained LQR problems between our constrained LQR method (CLQR) and a method using LAPACK to directly solve the KKT system of equations.

n	m	T	% Constrained	LAPACK (s)	CLQR (s)
40	10	250	0	0.089	0.031
40	10	250	90	0.095	0.040
40	10	125	90	0.046	0.021
9	2	250	0	0.002	0.004
9	2	250	50	0.002	0.007
9	2	125	50	0.001	0.003

Computation

We mentioned that one of the contributions of this method is its computational efficiency compared to existing results. Due to the dynamic-programming nature of this method, the computational time-dependence on trajectory length is linear, irrespective of the dimension of auxiliary constraints.

In each iteration of the dynamic programming backups, the heavy computations involve computing the null and range space representations of N_{u_t} and $N_{u_t}^\intercal$, respectively, and then computing the pseudo-inverse of $N_{u_t}P_{y_t}$. These operations can all be done by making use of one singular-value decomposition. The dimension of N_{u_t} is no greater than $(2n{+}m){\times}m$ where n is the dimension of the state and m is the dimension of the control signal. Computation complexity of the SVD is thus $O((2n{+}m)^2m{+}m^3)$ [100]. We also make use of a decomposition on the terms $\begin{bmatrix} h_{1_t} & H_{x_t} \end{bmatrix}$ to remove redundant constraints, which requires computations on the order of $O(n^3)$. The remaining computations are numerous matrix-matrix products and matrix-inversions with terms having dimension no larger than $2n + m \times n$. Thus, the overall order of the method presented here is $O(T(\kappa_1 n^3 + \kappa_2 n^2 m + \kappa_3 m^2 n + \kappa_4 m^3))$, where $\kappa_1, \kappa_2, \kappa_3, \kappa_4$ are some positive scalars.

Therefore, the method we present here has computational complexity which is roughly equivalent to known solutions based on using a banded-matrix solver on the system of KKT conditions [50] [270]. This is not surprising, since our method can be thought of as performing a specialized block-substitution method on the system of KKT conditions, and hence a specialized block-substitution solver for the particular structure arising in constrained optimal control problems.

In Table 6.2 we show a comparison of computation times between our method and the method 'DGBTRS' from the well-known linear algebra package LAPACK [9]. All times are taken as the minimum over 10 trials, run on a laptop with 2-core 1.7GHz Intel Core i7-4650U processor. The LAPACK method performs Gaussian elimination on the banded KKT system of equations, using a standard BLAS library [155]. We make this comparison for varying problem sizes and percentage of the number of independent constraints relative to the total number of degrees of freedom in the problem. For problems of relatively small size, we see that LAPACK offers superior speed, even in the standard unconstrained LQR case. However, as the problem size grows, we see that our method quickly becomes more efficient than the LAPACK solution.

Infinite Penalty Perspective

We here consider an alternative way to solve (6.10), the quadratic penalty approach. It is known that we can solve equality-constrained quadratic programs by solving an unconstrained problem, where the linear constraint terms are penalized in the objective as an infinitely weighted cost on the sum of squared constraint residuals [28]. Therefore, in light of our original problem (6.10), we could penalize the constraints (6.10d) and (6.10e) in this way, which would result in a standard (from a structural standpoint) LQR problem, where some of the cost terms are weighted infinitely high. The resulting problem would appear as

$$\begin{aligned}
\min_{u_0,\ldots,u_{T-1}} \quad & \mathit{constraint_penalized_cost}_T(x_T) + && (6.35a)\\
& \sum_{t=0}^{T-1} \mathit{constraint_penalized_cost}_t(x_t, u_t) && (6.35b)\\
\text{s.t.} \quad & \mathit{dynamics}_t(x_{t+1}, x_t, u_t) = 0 \ \forall t \in \{0, ..., T-1\} && (6.35c)\\
& x_0 = x_{init}. && (6.35d)
\end{aligned}$$

Here the modified cost functions are defined as

$$\begin{aligned}
&\mathit{constraint_penalized_cost}_t(x, u) = \\
&\qquad \mathit{cost}_t(x, u) + \frac{1}{\epsilon}\|\mathit{constraint}_t(x, u)\|_2^2
\end{aligned} \tag{6.36}$$

$$\begin{aligned}
&\mathit{constraint_penalized_cost}_T(x) = \\
&\qquad \mathit{cost}_T(x) + \frac{1}{\epsilon}\|\mathit{constraint}_T(x)\|_2^2.
\end{aligned} \tag{6.37}$$

Reiterating, if $\epsilon \to 0_+$, problem (6.35) converges to problem (6.10) [28]. However in practice, we cannot penalize the constraint terms by infinity (by letting $\epsilon \to 0_+$), but it may suffice to penalize the constraints by some very large constant. Because the optimal control problems we typically solve are based on approximate models of systems, solving an 'approximately' constrained system may sometimes be adequate. In these cases, one could consider solving the unconstrained penalized problem (6.35). The necessary computation for its solution is slightly less than the approach developed in section 6.3, as was shown in Table 6.2.

We illustrate this relationship between the two methods using a very simple example.

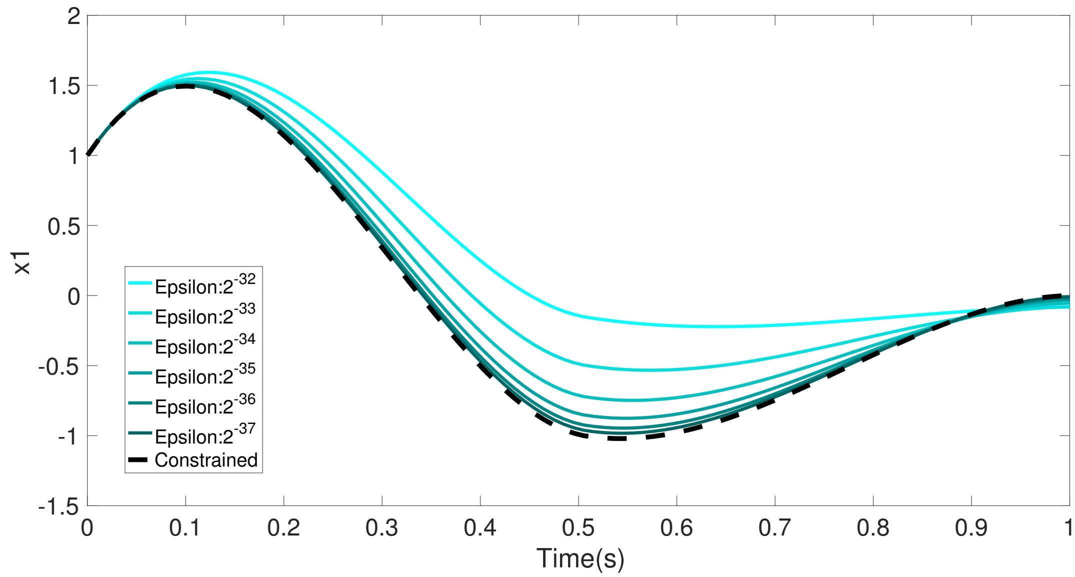


Figure 6.2: Limiting behavior of the penalty approach for problem (6.38).

Consider the constrained LQR problem for a discrete-time double integrator below:

$$\min_{u_0,\dots,u_{T-1}} \sum_{t=0}^{T-1} \|u_t\|_2^2 \tag{6.38a}$$

$$\text{s.t.} \quad x_{t+1} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ dt \end{bmatrix} u_t \tag{6.38b}$$

$$x_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}^\intercal \tag{6.38c}$$

$$x_{T/2} = \begin{bmatrix} -1 & -1 \end{bmatrix}^\intercal \tag{6.38d}$$

$$x_T = \begin{bmatrix} 0 & 0 \end{bmatrix}^\intercal \tag{6.38e}$$

For this example, we let $dt = 0.01$ and $T = 100$ to simulate a one second trajectory. In Figure 6.2, trajectories of the first element of x_t can be seen for the solution to the explicitly constrained formulation as well as solutions computed using the penalty formulation (6.35) for varying values of ϵ. As can be seen, as $\epsilon \to 0_+$, the solutions of the penalty method converge to that of the explicitly constrained method.

While this simpler approach might seem an enticing alternative to the approach outlined in section 6.3, we maintain that our method which handles constraints explicitly is still important. Our method ensures the optimal solution without guessing a sufficient value of ϵ. In applications where correct solutions are needed, such as using this method in the context of an SQP approach (discussed more below), iteratively updating the penalty parameter

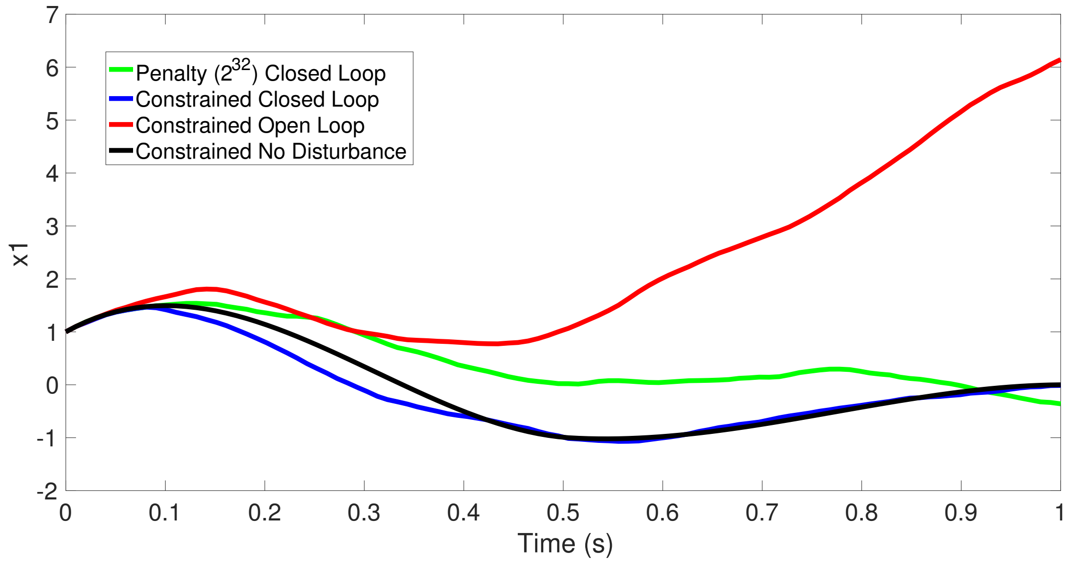


Figure 6.3: Robust constraint satisfaction for problem (6.38) subject to additive input noise.

until acceptable constraint satisfaction might be much slower than computing the analytic solution from the start.

Disturbance Rejection

Another benefit of the control policies we have generated is in robustly satisfying constraints. Consider again the example (6.38). Let us compare the performance of executing the open-loop control signal as would be generated when using a Gaussian elimination technique as discussed above, compared to executing the constrained feedback policies, in the presence of unforeseen disturbances. In Figure 6.3, we see the comparison of the open loop control policy compared to the feedback policy when executed on a 'true' system with dynamics when the input u_t is corrupted by unit Gaussian noise ($u_t \sim \mathcal{N}(0, 1)$). We see (as would be expected) that the open loop signal strays far from satisfying either of the equality constraints (6.38d) and (6.38e), where as by using the constrained feedback policies, they are still nearly satisfied. This is a purely empirical argument, but demonstrates a simple case in which the benefits of the generated control policies are seen. More in-depth analysis of the robustness properties of constraint-aware feedback policies can be seen in [133] for a time-invariant constraint, and a similar analysis could be done for the general constraint policies presented here, but is left for future work.

Application to Sequential Quadratic Programming

Due to the generality and computational efficiency of our method, we believe it is well-suited for algorithms for solving more complicated optimal control problems. In particular, consider the more general version of problem (6.10) where the cost functions might be non-quadratic or even non-convex, and the dynamic and auxiliary constraints might be non-linear. In this general form, computing solutions requires a non-convex optimization method. One prominent method for solving these types of problem is Sequential Quadratic Programming (SQP). A in-depth overview SQP methods can be found in [28] or [268].

When using an SQP approach to solving a non-convex version of (6.10), Newton's method is used to solve the KKT conditions of the problem [268]. Each iteration of Newton's method results in a linearly-constrained LQR problem, of which the solution provides an update to the solution of the non-convex problem. Therefore, because this procedure requires solving many constrained LQR problems, having an efficient means of computing the solutions to those subproblems is critical for an efficient solution to the non-convex problem.

If the solutions of constrained LQR subproblems generated in an SQP are only used as updates in an iterative procedure for generating a trajectory, it may seem unnecessary to generate feedback policies and an open-loop solution based approach might suffice. However, there has been much research into the advantages of *shooting* type methods for unconstrained variants of the nonlinear optimal control problem, such as in Differential Dynamic Programming [117]. These methods generate iterates by applying the open-loop controls updates on the nonlinear system dynamics, in effect projecting the iterate onto the manifold of dynamically feasible trajectories. A recent exploration into the benefits of these type of methods [95] has shown that generating iterates in this way can lead to improved rate of convergence of trajectories to solutions of the non-convex problem, but sometimes suffer instabilities when the underlying system dynamics are unstable. Using the feedback control *policies* to update the control signal as the nonlinear system trajectory diverges from the linear system trajectory such as in [230] and [157] can mitigate this instability while maintaining enhanced convergence properties.

Because our method is highly efficient, and because it can handle arbitrary constraints without making any assumptions about linear dependence or dimension, it is an excellent candidate for use in SQP algorithms for trajectory optimization. Therefore, using our method to compute solutions to sub-problems would be no worse than using a direct method in terms of versatility and computation time, and the feedback policies could potentially improve convergence as discussed in [95] and [94]. An in-depth analysis of how and when these policies can aid in convergence would be interesting, but is left for future work.

Conclusion

In summary, we have presented a method for computing feedback control policies for the general equality-constrained LQR problem. The method presented has a computational complexity that scales linearly with respect to the trajectory length. We demonstrated that in practice the computation of such policies is on the order of the fastest existing methods. We also showed that the control policies generated are useful in contexts of robustly satisfying constraints, and offered perspective on the use of our method in contexts of solving general trajectory optimization problems.

Chapter 7

Conclusions and Future Work

The contents of this dissertation have introduced a class of connected optimization problems ($EPNECs$), for which analysis and computational methods were developed. These results were applied to the context of a handful of problems arising in the context of autonomous system design, namely that of game-theoretic motion planing for autonomous vehicles, and solving optimization problems with ordered preferences.

The methodologies presented here apply more broadly to many other problems which could not be covered in this work. For example, problems ranging from motion planning through contact, to mechanism design and inverse game theory problems, to geometric problems with set-based constraints, all can be cast as $EPNECs$. Going forward, it will be of great interest to apply these techniques to those problems. Already, the work that has been developed here has enabled the computation to a wide range of problems which were previously inaccessible by existing approaches.

Regarding the methodologies themselves, in preceding chapters, particularly in Chapter 6, reference was made to some avenues for future work in this area. In what follows, three such avenues of future research are proposed and discussed.

7.1 Existence and Solvability

In chapters 2 and 3, some sufficient conditions were laid out for equilibrium problems with nested equilibrium constraints to have solutions. These conditions relied on a local piecewise linear representation of the equilibrium constraints, and then leveraged standard methods for determining whether solutions are guaranteed to exist to equilibrium problems with a local linear representation of the equilibrium constraints. Two approaches for establishing the guaranteed existence were proposed, one based on formulating the problem as a LCP and using corresponding existence theorems, or using general fixed-point arguments.

The problem with the LCP approach is that often existence theorems in the LCP literature covered in the references here focus on proving existence for any "q". In the case of the equilibrium problems of concern in this dissertation, we often do not care about proving existence for any q vector, but those lying in particular subspaces. Therefore often the LCP existence theorems are too general and therefore restrictive for our purposes.

On the contrary, the fixed point arguments are also often more general than needed for our purposes. Those arguments fail to exploit the linear structure of the equilibrium problems considered here.

It would be of strong interest to develop both non-trivial necessary conditions, as well as less restrictive sufficient conditions, for an equilibrium solution to exist to the $EPNECs$ presented in this work. For most practical problems, it cannot be predicted with the current tools whether or not equilibrium solutions exist. Solutions can only be attempted to found using computational search procedures such as Algorithm 4.

On that note, it would be of great interest to be able to characterize the class of $EPNECs$ such that Algorithm 4 is guaranteed to find solutions, if one exists. This would bestow greater confidence in the method by practitioners and widen its reach.

7.2 Unconstrained Shared Variables

One particular problem of great importance is that of shared control of decision variables. Consider as an example, a simple two-player game in which both players attempt to optimize over some shared variable $x \in \mathbb{R}^n$:

$$QP_1 := \underset{x \in \mathbb{R}^n}{\arg\min} \quad x^\intercal(\frac{1}{2}Q_1 x + q_1) \tag{7.1a}$$
$$\text{subject to} \quad A_1 x + b_1 \geq 0 \tag{7.1b}$$

$$QP_2 := \underset{x \in \mathbb{R}^n}{\arg\min} \quad x^\intercal(\frac{1}{2}Q_2 x + q_2) \tag{7.2a}$$
$$\text{subject to} \quad A_2 x + b_2 \geq 0 \tag{7.2b}$$

In the programs (7.1, 7.2), the entire variable x is optimized over by both players. When both programs are strictly convex, there exists a unique solution for both programs, which in general will be different. Therefore the concatenated necessary conditions of optimality for both players forms an over-constrained system, and it is not possible to find a solution which is optimal for both players. Instead, some kind of *fairness* constraint must be imposed upon the players, such that the loss each player experiences for having to play fair is equivalent.

One possible approach to this type of problem could be to have the deviation of cost for each player from their optimal value be equivalent:

$$(x_s^* - x_1^*)^\intercal(\frac{1}{2}Q_1(x_s^* - x_1^*) + q_1) = (x_s^* - x_2^*)^\intercal(\frac{1}{2}Q_2(x_s^* - x_2^*) + q_2), \tag{7.3}$$

where x_1^* solves QP_1, x_2^* solves QP_2, and x_s^* satisfies (7.1b,7.2b). Finding x_s^* results in a feasibility problem with quadratic equality constraints, and therefore is a non-convex problem. There are other reasons this formulation is not ideal, such as that it requires handling a quadratic equality constraint, which is something not required by either of the problems QP_1 or QP_2.

The development of a formulation which warrants both intuitive meaning for real-world problems (i.e. the solution does in fact constitute some notion of fairness), and also is amenable to computational solution approaches would have wide-ranging applicability. One such case in which such a development would be advantageous is described in the next section.

7.3 Equilibrium Problems with Nested, Non-Isolated Equilibrium Constraints

In Chapter 6, it was discussed that Algorithm 4 (and Algorithm 5) are not meaningful if the equilibrium points satisfying any of the equilibrium constraints in an L-layer $EPNEC$ are non-isolated, meaning there exist a continuum of solutions satisfying the constraint at some input parameter. The major problem associated with non-isolated solutions is that a piecewise linear representation of the mapping is no longer possible — in fact, the solution is impossible to represent as a function because the solution space of the equilibrium problem is locally a set-valued mapping. In the context of Problems of Ordered Preference, where each layer of the $EPNEC$ is only a single optimization problem, presumably Algorithm 4 could be extended to allow the variables x_{l+1} at layer l to be under specified. However, when each layer of the $EPNEC$ contains multiple optimization problems, this issue of under-specified shared variables discussed in the preceding section arises.

Resolving this issue and using it to extend Algorithm 4 to handle these under-constrained shared variables arising in the presence of non-isolated equilibria would greatly expand the applicability of the method. Specifically, some of the strict requirements on equilibria being isolated would no longer be necessary. This would apply to, for example, each of the connected optimization problems to be linear programs instead of quadratic programs.

Bibliography

[1] Joshua Achiam et al. “Constrained Policy Optimization”. In: *International Conference on Machine Learning*. 2017, pp. 22–31.

[2] Ali Ahmadzadeh et al. “Multi-vehicle path planning in dynamically changing environments”. In: *International Conference on Robotics and Automation*. IEEE. 2009, pp. 2449–2454.

[3] Anayo K Akametalu et al. “A minimum discounted reward hamilton-jacobi formulation for computing reachable sets”. In: *arXiv preprint arXiv:1809.00706* (2018).

[4] Anayo K Akametalu et al. “Reachability-based safe learning with Gaussian processes.” In: *53rd Conference on Decision and Control (CDC)*. Citeseer. 2014, pp. 1424–1431.

[5] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*. Vol. 26. Birkhäuser, 2012.

[6] Matthias Althoff and John M Dolan. “Set-based computation of vehicle behaviors for the online verification of autonomous vehicles”. In: *Intelligent Transportation Systems (ITSC)*. IEEE. 2011, pp. 1162–1167.

[7] Matthias Althoff and Bruce H Krogh. “Zonotope bundles for the efficient computation of reachable sets”. In: *2011 50th IEEE conference on decision and control and European control conference*. IEEE. 2011, pp. 6814–6821.

[8] Heni Ben Amor et al. “Interaction primitives for human-robot cooperation tasks”. In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 2831–2837.

[9] E. Anderson et al. *LAPACK Users’ Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999. ISBN: 0-89871-447-8 (paperback).

[10] Georges S Aoude et al. “Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns”. In: *Autonomous Robots* 35.1 (2013), pp. 51–76.

[11] Peter Arbenz et al. “A comparison of parallel solvers for diagonally dominant and general narrow-banded linear systems”. In: *Parallel and Distributed Computing Practices (PCDP) 2000*. Citeseer. 1999.

[12] Haoyu Bai et al. "Intention-aware online POMDP planning for autonomous driving in a crowd". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 454–460.

[13] Andrea Bajcsy et al. "A Scalable Framework For Real-Time Multi-Robot, Multi-Human Collision Avoidance". In: *arXiv preprint arXiv:1811.05929* (2018).

[14] Chris L Baker, Joshua B Tenenbaum, and Rebecca R Saxe. "Goal inference as inverse planning". In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 29. 2007.

[15] Tirthankar Bandyopadhyay et al. "Intention-aware motion planning". In: *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 475–491.

[16] Somil Bansal et al. "Hamilton-Jacobi Reachability: A Brief Overview and Recent Advances". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 2242–2253.

[17] Somil Bansal et al. "Safe sequential path planning of multi-vehicle systems under presence of disturbances and imperfect information". In: *Proc. Amer. Control Conf.* 2017, pp. 1–8.

[18] Tamer Başar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory*. 2nd. Vol. 23. SIAM, 1999.

[19] Tamer Başar and Geert Jan Olsder. *Dynamic noncooperative game theory*. SIAM, 1998.

[20] Randal W Beard and Timothy W McLain. "Multiple UAV cooperative search under collision avoidance and limited range communication constraints". In: *42nd Conference on Decision and Control (CDC)*. Vol. 1. IEEE. 2003, pp. 25–30.

[21] Kostas E Bekris and Lydia E Kavraki. "Greedy but safe replanning under kinodynamic constraints". In: *Robotics and Automation, 2007 IEEE International Conference on*. IEEE. 2007, pp. 704–710.

[22] Fethi Belkhouche. "Reactive path planning in a dynamic environment". In: *IEEE Transactions on Robotics* 25.4 (2009), pp. 902–911.

[23] Richard Bellman. *Dynamic programming*. Tech. rep. RAND CORP SANTA MONICA CA, 1956.

[24] John S Bellmgham et al. "Cooperative path planning for multiple UAVs in dynamic and uncertain environments". In: *Proc. of IEEE* (2002), pp. 2816–2822.

[25] Peter Benner, Enrique S Quintana-Orti, and Gregorio Quintana-Orti. "Solving linear-quadratic optimal control problems on parallel computers". In: *Optimisation Methods & Software* 23.6 (2008), pp. 879–909.

[26] Jur van den Berg. "Iterated LQR smoothing for locally-optimal feedback control of systems with non-linear dynamics and non-quadratic cost". In: *American Control Conference (ACC)*. IEEE. 2014, pp. 1912–1918.

[27] Felix Berkenkamp et al. "Safe model-based reinforcement learning with stability guarantees". In: *Advances in Neural Information Processing Systems*. 2017, pp. 908–918.

[28] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[29] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Vol. 5. Athena Scientific Belmont, MA, 1996.

[30] L Susan Blackford et al. *ScaLAPACK users' guide*. SIAM, 1997.

[31] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[32] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[33] Cynthia Breazeal, Jesse Gray, and Matt Berlin. "An embodied cognition approach to mindreading skills for socially intelligent robots". In: *The International Journal of Robotics Research* 28.5 (2009), pp. 656–680.

[34] Frank M Callier and Charles A Desoer. *Linear system theory*. Springer Science & Business Media, 2012.

[35] Mark Cannon, Weiheng Liao, and Basil Kouvaritakis. "Efficient MPC optimization using Pontryagin's minimum principle". In: *International Journal of Robust and Nonlinear Control* 18.8 (2008), pp. 831–844.

[36] Nilanjan Chakraborty et al. "An Implicit Time-Stepping Method for Multibody Systems with Intermittent Contact." In: *Robotics: Science and Systems*. 2007.

[37] Georgios C Chasparis and Jeff S Shamma. "Linear-programming-based multi-vehicle path planning with adversaries". In: *American Control Conference (ACC)*. IEEE. 2005, pp. 1072–1077.

[38] Jianyu Chen, Wei Zhan, and Masayoshi Tomizuka. "Constrained iterative LQR for on-road autonomous driving motion planning". In: *International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–7.

[39] Mo Chen, Sylvia Herbert, and Claire J Tomlin. "Fast reachable set approximations via state decoupling disturbances". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 191–196.

[40] Mo Chen, Jennifer C Shih, and Claire J Tomlin. "Multi-vehicle collision avoidance via hamilton-jacobi reachability and mixed integer programming". In: *55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 1695–1700.

[41] Mo Chen et al. "Decomposition of reachable sets and tubes for a class of nonlinear systems". In: *Transactions on Automatic Control* 63.11 (2018), pp. 3675–3688.

[42] Mo Chen et al. "Robust sequential path planning under disturbances and adversarial intruder". In: *arXiv preprint arXiv:1611.08364* (2016).

[43] Mo Chen et al. "Safe sequential path planning of multi-vehicle systems via double-obstacle Hamilton-Jacobi-Isaacs variational inequality". In: *2015 European Control Conference (ECC)*. IEEE. 2015, pp. 3304–3309.

[44] Glen Chou, Necmiye Ozay, and Dmitry Berenson. "Learning constraints from locally-optimal demonstrations under cost function uncertainty". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3682–3690.

[45] Yinlam Chow et al. "A Lyapunov-based Approach to Safe Reinforcement Learning". In: *Advances in neural information processing systems*. 2018.

[46] Yao-Li Chuang et al. "Multi-vehicle flocking: scalability of cooperative control algorithms using pairwise potentials". In: *Robotics and Automation, 2007 IEEE International Conference on*. IEEE. 2007, pp. 2292–2299.

[47] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. "ALGAMES: A fast solver for constrained dynamic games". In: *arXiv preprint arXiv:1910.09713* (2019).

[48] Richard W Cottle, Jong-Shi Pang, and Richard E Stone. *The linear complementarity problem*. SIAM, 2009.

[49] Navid Dadkhah and Bérénice Mettler. "Survey of motion planning literature in the presence of uncertainty: Considerations for UAV guidance". In: *Journal of Intelligent & Robotic Systems* 65.1-4 (2012), pp. 233–246.

[50] James W Demmel. *Applied numerical linear algebra*. Vol. 56. Siam, 1997.

[51] Bolei Di and Andrew Lamperski. "Local First-Order Algorithms for Constrained Nonlinear Dynamic Games". In: *2020 American Control Conference (ACC)*. IEEE. 2020, pp. 5358–5363.

[52] Bolei Di and Andrew Lamperski. "Newton's Method and Differential Dynamic Programming for Unconstrained Nonlinear Dynamic Games". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 4073–4078.

[53] Moritz Diehl, Hans Joachim Ferreau, and Niels Haverbeke. "Efficient numerical methods for nonlinear MPC and moving horizon estimation". In: *Nonlinear model predictive control*. Springer, 2009, pp. 391–417.

[54] Moritz Diehl et al. "Fast direct multiple shooting algorithms for optimal robot control". In: *Fast motions in biomechanics and robotics*. Springer, 2006, pp. 65–93.

[55] Hao Ding et al. "Human arm motion modeling and long-term prediction for safe and efficient human-robot-interaction". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2011, pp. 5875–5880.

[56] Jerry Ding et al. "Reachability calculations for automated aerial refueling". In: *47th Conference on Decision and Control (CDC)*. IEEE. 2008, pp. 3706–3712.

[57] Steven P Dirkse and Michael C Ferris. "The path solver: a nommonotone stabilization scheme for mixed complementarity problems". In: *Optimization methods and software* 5.2 (1995), pp. 123–156.

[58] Badis Djeridane and John Lygeros. "Neural approximation of PDE solutions: An application to reachability computations". In: *45th Conference on Decision and Control (CDC)*. IEEE. 2006, pp. 3034–3039.

[59] Engelbert Dockner, Gustav Feichtinger, and Steffen Jørgensen. "Tractable classes of nonzero-sum open-loop Nash differential games: theory and examples". In: *Journal of Optimization Theory and Applications* 45.2 (1985), pp. 179–197.

[60] Jack J Dongarra and Lennart Johnsson. "Solving banded systems on a parallel processor". In: *Parallel Computing* 5.1-2 (1987), pp. 219–246.

[61] Anca D Dragan and Siddhartha S Srinivasa. *Formalizing assistive teleoperation*. MIT Press, July, 2012.

[62] Axel Dreves et al. "On the solution of the KKT conditions of generalized Nash equilibrium problems". In: *SIAM Journal on Optimization* 21.3 (2011), pp. 1082–1108.

[63] Noel E Du Toit and Joel W Burdick. "Robot motion planning in dynamic, uncertain environments". In: *Transactions on Robotics* 28.1 (2012), pp. 101–115.

[64] Noel E Du Toit and Joel W Burdick. "Robot motion planning in dynamic, uncertain environments". In: *IEEE Transactions on Robotics* 28.1 (2012), pp. 101–115.

[65] Joseph C Dunn and Dimitri P Bertsekas. "Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems". In: *Journal of Optimization Theory and Applications* 63.1 (1989), pp. 23–38.

[66] Christophe Dutang. "Existence theorems for generalized Nash equilibrium problems: An analysis of assumptions". In: *Journal of Nonlinear Analysis and Optimization* 4.2 (2013), pp. 115–126.

[67] L. C. Evans and P. E. Souganidis. "Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations". In: *Indiana University mathematics journal* 33.5 (1984), pp. 773–797.

[68] L. C. Evans and P. E. Souganidis. "Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations". In: *Indiana University mathematics journal* 33.5 (1984), pp. 773–797.

[69] Lawrence C Evans and Panagiotis E Souganidis. *Differential Games and Representation Formulas for Solutions of Hamilton-Jacobi-Isaacs Equations.* Tech. rep. Wisconsin University Mathematics Research Center, 1983.

[70] Francisco Facchinei, Andreas Fischer, and Veronica Piccialli. "Generalized Nash equilibrium problems and Newton methods". In: *Mathematical Programming* 117.1-2 (2009), pp. 163–194.

[71] Francisco Facchinei and Christian Kanzow. "Generalized Nash equilibrium problems". In: *Annals of Operations Research* 175.1 (2010), pp. 177–211.

[72] Farbod Farshidian and J. Buchli. "Chance-Constrained Optimal Covariance Steering with Iterative Risk Allocation". In: *ArXiv* abs/1512.07173 (2015).

[73] Farbod Farshidian and J. Buchli. "Risk Sensitive, Nonlinear Optimal Control: Iterative Linear Exponential-Quadratic Optimal Control with Gaussian Noise". In: *ArXiv* abs/1512.07173 (2015).

[74] Farbod Farshidian et al. "Real-time motion planning of legged robots: A model predictive control approach". In: *arXiv preprint arXiv:1710.04029* (2017).

[75] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided cost learning: Deep inverse optimal control via policy optimization". In: *International Conference on Machine Learning*. 2016, pp. 49–58.

[76] Paolo Fiorini and Zvi Shiller. "Motion planning in dynamic environments using velocity obstacles". In: *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772.

[77] Jaime F Fisac and S Shankar Sastry. "The pursuit-evasion-defense differential game in dynamic constrained environments". In: *54th Conference on Decision and Control (CDC)*. IEEE. 2015, pp. 4549–4556.

[78] Jaime F Fisac et al. "Hierarchical game-theoretic planning for autonomous vehicles". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 9590–9596.

[79] Jaime F Fisac et al. "Probabilistically Safe Robot Planning with Confidence-Based Human Predictions". In: *Robotics: Science & Systems*. 2018.

[80] Jaime F Fisac et al. "Reach-avoid problems with time-varying dynamics, targets and constraints". In: *Proceedings of the 18th international conference on hybrid systems: computation and control*. 2015, pp. 11–20.

[81] Jaime F. Fisac* et al. "A general safety framework for learning-based control in uncertain robotic systems". In: *IEEE Transactions on Automatic Control* (2018).

[82] Thierry Fraichard and Hajime Asama. "Inevitable collision states: A step towards safer robots?" In: *Advanced Robotics* 18.10 (2004), pp. 1001–1024.

[83] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. "Real-time motion planning for agile autonomous vehicles". In: *Journal of guidance, control, and dynamics* 25.1 (2002), pp. 116–129.

[84] Goran Frehse et al. "SpaceEx: Scalable verification of hybrid systems". In: *International Conference on Computer Aided Verification*. Springer. 2011, pp. 379–395.

[85] David Fridovich-Keil, Jaime F Fisac, and Claire J Tomlin. "Safely probabilistically complete real-time planning and exploration in unknown environments". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7470–7476.

[86] David Fridovich-Keil, Vicenc Rubies-Royo, and Claire J Tomlin. "An iterative quadratic method for general-sum differential games with feedback linearizable dynamics". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2216–2222.

[87] David Fridovich-Keil et al. "Confidence-aware motion prediction for real-time collision avoidance". In: *The International Journal of Robotics Research* 39.2-3 (2019), pp. 250–265.

[88] David Fridovich-Keil et al. "Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1475–1481.

[89] David Fridovich-Keil et al. "Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1475–1481.

[90] David Fridovich-Keil* et al. "Planning, Fast and Slow: A Framework for Adaptive Real-Time Safe Trajectory Planning." In: *International Conference on Robotics and Automation (ICRA)* (2018).

[91] Gianluca Frison and John Bagterp Jørgensen. "Algorithms and methods for high-performance model predictive control". In: (2016).

[92] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 3067–3074.

[93] Eloy Garcia, David W Casbeer, and Meir Pachter. "Design and analysis of state-feedback optimal strategies for the differential game of active defense". In: *IEEE Transactions on Automatic Control* 64.2 (2019), pp. 553–568.

[94] Markus Giftthaler and Jonas Buchli. "A projection approach to equality constrained iterative linear quadratic optimal control". In: *Humanoid Robotics (Humanoids), 2017 IEEE-RAS 17th International Conference on*. IEEE. 2017, pp. 61–66.

[95] Markus Giftthaler et al. "A Family of Iterative Gauss-Newton Shooting Methods for Nonlinear Optimal Control". In: *arXiv preprint arXiv:1711.11006* (2017).

[96] Philip E. Gill, Walter Murray, and Michael A. Saunders. "SNOPT: An SQP algorithm for large-scale constrained optimization". In: *SIAM Rev.* 47 (2005), pp. 99–131.

[97] Philip E. Gill et al. *User's Guide for SNOPT 7.6: Software for Large-Scale Nonlinear Programming*. Center for Computational Mathematics Report CCoM 17-1. La Jolla, CA: Department of Mathematics, University of California, San Diego, 2017.

[98] Philip E. Gill et al. *User's Guide for SNOPT 7.7: Software for Large-Scale Nonlinear Programming*. Center for Computational Mathematics Report CCoM 18-1. La Jolla, CA: Department of Mathematics, University of California, San Diego, 2018.

[99] Rafal Goebel and Maxim Subbotin. "Continuous time constrained linear quadratic regulator-convex duality approach". In: *American Control Conference, 2005. Proceedings of the 2005*. IEEE. 2005, pp. 1401–1406.

[100] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. Johns Hopkins Series in the Mathematical Sciences 3. The Johns Hopkins University Press, 1989.

[101] Michael Green and David JN Limebeer. *Linear robust control*. Courier Corporation, 2012.

[102] Mark R Greenstreet and Ian Mitchell. "Reachability analysis using polygonal projections". In: *International Workshop on Hybrid Systems: Computation and Control*. Springer. 1999, pp. 103–116.

[103] Peng Hang et al. "An Integrated Framework of Decision Making and Motion Planning for Autonomous Vehicles Considering Social Behaviors". In: *IEEE Transactions on Vehicular Technology* (2020).

[104] Jason Hardy and Mark Campbell. "Contingency planning over probabilistic obstacle predictions for autonomous road vehicles". In: *IEEE Transactions on Robotics* 29.4 (2013), pp. 913–929.

[105] Peter E Hart, Nils J Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths". In: *Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[106] Kelsey P Hawkins et al. "Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration". In: *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*. IEEE. 2013, pp. 499–506.

[107] Hooshang Hemami and B t Wyman. "Modeling and control of constrained dynamic systems with application to biped locomotion in the frontal plane". In: *IEEE Transactions on Automatic Control* 24.4 (1979), pp. 526–535.

[108] Sylvia L. Herbert* et al. "FaSTrack: a Modular Framework for Fast and Guaranteed Safe Motion Planning". In: *56th Conference on Decision and Control (CDC)* (2017).

[109] Jorge Herrera de la Cruz, Benjamin Ivorra, and Ángel M Ramos. "An Algorithm for Solving a Class of Multiplayer Feedback-Nash Differential Games". In: *Mathematical Problems in Engineering* 2019 (2019).

[110] Saif A Al-Hiddabi. "Quadrotor control using feedback linearization with dynamic extension". In: *2009 6th International Symposium on Mechatronics and its Applications*. IEEE. 2009, pp. 1–3.

[111] John R Hoare and Lynne E Parker. "Using on-line conditional random fields to determine human intent for peer-to-peer human robot teaming". In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE. 2010, pp. 4914–4921.

[112] Gabriel M Hoffmann and Claire J Tomlin. "Decentralized cooperative collision avoidance for acceleration constrained vehicles". In: *47th Conference on Decision and Control (CDC)*. IEEE. 2008, pp. 4357–4363.

[113] Maxime Hubert et al. "Bouncing dynamics of a spring". In: *Physica D: Nonlinear Phenomena* 272 (2014), pp. 1–7.

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

[115] Rufus Isaacs. *Games of pursuit*. Tech. rep. Rand Corporation, 1951.

[116] Boris Ivanovic and Marco Pavone. "The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 2375–2384.

[117] David H Jacobson and David Q Mayne. "Differential dynamic programming". In: (1970).

[118] J Ye Jane. "Constraint qualifications and optimality conditions in bilevel optimization". In: *Bilevel Optimization*. Springer, 2020, pp. 227–251.

[119] Lucas Janson, Tommy Hu, and Marco Pavone. "Safe Motion Planning in Unknown Environments: Optimality Benchmarks and Tractable Policies". In: *Robotics: Science & Systems*. 2018.

[120] Shervin Javdani, Siddhartha S Srinivasa, and J Andrew Bagnell. "Shared autonomy via hindsight optimization". In: *arXiv preprint arXiv:1503.07619* (2015).

[121] Houyuan Jiang and Daniel Ralph. "QPECgen, a MATLAB generator for mathematical programs with quadratic objectives and affine variational inequality constraints". In: *Computational Optimization and Applications* 13.1-3 (1999), pp. 25–59.

[122] CD Johnson. "Stabilization of linear dynamical systems with respect to arbitrary linear subspaces". In: *Journal of Mathematical Analysis and Applications* 44.1 (1973), pp. 175–186.

[123] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey". In: *Journal of Artificial Intelligence Research* 4 (1996), pp. 237–285.

[124] Rudolf Emil Kalman et al. "Contributions to the theory of optimal control". In: *Bol. Soc. Mat. Mexicana* 5.2 (1960), pp. 102–119.

[125] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *International Journal of Robotics Research* 30.7 (2011), pp. 846–894.

[126] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *The international journal of robotics research* 30.7 (2011), pp. 846–894.

[127] Sertac Karaman and Emilio Frazzoli. "Sampling-based optimal motion planning for non-holonomic dynamical systems". In: *International Conference on Robotics and Automation (ICRA*. IEEE. 2013, pp. 5041–5047.

[128] Shahab Karimi and Ardalan Vahidi. "Receding Horizon Motion Planning for Automated Lane Change and Merge Using Monte Carlo Tree Search and Level-K Game Theory". In: *2020 American Control Conference (ACC)*. IEEE. 2020, pp. 1223–1228.

[129] Hassan K Khalil. *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002.

[130] Youngdae Kim and Michael C Ferris. "Solving equilibrium problems using extended mathematical programming". In: *Mathematical programming computation* 11.3 (2019), pp. 457–501.

[131] Youngdae Kim, Sven Leyffer, and Todd Munson. "Mpec methods for bilevel optimization problems". In: *Bilevel Optimization*. Springer, 2020, pp. 335–360.

[132] Nikita Kitaev et al. "Physics-based trajectory optimization for grasping in cluttered environments". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 3102–3109.

[133] Sangho Ko and Robert R Bitmead. "Optimal control for linear systems with state equality constraints". In: *Automatica* 43.9 (2007), pp. 1573–1582.

[134] Jonas Koenemann et al. "Whole-body model-predictive control applied to the HRP-2 humanoid". In: *International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 3346–3351.

[135] Sven Koenig and Maxim Likhachev. "Fast replanning for navigation in unknown terrain". In: *Transactions on Robotics* 21.3 (2005), pp. 354–363.

[136] Hema Swetha Koppula and Ashutosh Saxena. "Anticipating human activities for reactive robotic response." In: *International Conference on Intelligent Robots and Systems (IROS)*. 2013, p. 2071.

[137] Markus Koschi et al. "Set-Based Prediction of Pedestrians in Urban Environments Considering Formalized Traffic Rules". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (2018), pp. 2704–2711.

[138] G Kossioris et al. "Feedback Nash equilibria for non-linear differential games in pollution control". In: *Journal of Economic Dynamics and Control* 32.4 (2008), pp. 1312–1331.

[139] Henrik Kretzschmar et al. "Socially compliant mobile robot navigation via inverse reinforcement learning". In: *International Journal of Robotics Research* 35.11 (2016), pp. 1289–1307.

[140] Scott Kuindersma, Frank Permenter, and Russ Tedrake. "An efficiently solvable quadratic program for stabilizing dynamic locomotion". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 2589–2594.

[141] Ankur A Kulkarni. *Generalized Nash games with shared constraints: existence, efficiency, refinement and equilibrium constraints*. University of Illinois at Urbana-Champaign, 2010.

[142] Alexander B Kurzhanski and Pravin Varaiya. "Ellipsoidal techniques for reachability analysis: internal approximation". In: *Systems & Control Letters* 41.3 (2000), pp. 201–211.

[143] Alexander B Kurzhanski and Pravin Varaiya. "Ellipsoidal techniques for reachability analysis: internal approximation". In: *Systems & control letters* 41.3 (2000), pp. 201–211.

[144] Alexander B Kurzhanski and Pravin Varaiya. "On ellipsoidal techniques for reachability analysis. part ii: Internal approximations box-valued constraints". In: *Optimization Methods and Software* 17.2 (2002), pp. 207–237.

[145] Alexander B Kurzhanski and Pravin Varaiya. "On ellipsoidal techniques for reachability analysis. part ii: Internal approximations box-valued constraints". In: *Optimization methods and software* 17.2 (2002), pp. 207–237.

[146] Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*. Vol. 1. Wiley-Interscience New York, 1972.

[147] F. Laine and C. Tomlin. "Parallelizing LQR Computation Through Endpoint-Explicit Riccati Recursion". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. 2019, pp. 1395–1402. DOI: 10.1109/CDC40024.2019.9029974.

[148] Forrest Laine and Claire Tomlin. "Efficient Computation of Feedback Control for Constrained Systems". In: *arXiv preprint arXiv:1807.00794* (2018).

[149] Forrest Laine and Claire Tomlin. "Efficient computation of feedback control for equality-constrained LQR". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 6748–6754.

[150] Forrest Laine et al. *Multi-Hypothesis Interactions in Game-Theoretic Motion Planning*. 2020. arXiv: 2011.06047 [cs.RO].

[151] Forrest Laine et al. "The Computation of Approximate Generalized Feedback Nash Equilibria". In: *arXiv preprint arXiv:2101.02900* (2021).

[152] Emmett Lalish, Kristi A Morgansen, and Takashi Tsukamaki. "Decentralized reactive collision avoidance for multiple unicycle-type vehicles". In: *American Control Conference (ACC)*. IEEE. 2008, pp. 5055–5061.

[153] Joseph P LaSalle. "The time optimal control problem". In: *Contributions to the theory of nonlinear oscillations* 5 (1960), pp. 1–24.

[154] Przemyslaw A Lasota and Julie A Shah. "Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration". In: *Human factors* 57.1 (2015), pp. 21–33.

[155] Chuck L Lawson et al. "Basic linear algebra subprograms for Fortran usage". In: (1977).

[156] TY Li and Z Gajic. "Lyapunov iterations for solving coupled algebraic Riccati equations of Nash differential games and algebraic Riccati equations of zero-sum games". In: *New trends in dynamic games and applications*. Springer, 1995, pp. 333–351.

[157] Weiwei Li and Emanuel Todorov. "Iterative linear quadratic regulator design for nonlinear biological movement systems." In: *ICINCO*. 2004, pp. 222–229.

[158] Feng-Li Lian and Richard Murray. "Real-time trajectory generation for the cooperative path planning of multi-vehicle systems". In: *41st Conference on Decision and Control (CDC)*. Vol. 4. IEEE. 2002, pp. 3766–3769.

[159] Yucong Lin and Srikanth Saripalli. "Collision avoidance for UAVs using reachable sets". In: *International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2015, pp. 226–235.

[160] Alexander Liniger and John Lygeros. "A noncooperative game approach to autonomous racing". In: *IEEE Transactions on Control Systems Technology* 28.3 (2019), pp. 884–897.

[161] P-L Lions and Panagiotis E Souganidis. "Differential games, optimal control and directional derivatives of viscosity solutions of Bellman's and Isaacs' equations". In: *Journal on Control and Optimization* 23.4 (1985), pp. 566–583.

[162] Chang Liu et al. "Goal inference improves objective and perceived performance in human-robot collaboration". In: *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2016, pp. 940–948.

[163] Shih-Yuan Liu et al. "Evasion of a team of dubins vehicles from a hidden pursuer". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 6771–6776.

[164] J. Löfberg. "YALMIP : A Toolbox for Modeling and Optimization in MATLAB". In: *In Proceedings of the CACSD Conference*. Taipei, Taiwan, 2004.

[165] Tzon-Tzer Lu and Sheng-Hua Shiou. "Inverses of 2× 2 block matrices". In: *Computers & Mathematics with Applications* 43.1-2 (2002), pp. 119–129.

[166] R. Duncan Luce. *Individual choice behavior: A theoretical analysis*. Wiley, 1959.

[167] John N Maidens et al. "Lagrangian methods for approximating the viability kernel in high-dimensional systems". In: *Automatica* 49.7 (2013), pp. 2017–2029.

[168] Jim Mainprice and Dmitry Berenson. "Human-robot collaborative manipulation planning using early prediction of human motion". In: *International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2013, pp. 299–306.

[169] Jim Mainprice, Rafi Hayne, and Dmitry Berenson. "Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 885–892.

[170] Anirudha Majumdar et al. “Convex optimization of nonlinear feedback controllers via occupation measures”. In: *The International Journal of Robotics Research* 33.9 (2014), pp. 1209–1230.

[171] Zachary Manchester and Scott Kuindersma. “Variational Contact-Implicit Trajectory Optimization”. In: *Proceedings of the International Symposium on Robotics Research (ISRR), Puerto Varas, Chile*. 2017.

[172] José B Mare and José A De Doná. “Solution of the input-constrained LQR problem using dynamic programming”. In: *Systems & control letters* 56.5 (2007), pp. 342–348.

[173] Kostas Margellos and John Lygeros. “Hamilton–Jacobi formulation for reach–avoid differential games”. In: *Transactions on Automatic Control* 56.8 (2011), pp. 1849–1861.

[174] Mieke Massink and Nicoletta De Francesco. “Modelling free flight with collision avoidance”. In: *International Conference on Engineering of Complex Computer Systems*. IEEE. 2001, pp. 270–279.

[175] David Mayne. “A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems”. In: *International Journal of Control* 3.1 (1966), pp. 85–95.

[176] Eric Mazumdar and Lillian J Ratliff. “On the convergence of gradient-based learning in continuous games”. In: *ArXiv e-prints* (2018).

[177] Eric Mazumdar et al. “Policy-Gradient Algorithms Have No Guarantees of Convergence in Linear Quadratic Games”. In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 2020, pp. 860–868.

[178] Eric V Mazumdar, Michael I Jordan, and S Shankar Sastry. “On finding local nash equilibria (and only local nash equilibria) in zero-sum games”. In: *arXiv preprint arXiv:1901.00838* (2019).

[179] N Harris McClamroch and Danwei Wang. “Feedback stabilization and tracking of constrained robots”. In: *IEEE Transactions on Automatic Control* 33.5 (1988), pp. 419–426.

[180] Ian Mitchell. *A Toolbox of Level Set Methods*. `http://people.cs.ubc.ca/~mitchell/ToolboxLS/index.html`. 2009.

[181] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. “A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games”. In: *IEEE Transactions on automatic control* 50.7 (2005), pp. 947–957.

[182] Ian M. Mitchell, A. M. Bayen, and C. J. Tomlin. “A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games”. In: *Transactions on Automatic Control* 50.7 (2005), pp. 947–957. DOI: `10.1109/TAC.2005.851439`. URL: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1463302`.

[183] Teodor M Moldovan and Pieter Abbeel. “Risk aversion in Markov decision processes via near optimal Chernoff bounds”. In: *Advances in neural information processing systems*. 2012, pp. 3131–3139.

[184] Teodor Mihai Moldovan and Pieter Abbeel. “Safe exploration in Markov decision processes”. In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. Omnipress. 2012, pp. 1451–1458.

[185] H Mukai et al. “Sequential linear quadratic method for differential games”. In: *Proc. 2nd DARPA-JFACC Symposium on Advances in Enterprise Control*. Citeseer. 2000, pp. 159–168.

[186] Katta G Murty and Feng-Tien Yu. *Linear complementarity, linear and nonlinear programming*. Vol. 3. Citeseer, 1988.

[187] John F Nash et al. “Equilibrium points in n-person games”. In: *Proceedings of the national academy of sciences* 36.1 (1950), pp. 48–49.

[188] Andrew Y Ng, Stuart J Russell, et al. “Algorithms for inverse reinforcement learning.” In: *Icml*. 2000, pp. 663–670.

[189] Truong-Huy Dinh Nguyen et al. “CAPIR: Collaborative Action Planning with Intention Recognition.” In: *AIIDE*. 2011.

[190] Stefanos Nikolaidis et al. “Efficient model learning from joint-action demonstrations for human-robot collaborative tasks”. In: *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*. ACM. 2015, pp. 189–196.

[191] Haruki Nishimura et al. “RAT iLQR: A Risk Auto-Tuning Controller to Optimally Account for Stochastic Model Mismatch”. In: *arXiv preprint arXiv:2010.08174* (2020).

[192] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[193] Jorge Nocedal and Stephen J Wright. *Sequential quadratic programming*. Springer, 2006.

[194] Reza Olfati-Saber and Richard M Murray. “Distributed cooperative control of multiple vehicle formations using structural potential functions”. In: *IFAC world congress*. Vol. 15. 1. Citeseer. 2002, pp. 242–248.

[195] M. Ono, L. Blackmore, and B. C. Williams. “Chance constrained finite horizon optimal control with nonconvex constraints”. In: *Proceedings of the 2010 American Control Conference*. 2010, pp. 1145–1152.

[196] Jong-Shi Pang and Gesualdo Scutari. “Nonconvex Games with Side Constraints”. In: *SIAM J. on Optimization* 21.4 (Dec. 2011), pp. 1491–1522. ISSN: 1052-6234. DOI: 10.1137/100811787. URL: https://doi.org/10.1137/100811787.

[197] Jong-Shi Pang and Gesualdo Scutari. “Nonconvex Games with Side Constraints”. In: *SIAM J. on Optimization* 21.4 (Dec. 2011), pp. 1491–1522. ISSN: 1052-6234. DOI: 10.1137/100811787. URL: https://doi.org/10.1137/100811787.

[198] Christos H Papadimitriou. "The complexity of finding Nash equilibria". In: *Algorithmic game theory* 2 (2007), p. 30.

[199] Jung Hun Park, Soohee Han, and Wook Hyun Kwon. "LQ tracking controls with fixed terminal states and their application to receding horizon controls". In: *Systems & Control Letters* 57.9 (2008), pp. 772–777.

[200] C. Pek and M. Althoff. "Computationally Efficient Fail-safe Trajectory Planning for Self-driving Vehicles Using Convex Optimization". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (2018), pp. 1447–1454.

[201] C. Pek and M. Althoff. "Computationally Efficient Fail-safe Trajectory Planning for Self-driving Vehicles Using Convex Optimization". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 1447–1454.

[202] Stefania Pellegrinelli et al. "Human-robot shared workspace collaboration via hindsight optimization". In: *International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 831–838.

[203] Claudia Pérez-D'Arpino and Julie A Shah. "Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 6175–6182.

[204] Lasse Peters et al. "Inference-Based Strategy Alignment for General-Sum Differential Games". In: *arXiv preprint arXiv:2002.04354* (2020).

[205] Ovanes Petrosian and Anna Tur. "Hamilton-Jacobi-Bellman Equations for Non-cooperative Differential Games with Continuous Updating". In: *International Conference on Mathematical Optimization Theory and Operations Research*. Springer. 2019, pp. 178–191.

[206] Michael Posa, Cecilia Cantu, and Russ Tedrake. "A direct method for trajectory optimization of rigid bodies through contact". In: *The International Journal of Robotics Research* 33.1 (2014), pp. 69–81.

[207] Michael Posa, Scott Kuindersma, and Russ Tedrake. "Optimization and stabilization of trajectories for constrained dynamical systems". In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1366–1373.

[208] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*. Vol. 703. John Wiley & Sons, 2007.

[209] Morgan Quigley et al. "ROS: an Open-Source Robot Operating System". In: *ICRA Workshop on Open Source Software*. 2009.

[210] Deepak Ramachandran and Eyal Amir. "Bayesian inverse reinforcement learning". In: *Urbana* 51.61801 (2007), pp. 1–4.

[211] Lillian J Ratliff, Samuel A Burden, and S Shankar Sastry. "On the characterization of local Nash equilibria in continuous games". In: *Transactions on Automatic Control* 61.8 (2016), pp. 2301–2307.

[212] P. V. Reddy and G. Zaccour. "Feedback Nash Equilibria in Linear-Quadratic Difference Games With Constraints". In: *IEEE Transactions on Automatic Control* 62.2 (2017), pp. 590–604. DOI: 10.1109/TAC.2016.2555879.

[213] Puduru Viswanadha Reddy and Georges Zaccour. "Open-loop and feedback Nash equilibria in constrained linear–quadratic dynamic games played over event trees". In: *Automatica* 107 (2019), pp. 162–174.

[214] Arthur Richards and Jonathan P How. "Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility". In: *American Control Conference (ACC)*. Vol. 5. IEEE. 2003, pp. 4034–4040.

[215] Stefan Richter, Colin Neil Jones, and Manfred Morari. "Computational complexity certification for real-time MPC with input constraints based on the fast gradient method". In: *Transactions on Automatic Control* 57.6 (2012), pp. 1391–1403.

[216] Stefan Richter, Colin Neil Jones, and Manfred Morari. "Computational complexity certification for real-time MPC with input constraints based on the fast gradient method". In: *IEEE Transactions on Automatic Control* 57.6 (2012), pp. 1391–1403.

[217] D Reed Robinson et al. "An Efficient Algorithm for Optimal Trajectory Generation for Heterogeneous Multi-Agent Systems in Non-Convex Environments". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 1215–1222.

[218] Ugo Rosolia and Francesco Borrelli. "Learning model predictive control for iterative tasks. a data-driven control framework". In: *Transactions on Automatic Control* 63.7 (2018), pp. 1883–1896.

[219] Vicenc Rubies Royo et al. "Classification-based Approximate Reachability with Guarantees Applied to Safe Trajectory Tracking". In: *arXiv preprint arXiv:1803.03237* (2018).

[220] Vicenç Rubies-Royo et al. "A Classification-based Approach for Approximate Reachability". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7697–7704.

[221] Vicenç Rubies-Royo et al. "A Classification-based Approach for Approximate Reachability". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7697–7704.

[222] Dorsa Sadigh et al. "Information Gathering Actions over Human Internal State". In: *International Conference on Intelligent Robots and Systems (IROS)* (2016).

[223] Dorsa Sadigh et al. "Planning for autonomous cars that leverage effects on human actions." In: *Robotics: Science & Systems*. Vol. 2. Ann Arbor, MI, USA. 2016.

[224] Edward Schmerling et al. "Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction". In: *arXiv preprint arXiv:1710.09483* (2017).

[225] Edward Schmerling et al. “Multimodal probabilistic model-based planning for human-robot interaction”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–9.

[226] Tom Schouwenaars, Jonathan How, and Eric Feron. “Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees”. In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. 2004, p. 5141.

[227] Wilko Schwarting et al. “Stochastic Dynamic Games in Belief Space”. In: *arXiv preprint arXiv:1909.06963* (2019).

[228] Dexter RR Scobee and S Shankar Sastry. “Maximum likelihood constraint inference for inverse reinforcement learning”. In: *arXiv preprint arXiv:1909.05477* (2019).

[229] Pierre OM Scokaert and James B Rawlings. “Constrained linear quadratic regulation”. In: *IEEE Transactions on automatic control* 43.8 (1998), pp. 1163–1169.

[230] Athanasios Sideris and James E Bobrow. “An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems”. In: *IEEE Transactions on Automatic Control* 50.12 (2005), pp. 2043–2047.

[231] Athanasios Sideris and Luis A Rodriguez. “A Riccati approach for constrained linear quadratic optimal control”. In: *International Journal of Control* 84.2 (2011), pp. 370–380.

[232] Alp Simsek, Asuman Ozdaglar, and Daron Acemoglu. “On the uniqueness of solutions for nonlinear and mixed complementarity problems”. In: *Working Paper* (2005).

[233] Sumeet Singh et al. “Robust online motion planning via contraction theory and convex optimization”. In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2017.

[234] Sumeet Singh et al. “Robust online motion planning via contraction theory and convex optimization”. In: *Proc. IEEE Int. Conf. Robotics and Automation*. 2017.

[235] Sumeet Singh et al. “Robust Tracking with Model Mismatch for Fast and Safe Planning: an SOS Optimization Approach”. In: *arXiv preprint arXiv:1808.00649* (2018).

[236] Riccardo Spica et al. “A real-time game theoretic planner for autonomous two-player drone racing”. In: *IEEE Transactions on Robotics* 36.5 (2020), pp. 1389–1403.

[237] Alan W Starr and Yu-Chi Ho. “Further properties of nonzero-sum differential games”. In: *Journal of Optimization Theory and Applications* 3.4 (1969), pp. 207–219.

[238] Alan W Starr and Yu-Chi Ho. “Nonzero-sum differential games”. In: *Journal of Optimization Theory and Applications* 3.3 (1969), pp. 184–206.

[239] Anthony Stentz. “Optimal and efficient path planning for partially-known environments”. In: *International Conference on Robotics and Automation (ICRA)*. Vol. 94. 1994, pp. 3310–3317.

[240] David Stewart and Jeffrey C Trinkle. “An implicit time-stepping scheme for rigid body dynamics with coulomb friction”. In: *Robotics and Automation, 2000. Proceedings. ICRA’00. IEEE International Conference on*. Vol. 1. IEEE. 2000, pp. 162–169.

[241] Dušan M Stipanović et al. “Cooperative avoidance control for multiagent systems”. In: *Journal of Dynamic Systems, Measurement, and Control* 129.5 (2007), pp. 699–707.

[242] Che-Lin Su. *Equilibrium problems with equilibrium constraints: Stationarities, algorithms, and applications*. Stanford University, 2005.

[243] Ioan A Şucan, Mark Moll, and Lydia E Kavraki. “The open motion planning library”. In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82.

[244] Zachary N Sunberg and Mykel J Kochenderfer. “Online algorithms for POMDPs with continuous state, action, and observation spaces”. In: *Twenty-Eighth International Conference on Automated Planning and Scheduling*. 2018.

[245] P Svestka, JC Latombe, and LE Overmars Kavraki. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[246] Ryo Takei et al. “Time-optimal multi-stage motion planning with guaranteed collision avoidance via an open-loop game formulation”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 323–329.

[247] Akio Tanikawa, Hiro Mukai, and Min Xu. “Local Convergence of the Sequential Quadratic Method for Differential Games”. In: *Transactions of the Institute of Systems, Control and Information Engineers* 25.12 (2012), pp. 349–357.

[248] A. Tanwani and Q. Zhu. “Feedback Nash Equilibrium for Markov Jump Games under Differential-Algebraic Constraints with Application to Robust Control”. In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 1124–1129. DOI: 10.1109/CDC.2018.8619628.

[249] Aneel Tanwani and Quanyan Zhu. “Feedback Nash equilibrium for randomly switching differential-algebraic games”. In: *IEEE Transactions on Automatic Control* (2019).

[250] Yuval Tassa, Tom Erez, and Emanuel Todorov. “Synthesis and stabilization of complex behaviors through online trajectory optimization”. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 4906–4913.

[251] Yuval Tassa, Nicolas Mansard, and Emo Todorov. “Control-limited differential dynamic programming”. In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 1168–1175.

[252] Emanuel Todorov and Weiwei Li. “A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems”. In: *American Control Conference (ACC)*. IEEE. 2005, pp. 300–306.

[253] Peter Trautman and Andreas Krause. "Unfreezing the robot: Navigation in dense, interacting crowds". In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE. 2010, pp. 797–803.

[254] Vladimir Turetsky, Valery Y Glizer, and Josef Shinar. "Robust trajectory tracking: differential game/cheap control approach". In: *International Journal of Systems Science* 45.11 (2014), pp. 2260–2274.

[255] Jur Van den Berg, Ming Lin, and Dinesh Manocha. "Reciprocal velocity obstacles for real-time multi-agent navigation". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2008, pp. 1928–1935.

[256] Bart PG Van Parys et al. "Distributionally robust control of constrained stochastic systems". In: *IEEE Transactions on Automatic Control* 61.2 (2015), pp. 430–442.

[257] S. Vaskov et al. "Not-at-Fault Driving in Traffic: A Reachability-Based Approach". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 2785–2790.

[258] Marcell Vazquez-Chanlatte and Sanjit A Seshia. "Maximum Causal Entropy Specification Inference from Demonstrations". In: *International Conference on Computer Aided Verification*. Springer. 2020, pp. 255–278.

[259] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton University Press Princeton, NJ, 1945.

[260] A Wächter and L T Biegler. "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming". In: *Mathematical Programming* 106.1 (2006), pp. 25–57.

[261] Rahee Walambe et al. "Optimal trajectory generation for car-type mobile robot using spline interpolation". In: *IFAC-PapersOnLine* 49.1 (2016), pp. 601–606.

[262] Mingyu Wang et al. "Game Theoretic Planning for Self-Driving Cars in Competitive Scenarios". In: *Robotics: Science & Systems*. 2019.

[263] Zijian Wang, Riccardo Spica, and Mac Schwager. "Game Theoretic Motion Planning for Multi-Robot Racing". In: (2018).

[264] Zijian Wang, Riccardo Spica, and Mac Schwager. "Game Theoretic Motion Planning for Multi-robot Racing". In: *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 225–238.

[265] Zijian Wang, Tim Taubner, and Mac Schwager. "Multi-agent sensitivity enhanced iterative best response: A real-time game theoretic planner for drone racing in 3D environments". In: *Robotics and Autonomous Systems* 125 (2020), p. 103410.

[266] Tyler Westenbroek et al. "Feedback Linearization for Unknown Systems via Reinforcement Learning". In: *arXiv preprint arXiv:1910.13272* (2019).

[267] Tyler Westenbroek et al. "Technical Report: Adaptive Control for Linearizable Systems Using On-Policy Reinforcement Learning". In: *arXiv preprint arXiv:2004.02766* (2020).

[268] Stephen Wright and Jorge Nocedal. "Numerical optimization". In: *Springer Science* 35.67-68 (1999), p. 7.

[269] Stephen J Wright. *Applying new optimization algorithms to more predictive control*. Tech. rep. Argonne National Lab., IL (United States), 1996.

[270] Stephen J Wright. "Interior point methods for optimal control of discrete time systems". In: *Journal of Optimization Theory and Applications* 77.1 (1993), pp. 161–187.

[271] Stephen J Wright. "Solution of discrete-time optimal control problems on parallel computers". In: *Parallel Computing* 16.2-3 (1990), pp. 221–237.

[272] Albert Wu and Jonathan P How. "Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles". In: *Autonomous robots* 32.3 (2012), pp. 227–242.

[273] Zhaoming Xie, C Karen Liu, and Kris Hauser. "Differential dynamic programming with nonlinear constraints". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 695–702.

[274] Wenda Xu et al. "Motion planning under uncertainty for on-road autonomous driving". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 2507–2512.

[275] Brian Yamauchi. "A frontier-based approach for autonomous exploration". In: *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*. IEEE. 1997, pp. 146–151.

[276] Brian Yamauchi. "Frontier-based exploration using multiple robots". In: *Proceedings of the second international conference on Autonomous agents*. ACM. 1998, pp. 47–53.

[277] Luke Yoder and Sebastian Scherer. "Autonomous exploration for infrastructure modeling with a micro aerial vehicle". In: *Field and service robotics*. Springer. 2016, pp. 427–440.

[278] Ming-Yuan Yu, R. Vasudevan, and Matthew Johnson-Roberson. "Risk Assessment and Planning with Bidirectional Reachability for Autonomous Driving". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 5363–5369.

[279] Tie-Jun Yu, Ching-Fang Lin, and PC Muller. "Design of LQ regulator for linear systems with algebraic-equation constraints". In: *Decision and Control, 1996., Proceedings of the 35th IEEE Conference on*. Vol. 4. IEEE. 1996, pp. 4146–4151.

[280] Hao-jie Zhang et al. “An iterative linear quadratic regulator based trajectory tracking controller for wheeled mobile robot”. In: *Journal of Zhejiang University SCIENCE C* 13.8 (2012), pp. 593–600.

[281] Dingjiang Zhou et al. “Fast, On-line Collision Avoidance for Dynamic Vehicles using Buffered Voronoi Cells”. In: *Robotics and Automation Letters (RA-L)* 2 (2017), pp. 1047–1054. DOI: 10.1109/LRA.2017.2656241.

[282] Zhengyuan Zhou et al. “A general, open-loop formulation for reach-avoid games”. In: *51st Conference on Decision and Control (CDC)*. IEEE. 2012, pp. 6501–6506.

[283] Zhengyuan Zhou et al. “Cooperative pursuit with Voronoi partitions”. In: *Automatica* 72 (2016), pp. 64–72.

[284] Brian Ziebart et al. “Planning-based Prediction for Pedestrians”. In: Dec. 2009, pp. 3931–3936. DOI: 10.1109/IROS.2009.5354147.

[285] Brian D Ziebart et al. “Maximum entropy inverse reinforcement learning.” In: *AAAI*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.

[286] Brian D. Ziebart et al. “Planning-based prediction for pedestrians”. In: *International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2009, pp. 3931–3936.