First-Order Algorithms for Unconstrained/Constrained Dynamic Games and Identification of Linear Dynamic Systems with Multiplicative Noise

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Bolei Di

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Advisor, Andrew Lamperski

February, 2021

Acknowledgements

There are many people that have earned my gratitude for their contribution to my time in graduate school. Thanks to Prof. Tryphon Georgiou, Murti Salapaka, Nicola Elia, Richard Linares, Volkan Isler, and Mihailo Jovanovic. Your knowledge, insight, advice, and encouragement have been the constant, reliable guidance for me along the road.

Last but not least, my sincerest thanks to my advisor, Prof. Andrew Lamperski. Thanks for teaching me the highest academic and ethical standard, being the greatest role model of a colleague, a friend, and a noble person. I will carry on these lessons for the rest of my life.

Dedication

I dedicate this dissertation to my dear wife, Kaka Siu, who has supported me and taken care of our family over all the years of my Ph.D. study. Marrying you was the best decision I have made in my life. Thanks to the warmest company provided by our furry family members, Keno, Ripley, Willa, Kadelyn, and Dodo.

I also dedicate the dissertation to my parents, who have offered the most selfless support throughout my life while I explore the world and search for my path. Words cannot express the gratitude that I hold for being your son.

Abstract

The thesis consists of two parts or works. The first part focuses on numerical methods for computing the Nash equilibria of unconstrained and constrained dynamics games, and the second one studies linear system identification with multiplicative noise. We approach both topics from the angles of optimal control and optimization theory.

Dynamic games arise when multiple agents with differing objectives control a dynamic system. They model a wide variety of applications in economics, defense, energy systems, etc. However, compared to single-agent control problems, the computational methods for dynamic games are relatively limited. We focus on the state space formulation of dynamic games. Extra constraints that limit the space of actions based on current state and other players' actions at each step, makes the distinction between our unconstrained and constrained dynamic games. As in the single-agent case, only specific dynamic games can be solved exactly, so approximation algorithms are required. We focus on iterative numerical methods for finding the Nash equilibria of constrained/unconstrained full-information non-zero-sum dynamic games. We make clear distinction between open-loop Nash equilibrium (OLNE) and feedback Nash equilibrium (FNE), which are very different in the structures. Therefore, we need different numerical methods.

In the unconstrained case, we show how the stagewise Newton method, the approximated Bellman recursion (ABR) and the popular differential dynamic programming (DDP) originated from single-agent optimization or optimal control, can be adapted to the multi-player dynamic games. While the Newton step method works for OLNE, the other methods are related to FNE. We show that the Newton's step can be solved in a computationally efficient manner and inherits its original quadratic convergence rate to open-loop Nash equilibria, and that the ABR and DDP methods are very similar and can be used to find local feedback $O(\varepsilon^2)$-Nash equilibria.

For the constrained case, we show how to extend the projected gradient and Douglas-Rachford (DR) splitting methods that originated from constrained optimization and variational inequalities to solve the OLNEs. The resulting algorithms converge locally to open-loop Nash equilibria (OLNE) at linear rates. Furthermore, we extend the approximated Bellman recursion we proposed for unconstrained dynamic games method to find local FNE for constrained dynamic games. In the case of linear dynamics and polyhedral constraints, we show that this local

feedback policy is a local approximated feedback Nash equilibrium (FNE). All of our methods exploit the temporal structure of a dynamic system and therefore has a linear complexity with respect to the number of stages, which is a major improvement on the previously existing methods.

Linear systems with multiplicative noise (LSMN) have been approached from robust control, filtering and optimal control, system identification, and reinforcement learning since LSMN's early emergence in the 1960s. We focus on the improvement of system identification methods for LSMN by offering an algorithm based on least-squares estimation, which estimates both the first and second moments of the system parameters, and offers a probability bound on the estimates. Our proposed method can be applied to a more general problem formulation than the existing ones. We further develop an online scheme for estimation and robust control scheme based on the estimation and bound.

Overall, the thesis is focusing on proposing novel numerical algorithms for control/game problems that have not been satisfyingly solved. The effectiveness of the proposed algorithms in terms of their scope, convergence and complexity is analyzed. Inspirations of these new numerical algorithms typically came from existing methods for optimization, optimal control and variational inequality problems. For each topic we focus on, the problem formulation, proposed algorithms and analysis of the algorithms are the three major components.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Dynamic Games: Extension of Control Theory

We study the finite-horizon, constrained/unconstrained, discrete-time dynamic games with full information, from the angle that they are extending optimal control to multiple agents aiming at optimizing different objective functions. The dynamic system can be naturally discrete-time or emerge from the discretization of a differential game [1, 2, 3, 4]. Dynamic games have many applications including pursuit-evasion [5], active-defense [6, 7], economics [8] and the smart grid [9]. Despite a wide array of applications, the computational methods for dynamic games are considerably less developed than the single-agent case of optimal control.

The most common solution concepts are *open-loop Nash equilibria (OLNE)* and *feedback Nash equilibria (FNE)* [10, 1, 11]. The prior assumes no feedback structure, players predetermine their actions, while for the latter one, players can make their decisions based on measurements of the states at each step. Even given the same basic problem, because of this difference in the information structure, these different solutions can be very different, both conceptually and numerically.

Early works on OLNEs gave conditions for the existence and uniqueness of OLNEs for convex cost games [12][13][14]. Most pioneering works suggested directly using the gradient descent method to solve for the equilibria. However, there are fewer results related to feedback Nash equilibrium. Because of the constrained v.s. unconstrained structures of the problems, we need different methods or adaptation of the same methods.

1.2 Background of Constrained / Unconstrained Dynamic Games

This section gives an overview of related numerical methods for dynamic games. We directly generalize the Newton step and dynamic programming style algorithms such as differential dynamic programming. Similar approaches to ours are very limited, the only one we are aware of suffers from the curse of dimensionality w.r.t. the number of players [15]. Therefore, we focus on more broadly related methods and development in this section.

We will discuss methods for static games, including general Nash equilibrium problems (GNEP) [16], Newton's method [17], Nikaido-Isoda relaxation algorithms [18, 19, 20, 21, 22], and extremum seeking [23, 24, 25]. We will also discuss methods for special dynamic games, such as linear-quadratic games [26, 27, 28, 29, 30, 31], potential games [32, 33, 34, 35, 36, 37], and zero-sum games [38, 39]. We will also discuss general methods based on Pontryagin's Maximum Principle [11, 1, 3]. Specific to constrained dynamic games, we focus on gradient descent method [40, 41, 42, 43, 44] and Douglas-Rachford splitting method in the optimization and variational inequality (VI) community. These existing methods for games suffer from different reasons when applied to nonlinear dynamic games or only handle special cases. For a broad overview of recent developments, see [45].

General Nash equilibrium problems (GNEPs) are games with constraints that may be coupled [16]. GNEPs are reformulated to a set of necessary conditions via KKT conditions, which is in the form of variational inequalities (VI). These inequalities can be solved via generic VI methods or classic feasibility problem methods, such as Newton's method [17] or others [46]. In particular, Newton's method converts the complementarity conditions to equality constraints via complementarity functions. While these static methods for GNEPs can be applied to dynamic games, each iteration will have a computational complexity of $O(T^3)$ where T is the number of stages, because of the unexploited dynamic structure. Our proposed stagewise Newton's method for open-loop Nash equilibria is closely in-line with Newton's method for quasi-variational inequalities [17] but more specialized and faster because it exploits the dynamic structure and each step has a complexity of T.

The Nikaido-Isoda relaxation algorithm (NIRA) is another method for solving GNEPs [18, 19, 20, 21, 22]. The iteration of this method is based on weighted average of the current action and the *best response function*, which returns the set of players' actions that minimize each of their cost unilaterally given the current actions. The method converts the relatively hard

root-finding nature of solving for a NE to an optimization problem. However, the convergence conditions are very restrictive.It also does not utilize the dynamic structure, therefore does not scale well w.r.t. number of stages when applied to dynamic games.

Methods for finding Nash equilibria of static games via extremum seeking were presented in [47, 23, 24, 25]. In particular, the controllers drive the system to a Nash equilibrium. The work expands from linear systems to general nonlinear systems. For these works, each agent only requires measurements of its own cost. Our method requires each agent to have explicit model information, but gives equilibria for finite-horizon dynamic games. This is particularly important for games in which trajectories from initial to final states are desired.

As in optimal control, linear-quadratic (LQ) systems for games are well-understood compared to general systems and serve as the backbone for many solution methods [26, 27, 28, 29, 30, 31]. The existence of FNEs for linear-quadratic systems, and their analytic computation by coupled Riccati equations, is well understood [10, 48, 27]. The solution has also been extended to infinite horizon and distributed information cases [49, 50]. For a detailed description of the method for solving linear-quadratic games see [51]; for the complete set of sufficient conditions for discrete-time linear-quadratic games see [10].

In a potential game, a single potential function can be used to describe the marginal costs for each player [52, 32, 33, 34, 35, 36, 37]. Based on this property, potential games can often be solved using methods of single-agent optimization or optimal control. This line of work has been extended to constrained stochastic dynamic potential games [53]. However, the prerequisite that the game problem has a potential function is very restrictive.

Zero-sum games are another class of well-studied problems. Two player zero-sum differential games date back to the work of Issacs [54]. Extensions such as stochastic zero-sum dynamic games also exist [55, 56]. This stream of work is very closely related to robust control, in which a controller aims to perform well in the worst-case [57]. Work closely related to this paper is [38, 39], which applies DDP to zero-sum games. Our paper can be seen as a generalization of [39] to multi-player nonzero-sum games with theoretical justification.

The standard solution method for OLNEs of nonlinear dynamic games is via Pontryagin's Minimum Principle (PMP) for either continuous or discrete-time problems, as recognized by the community [3, 1, 11, 2, 4, 58]. Although the PMP allows us to analyze the existence of solutions and solve for analytical solutions for a few simple games, the resulting boundary value problem (BVP) with optimization is, in general, hard to solve [11]. A more approachable

reformulation of the necessary conditions is concatenating the KKT conditions of each player [16, 11], in which case, we arrive at a structured nonlinear programming (NLP), or *feasibility problem*. Though it has been known for years that such necessary conditions exist for games, we have not found works on developing specialized algorithms for solving these conditions, and generic solvers suffer from high complexity since they do not utilize the dynamic structure. Unlike its counterpart in optimal control, the KKT conditions for games require users to solve a root-finding problem, for which the conditions for the existence of solution and conditions for convergence of algorithm have not been developed.

Projected gradient algorithms alternate between taking the gradient descent steps and projecting onto constraints. They have been well studied for optimization [40] and variational inequality (VI) problems [41], and proven to converge linearly in both cases. This paper describes how the projected gradient method can be applied to constrained dynamic games, including convergence conditions and convergence rate of the algorithm.

Douglas-Rachford (DR) splitting methods alternate between solving two VI problems when applied to constrained game problems. Typically one problem corresponds to an unconstrained optimization or game problem, while the other corresponds to projection onto constraints [41]. As with the projected gradient method, the DR method converges linearly [59]. In optimization, the DR method is closely related to the alternating direction method of multipliers (ADMM) [42] and has been extended to (single-agent) optimal control problems [43, 44].

1.3 Identification of Linear Systems with Multiplicative Noise

Linear systems with multiplicative noise (LSMN) are more general than linear systems with additive noise. Compared to the later, LSMN covers more dynamic systems, specifically when noise is coupled with system state, or when the system parameters are naturally uncertain or vary in a range.

For systems with multiplicative noise, robust and optimal control require more knowledge than the mean of the noise, typically the covariance or second moments. We approach LSMN from the system identification perspective, which aims to estimate the first and second moments of system parameters and provide a bound on the estimation. We first develop the self-normalizing and Euclidean bound for the least-squares estimation of a simple linear equation estimation problem with multiplicative noise, extending the confidence bounds result for the

additive noise case [60]. Then we developed a scheme for estimating and bounding the parameters of a linear dynamic problem based on the Euclidean norm bound. The obtained bound can be then applied to robust/optimal control. We show how it can be applied to robust control in particular.

The study of linear systems with multiplicative noise (LSMN) dates back to the 1960s [61]. It has been studied from the perspectives of robust control [62], filtering and control [63], and reinforcement learning [64]. Compared to additive noise, LSMN is a more powerful model that can capture state and input dependent noise, a situation that occurs in a wide range of modern applications of the system theory, including robotics [65], networked communication systems [66, 67] and power networks [68].

Some general nonlinear system identification methods that are based on maximum likelihood, expectation-maximization (EM) algorithms, and the Bayesian inference, can be specialized to LSMN [69, 70, 71]. However, the methods are computationally more expensive, require a priori assumptions of the distribution of the uncertainties which might worsen the control performance, and do not offer a bound of the estimation.

Most recently, [72] studied identification of linear system with multiplicative noise, which shares the same background and is the most relevant to our work. The work suggested a multiple-trajectory averaging least-squares (MALS) method. It developed the complete dynamics for the second moments of the system matrices, collects data with a special multiple-trajectory scheme, runs least-squares on the entire dynamics and obtains an estimate of the mean and covariance of the system parameters. MALS suffers from a few restrictions, such as assuming independent system matrices A and B, assuming controllability of the dynamics of the second moments, and requiring multiple trajectories with the same initial state running open-loop policy. Our proposed method gets rid of the aforementioned restrictions and also gives a probability bound on the estimations.

1.4 Thesis Outline

- Chapter 1 gives a brief overview of our problems, our methods for tackling them, and related background in the literature.
- Chapter 2 contains our work on the unconstrained dynamic games. Including problem formulation, algorithms development and analysis.

- Chapter 3 focuses on constrained dynamic games, which is an extension of Chapter 2. The problems are formulated similarly, and corresponding algorithms are proposed and analyzed.
- Chapter 4 switches gear to linear systems with multiplicative noise, where we start with studying a static vector-scalar least-squares estimator and then apply it to linear dynamic systems.
- Chapter 5 concludes the thesis.
- Appendix A, B and C contain the technical proofs.

Chapter 2

Newton's Method, Approximated Bellman Recursion and Differential Dynamic Programming for Unconstrained Dynamic Games

Our numerical methods for unconstrained games are along the approach of dynamic programming, Bellman recursion, and quadratic approximation. In particular, we extend the classic Newton method, Bellman recursion, and the differential dynamic programming (DDP) method.

We tackle the open-loop Nash equilibria (OLNE) by computing the Newton step to solving the necessary condition of an OLNE. Based on the temporal structure, we develop a stagewise recursion for solving the Newton step, therefore we refer to our method as stagewise Newton method, which inherits the convergence analysis of Newton step for solving nonlinear equations.

For the study of feedback Nash equilibria, we extend both the classic Bellman recursion method and DDP to dynamic problems, which are of great theoretical and practical interest. The proposed approximated Bellman recursion method first approximates the original problem with a local quadratic dynamic game, then performs a Bellman recursion of the approximated game, while the DDP method solves the quadratically approximated Bellman recursion for the original game. Both methods find approximated local FNE [51, 11]. Other than deriving their

basic algorithmic forms, we prove that they both generate local FNE.

This chapter is organized as following. Section 2.1 starts with defining our problems and solution concepts, specifically, the deterministic, non-zero-sum, unconstrained dynamic games and its typical solutions, open-loop Nash equilibria (OLNE) and feedback Nash equilibria (FNE). Section 2.2 proposes the stagewise Newton method for solving the OLNEs. Section 2.32.4 and 2.5 proceed to propose the approximated Bellman recursion (ABR) and differential dynamic programming (DDP) methods for solving local feedback Nash equilibria. Eventually, Section 2.6 and 2.7 discuss the implementation details of the algorithms and demonstrate them with numerical examples.

2.1 Deterministic Nonlinear Dynamic Game Problem Formulation

In this section, we introduce deterministic finite-horizon nonlinear game problems, the notation for dynamic games, solution concepts and the existence condition of local OLNEs.

2.1.1 Problem Formulation

Problem 1. ***Nonlinear dynamic game***
Each player tries to minimize their own cost

$$J_{n,t}(x,u) = \sum_{k=t}^{T} c_{n,k}(x_k, u_{:,k}), \quad n = 1,2,...,N \tag{2.1}$$

Subject to dynamic constraints

$$x_{k+1} = f_k(x_k, u_{:,k}), \quad k = t, t+1, \cdots, T-1 \tag{2.2a}$$

$$x_0 \textit{ is fixed.} \tag{2.2b}$$

Here, $0 \leq t \leq T$ is the starting point for a game. When $t = 0$, we call it the *full game*, and $t > 0$, a *subgame* t. As indicated by the notation, we consider a full game of $T+1$ steps played by N players. The state of the system at time k is denoted by $x_k \in \mathbb{R}^{n_x}$. Player n's input at time k is given by $u_{n,k} \in \mathbb{R}^{n_{u_n}}$. The vector of all players' actions at time k is denoted by $u_{:,k} = [u_{1,k}^\top, u_{2,k}^\top, \ldots, u_{N,k}^\top]^\top \in \mathbb{R}^{n_u}$. The cost for player n at time k is $c_{n,k}(x_k, u_{:,k})$. In later analysis, some other notation will be helpful. The vector player n's actions over all time is denoted by $u_{n,:} = [u_{n,0}^\top, u_{n,1}^\top, \ldots, u_{n,T}^\top]^\top$. The vector of all actions other than those of player n

is denoted by $u_{-n,:} = [u_{1,:}^\top, \ldots, u_{n-1,:}^\top, u_{n+1,:}^\top, \ldots, u_{N,:}^\top]^\top$. The vector of all states is denoted by $x = [x_0^\top, x_1^\top, \ldots, x_T^\top]^\top$ while the vector of all inputs is given by $u = [u_{1,:}^\top, u_{2,:}^\top, \ldots, u_{N,:}^\top]^\top$.

Note that since the dynamics are deterministic, the cost for each player can be expressed as a function of all actions and the initial state, i.e. $J_{n,t}(x_t, u_{:,t:})$. Note that the dynamics are implicitly substituted to eliminate the dependency on x when we use $J_{n,t}(x_t, u_{:,t:})$ and the subscript t is omitted when we refer to the values of the full game. We assume $J_{n,t}(x_t, u_{:,t:})$ is twice differentiable. One set of sufficient conditions for the differentiability of $J_{n,t}(x_t, u_{:,t:})$ is that both the cost $c_{n,k}(x_k, u_{:,k})$ and the dynamics $f_k(x_k, u_{:,k})$ share at least the same differentiability, which is not very restrictive since most physical systems are governed by ordinary differential equations.

2.1.2 Open-loop Nash Equilibrium (OLNE)

When we discuss open-loop equilibria in the thesis, we will fix the initial condition x_0, the initial time $t = 0$ and consider the full game. For notational simplicity, we drop the t in $J_{n,t}(x, u)$.

Definition 1. ***(Local) open-loop Nash equilibrium***
An open-loop Nash equilibrium *(OLNE) for the full game problem 1 is a set of inputs $u^\star$ such that*

$$J_n(u_{n,:}, u_{-n,:}^\star) \geq J_n(u^\star),\ n = 1, 2, \ldots, N \tag{2.3}$$

for all $u_{n,:}$. Furthermore, if (2.3) *only holds for $u_{n,:}$ in a neighborhood of $u_{n,:}^\star$, it is called a* local open-loop Nash equilibrium *(local-OLNE).*

The equilibrium is called a *strict local Nash equilibrium* if the inequality in (2.3) is strict for all $u_{n,:} \neq u_{n,:}^\star$ in a neighborhood of $u_{n,:}^\star$. For unconstrained games, the following condition is necessary for a local Nash equilibrium.

Problem 2. ***Necessary conditions.*** *Find $u^\star$ such that*

$$\mathcal{J}(u^\star) \equiv \left[\frac{\partial J_1}{\partial u_{1,:}} \quad \frac{\partial J_2}{\partial u_{2,:}} \quad \cdots \quad \frac{\partial J_N}{\partial u_{N,:}} \right]^\top \Bigg|_{u^\star} = 0 \tag{2.4}$$

A trajectory of actions u satisfying (2.4) is referred to as a *stationary trajectory* of the open-loop dynamic game. Solving such necessary conditions is standard which also arises in other works [73, 74, 11, 45].

2.1.3 Feedback Nash Equilibrium (FNE)

In the case when state feedback information is available and each player acts according to a feedback strategy $u_{n,k} = \phi_{n,k}(x_k)$, feedback Nash equilibrium (FNE) can be defined. We consider only the case when players have a single step state feedback based on the state, do not have memory, or implement any inference of other players.

The collection of all players' strategy at step k is denoted as $\phi_{:,k}(\cdot)$. All players' strategies except for player n is denoted $\phi_{-n,k}(\cdot)$.

Definition 2. ***(Local) feedback Nash equilibrium (FNE)***
A collection of feedback policies $u_{n,k} = \phi^{\star}_{n,k}(x_k)$ is said to be a feedback Nash equilibrium (FNE) to the full game if no player can benefit from changing their policy unilaterally for any subgame, i.e.,

$$J_{n,t}(x_t, \phi^{\star}_{:,t:}) \leq J_{n,t}(x_t, [\phi_{n,t:}, \phi^{\star}_{-n,t:}]),\ \forall t \in \{0, 1, ..., T\} \tag{2.5}$$

where $J_{n,t}(x_t, \phi_{:,t:})$ indicates the total cost of player n when all players follow policy $\phi_{:,t:}$ for subgame t.

Furthermore, the FNE is local around $\bar{u}_{n,k} = \phi^{*}_{n,k}(\bar{x}_k)$ $\forall n \in \{1, 2, ..., N\}, \forall k \in \{0, 1, \ldots, T-1\}$, if (2.5) *holds only locally and the resulting trajectories remain in a neighborhood of $[\bar{x}, \bar{u}]$.*

Ideally, feedback Nash equilibrium can be solved via Bellman recursion, which originated from optimal control and was extended to dynamic games [51, 11]. Instead of solving for the minimizing action at each stage, equilibria of stage-wise games are computed via the following recursion.

$$V^{\star}_{n,T+1}(x_{T+1}) = 0 \tag{2.6a}$$

$$Q^{\star}_{n,k}(x_k, u_{:,k}) = c_{n,k}(x_k, u_{:,k}) + V^{\star}_{n,k+1}(f_k(x_k, u_{:,k})) \tag{2.6b}$$

$$V^{\star}_{n,k}(x_k) = \min_{u_{n,k}} Q^{\star}_{n,k}(x_k, u_{:,k}) \tag{2.6c}$$

$$k = 0, 1, ..., T \tag{2.6d}$$

Here $V^{\star}_{n,k}(x_k)$ and $Q^{\star}_{n,k}(x_k, u_{:,k})$ are referred to as *equilibrium value functions* for player n at time step k. In particular, if a solution to the Bellman recursion is found, the corresponding optimal strategy for player n at time k would be the $u_{n,k}$ which minimizes $Q^{\star}_{n,k}(x_k, u_{:,k})$. Note that (2.6c) defines a static game with respect to the $u_{:,k}$ variable at step k, which we call the *stagewise*

game at k. A well known verification theorem states that a feedback policy $u_{:,k} = \phi^{\star}_{:,k}(x_k),\ k = 0, 1, ..., T$ solving the sequence of static games defined by (2.6c), is a subgame perfect FNE for the dynamic game [11, 51]. For general dynamic games, the Bellman equations are not tractable. Note that the game ends at $k = T$, and setting $V^{\star}_{n,T+1}(x_{T+1}) = 0$ is only for ease of describing the Bellman recursion.

Suppose following the FNE $\phi^{\star}_{:,k}(x_k)$ generates a trajectory of $\bar{x}^{\star}, \bar{u}^{\star}$, a necessary condition for the FNE can be directly derived from (2.6c), which is

$$\frac{\partial}{\partial u_{n,k}} Q^{\star}_{n,k}(\bar{x}^{\star}_k, \bar{u}^{\star}_{:,k}) = 0, \forall n, k \tag{2.7}$$

If condition (2.7) were not true, then player n at time k can reduce their cost by adding a constant term to their strategy $\phi^{\star}_{n,k}(x_k)$, contradicting it being an FNE.

2.1.4 Existence of Solutions and Convergence Conditions

To guarantee the existence of local OLNEs and our stagewise Newton method's convergence to them, we assume that $\mathcal{J}(u)$ satisfies the smoothness and non-degeneracy conditions required by Newton's method locally [40]. Since Newton's method is widely applied to optimization problems, which is similar to our assumption, so our assumptions are fairly general.

Assumption 1 (Smoothness)**.** *The vector-valued function, $\mathcal{J}(u)$, is differentiable with locally Lipschitz derivatives.*

Assumption 2 (Non-degeneracy)**.** *The Jacobian $\frac{\partial \mathcal{J}(u^{\star})}{\partial u}$ is invertible.*

A sufficient condition for the smoothness assumptions is that the functions f_k and $c_{n,k}$ are twice continuously differentiable with Lipschitz second derivatives. For either of our methods, we will solve a sequence of stagewise quadratic games. As we will see, a sufficient condition for invertibility of $\frac{\partial \mathcal{J}(u^{\star})}{\partial u}$ is the unique solvability of the stagewise games near the equilibrium.

The non-degeneracy and smoothness conditions guarantee that Newton's method converges locally to a stationary trajectory satisfying (2.4). The following assumption guarantees that this stationary trajectory is a strict local Nash equilibrium.

Assumption 3. *Each player's Hessian, $\frac{\partial^2 J_{n,0}(u^{\star})}{\partial^2 u_{n,:}}$, is locally positive definite, i.e. each player's cost $J_n(u)$ is strictly convex w.r.t. their actions $u_{n,:}$.*

This assumption is a natural specialization of the definition of Nash equilibrium when the cost functions are second-order differentiable. For an OLNE to exist for such problems, this assumption is natural and necessary.

2.1.5 Notation of Derivatives

We define the following shorthand notation for first and second order derivatives of both the dynamics and cost functions for given trajectory $\bar{x}, \bar{u}$, which are used in the OLNE recursion and both ABR method and DDP method for FNE. The derivatives show up because we are using quadratic approximations around trajectories.

$$A_k = \frac{\partial f_k(x_k, u_{:,k})}{\partial x_k}\Big|_{\bar{x},\bar{u}} \qquad B_k = \frac{\partial f_k(x_k, u_{:,k})}{\partial u_{:,k}}\Big|_{\bar{x},\bar{u}} \tag{2.8a}$$

$$G_k^l = \begin{bmatrix} \frac{\partial^2 f_k^l}{\partial x_k^2} & \frac{\partial^2 f_k^l}{\partial x_k \partial u_{:,k}} \\ \frac{\partial^2 f_k^l}{\partial u_{:,k} \partial x_k} & \frac{\partial^2 f_k^l}{\partial u_{:,k}^2} \end{bmatrix}\Bigg|_{\bar{x},\bar{u}}, \quad l = 1, 2, \ldots, n_x \tag{2.8b}$$

$$R_k(\delta x_k, \delta u_{:,k}) = \begin{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top G_k^1 \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top G_k^{n_x} \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix} \end{bmatrix} \tag{2.8c}$$

$$\begin{aligned} M_{n,k} &= \begin{bmatrix} 2c_{n,k} & \frac{\partial c_{n,k}}{\partial x_k} & \frac{\partial c_{n,k}}{\partial u_{:,k}} \\ \frac{\partial c_{n,k}}{\partial x_k}^\top & \frac{\partial^2 c_{n,k}}{\partial x_k^2} & \frac{\partial^2 c_{n,k}}{\partial x_k \partial u_{:,k}} \\ \frac{\partial c_{n,k}}{\partial u_{:,k}}^\top & \frac{\partial^2 c_{n,k}}{\partial u_{:,k} \partial x_k} & \frac{\partial^2 c_{n,k}}{\partial u_{:,k}^2} \end{bmatrix}\Bigg|_{\bar{x},\bar{u}} \\ &= \begin{bmatrix} M_{n,k}^{11} & M_{n,k}^{1x} & M_{n,k}^{1u} \\ M_{n,k}^{x1} & M_{n,k}^{xx} & M_{n,k}^{xu} \\ M_{n,k}^{u1} & M_{n,k}^{ux} & M_{n,k}^{uu} \end{bmatrix}. \end{aligned} \tag{2.8d}$$

2.2 Stagewise Newton Method for Local Open-loop Nash Equilibrium

In this section, we propose the stagewise Newton method that solves the (local) open-loop Nash equilibrium. Section 2.2.1 gives the locally quadratically approximated dynamic game

(LQADG), which is of crucial importance for solving both OLNE and FNE in later sections. Section 2.2.2 and section 2.2.3 solve the OLNE of the LQADG, which is a linear algebraic process. While section 2.2.2 reformulates the LQADG to a simpler form, section 2.2.3 solves its OLNE in a recursive stagewise form. In the end, Section 2.2.4 summarizes the solved OLNE in an algorithm. In our algorithm, the OLNE to LQADG that is solved in a recursive, stagewise manner, is exactly a Newton step of solving the necessary condition of the original game, therefore the name *stagewise Newton method*.

2.2.1 Newton Step and Locally Quadratically Approximated Dynamic Game

With a given trajectory $\bar{u}$, the Newton step for solving the necessary condition (2.4) is given by:

$$\frac{\partial \mathcal{J}(\bar{u})}{\partial u}\delta u^N = -\mathcal{J}(\bar{u}). \tag{2.9}$$

We implement the Newton step as our open-loop Nash equilibrium update step. This rule leads to a quadratic convergence to a root of (2.4) when $\frac{\partial \mathcal{J}(u)}{\partial u}$ is locally Lipschitz and invertible [40]. The next two lemmas give game-theoretic interpretations of the Newton step.

Lemma 1. *If Assumptions 1- 3 hold, then solving* (2.9) *is equivalent to solving the quadratic game defined by:*

$$\min_{\delta u_{n,:}} \quad J_n(\bar{u}) + \frac{\partial J_n(\bar{u})}{\partial u}\delta u + \frac{1}{2}\delta u^\top \frac{\partial^2 J_n(\bar{u})}{\partial u^2}\delta u \tag{2.10}$$

Proof. Under the strict local equilibrium assumptions, (2.10) has a unique solution which is found by differentiating with respect to $\delta u_{n,:}$ and setting the result to 0. Stacking these equations leads precisely to (2.9). □

Throughout the discussion of OLNE, we will assume that Assumptions 1-3 hold. The next lemma shows that (2.10) can be expressed as a quadratic dynamic game, which is our locally quadratically approximated dynamic game (LQADG) to our original game Problem 1 around $\bar{x}, \bar{u}$. It is proven in Appendix A.1.

Lemma 2. *The quadratic game defined in* (2.10) *is equivalent to the dynamic game defined by:*

$$\min_{\delta u_{n,:}} \delta J_n(\delta x, \delta u) = \sum_{k=0}^{T} \delta c_{n,k}(\delta x_k, \delta u_k) = \frac{1}{2}\sum_{k=0}^{T}\left(\begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top M_{n,k} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix} + M_{n,k}^{1x}\Delta x_k\right) \tag{2.11a}$$

subject to

$$\delta x_0 = 0 \tag{2.11b}$$

$$\Delta x_0 = 0 \tag{2.11c}$$

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_{:,k} \tag{2.11d}$$

$$\Delta x_{k+1} = A_k \Delta x_k + R_k(\delta x_k, \delta u_{:,k}) \tag{2.11e}$$

$$k = 0, 1, \ldots, T \tag{2.11f}$$

where A_k, B_k, $M_{n,k}$, $R_k(\delta x_k, \delta u_{:,k})$ are defined in Section 2.1.5, which are constants for given trajectory $\bar{x}, \bar{u}$. The states of the dynamic game are given by δx_k and Δx_k as

$$\delta x_k = \sum_{i=0}^{T} \frac{\partial x_k}{\partial u_{:,i}}\Big|_{\bar{x},\bar{u}} \delta u_{:,i} \tag{2.12a}$$

$$\Delta x_k^l = \sum_{i=0}^{T} \sum_{j=0}^{T} \delta u_{:,i}^\top \frac{\partial^2 x_k^l}{\partial u_{:,i} \partial u_{:,j}}\Big|_{\bar{x},\bar{u}} \delta u_{:,j},\ l = 1, 2, \ldots, n_x \tag{2.12b}$$

This local quadratic approximation to our original problem is key in our work. Computing its open-loop Nash equilibrium and feedback Nash equilibrium result in update rules for our stagewise Newton's method for local OLNE and approximated Bellman recursion for FNE, respectively.

2.2.2 Reformulation of the Locally Approximated Game

To solve the OLNE of game (2.11), we need to reformulate it to an equivalent but simpler form. We start with eliminating Δx_k in (2.11), rewriting the related cost $\frac{1}{2}\sum_{k=0}^{T} M_{n,k}^{1x} \Delta x_k$ in terms of δx_k and δu_k. We introduce time-series matrices $\Omega_{n,k}$ and $D_{n,k}$ that are computed with end condition $\Omega_{n,T+1} = 0$, and for $k = T, T-1, ..., 0$, follow the backward recursion for every player.

$$\Omega_{n,k} = M_{n,k}^{1x} + \Omega_{n,k+1} A_k \tag{2.13a}$$

$$D_{n,k} = \sum_{l=1}^{n_x} \Omega_{n,k+1}^l G_k^l \in \mathbb{R}^{(n_x+n_u)\times(n_x+n_u)} \tag{2.13b}$$

$$\hat{M}_{n,k} = M_{n,k} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & D_{n,k} \end{bmatrix} \tag{2.13c}$$

where the zeros in (2.13c) has the shape such that the dimension matches $M_{n,k}$. Given $\hat{M}_{n,k}$, the problem (2.11) is equivalent to the following (2.14). The proof is provided in Appendix A.2.

$$\min_{\delta u_{n,:}} \sum_{k=0}^{T} \delta c_{n,k}(\delta x_k, \delta u_{:,k}) = \frac{1}{2} \sum_{k=0}^{T} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top \hat{M}_{n,k} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix} \tag{2.14a}$$

subject to

$$\delta x_0 = 0 \tag{2.14b}$$

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_{:,k} \tag{2.14c}$$

$$k = 0, 1, \ldots, T \tag{2.14d}$$

We further simplify (2.14) by merging the constant 1 to the state x_k, so the problem becomes

$$\min_{\delta u_{n,:}} \sum_{k=0}^{T} \delta J_n(\delta \hat{x}, \delta u) = \delta c_{n,k}(\delta \hat{x}_k, \delta \hat{u}_{:,k}) = \frac{1}{2} \sum_{k=0}^{T} \begin{bmatrix} \delta \hat{x}_k \\ \delta u_{:,k} \end{bmatrix}^\top \hat{M}_{n,k} \begin{bmatrix} \delta \hat{x}_k \\ \delta u_{:,k} \end{bmatrix} \tag{2.15a}$$

subject to

$$\delta \hat{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{2.15b}$$

$$\delta \hat{x}_{k+1} = \hat{A}_k \delta \hat{x}_k + \hat{B}_k \delta u_{:,k} \tag{2.15c}$$

$$k = 0, 1, \ldots, T \tag{2.15d}$$

where

$$\delta \hat{x}_k = \begin{bmatrix} 1 \\ \delta x_k \end{bmatrix} \quad \hat{A} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & A \end{bmatrix} \quad \hat{B} = \begin{bmatrix} 0 \\ B \end{bmatrix} \tag{2.15e}$$

(2.15) is a linear quadratic game that is equivalent to (2.11), and whose OLNE can be solved analytically. For notational purpose, the blocks related to x in $\hat{M}_{n,k}$ contains the blocks in $M_{n,k}$ related to constants.

$$\hat{M}^{xx}_{n,k} = \begin{bmatrix} M^{11}_{n,k} & M^{1x}_{n,k} \\ M^{x1}_{n,k} & M^{xx}_{n,k} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & D^{xx}_{n,k} \end{bmatrix} \quad \hat{M}^{ux}_{n,k} = \begin{bmatrix} M^{u1}_{n,k} & M^{ux}_{n,k} \end{bmatrix} + \begin{bmatrix} 0 & D^{ux}_{n,k} \end{bmatrix} \tag{2.16}$$

2.2.3 Solution to the Reformulated Game

Deriving the gradient of the open-loop game and solve the OLNE of (2.15) is a lengthy algebraic process. The details are in Appendix A.3. We simply state the results here. The playerwise gradient is

$$\frac{\partial \delta J_n(\delta\hat{x}_0, \delta u)}{\partial \delta u_{n,:}} = F_n^\top \begin{bmatrix} \hat{M}_{n,0}^{xx}\delta\hat{x}_0 + \hat{M}_{n,0}^{xu}\delta u_{:,0} \\ \hat{M}_{n,1}^{xx}\delta\hat{x}_1 + \hat{M}_{n,0}^{xu}\delta u_{:,1} \\ \vdots \\ \hat{M}_{n,N}^{xx}\delta\hat{x}_T + \hat{M}_{n,N}^{xu}\delta u_{:,T} \end{bmatrix} + \begin{bmatrix} \hat{M}_{n,0}^{u_n x}\delta\hat{x}_0 + \hat{M}_{n,0}^{u_n u}\delta u_{:,0} \\ \hat{M}_{n,1}^{u_n x}\delta\hat{x}_1 + \hat{M}_{n,1}^{u_n u}\delta u_{:,1} \\ \vdots \\ \hat{M}_{n,N}^{u_n x}\delta\hat{x}_T + \hat{M}_{n,N}^{u_n u}\delta u_{:,T} \end{bmatrix} \tag{2.17a}$$

$$= F_n^\top \hat{\mathcal{M}}_n^x + \hat{\mathcal{M}}_n^{u_n} \tag{2.17b}$$

where

$$F_n = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ \hat{B}_{n,0} & 0 & 0 & \cdots & 0 & 0 \\ \hat{A}_1\hat{B}_{n,0} & \hat{B}_{n,1} & 0 & \cdots & 0 & 0 \\ \hat{A}_2\hat{A}_1\hat{B}_{n,0} & \hat{A}_2\hat{B}_{n,1} & \hat{B}_{n,2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \Pi_{k=1}^{T-1}\hat{A}_k\hat{B}_{n,0} & \Pi_{k=2}^{T-1}\hat{A}_k\hat{B}_{n,1} & \cdots & \hat{A}_{T-1}\hat{B}_{n,T-2} & \hat{B}_{n,T-1} & 0 \end{bmatrix} \tag{2.18a}$$

$$\hat{M}_{n,k}^{xu} = \begin{bmatrix} \hat{M}_{n,k}^{xu_1} & \hat{M}_{n,k}^{xu_2} & \cdots & \hat{M}_{n,k}^{xu_N} \end{bmatrix} \quad \hat{M}_{n,k}^{u_n u} = \begin{bmatrix} \hat{M}_{n,k}^{u_n u_1} & \hat{M}_{n,k}^{u_n u_2} & \cdots & \hat{M}_{n,k}^{u_n u_N} \end{bmatrix} \tag{2.18b}$$

We further set the gradient to zero and solve the equation in a backward, stagewise form. We solved the OLNE of the last steps $u_{:,T-2:T}^\star$, found a pattern and proved with induction that the pattern applies to all the rest of the steps. The final recursion is summarized in Section 2.2.4. Because the resulting update step is exactly the same as a Newton step for solving (2.4) and is computed in a stagewise form, we name it *stagewise Newton's method*.

2.2.4 Stagewise Newton Method for Open-loop Nash Equilibrium

We layout the stagewise Newton method for computing the OLNE in this section. The recursion can be summarized with the end condition $\hat{S}_{n,T+1} = 0$ of a time-series of matrices $\hat{S}_{n,k}$, and

perform a backward recursion for $k = T, T-1, ..., 0$.

$$\hat{S}^{BB}_{k+1} = \begin{bmatrix} \hat{B}^\top_{1,k}\hat{S}_{1,k+1}\hat{B}_{1,k} & \hat{B}^\top_{1,k}\hat{S}_{1,k+1}\hat{B}_{2,k} & \cdots & \hat{B}^\top_{1,k}\hat{S}_{1,k+1}\hat{B}_{N,k} \\ \hat{B}^\top_{2,k}\hat{S}_{2,k+1}\hat{B}_{1,k} & \hat{B}^\top_{2,k}\hat{S}_{2,k+1}\hat{B}_{2,k} & \cdots & \hat{B}^\top_{2,k}\hat{S}_{2,k+1}\hat{B}_{N,k} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{B}^\top_{N,k}\hat{S}_{N,k+1}\hat{B}_{1,k} & \hat{B}^\top_{N,k}\hat{S}_{N,k+1}\hat{B}_{2,k} & \cdots & \hat{B}^\top_{N,k}\hat{S}_{N,k+1}\hat{B}_{N,k} \end{bmatrix} \tag{2.19a}$$

$$\hat{S}^{BA}_{k+1} = \begin{bmatrix} \hat{B}^\top_{1,k}\hat{S}_{1,k+1}\hat{A}_k \\ \hat{B}^\top_{2,k}\hat{S}_{2,k+1}\hat{A}_k \\ \vdots \\ \hat{B}^\top_{N,k}\hat{S}_{N,k+1}\hat{A}_k \end{bmatrix} \tag{2.19b}$$

$$\hat{K}_k = -\left(\hat{S}^{BB}_{k+1} + \begin{bmatrix} \hat{M}^{u_1u_1}_{1,k} & \hat{M}^{u_1u_2}_{1,k} & \cdots & \hat{M}^{u_1u_N}_{1,k} \\ \hat{M}^{u_2u_1}_{2,k} & \hat{M}^{u_2u_2}_{2,k} & \cdots & \hat{M}^{u_2u_N}_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{M}^{u_Nu_1}_{N,k} & \hat{M}^{u_Nu_2}_{N,k} & \cdots & \hat{M}^{u_Nu_N}_{N,k} \end{bmatrix} \right)^{-1} \left(\hat{S}^{BA}_{k+1} + \begin{bmatrix} \hat{M}^{u_1x}_{1,k} \\ \hat{M}^{u_2x}_{2,k} \\ \vdots \\ \hat{M}^{u_Nx}_{N,k} \end{bmatrix} \right) \tag{2.19c}$$

$$S_k = \hat{M}^{xx}_{n,k} + \hat{M}^{xu}_{n,k}\hat{K}_k + \hat{A}^\top_k \hat{S}_{k+1}(\hat{A}_k + \hat{B}_k\hat{K}_k) \tag{2.19d}$$

After the backward pass, a forward pass $k = 0, 1, ..., T-1$ should be performed to obtain the open-loop Nash equilibrium actions.

$$\delta u^\star_{:,k} = \hat{K}_k \delta\hat{x}_k \tag{2.20a}$$

$$\delta\hat{x}_{k+1} = \hat{A}_k\delta\hat{x}_k + \hat{B}_k\delta u^\star_{:,k} \tag{2.20b}$$

We summarize the overall algorithm.

Algorithm 1 Stagewise Newton's method for OLNE of nonlinear dynamic games

For a dynamic game of Problem 1, generate an initial trajectory $\bar{x}^0, \bar{u}^0$, set $i = 0$

loop

Compute and simplify the LQADG

Compute the derivatives $A_k, B_k, G_k^l, M_{n,k}$ following (2.8)

Compute $\hat{M}, \hat{A}, \hat{B}$ following (2.13) (2.15e) and formulate the simplified problem (2.15)

Backward Pass

Compute $\hat{S}_{n,k}, \hat{K}_k$ by performing the backward pass (2.19)

Forward Pass

Generate the Newton step $\delta u^\star$ with the policy defined by K_k by performing the forward pass (2.20)

Update the trajectory $\bar{u}^{i+1} = \bar{u}^i + \delta u^\star$

Set $i = i + 1$

Check convergence

end loop

$\bar{u}$ is the OLNE of the original game Problem 1

2.2.5 Uniqueness and Convergence of Stagewise Newton's Method

Since our stagewise Newton's method is the exact Newton step for solving the necessary condition, only in a computationally cheaper manner, it is clear that they should share the same quadratic convergence.

Theorem 1. ***Uniqueness and existence of local open-loop Nash Equilibrium.***
If Assumptions 1- 3 hold locally in a region of u, (2.9) *has a local unique solution $u^\star$ within the region. Starting from a nearby trajectory $\bar{u}$, Algorithm 1 converges to $u^\star$ quadratically.*

Proof. Assumptions 1- 3 guarantee that the Newton step (2.9) converges quadratically to a unique solution locally, and since Algorithm 1 computes the exact Newton step, it inherits Newton step's property. □

Note that OLNEs are not subgame perfect. Despite this weakness, OLNEs are still valuable in cases where no feedback information is available, a model predictive control (MPC) style strategy is applied or simply the system is sufficiently deterministic.

2.3 Approximated Bellman Recursion (ABR) Method for FNE

This section describes the approximated Bellman recursion method for solving the local feedback Nash equilibrium of dynamic games Problem 1. Section 2.3.1 gives a high-level description of the algorithms, while Section 2.3.2 describes the explicit matrix calculations.

2.3.1 Algorithm Overview

Both the approximated Bellman and DDP methods work iteratively starting with a nominal trajectory $\bar{u}$. With each iteration, the trajectory is updated and becoming closer and closer to a stationary trajectory of each method, around which a local FNE can be found. For problems with multiple stationary trajectories, the choice of initial $\bar{u}$ determines which one is finally converged to. Both algorithms offer an affine feedback policy solution for each of the iteration to update the trajectory. The affine feedback policies are generated differently for the two methods. The final feedback policies around the stationary trajectory are local feedback Nash equilibria, which we will discuss in detail in Section 2.5.

Because of their similarity, the procedures can be organized in one pseudo code as in Algorithm 2. An initial trajectory of $\bar{x}$ should be found by running the system with actions $\bar{u}$, which are needed to compute derivatives in (2.8).

Algorithm 2 ABR and DDP methods for Nonlinear Dynamic Games

Generate an initial trajectory $\bar{x}, \bar{u}$

loop

Backward Pass:

if ABR method **then**

Compute the derivatives $A_k, B_k, G_k^l, M_{n,k}$ following (2.8)

Form the approximated dynamic game (2.21)

Compute K_k and s_k from (2.25).

end if

if DDP **then**

Form the differential dynamic programming around $\bar{x}, \bar{u}$ (2.30)

Compute $\tilde{K}_k$ and $\tilde{s}_k$ from (2.31).

end if

Forward Pass:

Generate a new trajectory using the affine policy defined by K_k, s_k or $\tilde{K}_k, \tilde{s}_k$

Check convergence

end loop

Obtaining stationary trajectory $u^\star$, $x^\star$ and feedback policy $\tilde{K}_k^\star$, $\tilde{s}_k^\star$ or $K_k^\star$, $s_k^\star$, which form the local FNE

Recall the local quadratic approximation defined in Lemma 2, Section 2.2.

$$\min_{u_{n,:}} \frac{1}{2} \sum_{k=0}^{T} \left(\begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top M_{n,k} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix} + M_{n,k}^{1x} \Delta x_k \right) \tag{2.21a}$$

subject to

$$\delta x_0 = 0 \tag{2.21b}$$

$$\Delta x_0 = 0 \tag{2.21c}$$

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_{:,k} \tag{2.21d}$$

$$\Delta x_{k+1} = A_k \Delta x_k + R_k(\delta x_k, \delta u_{:,k}) \tag{2.21e}$$

$$k = 0, 1, \ldots, T \tag{2.21f}$$

where A_k, B_k, $M_{n,k}$, $R_k(\delta x_k, \delta u_{:,k})$ are defined in Section 2.1.5, which are constants for given trajectory $\bar{u}$.

Note that the states of the dynamic game are given by δx_k and Δx_k as

$$\delta x_k = \sum_{i=0}^{T} \frac{\partial x_k}{\partial u_{:,i}}\Big|_{\bar{x},\bar{u}} \delta u_{:,i} \tag{2.22a}$$

$$\Delta x_k^l = \sum_{i=0}^{T}\sum_{j=0}^{T} \delta u_{:,i}^{\top} \frac{\partial^2 x_k^l}{\partial u_{:,i}\partial u_{:,j}}\Big|_{\bar{x},\bar{u}} \delta u_{:,j},\ l = 1,2,\ldots,n_x \tag{2.22b}$$

It turns out that the Bellman equations (2.6) associated with problem (2.21) can be solved analytically and the resulting value functions have quadratic forms. The next lemma describes the explicit solution to (2.21) based on Bellman equation (2.6). It is proved in Appendix A.4.

Lemma 3. *The equilibrium value functions for the dynamic game defined by* (2.21) *are denoted as* $\hat{V}_{n,k}^{\bar{u}}(\cdot)$ *and* $\hat{Q}_{n,k}^{\bar{u}}(\cdot,\cdot)$*, which can be expressed as*

$$\hat{V}_{n,k}^{\bar{u}}(\delta x_k, \Delta x_k) = \frac{1}{2}\left(\begin{bmatrix}1\\ \delta x_k\end{bmatrix}^{\top} S_{n,k} \begin{bmatrix}1\\ \delta x_k\end{bmatrix} + \Omega_{n,k}\Delta x_k\right) \tag{2.23a}$$

$$\hat{Q}_{n,k}^{\bar{u}}(\delta x_k, \Delta x_k, \delta u_{:,k}) = \frac{1}{2}\left(\begin{bmatrix}1\\ \delta x_k\\ \delta u_{:,k}\end{bmatrix}^{\top} \Gamma_{n,k} \begin{bmatrix}1\\ \delta x_k\\ \delta u_{:,k}\end{bmatrix} + \Omega_{n,k}\Delta x_k\right) \tag{2.23b}$$

where the matrices $S_{n,k}$*,* $\Gamma_{n,k}$*, and* $\Omega_{n,k}$ *can be computed in a backward pass. Note that we use the superscript* $^{\bar{u}}$ *to indicate the nominal trajectory that we are approximating the original problem around. Detailed descriptions are given by* (2.25) *in Section 2.3.2.*

The local game defined by (2.23b) *is now a quadratic game in the* $\delta u_{:,k}$ *variables, whose feedback Nash equilibrium is*

$$u_{:,k} = \bar{u}_{:,k} + K_k \delta x_k + s_k. \tag{2.24}$$

We define a *stationary trajectory of the ABR method* as $\bar{x}, \bar{u}$ such that, the resulting policy (2.24) has zero constant terms $s_k = 0, \forall k$.

2.3.2 Details of Approximated Bellman Recursion

The following lemma explains how to compute $S_{n,k}$ and $\Gamma_{n,k}$, which is the major computational cost for our method. The procedure is a backward recursion of T steps in time consists of

mostly matrix multiplications, additions and one matrix inversion. See Section 2.6.2 and 2.6.4 for detailed analysis.

Lemma 4. *The matrices $S_{n,k}$, $\Gamma_{n,k}$, and $\Omega_{n,k}$ in* (2.23) *are computed recursively by $S_{n,T+1} = 0$, $\Omega_{n,T+1} = 0$, and*

$$\Omega_{n,k} = M^{1x}_{n,k} + \Omega_{n,k+1} A_k \tag{2.25a}$$

$$D_{n,k} = \sum_{l=1}^{n_x} \Omega^l_{n,k+1} G^l_k \tag{2.25b}$$

$$\begin{aligned}\Gamma_{n,k} &= M_{n,k} \\ &\quad + \begin{bmatrix} S^{11}_{n,k+1} & S^{1x}_{n,k+1} A_k & S^{1x}_{n,k+1} B_k \\ A_k^\top S^{x1}_{n,k+1} & A_k^\top S^{xx}_{n,k+1} A_k + D^{xx}_k & A_k^\top S^{xx}_{n,k+1} B_k + D^{xu}_k \\ B_k^\top S^{x1}_{n,k+1} & B_k^\top S^{xx}_{n,k+1} A_k + D^{ux}_k & B_k^\top S^{xx}_{n,k+1} B_k + D^{uu}_k \end{bmatrix}\end{aligned} \tag{2.25c}$$

$$= \begin{bmatrix} \Gamma^{11}_{n,k} & \Gamma^{1x}_{n,k} & \Gamma^{1u_1}_{n,k} & \Gamma^{1u_2}_{n,k} & \cdots & \Gamma^{1u_N}_{n,k} \\ \Gamma^{x1}_{n,k} & \Gamma^{xx}_{n,k} & \Gamma^{xu_1}_{n,k} & \Gamma^{xu_2}_{n,k} & \cdots & \Gamma^{xu_N}_{n,k} \\ \Gamma^{u_1 1}_{n,k} & \Gamma^{u_1 x}_{n,k} & \Gamma^{u_1 u_1}_{n,k} & \Gamma^{u_1 u_2}_{n,k} & \cdots & \Gamma^{u_1 u_N}_{n,k} \\ \Gamma^{u_2 1}_{n,k} & \Gamma^{u_2 x}_{n,k} & \Gamma^{u_2 u_1}_{n,k} & \Gamma^{u_2 u_2}_{n,k} & \cdots & \Gamma^{u_2 u_N}_{n,k} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \Gamma^{u_N 1}_{n,k} & \Gamma^{u_N x}_{n,k} & \Gamma^{u_N u_1}_{n,k} & \Gamma^{u_N u_2}_{n,k} & \cdots & \Gamma^{u_N u_N}_{n,k} \end{bmatrix} \tag{2.25d}$$

$$F_k = \begin{bmatrix} \Gamma^{u_1 u}_{1k} \\ \Gamma^{u_2 u}_{2k} \\ \vdots \\ \Gamma^{u_N u}_{Nk} \end{bmatrix} = \begin{bmatrix} \Gamma^{u_1 u_1}_{1k} & \Gamma^{u_1 u_2}_{1k} & \cdots & \Gamma^{u_1 u_N}_{1k} \\ \Gamma^{u_2 u_1}_{2k} & \Gamma^{u_2 u_2}_{2k} & \cdots & \Gamma^{u_2 u_N}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \Gamma^{u_N u_1}_{Nk} & \Gamma^{u_N u_2}_{Nk} & \cdots & \Gamma^{u_N u_N}_{Nk} \end{bmatrix} \tag{2.25e}$$

$$P_k = \begin{bmatrix} \Gamma^{u_1 x}_{1k} \\ \Gamma^{u_2 x}_{2k} \\ \vdots \\ \Gamma^{u_N x}_{Nk} \end{bmatrix}, \quad H_k = \begin{bmatrix} \Gamma^{u_1 1}_{1k} \\ \Gamma^{u_2 1}_{2k} \\ \vdots \\ \Gamma^{u_N 1}_{Nk} \end{bmatrix} \tag{2.25f}$$

$$s_k = -F_k^{-1} H_k, \quad K_k = -F_k^{-1} P_k \tag{2.25g}$$

$$S_{n,k} = \begin{bmatrix} 1 & 0 & s_k^\top \\ 0 & I & K_k^\top \end{bmatrix} \Gamma_{n,k} \begin{bmatrix} 1 & 0 \\ 0 & I \\ s_k & K_k \end{bmatrix} \tag{2.25h}$$

for $k = T, T-1, \ldots, 0$.

Proof. By construction we must have $S_{n,T+1} = 0$. Plugging (2.11d) and (2.11e) into (2.23a) gives the backward iteration of (2.25a)(2.25b)(2.25c). Since $u_{:,k} = \bar{u}_{:,k} + \delta u_{:,k}$ and $\bar{u}_{:,k}$ is constant, the static game defined in (2.23b) can be solved in the $\delta u_{:,k}$ variables. Differentiating (2.23b) by $\delta u_{n,k}$, collecting the derivatives for all players and setting them to zero leads to the necessary condition for an equilibrium:

$$F_k \delta u_{:,k} + P_k \delta x_k + H_k = 0. \tag{2.26}$$

Thus, the matrices for the equilibrium strategy are given in (2.25g). Plugging (2.25g) into (2.23b) leads to (2.25h). □

2.3.3 Comparing ABR and stagewise Newton

The form of the backward recursions of the stagewise Newton (2.19) and ABR (2.25) are similar. In particular, the last step when step $k = T$, they are the same. For the stagewise Newton for OLNE, $\hat{S}_{T+1} = 0, \hat{M}_{n,T} = M_{n,T}$, therefore

$$\delta u_{:,T} = \hat{K}_T \delta \hat{x}_T = \begin{bmatrix} \hat{K}_{1,T} \\ \hat{K}_{2,T} \\ \vdots \\ \hat{K}_{N,T} \end{bmatrix} \delta \hat{x}_T = - \begin{bmatrix} \hat{M}_{1,T}^{u_1 u_1} & \hat{M}_{1,T}^{u_1 u_2} & \cdots & \hat{M}_{1,T}^{u_1 u_N} \\ \hat{M}_{2,T}^{u_2 u_1} & \hat{M}_{2,T}^{u_2 u_2} & \cdots & \hat{M}_{2,T}^{u_2 u_N} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{M}_{N,T}^{u_N u_1} & \hat{M}_{N,T}^{u_N u_2} & \cdots & \hat{M}_{N,T}^{u_N u_N} \end{bmatrix}^{-1} \begin{bmatrix} \hat{M}_{1,T}^{u_1 x} \\ \hat{M}_{2,T}^{u_2 x} \\ \vdots \\ \hat{M}_{N,T}^{u_N x} \end{bmatrix} \delta \hat{x}_T \tag{2.27a}$$

$$= - \begin{bmatrix} M_{1,T}^{u_1 u_1} & M_{1,T}^{u_1 u_2} & \cdots & M_{1,T}^{u_1 u_N} \\ M_{2,T}^{u_2 u_1} & M_{2,T}^{u_2 u_2} & \cdots & M_{2,T}^{u_2 u_N} \\ \vdots & \vdots & \ddots & \vdots \\ M_{N,T}^{u_N u_1} & M_{N,T}^{u_N u_2} & \cdots & M_{N,T}^{u_N u_N} \end{bmatrix}^{-1} \begin{bmatrix} M_{1,T}^{u_1 1} & M_{1,T}^{u_1 x} \\ M_{1,T}^{u_2 1} & M_{1,T}^{u_2 x} \\ \vdots & \vdots \\ M_{1,T}^{u_N 1} & M_{1,T}^{u_N x} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_T \end{bmatrix} \tag{2.27b}$$

$$= - \begin{bmatrix} M_{1,T}^{u_1 u_1} & M_{1,T}^{u_1 u_2} & \cdots & M_{1,T}^{u_1 u_N} \\ M_{2,T}^{u_2 u_1} & M_{2,T}^{u_2 u_2} & \cdots & M_{2,T}^{u_2 u_N} \\ \vdots & \vdots & \ddots & \vdots \\ M_{N,T}^{u_N u_1} & M_{N,T}^{u_N u_2} & \cdots & M_{N,T}^{u_N u_N} \end{bmatrix}^{-1} \left(\begin{bmatrix} M_{1,T}^{u_1 x} \\ M_{1,T}^{u_2 x} \\ \vdots \\ M_{1,T}^{u_N x} \end{bmatrix} \delta x_T + \begin{bmatrix} M_{1,T}^{u_1 1} \\ M_{1,T}^{u_2 1} \\ \vdots \\ M_{1,T}^{u_N 1} \end{bmatrix} \right) \tag{2.27c}$$

which is exactly the same gain and constant term compared to (2.25g) when $k = T$. The difference between stagewise Newton for OLNE and ABR is that, the former updates the actions directly and the latter computes a feedback policy and updates the nominal trajectory.

2.4 DDP Method for Dynamic Games

This section describes the differential dynamic programming algorithm for solving the local FNE of dynamic games of the form in Problem 1. Subsection 2.4.1 gives a high-level description of the algorithms, while Subsection 2.4.2 describe the explicit matrix calculations.

2.4.1 Algorithm Overview

The idea of the differentiable dynamic programming (DDP) is to maintain quadratic approximations of $V^*_{n,k}$ and $Q^*_{n,k}$ around a trajectory $\bar{u}$ denoted by $\tilde{V}^{\bar{u}}_{n,k}$ and $\tilde{Q}^{\bar{u}}_{n,k}$, respectively.

We need some notation for our approximations. For a scalar-valued function, $h(z)$, we denote the quadratic approximation near $\bar{z}$ by:

$$\text{quad}(h(z))_{\bar{z}} = \frac{1}{2}\begin{bmatrix} 1 \\ \delta z \end{bmatrix}^\top \begin{bmatrix} 2h(\bar{z}) & \frac{\partial h(\bar{z})}{\partial z} \\ \frac{\partial h(\bar{z})}{\partial z}^\top & \frac{\partial^2 h(\bar{z})}{\partial z^2} \end{bmatrix} \begin{bmatrix} 1 \\ \delta z \end{bmatrix} \tag{2.28a}$$

$$\delta z = z - \bar{z}. \tag{2.28b}$$

If $h : \mathbb{R}^n \to \mathbb{R}^m$ we form the quadratic approximation by stacking all of the quadratic approximations of the entries:

$$\text{quad}(h(z))_{\bar{z}} = [\text{quad}(h_1(z))_{\bar{z}},\ \ldots,\ \text{quad}(h_m(z))_{\bar{z}}]^\top \tag{2.29}$$

Let $\bar{x}_k$ and $\bar{u}_{:,k}$ be a trajectory of states and actions satisfying the dynamic equations from (2.2) and $z_k = [x_k^\top, u_{:,k}^\top]^\top$. The differential dynamic programming around this trajectory is given by:

$$\tilde{V}^{\bar{u}}_{n,T+1}(x_{T+1}) = 0 \tag{2.30a}$$

$$\tilde{Q}^{\bar{u}}_{n,k}(z_k) = \text{quad}(c_{n,k}(z_k) + \tilde{V}^{\bar{u}}_{n,k+1}(f_k(z_k)))_{\bar{z}_k} \tag{2.30b}$$

$$\tilde{V}^{\bar{u}}_{n,k}(x_k) = \min_{u_{n,k}} \tilde{Q}^{\bar{u}}_{n,k}(x_k, u_{:,k}). \tag{2.30c}$$

The quadratic approximation is possible because $f_k(z_k)$ and $c_{n,k}(z_k)$ are twice differentiable. Similar to approximated Bellman recursion method and Lemma 3 in Section 2.3.1, the form of solution to (2.30c) is

$$u_{:,k} = \bar{u}_{:,k} + \tilde{K}_k \delta x_k + \tilde{s}_k. \tag{2.31}$$

In the notation defined above, we have that $\delta x_k = x_k - \bar{x}_k$. See Lemma 5 for details and proof. We define a *stationary trajectory of the DDP method* as $\bar{x}, \bar{u}$ such that, the resulting policy (2.31) has zero constant terms $\tilde{s}_k = 0, \forall k$.

2.4.2 Details of DDP method

Using the notation from (2.8), (2.28), (2.29) and z_k, the second-order approximations of the dynamics and cost are given by:

$$\text{quad}(f_k(z_k))_{\bar{z}_k} = f_k(\bar{z}_k) + A_k \delta x_k + B_k \delta u_{:,k} + R_k(\delta z_k) \tag{2.32a}$$

$$\text{quad}(c_{n,k}(z_k))_{\bar{z}_k} = \begin{bmatrix} 1 \\ \delta z_k \end{bmatrix}^\top M_{n,k} \begin{bmatrix} 1 \\ \delta z_k \end{bmatrix}. \tag{2.32b}$$

By construction $\tilde{V}^{\bar{u}}_{n,k}(x_k)$ and $\tilde{Q}^{\bar{u}}_{n,k}(x_k, u_{:,k})$ are quadratic, so there must be matrices $\tilde{S}_{n,k}$ and $\tilde{\Gamma}_{n,k}$ such that

$$\tilde{V}^{\bar{u}}_{n,k}(x_k) = \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_k \end{bmatrix}^\top \tilde{S}_{n,k} \begin{bmatrix} 1 \\ \delta x_k \end{bmatrix} \tag{2.33a}$$

$$\tilde{Q}^{\bar{u}}_{n,k}(x_k, u_{:,k}) = \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top \tilde{\Gamma}_{n,k} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}. \tag{2.33b}$$

The following lemma answers the question of how to compute $\tilde{S}_{n,k}$ and $\tilde{\Gamma}_{n,k}$. The computation is very similar to that of Lemma 4 in Section 2.3.2. See Section 2.6 for detailed analysis on the numerical computation aspects.

Lemma 5. *The matrices in* (2.33) *are defined recursively by* $\tilde{S}_{n,T+1} = 0$ *and:*

$$\tilde{D}_{n,k} = \sum_{l=1}^{n_x} \tilde{S}^{1x^l}_{n,k+1} G^l_k \tag{2.34a}$$

$$\begin{aligned} \tilde{\Gamma}_{n,k} &= M_{n,k} \\ &+ \begin{bmatrix} \tilde{S}^{11}_{n,k+1} & \tilde{S}^{1x}_{n,k+1} A_k & \tilde{S}^{1x}_{n,k+1} B_k \\ A_k^\top \tilde{S}^{x1}_{n,k+1} & A_k^\top \tilde{S}^{xx}_{n,k+1} A_k + \tilde{D}^{xx}_{n,k} & A_k^\top \tilde{S}^{xx}_{n,k+1} B_k + \tilde{D}^{xu}_{n,k} \\ B_k^\top \tilde{S}^{x1}_{n,k+1} & B_k^\top \tilde{S}^{xx}_{n,k+1} A_k + \tilde{D}^{ux}_{n,k} & B_k^\top \tilde{S}^{xx}_{n,k+1} B_k + \tilde{D}^{uu}_{n,k} \end{bmatrix} \end{aligned} \tag{2.34b}$$

$$= \begin{bmatrix} \tilde{\Gamma}^{11}_{n,k} & \tilde{\Gamma}^{1x}_{n,k} & \tilde{\Gamma}^{1u_1}_{n,k} & \tilde{\Gamma}^{1u_2}_{n,k} & \cdots & \tilde{\Gamma}^{1u_N}_{n,k} \\ \tilde{\Gamma}^{x1}_{n,k} & \tilde{\Gamma}^{xx}_{n,k} & \tilde{\Gamma}^{xu_1}_{n,k} & \tilde{\Gamma}^{xu_2}_{n,k} & \cdots & \tilde{\Gamma}^{xu_N}_{n,k} \\ \tilde{\Gamma}^{u_1 1}_{n,k} & \tilde{\Gamma}^{u_1 x}_{n,k} & \tilde{\Gamma}^{u_1 u_1}_{n,k} & \tilde{\Gamma}^{u_1 u_2}_{n,k} & \cdots & \tilde{\Gamma}^{u_1 u_N}_{n,k} \\ \tilde{\Gamma}^{u_2 1}_{n,k} & \tilde{\Gamma}^{u_2 x}_{n,k} & \tilde{\Gamma}^{u_2 u_1}_{n,k} & \tilde{\Gamma}^{u_2 u_2}_{n,k} & \cdots & \tilde{\Gamma}^{u_2 u_N}_{n,k} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{\Gamma}^{u_N 1}_{n,k} & \tilde{\Gamma}^{u_N x}_{n,k} & \tilde{\Gamma}^{u_N u_1}_{n,k} & \tilde{\Gamma}^{u_N u_2}_{n,k} & \cdots & \tilde{\Gamma}^{u_N u_N}_{n,k} \end{bmatrix} \tag{2.34c}$$

$$\tilde{F}_k = \begin{bmatrix} \tilde{\Gamma}^{u_1 u}_{1,k} \\ \tilde{\Gamma}^{u_2 u}_{2,k} \\ \vdots \\ \tilde{\Gamma}^{u_N u}_{N,k} \end{bmatrix} = \begin{bmatrix} \tilde{\Gamma}^{u_1 u_1}_{1,k} & \tilde{\Gamma}^{u_1 u_2}_{1,k} & \cdots & \tilde{\Gamma}^{u_1 u_N}_{1,k} \\ \tilde{\Gamma}^{u_2 u_1}_{2,k} & \tilde{\Gamma}^{u_2 u_2}_{2,k} & \cdots & \tilde{\Gamma}^{u_2 u_N}_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\Gamma}^{u_N u_1}_{N,k} & \tilde{\Gamma}^{u_N u_2}_{N,k} & \cdots & \tilde{\Gamma}^{u_N u_N}_{N,k} \end{bmatrix} \tag{2.34d}$$

$$\tilde{P}_k = \begin{bmatrix} \tilde{\Gamma}^{u_1 x}_{1,k} \\ \tilde{\Gamma}^{u_2 x}_{2,k} \\ \vdots \\ \tilde{\Gamma}^{u_N x}_{N,k} \end{bmatrix}, \quad \tilde{H}_k = \begin{bmatrix} \tilde{\Gamma}^{u_1 1}_{1,k} \\ \tilde{\Gamma}^{u_2 1}_{2,k} \\ \vdots \\ \tilde{\Gamma}^{u_N 1}_{N,k} \end{bmatrix} \tag{2.34e}$$

$$\tilde{s}_k = -\tilde{F}_k^{-1}\tilde{H}_k, \quad \tilde{K}_k = -\tilde{F}_k^{-1}\tilde{P}_k \tag{2.34f}$$

$$\tilde{S}_{n,k} = \begin{bmatrix} 1 & 0 & \tilde{s}_k^\top \\ 0 & I & \tilde{K}_k^\top \end{bmatrix} \tilde{\Gamma}_{n,k} \begin{bmatrix} 1 & 0 \\ 0 & I \\ \tilde{s}_k & \tilde{K}_k \end{bmatrix}, \tag{2.34g}$$

for $k = T, T-1, \ldots, 0$.

Proof. By construction we must have $\tilde{S}_{n,T+1} = 0$. Plugging (2.32a) into (2.30b) and dropping all cubic and higher terms gives (2.34a)(2.34b). Since $u_{:,k} = \bar{u}_{:,k} + \delta u_{:,k}$ and $\bar{u}_{:,k}$ is constant, the static game defined in (2.30c) can be solved in the $\delta u_{:,k}$ variables. Differentiating (2.33b) by $\delta u_{n,k}$, collecting the derivatives for all players and setting them to zero leads to the necessary condition for an equilibrium:

$$\tilde{F}_k \delta u_{:,k} + \tilde{P}_k \delta x_k + \tilde{H}_k = 0. \tag{2.35}$$

Thus, the matrices for the equilibrium strategy are given in (2.34f). Plugging (2.31) into (2.33b) leads to (2.34g). □

We can see that the matrices used in the recursions for both DDP and approximated Bellman

recursion methods are very similar in structure. Indeed, the iterations are identical aside from the definitions of the $D_{n,k}$ and $\tilde{D}_{n,k}$ matrices.

Remark 1. *The calculations of ABR and DDP for games are similar to those arising in single-agent optimal control. The difference is that the game case inverts the matrices F_k and $\tilde{F}_k$ which are constructed from submatrices of the value function matrices, $\Gamma_{n,k}$ and $\tilde{\Gamma}_{n,k}$. In contrast, the single agent algorithms invert $\Gamma_{n,k}$ and $\tilde{\Gamma}_{n,k}$ directly. It is due to this difference, that the proof for game scenario requires separate though similar treatment to those of [75, 76].*

2.5 Equilibria for the FNE Methods

Local FNEs are practical for stochastic applications as is as long as the system does not deviate too far from the nominal trajectory. We focus on the local FNE of Problem 1 that can be found via ABR and DDP in this section. In general, it is very hard to find conditions that guarantee the existence of feedback Nash equilibria or stationaries trajectories. Our argument in this section makes assumption of their existence.

Suppose the ABR and DDP find their respective stationary trajectories $\bar{u}^\star, \bar{x}^\star$ and $\tilde{u}^\star, \tilde{x}^\star$. We study the two closed-loop policies found by the ABR $u_{:,k} = \hat{\phi}_k^\star(x_k) = \bar{u}_{:,k}^\star + K_k^\star \delta x_k + s_k^\star$ and DDP $u_{:,k} = \tilde{\phi}_k^\star(x_k) = \tilde{u}_{:,k}^\star + \tilde{K}_k^\star \delta x_k + \tilde{s}_k^\star$. Our second theorem states that the feedback policies generated by approximated Bellman and DDP around their respective stationary trajectories are approximated local feedback Nash equilibria.

Theorem 2. *The feedback policies by approximated Bellman recursion $\hat{\phi}_k^\star(\cdot)$ and DDP $\tilde{\phi}_k^\star(\cdot)$ around the stationary trajectory are local feedback $O(\varepsilon^2)$-Nash equilibria in the sense of Definition 2. More specifically,*

$$J_{n,t}(x_t, \hat{\phi}_{:,t:}^\star) \leq J_{n,t}(x_t, \phi_{n,t:}, \hat{\phi}_{-n,t:}^\star) + O(\varepsilon^2),\ \forall t, n$$
$$J_{n,t:}(x_t, \tilde{\phi}_{:,t:}^\star) \leq J_{n,t}(x_t, \phi_{n,t:}, \tilde{\phi}_{-n,t:}^\star) + O(\varepsilon^2),\ \forall t, n$$

Proof. We first prove the results corresponding to ABR method. Fix a player n, and assume that the other players are using the strategy profile $\hat{\phi}_{-n,t:}$ for a subgame t. Then the optimal policy of player n that minimizes $J_{n,t:}(x_t, \phi_{n,t:}, \phi_{-n,t:}^\star)$ can be computed from the following optimal control problem:

$$\min_{u_n} \quad \sum_{k=t}^{T} c_{n,k}(x_k, u_{:,k}) \tag{2.37a}$$

$$\text{s.t.} \quad u_{-n,k} = \hat{\phi}^{\star}_{-n,k}(x_k) \tag{2.37b}$$

$$x_{k+1} = f_x(x_k, u_{:,k}) \tag{2.37c}$$

$$k = t, t+1, ..., T-1 \tag{2.37d}$$

$$x_t \text{ is given.} \tag{2.37e}$$

The optimal control problem is reduced from the dynamic game by substituting the equilibrium policy of other players. To show that the approximated Bellman recursion method is an approximate feedback equilibrium, it suffices to show that $\hat{\phi}_n, t$: is approximately optimal for this problem, for all n and t.

The quadraticization of problem (2.37) around $[\bar{x}, \bar{u}]$ is the same as the part of player n in the quadraticized dynamic game as in (2.21). Since it is assumed that the other players are playing their equilibrium strategies for the quadraticized game, the optimal strategy for this quadraticized control problem is precisely given by player n's solution from the approximated Bellman recursion method $\hat{\phi}_{n,t:}$. The approximate optimality of $\hat{\phi}_{n,t:}$ now follows from Lemma 9 in Appendix A.5. DDP is quadratically close to ABR as evident by the Section A.8, therefore shares the same feedback $O(\varepsilon^2)-$Nash equilibrium. □

2.6 Implementation Details of the Algorithms

The stagewise Newton method for OLNE is a computationally cheap way of performing the Newton step and inherits all Newton's step's properties, which requires little extra care for implementation compared with the two local FNE methods as our experiments indicat. Despite their different origins, the two FNE methods are almost the same for applications. In general, it is hard to tell which method works better for a specific application beforehand.

2.6.1 Choice of Initial Trajectory

Similar to Newton's method for nonlinear optimization, it is hard to find general global convergence condition except for convexity. In cases when there are multiple OLNEs, the algorithm might converge to different ones when started from different initial trajectories. It is reasonable to think that in general, we need to start with a trajectory that is fairly close to the actual stationary trajectory. Finding a good initial trajectory should require expert knowledge of the specific dynamic game and some heuristic methods. For example, a complicated game can be simplified

via ignoring or approximating some complex terms in dynamics and cost functions and solved, then the resulting can be used as an initial trajectory for the original game. We acknowledge this difficulty, but it is beyond the scope of this thesis.

2.6.2 Computing Derivatives

For complicated nonlinear dynamics and costs, modern algorithmic differentiation (AD) software packages, such as Tensorflow [77], Pytorch [78], CasADi [79], are strongly recommended. Automatic differentiation is widely applied in the training of modern neural networks. With the help of AD and given trajectory $\bar{u}, \bar{x}$, the derivatives matrices A_k, B_k, G_k^l and $M_{n,k}$ in (2.8) can be easily computed. Compared to modern neural networks, the complexity of dynamic game formulations are relatively very low, therefore the computation cost of this step should not be bottle neck for our application.

With the help of automatic differentiation, there is one more way to implement the DDP algorithm. In the backward pass for the DDP method, the quadratic approximation in (2.30b) and $\tilde{\Gamma}_k$ can be directly computed, without computing the derivatives of a single step A_k, G_k^l, $M_{n,k}$ in (2.8) or keeping $\tilde{D}_{n,k}$ in (2.34a). It is not obvious a priori whether this approach will be more efficient than computing (2.8). It will depend on the specific hardware and problem.

2.6.3 Regularization

To ensure that the algorithms converge regardless of initial condition, a Levenberg-Marquardt style regularization should be employed. Such regularization has been used in DDP algorithms for optimal control to ensure that the required inverses exist and that the solution improves [80, 81]. We found in practice that regularization is essential to the stability for both FNE algorithms. We only use the notation for DDP for simplicity in this section but the same insights hold for approximated Bellman recursion method. At each step k of the backward pass, we checked the minimal eigenvalue $e_{n,k}$ of matrix $\tilde{\Gamma}_{n,k}$. If the minimal eigenvalue $e_{n,k}$ is less than a positive value λ, we reset $\tilde{\Gamma}_{n,k}$ as

$$\tilde{\Gamma}_{n,k} \leftarrow \tilde{\Gamma}_{n,k} + (\lambda - e_{n,k})I, \quad n = 1, 2, \cdots, N \tag{2.38}$$

where I is an identity matrix. The regularization penalizes large steps in δx and δu. Thus, it improves the stability of the algorithm, but sacrifices speed of convergence. The λ in our examples are chosen via experimental trials and kept constant for our examples in section 2.7. When

the algorithm was insufficiently regulated, the trajectories over iterations did not converge from the initial trajectory for our examples. Changing the regularization over iterations while guaranteeing quadratic convergence has been studied in differential dynamic programming literature and is referred to as *adaptive shift for DDP* [80]. Since it is not the focus of this thesis, we settled at a constant regularization that enabled smooth and steady improvement of trajectories over iterations for our examples.

2.6.4 Computational Complexity

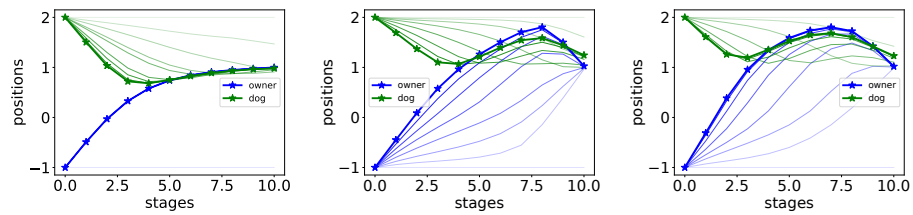
The computational cost of all algorithms come mainly from evaluating multiple derivatives according to (2.8) and do backward passes according to either (2.19), (2.25) or (2.34). The former requires T evaluations of A_k, B_k, which are the first order derivatives of the dynamic, Tn_x evaluations of G_k^l, which is the second order derivative of the dynamic. TN evaluations of $M_{n,k}$ are also required as they contain the first and second order derivatives of the cost functions. The latter consists of mainly matrix multiplication and solving linear equations in either (2.19c), (2.25g) or (2.34f). The overall complexity for all, depending on the implementation, can roughly vary from $O(\min(n_u^2, n_x^2))$ to $O(\max(n_u^3, n_x^3))$ for each step in the backward pass, which is constant for given system and agents. The advantage of the algorithms over other general GNEP methods is that, the complexity w.r.t. number of stages is linear, i.e. $O(T)$, since the only dependency on it is that we need to do each stagewise computation for T times. As can be seen based on the complexity analysis, the algorithms are better suited for longer horizon (or finer discretization of continuous problems) dynamic games, rather than games with a large or infinite number of agents.

2.7 Numerical Examples

We apply the proposed algorithms for deterministic nonlinear dynamic games to two examples in this section. We compare the performances of both algorithms on a simple toy example first, and then apply the DDP method to a more complicated problem. We gain proof of concept that both methods performs reasonably close in practice and that they can be extended to complicated models.

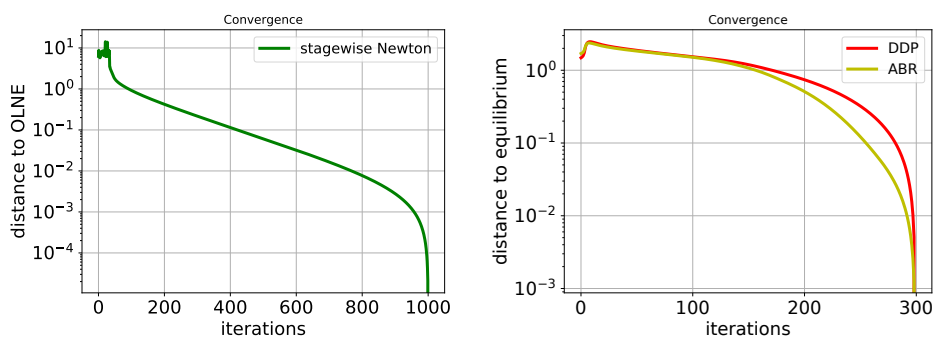
2.7.1 Owner-dog Dynamic Game

First we look at a toy example, which is implemented in Python and all derivatives of nonlinear functions are computed via Tensorflow [77].



(a) stagewise Newton OLNE (b) DDP method (c) ABR

Figure 2.1: Owner-dog dynamic game equilibrium trajectories. Lighter colored trajectories are earlier in the overall iterations. The starred trajectory is the final equilibrium solution. For DDP and ABR, we sampled 8 trajectories uniformly spaced out of 300. For stagewise Newton, we sampled 10 trajectories from 1000 iterations.



(a) stagewise Newton OLNE (b) local FNE methods

Figure 2.2: This shows the 2-norm distance between inputs $\bar{u}$ and the final stationary point $u^\star$ over iterations for all algorithms.

We consider a simple 1-D owner-dog problem, with horizon $T = 10$ and initial state $x_{:,0} = [-1, 2]$ where the dynamics of the owner and the dog are given respectively by

$$x_{0,k+1} = x_{0,k} + \tanh(u_{0,k}) \tag{2.39a}$$

$$x_{1,k+1} = x_{1,k} + \tanh(u_{0,k}) \tag{2.39b}$$

$$k = 0, 1, \cdots, T-1 \tag{2.39c}$$

The owner cares about going to $x_{0,k} = 1$ and that the dog can stay at $x_{1,k} = 2$. The dog, however, only tries to catch up with the owner. Each player also concerns themselves with the energy consumption, therefore has a cost term related to the magnitude of its input. Their cost functions are formulated as

$$c_{0,k}(x,u) = \text{sigmoid}((x_{0,k}-1)^2) + 40(x_{1,k}-2)^2 + (u_{0,k})^2 \tag{2.40a}$$

$$c_{1,k}(x,u) = \tanh^2(x_{0,k} - x_{1,k}) + (u_{1,k})^2 \tag{2.40b}$$

$$k = 0, 1, ..., T-1 \tag{2.40c}$$

We use a different terminal cost that penalizes much more heavily the owner for not reaching to their target $x_{0,T} = 1$.

$$c_{0,T}(x,u) = 100\ \text{sigmoid}((x_{0,T}-1)^2) + 40(x_{1,T}-2)^2 \tag{2.41a}$$

$$c_{1,T}(x,u) = \tanh^2(x_{0,T} - x_{1,T}) \tag{2.41b}$$

Nonlinear functions are added to the dynamics and costs to create a nonlinear game rather than for explicit physical meaning. We initialize a trajectory with zero input and initial state, i.e. $\bar{u} = [0., 0., \ldots, 0.]$ and $\bar{x} = [-1, 2, -1, 2, \ldots, -1, 2]$. For the local FNE methods, we used an identity regularization matrix with a magnitude of $\lambda = 30$ as in (2.38) and performed 300 iterations from the initial trajectory. Note that we started the iteration with a trajectory that is far from a local equilibrium, therefore we do not expect the updates generated by both algorithms to be close. For the stagewise Newton method, we simulated 1000 iterations.

Fig. 2.1 shows the stationary trajectories found via all algorithms. In order to keep the dog around $x_{1,k} = 2$, the owner has to overshoot and then come back to $x_{0,T} = 1$ for the local FNE methods. For the stagewise Newton method for OLNE, we observe a different stationary trajectory, highlighting the fact that OLNE and FNE can be very different due to the different structure of their formulation. The dog learns to get closer to the owner over iterations in all cases, which is what we would expect given how the problem is formulated. The approximated Bellman recursion step generated a smoother trajectory in this particular case than DDP. Fig. 2.2 shows the distances of input to the final equilibrium over all 300 iterations for the two local FNE methods and 1000 iterations for the OLNE method. As can be seen that the error reduces sub-linearly on a log scaled plot for all of them, which is evidence that the algorithms converge

quadratically. The ABR method converges quicker in this particular case. Note that the two methods did not converge to the same trajectory, which is because we fixed a regularization λ and started far off the equilibrium. By tuning the regularization with iterations or start from a closer trajectory to the equilibrium, we should improve the situation, which is beyond the scope of this thesis.

2.7.2 Planar Robots Target Reaching

Here we consider an experimental setup in which three planar robots try to reach each of their own targets, while avoiding collisions with other robots. The problem is set up such that, if the robots ignore the existence of others and run its own optimal trajectory, they will collide. We apply the proposed DDP algorithm for game to solve for the equilibrium trajectories. This example is implemented in Python and derivatives are computed via PyTorch [78].

Robots are modeled as circles on a plane with the location of its center and a diameter. All robots share the same dynamics given by (2.42). The state $x_{:,k}$ collects all vehicles' positions and $x_{n,k}$ picks the nth vehicle's position at step k.

$$x_{n,k+1} = x_{n,k} + (\tanh u_{n,k})\mathrm{d}t, \quad n = 1,2,\ldots,N \tag{2.42}$$

where n enumerates all robots. The step cost of each robot consists of a goal cost, a control cost and, and an avoidance cost. To compute the avoidance cost, we check the distances among robots at each step, when the robots do collide and the distances become negative, we set it to the small positive number, 0.01, for numerical stability. Cost functions follow (2.43).

$$\begin{aligned} c_{n,k}(x_k,u_k) =& \alpha(1-e^{-||x_{n,k}-g_n||^2}) + ||u_{n,k}||^2 + \\ & \beta\sum_{i\neq n}[-\log(1-e^{-\max(||x_{i,k}-x_{n,k}||-r_i-r_n,\ 0.01)})] \end{aligned} \tag{2.43}$$

$$n = 1,2,3 \tag{2.44}$$

where g_n is the target of the nth vehicle, r_n is the radius of the nth robot, which are constants given the problem, α and β are parameters controlling the relative weights of these three cost terms. The costs are coupled via the distances between robots.

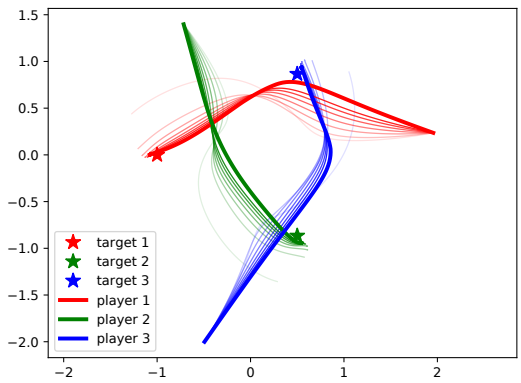


Figure 2.3: Robots trajectories over iterations. As can be seen that the robots are taking indirect routes to their targets to avoid colliding into each other. As we optimize over the trajectory via the proposed algorithm, the trajectory becomes smoother and the end location closer to the targets.

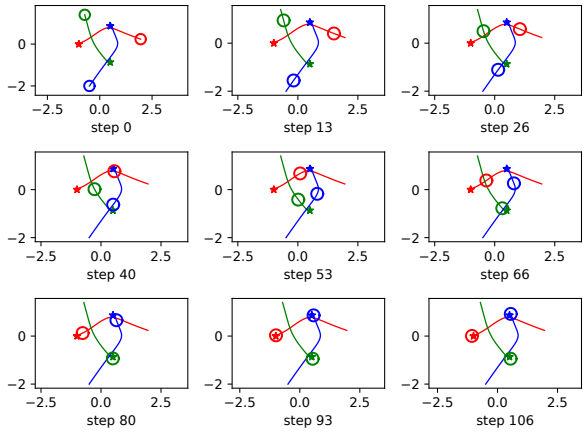


Figure 2.4: Snapshots of robots position of equilibrium trajactory. Robots are avoiding each other and keeping proper distances from each other.

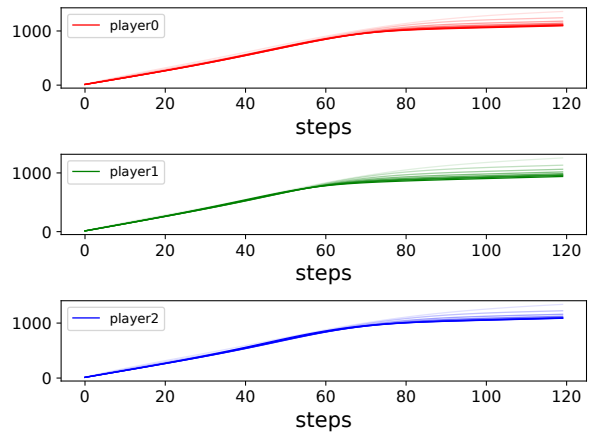


Figure 2.5: Cumulative costs. The cumulative cost for all robots reduces over iterations.

In our particular implementation, we used the parameters $T = 119$, $\mathrm{d}t = 0.04$, $\alpha = 10$, $\beta = 3$ and $r_i = 0.25$ for all robots. The targets are chosen as $g_1 = [-1, 0]^\top$, $g_2 = [0.5, -0.866]^\top$ and $g_3 = [0.5, 0.866]^\top$. The robots are initialize at

$$x_0 = [1.96, 0.24, -0.72, 1.39, -0.49, -2.00]^\top \tag{2.45}$$

Figures 2.3, 2.4 and 2.5 show the result of implementing DDP method to this problem where lighter color means values from earlier iterations. An initial trajectory was generated with a naïve push-pull control, which assumes each robot is being pulled to its target by an input that is proportional to the distance to the target, and being pushed away from other robots by an input that is inversely proportional to the squared distance to the other robots. This simple control scheme enables the robots to reach their targets given sufficient horizon but requires large inputs and takes sharper turns therefore is far from optimal. Collisions did not happen in any iteration. A regularization magnitude of $\lambda = 10$ as in (2.38) was implemented in this example. Due to heavy computational complexity, we only ran the algorithm for 7 iterations. Computing Jacobians in (2.30b) took up the majority of the program's runtime.

2.8 Conclusion and Future Directions

In this chapter we have shown how Newton's method, Bellman recursion and differential dynamic programming extend to dynamic games. In particular, the stagewise Newton's method

can find local open-loop Nash equilibria, ABR and DDP methods can find local feedback Nash equilibria. Convergence of the stagewise Newton' method is proven and the nature of all the equilibria are studied. We demonstrated the performance of all algorithms with nonlinear dynamic games in simulation. Many extensions are possible. We will examine larger examples and work on numerical scaling. Derivative-free methods with convergence guarantee are also attractive since they can be computationally faster and eliminate the dependence on analytical models. All methods can be applied as part of projected gradient descent style methods or operator splitting method [43] to constrained dynamic games. Additionally, handling scenarios in which agents have imperfect model information will be of great practical importance.

Chapter 3

First-Order Algorithms for Constrained Dynamic Games

We study constrained dynamic games in this chapter. Because we allow coupling w.r.t. different players' actions in the constraints, we are studying the open-loop/feedback generalized Nash equilibrium problems (GNEP).

We examined typical numerical methods from optimization and variational inequality communities and found that gradient descent and Douglas-Rachford splitting method can be adapted to constrained dynamic games. Similar to their original implementation, both methods, after adapted to dynamic games, alternate between solving two different control/game problems and converge to the final open-loop Nash equilibrium solution.

We describe efficient implementations of projected gradient and Douglas-Rachford splitting methods for computing OLNEs of constrained dynamic games. In both cases, we show how to exploit the temporal structure to achieve iterations of $O(T)$ complexity, where T is the total number of stages.

For the feedback information structure, we adapted the approximated Bellman recursion to solve a feedback policy around given trajectories. In the case of games with linear dynamics, polyhedral constraints and proper stationary trajectory, we show that it is an approximate (generalized) feedback Nash equilibrium.

The chapter is organized as following. We begin in Section 3.1 by stating the basic problem

formulation. Section 3.2 and 3.3 explains two methods that solve the open-loop Nash equilibrium, the projected gradient and the operator splitting method. The discussion about feedback Nash equilibrium starts from Section 3.4, which first lay down some ground related to linearly constrained quadratic static/dynamic games.

3.1 Problem Formulations and Solution Concepts

We introduce some standard notation, formulate the constrained dynamic game problems, and introduce the associated solution concepts.

3.1.1 Dynamic and Static Game Formulation

The main problem of interest is a constrained, deterministic, full-information dynamic game with N players of the form below.

Problem 3. ***Constrained nonlinear dynamic game***
Each player aims to minimize their own cost

$$\min_{u_{n,t:}} \quad J_{n,t}(x,u) = \sum_{k=t}^{T} c_{n,k}(x_k, u_{:,k}) \tag{3.1}$$

Subject to dynamic constraints $f_k(\cdot)$ and extra constraints $g_{n,k}(\cdot)$

$$\begin{aligned} s.t. \quad & x_{k+1} = f_k(x_k, u_{:,k}) && (3.2a)\\ & g_k(x_k, u_{:,k}) \leq 0, && (3.2b)\\ & x_t \text{ is fixed.} && (3.2c)\\ & k = t, t+1, \ldots, T-1. && (3.2d) \end{aligned}$$

Here, $0 \leq t \leq T$ is the starting point for a game. When $t = 0$, we call it the *full game*, and $t > 0$, a *subgame* t. Here, the state of the system at time k is denoted by $x_k \in \mathbb{R}^{n_x}$. Player n's action at time k is given by $u_{n,k} \in \mathbb{R}^{n_{u_n}}$. The vector of all players' actions at time k is denoted $u_{:,k} = [u_{1,k}^\top, u_{2,k}^\top, \ldots, u_{N,k}^\top]^\top \in \mathbb{R}^{n_u}$. The cost for player n at time k is $c_{n,k}(x_k, u_{:,k})$. This encodes the fact that the cost for each player can depend on the actions of all the players. We assume that the costs are twice differentiable with locally Lipschitz Hessians.

In later analysis, some other notation is helpful. The actions of player n from time t to T in a vector is denoted $u_{n,t:} = [u_{n,t}^\top, u_{n,t+1}^\top, \ldots, u_{n,T}^\top]^\top$. The vector of actions other than those of

player n from time t to T is denoted by $u_{-n,t:} = [u_{1,t:}^\top, \ldots, u_{n-1,t:}^\top, u_{n+1,t:}^\top, \ldots, u_{N,t:}^\top]^\top$. The vector of states from time t to T is denoted $x_{t:} = [x_t^\top, x_{t+1}^\top, \ldots, x_T^\top]^\top$ while the vector of all players' actions from t to T is given by $u_{:,t:} = [u_{1,t:}^\top, u_{2,t:}^\top, \ldots, u_{N,t:}^\top]^\top$. x, u denote all states and actions collected over all time in a vector.

We denote the set of trajectories $[x_{:,t:}, u_{:,t:}]$ satisfying the dynamic constraint with given x_t as $\mathcal{D}_t(x_t)$ and the extra constraints $\mathcal{G}_t(x_t)$. Feasible sets of state and actions are denoted $\mathcal{X}_t(x_t)$ and $\mathcal{U}_t(x_t)$ for subgame t. The subscript t or initial condition x_0 can be suppressed later in this chapter indicating values for the full game.

Note that the dynamics are deterministic, the cost for each player in a subgame can be expressed as a function of all following actions and given state x_t, i.e., $J_{n,t}(x_t, u_{:,t:})$. The dynamics are implicitly substituted to eliminate x when we use such notation. Problem 3 can be written in an equivalent static game form for ease of notation and analysis.

Problem 4. ***Static form of game Problem 3***
A static form equivalent to Problem 3 is denoted

$$\min_{u_{n,t:}} \quad J_{n,t}(x_t, u_{:,t:}) \tag{3.3a}$$

$$s.t. \quad u_{:,t:} \in \mathcal{U}_t(x_t) \tag{3.3b}$$

3.1.2 Solution Concept of Games

We focus on local open-loop Nash equilibria and feedback Nash equilibria in this chapter. Technically, we are studying generalized Nash equilibria, since players' actions can be coupled in the constraints [73]. However, for simplicity, we will refer to them as Nash equilibria.

Definition 3. ***(Local) open-loop Nash equilibrium***
An open-loop Nash equilibrium (OLNE) *for subgame t of problem 3 and 4 with one specific state x_t is a set of actions $u_{:,t:}^\star \in \mathcal{U}_t(x_t)$ such that*

$$J_{n,t}(x_t, [u_{n,t:}, u_{-n,t:}^\star]) \geq J_n(x_t, u_{:,t:}^\star),\ n = 1, 2, \ldots, N \tag{3.4}$$

Furthermore, if (3.4) *only holds for $u_{n,t:} \in \mathcal{U}_{n,t}(x_t)$ in a neighborhood of $u_{n,t:}^\star$, it is called a* local open-loop Nash equilibrium (local-OLNE).

Problem 3 and Problem 4 are equivalent in terms of a local OLNE. An OLNE does not dynamically adjust if the state changes. In contrast, a feedback Nash equilibrium for dynamic

games Problem 3 requires players to be able to measure the state x_k at each step and execute a step-by-step policy $u_{:,k} = \phi^\star_{:,k}(x_k)$ to account for changes in the state. FNE has the valuable property of being subgame perfect [11].

Definition 4. ***(Local) feedback Nash equilibrium***
A collection of feedback policies $u_{n,k} = \phi^\star_{n,k}(x_k)$, with $\bar{u}_{n,k} = \phi^*_{n,k}(\bar{x}_k)$ $\forall n \in \{1,2,...,N\}, \forall k \in \{0,1,\ldots,T-1\}$ is said to be a feedback Nash equilibrium of the full game if no player can benefit from changing their policy unilaterally for any subgame t, i.e.,

$$J_{n,t}(x_t, \phi^\star_{:,t:}) \leq J_{n,t}(x_t, [\phi_{n,t:}, \phi^\star_{-n,t:}]),\ \forall t \in \{0,1,...,T\}, \tag{3.5}$$

where $J_{n,t}(x_t, \phi_{:,t:})$ indicates the total cost of player n when all players follow policy $\phi_{:,t:}$ for subgame t. All policies should be compatible with the constraints. Furthermone, if (3.5) *only holds around $[\bar{x}, \bar{u}] \in \mathcal{D} \cap \mathcal{G}$ and the resulting trajectories remain in a neighborhood of $[\bar{x}, \bar{u}]$, it is called a* local feedback Nash equilibrium (local-FNE).

This definition only applies to Problem 3 because Problem 4 does not have explicit state information. Ideally, an FNE is solved via the Bellman recursion, which originated from optimal control problems, and was extended to dynamic games [51, 11]. Instead of solving the minimizing action at each stage, equilibrium of stagewise games are computed via the following recursion

$$V^\star_{n,T+1}(x_{T+1}) = 0 \tag{3.6a}$$

$$Q^\star_{n,k}(x_k, u_{:,k}) = c_{n,k}(x_k, u_{:,k}) + V^\star_{n,k+1}(f_k(x_k, u_{:,k})) \tag{3.6b}$$

$$V^\star_{n,k}(x_k) = \min_{\phi_{n,k}} Q^\star_{n,k}(x_k, \phi_{n,k}(x_k)),\ \text{s.t. } g_k(x_k, \phi_{:,k}(x_k)) \leq 0 \tag{3.6c}$$

$$k = 0,1,...,T. \tag{3.6d}$$

Here $V^\star_{n,k}(x_k)$ and $Q^\star_{n,k}(x_k, u_{:,k})$ are referred to as *equilibrium value functions* for player n at time step k. Note that (3.6c) defines a parametric static game in terms of the $u_{:,k}$ variable at step k and parameterized by x_k. For dynamic games with quadratic costs, linear dynamics and polyhedral constraints, the analytical FNE can be obtained in theory as described in Section 3.4 and Appendix B.2. In general, this backward recursion is intractable, so we focus on solving it approximately around a stationary trajectory. Note that the game ends at $k = T$, and setting $V^\star_{n,T+1}(x_{T+1}) = 0$ is only for ease of notation and that by construction, $V^*_{n,t}(x_t) = J_{n,t}(x_t, \phi^*_{:,t:})$ for $t = 0,\ldots,T$.

3.1.3 Variational Inequality (VI) Formulation

For continuous-variable games, variational inequalities (VIs) give a necessary condition for the solution of generalized Nash equilibrium problems [41]. We describe the VI formulation of the full game, which starts at $t = 0$. The formulation for games starting at $t \geq 1$ is simular. We omit the dependency on x_0, stack all of the gradient vector $\frac{\partial J_{n,0}(u,x_0)}{\partial_{u_{n,:}}}$ and define

$$\mathcal{J}(u) = \begin{bmatrix} \frac{\partial J_{1,0}}{\partial u_{1,:}} & \frac{\partial J_{2,0}}{\partial u_{2,:}} & \cdots & \frac{\partial J_{N,0}}{\partial u_{N,:}} \end{bmatrix}^\top. \tag{3.7}$$

Note that $\mathcal{J}(u)$ has the same dimension as u. When the dependence on x_0 must be emphasized, we denote the corresponding function by $\mathcal{J}(x_0, u)$. The VI formulation of the full game, Problem 4, is as following.

Problem 5. ***VI formulation of static game***

$$\textit{Find } u^\star \in \mathcal{U} \textit{ s.t. } \mathcal{J}^\top(u)(u - u^\star) \geq 0, \ \forall u \in \mathcal{U}. \tag{3.8}$$

Note that $u^\star$ is the solution of VI formulation is only necessary to $u^\star$ being a local OLNE to Problem 4. To ensure sufficiency, it should also be checked if each player's action is a local minimizer to their objective. Two sufficient conditions for player n's cost to be locally minimized are 1) $\frac{\partial J_n}{\partial u_{n,:}}(u^\star) \neq 0$ and 2) $\frac{\partial J_n}{\partial u_{n,:}}(u^\star) = 0$ and $J_n(u_{n,:})$ is locally convex. If $u^\star$ is a local minimizer for all players, then it is also a local OLNE solution to Problem 4. We refer to this procedure as *playerwise local minimizer check*.

3.1.4 Sufficient Conditions for Existence of OLNEs

A commonly applicable sufficient condition for a solution of the VI problem to exist is that $\mathcal{J}(u)$ is locally strongly monotone and $\mathcal{U}$ is convex. Another sufficient condition is that, $\mathcal{U}$ is compact convex and $\mathcal{J}(u)$ is continuous on $\mathcal{U}$ [41].

In the case if $J_n(x_0, u)$ is convex w.r.t. $u_{n,:}$, the playerwise local minimizer check described above is satisfied for all players, and therefore the game has an OLNE solution. If $\mathcal{J}(x_0, u)$ is continuous in x_0 (which is guaranteed by our differentiability assumption) and locally strongly monotone with respect to u, the dynamic games can be solved for all states near x_t.

3.2 The Projected Gradient Method

The projected gradient method for monotone VI problems was explained in details in [41]. We briefly describe the method and its basic application to Problem 5, which helps find an OLNE for Problem 4. We show how this leads to an algorithm for constrained dynamic games.

3.2.1 Projected Gradient Method for VI

The projection algorithm follows an iterative update

$$u^{t+1} = \Pi_{\mathcal{U}}(u^t - \rho \mathcal{J}(u)), \tag{3.9}$$

where ρ is a step size and Π is the projection operator. The algorithm converges linearly with constant $(1+\rho^2 L^2 - 2\rho\mu)$ when $\mathcal{J}(u)$ is μ-strongly monotone, L-Lipschitz, i.e.,

$$(\mathcal{J}(u) - \mathcal{J}(v))^\top (u - v) \geq \mu \|u - v\|^2 \tag{3.10a}$$

$$\|\mathcal{J}(u) - \mathcal{J}(v)\| \leq L \|u - v\|, \tag{3.10b}$$

and that $\rho L^2 \leq 2\mu$ [41]. There exists a small ρ that guarantees the linear convergence, although smaller ρ leads to slower convergence. To apply the projection method to constrained dynamic games, we need procedures to compute the gradient $\mathcal{J}(u)$ and the projection onto $\mathcal{U}$.

Given feasible $\bar{u}$ and $\bar{x}$ such that $[\bar{x}, \bar{u}] \in \mathcal{D}$, the gradient $\mathcal{J}(\bar{u})$ can be found efficiently by first performing a backward pass (3.11)[82], then extracting and stacking corresponding elements in $\left.\frac{\partial J_n(u,x_0)}{\partial u_{:,k}}\right|_{\bar{u}}$.

$$\Omega_{n,T+1} = 0 \tag{3.11a}$$

$$\Omega_{n,k} = M^{1x}_{n,k} + \Omega_{n,k+1} A_k \tag{3.11b}$$

$$\left.\frac{\partial J_n(u,x_0)}{\partial u_{:,k}}\right|_{\bar{u}} = M^{1u}_{n,k} + \Omega_{n,k+1} B_k \tag{3.11c}$$

$$k = T, T-1, \ldots, 0, \tag{3.11d}$$

where A_k, B_k, M_k are derivatives of the dynamics and cost evaluated around $\bar{x}, \bar{u}$ defined in (3.34) in Section 3.5.

The projection onto $\mathcal{U}$ requires solving a constrained optimal control problem with quadratic step costs, which can be solved via classic optimal control methods [83].

Problem 6. ***Constrained optimal control around nominal trajectory*** $\bar{u}$

$$\min_{u} \quad J(u) = \sum_{k=0}^{T} \|u_{:,k} - \bar{u}_{:,k}\|^2 \tag{3.12a}$$

$$s.t. \quad g_k(x_k, u_{:,k}) \leq 0, \tag{3.12b}$$

$$x_{k+1} = f_k(x_k, u_{:,k}) \tag{3.12c}$$

$$k = 0, 1, ..., T-1. \tag{3.12d}$$

The notation J is overloaded and is different from that of the formulations of Problem 3 and 4. Note that a trajectory found via projected gradient method is a solution to the VI problem but not necessarily the game. A playerwise local minimizer check needs to be done as discussed in Section 3.1.3. We summarize the projected gradient method in Algorithm 3.

Algorithm 3 Projected Gradient Method for Constrained Dynamic Games

```
Generate an initial trajectory of actions ū
loop
    Compute gradient 𝒥(ū) according to (3.11)
    Update ū ← ū − ρ𝒥(ū)
    Solve Problem 6 around ū, assign the solution to u
    Compare ū and u to check convergence
    Set ū ← u
end loop
Perform convexity check for ū
```

3.3 The Douglas-Rachford Operator Splitting Method

This section concerns with solving an OLNE with the Douglas-Rachford splitting method, which is an alternative to the projected gradient method. The DR splitting method, in its most general form, finds the vector w that solves a monotone inclusion problem of the form

$$0 \in \Theta w + \Phi w \tag{3.13}$$

where Θ and Φ are two *maximally monotone* operators in this section. The DR algorithm is defined by the iteration

$$w^{t+1} = [(1-\alpha)I + \alpha R_\Theta R_\Phi] w^t. \tag{3.14}$$

$R_\Theta := 2r_\Theta - I$ is called the *reflected resolvent* of Θ where r_Θ is the *resolvent* of Θ. The same notation applies for operator Φ. α is a constant such that $\alpha \in (0,1)$. The key steps are solving for the two operators' resolvents, which we elaborate in the case of constrained dynamic games in this section. The DR splitting method has been proven to converge linearly in cases when at least one operator has stronger properties such as strong monotonicity and Lipschitz continuity. See [59] for more details.

3.3.1 VI Reformulation of Static Game with States

For ease of notation we interpret $[x,u]$ as a column vector vertically stacking x and u in this section. We create an extended gradient $\mathcal{J}_{xu}([x,u])$ prepending n_x of zeros in front of $\mathcal{J}(u)$, i.e.,

$$\mathcal{J}_{xu}([x,u]) = \begin{bmatrix} \mathbf{0} \\ \mathcal{J}(u) \end{bmatrix} \in \mathbb{R}^{n_x+n_u} \tag{3.15}$$

and use it to reformulate Problem 5 equivalently as

Problem 7. ***VI formulation of static game with extended gradient***

$$\begin{aligned} \textit{Find} \quad & [x^\star, u^\star] \in \mathcal{D} \cap \mathcal{G} && \text{(3.16a)} \\ \textit{s.t.} \quad & \mathcal{J}_{xu}^\top([x,u])([x,u] - [x^\star,u^\star]) \geq 0 && \text{(3.16b)} \\ & \forall [x,u] \in \mathcal{D} \cap \mathcal{G} && \text{(3.16c)} \end{aligned}$$

Problem 7 is equivalent to an inclusion problem, when $\mathcal{D}$ and $\mathcal{G}$ are convex and $\mathcal{J}_{xu}$ is strongly monotone as explained in Appendix B.1.1. Note that x is also a decision variable in this formulation. This VI problem can also be formulated as an inclusion problem such that we can apply the DR-splitting method.

Problem 8. ***Inclusion problem form of the static game Problem 4***

$$0 \in (\eta \mathcal{J}_{xu} + \mathcal{N}_\mathcal{D} + \mathcal{N}_\mathcal{G})([x^\star, u^\star]) \tag{3.17}$$

where $\mathcal{N}_\mathcal{D}$ and $\mathcal{N}_\mathcal{G}$ indicate the normal cone operators of $\mathcal{N}$ and $\mathcal{G}$, and η is a positive regularization constant. The smaller the η, the more regularized the problems are, but the slower the convergence would be. η should be tuned so that the relevant problems in Section 3.3.2,

3.3.3 and 3.3.4 are either strongly convex of monotone for better solvability. Note that there are three adding operators in Problem 8, in contrast to (3.13) which has only two. We can single out any one operator and combine the other two to form an inclusion problem with two adding operators and apply the DR algorithm. The algorithm requires alternately solving two problems that are the resolvents of different operators. Which combination to chose ultimately depends on which resolvents are easier to solve, which varies with the specific dynamic game and available solvers. Ideally, when both resolvents are analytically solvable, the DR algorithm is preferred.

We summarize the implementation of the DR splitting in Algorithm 4 and elaborate the optimization/game problems derived from the resolvents later in this section. We use y, z to indicate a nominal trajectory that follows the same convention as x, u. The notation J is overloaded to indicate costs for these different resolvent problems. For details regarding how the problems are derived from resolvents, see Appendix B.1.2.

Algorithm 4 Douglas-Rachford for Constrained Dynamic Games

Generate an initial trajectory $\bar{x}, \bar{u}$, set $y = \bar{x}, z = \bar{u}$

Pick one scheme from Section 3.3.2, 3.3.3 or 3.3.4

loop

Set $y = \bar{x}, z = \bar{u}$

Solve the first problem in the selected section with y, z, assign the resulting trajectory to $\tilde{x}, \tilde{u}$

Reset the intermediate values $\tilde{x}, \tilde{u}$ and y, z

$$[y, z] \leftarrow 2[\tilde{x}, \tilde{u}] - [y, z] \tag{3.18a}$$

Solve the second problem in the selected section with y, z, assign the resulting trajectory to $\tilde{x}, \tilde{u}$

Reset the intermediate values $\tilde{x}, \tilde{u}$ and y, z

$$[y, z] \leftarrow 2[\tilde{x}, \tilde{u}] - [y, z] \tag{3.19}$$

Compute weighted average

$$[\bar{x}, \bar{u}] \leftarrow (1 - \alpha)[\bar{x}, \bar{u}] + \alpha[y, z] \tag{3.20}$$

Check for convergence with $\bar{x}, \bar{u}$ and $\tilde{x}, \tilde{u}$

end loop

3.3.2 Singling out $\mathcal{N}_{\mathcal{G}}$

Problem 8 becomes the following when singling out $\mathcal{N}_{\mathcal{G}}$

$$0 \in ([\eta \mathcal{J}_{xu} + \mathcal{N}_{\mathcal{D}}] + \mathcal{N}_{\mathcal{G}})([x^\star, u^\star]) \tag{3.21}$$

The resolvent of $\eta \mathcal{J}_{xu} + \mathcal{N}_{\mathcal{D}}$ corresponds to solving a regularized, unconstrained dynamic game and the resolvent of $\mathcal{N}_{\mathcal{G}}$ is the projection onto $\mathcal{G}$.

Problem 9. ***Regularized unconstrained dynamic game around nominal trajectory*** y, z

$$\min_{u_{n,:}} \quad J_n(x, u) = \sum_{k=0}^{T} c_{n,k}(x_k, u_{:,k}) + \frac{1}{2\eta} \|x_k - y_k\|^2$$

$$+\frac{1}{2\eta}\|u_{:,k}-z_{:,k}\|^2$$

$$n=1,2,\ldots,N \tag{3.22a}$$

$$s.t. \quad x_{k+1}=f_k(x_k,u_{:,k}) \tag{3.22b}$$

This problem can be solved via our previously proposed stagewise Newton method in Chapter 2 with linear complexity in T and quadratic convergence.

Problem 10. ***Projection of a nominal trajectory*** y,z ***onto the extra constraints set***

$$[x,u]=\Pi_{\mathcal{G}}([y,z]) \tag{3.23}$$

When the extra constraints sets are convex at each stage, this projection is equivalent to solving the projection onto a convex set at each stage.

3.3.3 Singling out $\mathcal{N}_{\mathcal{D}}$

Problem 8 becomes the following when singling out $\mathcal{N}_{\mathcal{D}}$

$$0\in([\eta\mathcal{J}_{xu}+\mathcal{N}_{\mathcal{G}}]+\mathcal{N}_{\mathcal{D}})([x^\star,u^\star]) \tag{3.24}$$

The resolvent of $\eta\mathcal{J}_{xu}+\mathcal{N}_{\mathcal{G}}$ corresponds to solving a series of regularized, constrained static games and the resolvent of $\mathcal{N}_{\mathcal{D}}$ is the projection onto $\mathcal{D}$, which is an optimal control problem.

Problem 11. ***Regularized constrained static games around nominal trajectory*** y,z

$$\min_{x_k,u_{:,k}} \quad J_n(x,u)=\sum_{k=0}^{T}c_{n,k}(x_k,u_{:,k})+\frac{1}{2\eta}\|x_k-y_k\|^2$$

$$+\frac{1}{2\eta}\|u_{:,k}-z_{:,k}\|^2 \tag{3.25a}$$

$$s.t. \quad g_k(x_k,u_{:,k})\leq 0, \tag{3.25b}$$

$$k=0,1,\ldots,T-1 \tag{3.25c}$$

Note that the objectives are not coupled cross time, therefore this is equivalent to solving $T+1$ constrained static games, which can be solved via solving the KKT conditions [73] or other static game methods. Note that there is an additional player with decision variable x_k at each stage.

Problem 12. ***Projection of a nominal trajectory*** y, z ***onto the dynamics***

$$\min_{u} \quad J(x,u) = \sum_{k=0}^{T} \|x_k - y_k\|^2 + \|u_{:,k} - z_{:,k}\|^2 \tag{3.26a}$$

$$s.t. \quad x_{k+1} = f_k(x_k, u_{:,k}) \tag{3.26b}$$

This is an optimal control problem with quadratic step cost, which can be solved via classic optimal control methods [84].

3.3.4 Singling out $\eta \mathcal{J}_{xu}$

Problem 8 becomes the following when singling out $\eta \mathcal{J}_{xu}$

$$0 \in ([\mathcal{N}_{\mathcal{D}} + \mathcal{N}_{\mathcal{G}}] + \eta \mathcal{J}_{xu})([x^\star, u^\star]) \tag{3.27}$$

The resolvent of $\mathcal{N}_{\mathcal{D}} + \mathcal{N}_{\mathcal{G}}$ corresponds to solving a constrained optimal control problem and the resolvent of $\mathcal{J}_{xu}$ is a series of unconstrained optimization.

Problem 13. ***Constrained optimal control around nominal trajectory*** y, z

$$\min_{u_{:,k}} \quad J(x,u) = \sum_{k=0}^{T} \|x_k - y_k\|^2 + \|u_{:,k} - z_{:,k}\|^2 \tag{3.28a}$$

$$s.t. \quad g_k(x_k, u_{:,k}) \leq 0, \tag{3.28b}$$

$$x_{k+1} = f_k(x_k, u_{:,k}) \tag{3.28c}$$

$$k = 0, 1, ..., T-1 \tag{3.28d}$$

This is an constrained optimal control problem with quadratic step cost, which can be solved via classic optimal control methods [84].

Problem 14. ***Regularized unconstrained games around a nominal trajectory*** y, z

$$\min_{x_k, u_{n,k}} \quad J_n(x,u) = \sum_{k=0}^{T} c_{n,k}(x_k, u_{:,k}) + \frac{1}{2\eta} \|x_k - y_k\|^2 + \frac{1}{2\eta} \|u_{:,k} - z_{:,k}\|^2 \tag{3.29a}$$

Because the objectives are not coupled cross time, this becomes $T+1$ unconstrained static games, which can be solved via classic VI methods. Similar to Problem 11, an additional player with decision variable x_k at each stage is added.

3.4 Parametric Games and Feedback Equilibria

Solving a parametric game is the backbone of solving an explicit feedback Nash equilibrium of a game. We briefly describe the results of analytically solvable parametric games and dynamic games in this section. Section 3.4.1 explains the result for linear equality constrained quadratic parametric games, which is directly applied to approximated Bellman recursion method in Section 3.5. Section 3.4.2 describes the form of feedback equilibrium of linearly constrained quadratic dynamic games, which is piecewise affine but can be exponentially complex. We assume each player's cost function $J_n(x,u)$ is continuous and strictly convex throughout this section.

The detailed development of the results can be found in the Appendix B.2, which is extending the analytical solution of linearly constrained, quadratic objective optimization and optimal control in Chapter 7 of [83].

3.4.1 Linear Equality Constrained Quadratic Parametric Game

Problem 15 is a basic form that is encountered when approximating a constrained dynamic game, where $u = [u_1^\top, u_2^\top, ..., u_N^\top,]^\top$ collects all players' action and x is a vector parameter. An FNE policy $u = \phi(x)$ is desired. This game has an explicit analytical solution with simple assumptions.

Problem 15. ***Linear equality constrained quadratic parametric game***

$$\min_{u_n} \quad J_n(x,u) = \frac{1}{2}\begin{bmatrix}1\\x\\u\end{bmatrix}^\top \begin{bmatrix}\Gamma_n^{11} & \Gamma_n^{1x} & \Gamma_n^{1u}\\ \Gamma_n^{x1} & \Gamma_n^{xx} & \Gamma_n^{xu}\\ \Gamma_n^{u1} & \Gamma_n^{ux} & \Gamma_n^{uu}\end{bmatrix}\begin{bmatrix}1\\x\\u\end{bmatrix} \tag{3.30a}$$

$$s.t. \quad Wx + Su + p = 0 \tag{3.30b}$$

The following lemma describes its solution.

Lemma 6. *Given playerwise convexity of objective functions, i.e.,* Γ_n^{uu} *are positive definite, Problem 15 has a unique affine feedback Nash equilibrium as in* (3.31a) *if* F *is invertible and* S *has full row rank.*

$$u^\star = Kx + s \tag{3.31a}$$

$$\lambda^\star = (SF^{-1}S^\top)^{-1}(W - SF^{-1}P)x + \tag{3.31b}$$

$$(SF^{-1}S^\top)^{-1}(p - SF^{-1}H) \tag{3.31c}$$

where

$$K = -F^{-1}S^\top(SF^{-1}S^\top)^{-1}(W - SF^{-1}P) - F^{-1}P \tag{3.32a}$$

$$s = -F^{-1}S^\top(SF^{-1}S^\top)^{-1}(p - SF^{-1}H) - F^{-1}H \tag{3.32b}$$

$$F = \begin{bmatrix} \Gamma_1^{u_1u_1} & \Gamma_1^{u_1u_2} & \dots & \Gamma_1^{u_1u_N} \\ \Gamma_2^{u_2u_1} & \Gamma_2^{u_2u_2} & \dots & \Gamma_2^{u_2u_N} \\ \vdots & \vdots & \ddots & \vdots \\ \Gamma_N^{u_Nu_1} & \Gamma_N^{u_Nu_2} & \dots & \Gamma_N^{u_Nu_N} \end{bmatrix} \tag{3.32c}$$

$$P = \begin{bmatrix} \Gamma_1^{u_1x} \\ \Gamma_2^{u_2x} \\ \vdots \\ \Gamma_N^{u_Nx} \end{bmatrix} \quad H = \begin{bmatrix} \Gamma_n^{u_11} \\ \Gamma_n^{u_21} \\ \vdots \\ \Gamma_n^{u_N1} \end{bmatrix} \tag{3.32d}$$

3.4.2 Linearly Constrained Quadratic Dynamic Games

A linearly constrained quadratic dynamic game is formulated as following. It is one of the most complicated forms of dynamic games of which we can acquire analytical FNE solution in theory.

Problem 16. ***Linearly Constrained Quadratic Dynamic Games***

$$\min_{u_n} \quad J_n(x,u) = \sum_{k=0}^{T} \frac{1}{2} \begin{bmatrix} 1 \\ x_k \\ u_{:,k} \end{bmatrix}^\top \begin{bmatrix} \Gamma_{n,k}^{11} & \Gamma_{n,k}^{1x} & \Gamma_{n,k}^{1u} \\ \Gamma_{n,k}^{x1} & \Gamma_{n,k}^{xx} & \Gamma_{n,k}^{xu} \\ \Gamma_{n,k}^{u1} & \Gamma_{n,k}^{ux} & \Gamma_{n,k}^{uu} \end{bmatrix} \begin{bmatrix} 1 \\ x_k \\ u_{:,k} \end{bmatrix} \tag{3.33a}$$

$$s.t. \quad x_{k+1} = A_k x_k + B_k u_{:,k} + b_k \tag{3.33b}$$

$$W_{n,k}x + S_{n,k}u + p_{n,k} \leq 0 \tag{3.33c}$$

This problem can be viewed as a static game in u parameterized by x. As developed in detail in Appendix B.2, the explicit FNE solution found via Bellman recursion to this problem is a piecewise affine policy, with polyhedral domains, and the value functions are piecewise quadratic. However, the required number of polyhedral domains on the space $\mathcal{X}_k$ can grow

exponentially, causing the procedure to be computationally prohibiting. It is reasonable to believe that the feedback Nash equilibrium of general constrained dynamic games can be more complex. This fact drives us to seek local FNE.

3.5 Local Feedback Equilibrium

OLNEs solved via projected gradient or DR splitting might not be applicable to systems with noise and general feedback Nash equilibrium can be hard to find. Thus, we aim to find a local feedback Nash equilibrium around local stationary trajectories that can accommodate disturbances of the system to a certain degree. As discussed above, even for linear-quadratic dynamic games with polyhedral constraints, explicit feedback strategies can require exponential computational complexity. In this section, we extend the approximated Bellman recursion method for unconstrained dynamic games from our previous chapter to constrained dynamic games. When the constraints are polyhedral, we prove that the feedback policy is indeed a local $O(\varepsilon^2)-$FNE.

3.5.1 Approximated Bellman Recursion (ABR) for Local Feedback Policy

We introduce the approximated Bellman recursion for computing a local feedback policy for Problem 3. The key difference from unconstrained dynamic game is that, at each step, an approximated linear equality constrained quadratic game Problem 15 is solved, instead of the unconstrained quadratic game.

For a given trajectory $\bar{x}, \bar{u}$ with known active constraints at each step. The set of indices of active constraints in $g_k(\bar{x}_k, \bar{u}_{:,k})$ is denoted $\bar{a}_k, \forall k$ and we use $g_k^{\bar{a}}(x_k, u_{:,k})$ to indicate the active constraints at $\bar{x}, \bar{u}$, therefore $g_k^{\bar{a}}(\bar{x}_k, \bar{u}_{:,k}) = 0$. We inherit the following notation of derivatives for from the previous chapter on unconstrained dynamic games in Section 2.1.5.

$$A_k = \frac{\partial f_k(x_k, u_{:,k})}{\partial x_k}\Big|_{\bar{x},\bar{u}} \qquad B_k = \frac{\partial f_k(x_k, u_{:,k})}{\partial u_{:,k}}\Big|_{\bar{x},\bar{u}} \tag{3.34a}$$

$$G_k^l = \begin{bmatrix} \frac{\partial^2 f_k^l}{\partial x_k^2} & \frac{\partial^2 f_k^l}{\partial x_k \partial u_{:,k}} \\ \frac{\partial^2 f_k^l}{\partial u_{:,k} \partial x_k} & \frac{\partial^2 f_k^l}{\partial u_{:,k}^2} \end{bmatrix}\Bigg|_{\bar{x},\bar{u}}, \quad l = 1, 2, \ldots, n_x \tag{3.34b}$$

$$R_k(\delta x_k, \delta u_{:,k}) = \begin{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top G_k^1 \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top G_k^{n_x} \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix} \end{bmatrix} \tag{3.34c}$$

$$M_{n,k} = \begin{bmatrix} 2c_{n,k} & \frac{\partial c_{n,k}}{\partial x_k} & \frac{\partial c_{n,k}}{\partial u_{:,k}} \\ \frac{\partial c_{n,k}}{\partial x_k}^\top & \frac{\partial^2 c_{n,k}}{\partial x_k^2} & \frac{\partial^2 c_{n,k}}{\partial x_k \partial u_{:,k}} \\ \frac{\partial c_{n,k}}{\partial u_{:,k}}^\top & \frac{\partial^2 c_{n,k}}{\partial u_{:,k} \partial x_k} & \frac{\partial^2 c_{n,k}}{\partial u_{:,k}^2} \end{bmatrix} \Bigg|_{\bar{x},\bar{u}} \tag{3.34d}$$

$$= \begin{bmatrix} M_{n,k}^{11} & M_{n,k}^{1x} & M_{n,k}^{1u} \\ M_{n,k}^{x1} & M_{n,k}^{xx} & M_{n,k}^{xu} \\ M_{n,k}^{u1} & M_{n,k}^{ux} & M_{n,k}^{uu} \end{bmatrix}.$$

We introduce new notation related to the derivatives of the active constraints $a_k, \forall k \in \{0, 1, 2, ..., T\}$.

$$W_k^{\bar{a}} = \frac{\partial g_k^{\bar{a}}(\bar{x}_k, \bar{u}_{:,k})}{\partial x} \tag{3.35a}$$

$$S_k^{\bar{a}} = \frac{\partial g_k^{\bar{a}}(\bar{x}_k, \bar{u}_{:,k})}{\partial u} \tag{3.35b}$$

$$p_k^{\bar{a}} = g_k^{\bar{a}}(\bar{x}_k, \bar{u}_{:,k}) \tag{3.35c}$$

We can form a local linear-quadratic approximation that is still analytically solvable via dynamic programming as following.

Problem 17. ***Linear-quadratic approximation to Problem 3***

$$\min_{u_{n,:}} \frac{1}{2} \sum_{k=0}^{T} \left(\begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top M_{n,k} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix} + M_{n,k}^{1x} \Delta x_k \right) \tag{3.36a}$$

subject to

$$\delta x_0 = 0 \tag{3.36b}$$

$$\Delta x_0 = 0 \tag{3.36c}$$

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_{:,k} \tag{3.36d}$$

$$\Delta x_{k+1} = A_k \Delta x_k + R_k(\delta x_k, \delta u_{:,k}) \tag{3.36e}$$

$$W_k^{\bar{a}} \delta x_k + S_k^{\bar{a}} \delta u_k + p_k^{\bar{a}} = 0 \tag{3.36f}$$

$$k = 0, 1, \ldots, T \tag{3.36g}$$

where the states of the dynamic game are given by δx_k and Δx_k as

$$\delta x_k = \sum_{i=0}^{T} \frac{\partial x_k}{\partial u_{:,i}}\Big|_{\bar{x},\bar{u}} \delta u_{:,i} \tag{3.37a}$$

$$\Delta x_k^l = \sum_{i=0}^{T} \sum_{j=0}^{T} \delta u_{:,i}^\top \frac{\partial^2 x_k^l}{\partial u_{:,i} \partial u_{:,j}}\Big|_{\bar{x},\bar{u}} \delta u_{:,j},\ l = 1, 2, \ldots, n_x \tag{3.37b}$$

The following lemma describes the solution to Problem 17.

Lemma 7. *The equilibrium value functions found via the Bellman recursion* (3.6) *for the dynamic game Problem* (17) *are denoted as $V_{n,k}^{\bar{u}}(\cdot)$ and $Q_{n,k}^{\bar{u}}(\cdot,\cdot)$, which can be expressed as*

$$V_{n,k}^{\bar{u}}(\delta x_k, \Delta x_k) = \frac{1}{2}\left(\begin{bmatrix} 1 \\ \delta x_k \end{bmatrix}^\top \Lambda_{n,k} \begin{bmatrix} 1 \\ \delta x_k \end{bmatrix} + \Omega_{n,k} \Delta x_k \right) \tag{3.38a}$$

$$Q_{n,k}^{\bar{u}}(\delta x_k, \Delta x_k, \delta u_{:,k}) = \frac{1}{2}\left(\begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top \Gamma_{n,k} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix} + \Omega_{n,k} \Delta x_k \right) \tag{3.38b}$$

where the matrices $\Lambda_{n,k}$, $\Gamma_{n,k}$, and $\Omega_{n,k}$ can be computed in a backward pass, which also finds a local feedback policy of the form $\delta u_{:,k} = K_k \delta x_k + s_k$ around this trajectory. When $s_k = 0, \forall k$, we call the corresponding trajectory $\bar{x}, \bar{u}$ a stationary trajectory of the ABR method.

The matrices $\Lambda_{n,k}$, $\Gamma_{n,k}$, and $\Omega_{n,k}$ in (3.38) *are computed recursively by $\Lambda_{n,T+1} = 0$, $\Omega_{n,T+1} = 0$, and*

$$\Omega_{n,k} = M_{n,k}^{1x} + \Omega_{n,k+1} A_k \tag{3.39a}$$

$$D_{n,k} = \sum_{l=1}^{n_x} \Omega_{n,k+1}^l G_k^l \tag{3.39b}$$

$$\begin{aligned} \Gamma_{n,k} &= M_{n,k} \\ &+ \begin{bmatrix} \Lambda_{n,k+1}^{11} & \Lambda_{n,k+1}^{1x} A_k & \Lambda_{n,k+1}^{1x} B_k \\ A_k^\top \Lambda_{n,k+1}^{x1} & A_k^\top \Lambda_{n,k+1}^{xx} A_k + D_k^{xx} & A_k^\top \Lambda_{n,k+1}^{xx} B_k + D_k^{xu} \\ B_k^\top \Lambda_{n,k+1}^{x1} & B_k^\top \Lambda_{n,k+1}^{xx} A_k + D_k^{ux} & B_k^\top \Lambda_{n,k+1}^{xx} B_k + D_k^{uu} \end{bmatrix} \end{aligned} \tag{3.39c}$$

$$= \begin{bmatrix} \Gamma^{11}_{n,k} & \Gamma^{1x}_{n,k} & \Gamma^{1u_1}_{n,k} & \Gamma^{1u_2}_{n,k} & \cdots & \Gamma^{1u_N}_{n,k} \\ \Gamma^{x1}_{n,k} & \Gamma^{xx}_{n,k} & \Gamma^{xu_1}_{n,k} & \Gamma^{xu_2}_{n,k} & \cdots & \Gamma^{xu_N}_{n,k} \\ \Gamma^{u_1 1}_{n,k} & \Gamma^{u_1 x}_{n,k} & \Gamma^{u_1 u_1}_{n,k} & \Gamma^{u_1 u_2}_{n,k} & \cdots & \Gamma^{u_1 u_N}_{n,k} \\ \Gamma^{u_2 1}_{n,k} & \Gamma^{u_2 x}_{n,k} & \Gamma^{u_2 u_1}_{n,k} & \Gamma^{u_2 u_2}_{n,k} & \cdots & \Gamma^{u_2 u_N}_{n,k} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \Gamma^{u_N 1}_{n,k} & \Gamma^{u_N x}_{n,k} & \Gamma^{u_N u_1}_{n,k} & \Gamma^{u_N u_2}_{n,k} & \cdots & \Gamma^{u_N u_N}_{n,k} \end{bmatrix} \tag{3.39d}$$

$$F_k = \begin{bmatrix} \Gamma^{u_1 u}_{1k} \\ \Gamma^{u_2 u}_{2k} \\ \vdots \\ \Gamma^{u_N u}_{Nk} \end{bmatrix} = \begin{bmatrix} \Gamma^{u_1 u_1}_{1k} & \Gamma^{u_1 u_2}_{1k} & \cdots & \Gamma^{u_1 u_N}_{1k} \\ \Gamma^{u_2 u_1}_{2k} & \Gamma^{u_2 u_2}_{2k} & \cdots & \Gamma^{u_2 u_N}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \Gamma^{u_N u_1}_{Nk} & \Gamma^{u_N u_2}_{Nk} & \cdots & \Gamma^{u_N u_N}_{Nk} \end{bmatrix} \tag{3.39e}$$

$$P_k = \begin{bmatrix} \Gamma^{u_1 x}_{1k} \\ \Gamma^{u_2 x}_{2k} \\ \vdots \\ \Gamma^{u_N x}_{Nk} \end{bmatrix}, \quad H_k = \begin{bmatrix} \Gamma^{u_1 1}_{1k} \\ \Gamma^{u_2 1}_{2k} \\ \vdots \\ \Gamma^{u_N 1}_{Nk} \end{bmatrix} \tag{3.39f}$$

$$K_k = -F_k^{-1} S_k^\top (S_k F_k^{-1} S_k^\top)^{-1} (W_k - S_k F_k^{-1} P_k) + F_k^{-1} P_k \tag{3.39g}$$

$$s_k = -F_k^{-1} S_k^\top (S_k F_k^{-1} S_k^\top)^{-1} (p_k - S_k F_k^{-1} H_k) + F_k^{-1} H_k \tag{3.39h}$$

$$\Lambda_{n,k} = \begin{bmatrix} 1 & 0 & s_k^\top \\ 0 & I & K_k^\top \end{bmatrix} \Gamma_{n,k} \begin{bmatrix} 1 & 0 \\ 0 & I \\ s_k & K_k \end{bmatrix} \tag{3.39i}$$

for $k = T, T-1, \ldots, 0$.

Proof. The approximated Bellman recursion method for unconstrained dynamic game and its solution was proved in previous chapter. The only difference in the dynamic programming procedure for Problem 17 is that an equality constrained static quadratic game is solved at each time step, resulting in different expressions for the feedback parameters K_k and s_k, which are justified by the solution of Problem 15. □

Note that the differential dynamic programming (DDP) method for unconstrained dynamic games can be adapted to constrained games in a similar fashion, for the sake of completeness of generalizing our previous work. As in the unconstrained dynamic game case, DDP or ABR do not manifest obvious advantages over each other.

3.5.2 Remarks on the Feedback Policy by the ABR Method

The ABR method solves a linear equality constrained quadratic dynamic game that approximates Problem 3 around $\bar{x}, \bar{u}$ and finds a local feedback policy. Without constraints $g_k(x_k, u_{:,k})$, i.e., setting $S_k = 0, W_k = 0, p_k = 0$, the method reduces to the unconstrained version of ABR. However, unlike its counterpart for unconstrained game, because of the introduction of constraints, the policy found might not be feasible, therefore we cannot perform the iterative process. We consider in this thesis the case when $\bar{x}, \bar{u}$ is a stationary trajectory. To find the stationary trajectory, we can apply the idea of interior-point method, wrap the constraints with log-barrier functions and add to the cost of each player, creating unconstrained dynamic games and perform the ABR iteration. In this case, the feedback policy simplifies to $\delta u_{:,k} = K_k \delta x_k$. Furthermore, the matrices are computed in $O(T)$ complexity. The possibility of infeasible policy remains and is overcome with tightened constraints as in Section 3.5.3.

The proposed algorithm neglects the inactive constraints. A problem arises, in that deviations in the state can cause these neglected constraints to become violated. A simple method to ensure feasibility is to tighten the inequality constraints, as is common in model predictive control [83]. We will see below that in the case of polyhedral constraints, the feedback policy is indeed an local $O(\varepsilon^2)$-FNE.

3.5.3 Problems with Polyhedral Constraints

In this section, we introduce a special class of dynamic problems, restricting to affine dynamics and polyhedral constraints, such that the approximated Bellman method can be utilized to find a local $O(\varepsilon^2)$-FNE for a partially tightened problem (defined below). We first introduce a fully tightened version of a polyhedrally constrained linear dynamic game

Problem 18. ***Game with tightened polyhedral constraints***

$$\min_{u_{n,:}} \quad J_{n,t}(x,u) = \sum_{k=t}^{T} c_{n,k}(x_k, u_{:,k}) \tag{3.40a}$$

$$s.t. \; x_{k+1} = A_k x_k + B_k u_{:,k} + b_k \tag{3.40b}$$

$$W_k x_k + S_k u_{:,k} + p_k \leq -\gamma_k, \tag{3.40c}$$

$$x_0 \textit{ is fixed.} \tag{3.40d}$$

Here $\gamma_k > 0$ are vectors used to tighten the inequality constraints. As above, assume that $\bar{x}, \bar{u}$ is a stationary trajectory of ABR. We use superscripts $^{\bar{a}}$ and $^{\bar{i}}$ to denote values associated with active and inactive constraints on $\bar{x}$ and $\bar{u}$. If we find a local feedback policy using the ABR method, it can be shown that it is feasible for the following partially tightened problem:

Problem 19. ***Game with partially tightened polyhedral constraints***

$$\min_{u_{n,:}} \quad J_{n,t}(x,u) = \sum_{k=t}^{T} c_{n,k}(x_k, u_{:,k}) \tag{3.41a}$$

$$s.t. \; x_{k+1} = A_k x_k + B_k u_{:,k} + b_k \tag{3.41b}$$

$$W_k^{\bar{a}} x_k + S_k^{\bar{a}} u_{:,k} + p_k^{\bar{a}} \leq -\gamma_k^{\bar{a}} \tag{3.41c}$$

$$W_k^{\bar{i}} x_k + S_k^{\bar{i}} u_{:,k} + p_k^{\bar{i}} \leq 0 \tag{3.41d}$$

$$x_t \text{ is fixed.} \tag{3.41e}$$

Recall that the dynamics, constraints, and control law are all affine. Thus, when a local variation of the state $\|x_k - \bar{x}\| = \varepsilon$ happens with a sufficiently small ε, the trajectories remain feasible. The next theorem summarizes this local feedback Nash equilibrium result. The detailed proof can be found in Appendix B.1.4.

Theorem 3. *There exists a sufficiently small ε, such that if $\|x_t - \bar{x}_t\| \leq \varepsilon$, the stationary trajectory $\bar{u}, \bar{x}$, active/inactive constraints and local feedback policy $\phi_{:,k}(x_k) = K_k x_k + s_k$ found for Problem 18, is a local feedback $O(\varepsilon^2)$-Nash equilibrium for Problem 19:*

$$J_{n,t}(x_t, \phi_{:,t:}) \leq J_{n,t}(x_t, [\psi_{n,t:}, \phi_{-n,t:}]) + O(\varepsilon^2), \; \forall t \in \{0, 1, ..., T\} \tag{3.42}$$

for any $\psi_{n,:}$ such that the resulting trajectories are feasible for (3.41) *and remain in a neighborhood of* $[\bar{x}, \bar{u}]$.

3.6 Numerical Examples

We demonstrate the approximated local feedback policy around an OLNE with a common-property fishery resource problem and Douglas-Rachford algorithm with a linear quadratic games with analytically projectable convex constraints.

3.6.1 A Common-Property Fishery Resource Problem

The common-property fishery game was considered in Chapter 13, [2], which is a classic renewable resource manage problem that dates back to 1970s. The analysis in [2] settled at the conclusion that efficient players will drive some opponents out of the competition and maintain at the bionomic equilibrium with zero sustained economic rent. We demonstrate the dynamic equilibrium of two players jointly utilize the resource for a given period of time.

The game is discretized. Scalars x_k and $u_{1,k}$, $u_{2,k}$ denote the biomass of fish and fishing effort of two players. The fishing effort is constrained by $0 \leq u_{n,k} \leq u_n^{\max}$. The dynamics of the system from time 0 to T is given by

$$x_{k+1} = x_k + \left[g(x_k) - \sum_{n=1}^{2} q_n u_{n,k} x_k \right] \mathrm{d}t,\ k = 0, 1, ..., T/\mathrm{d}t \tag{3.43}$$

where the natural growth rate $g(x_k)$ is

$$g(x_k) = \frac{r}{h^2}(2hx_k - x_k^2) + w_k \tag{3.44}$$

h is half the maximal biomass the environment can sustain and r is the maximal growth rate which happens when $x_k = h$. q_n is the catchability coefficient for player n. w_k is a I.I.D. Gaussian noise we injected for simulating a noisy system. Step profit of each player is modeled as

$$c_{n,k}(x_k, u_{:,k}) = (p_n q_n x_k - e_n)\, u_{n,k} \mathrm{d}t \tag{3.45}$$

where p_n is the unit price of landed fish and e_n is the unit cost of effort.

Constants chosen are chosen in favor of player 1.

$$\begin{aligned} &u_1^{\max} = 0.4,\ u_2^{\max} = 0.3,\ r = 8,\ h = 100, \mathrm{d}t = 0.1,\ T = 100, \\ &q_1 = q_2 = 0.1,\ p_1 = p_2 = 1,\ e_1 = 9,\ e_2 = 11 \end{aligned} \tag{3.46}$$

The *turnpike* for a player is the most profitable level of fish biomass if the player is managing the resource alone. And the *bionomic equilibrium* for a player is the minimal fish biomass that a player can turn a profit.

We first applied the projected gradient method with stepsize 0.01 for 1,000 iterations on the deterministic system (neglecting w_k), finding the OLNE shown in Fig. 3.1. The cumulative profit and convergence of actions is summarized in Fig 3.2. We further applied the ABR method, found the local feedback policy and implemented it for the noisy system, setting the variance of

zero-mean Gaussian noise w_k to $\mathbb{E}\{w_k^2\} = 2$. The comparison between OLNE and ABR policy around it for noisy system is shown in Fig. 3.3.

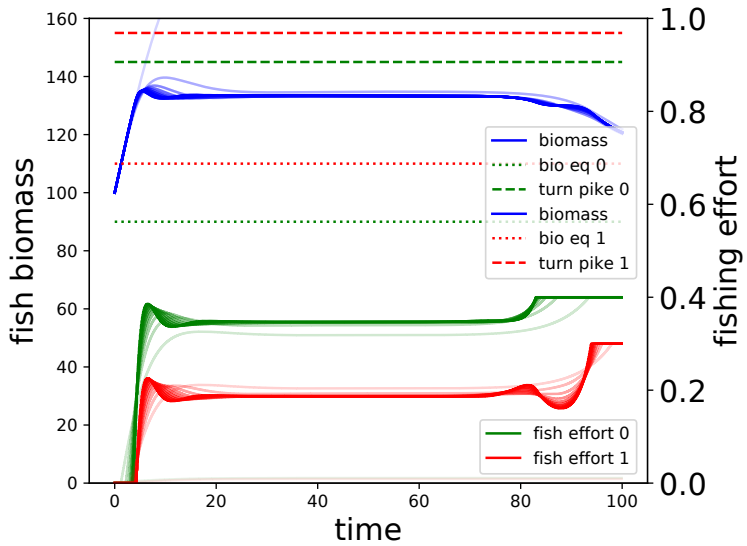
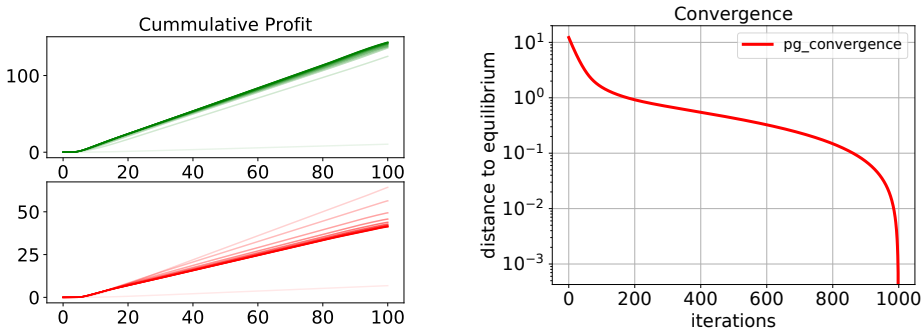


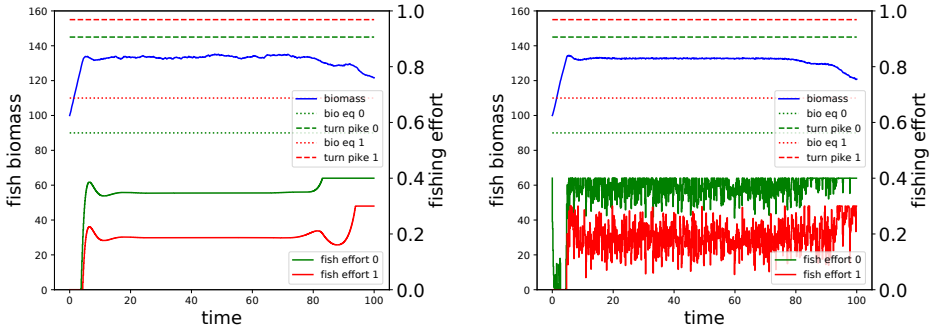
Figure 3.1: OLNE of projected gradient iterations. 10 trajectories were sampled from 1,000 iterations and shown in Fig. 3.1 with more transparent curves indicating earlier trajectories in the iteration. As can be seen, both players would wait at the beginning for the level of fish biomass to rise even after it passes their bionomic equilibria, since they are managing the resource on a longer term. Two players' effort stabilizes in the middle section, which we believe to be the infinite horizon equilibrium for the game, which is not within the scope of this thesis. In the end, player 1 does not care about longer term profit, so they maximize the effort. Player 2 would like to keep the biomass further away their bionomic equilibrium before the final dash, so they reduced effort from time 80 to around 93.



(a) Cumulative Profit

(b) Convergence

Figure 3.2: Cumulative profit and convergence. The game is formulated favoring player 1, it is not surprising that over iterations, player 1's profit increases while player 2's decreases as in Fig. 3.2(a). Fig. 3.2(b) shows the distance to the final OLNE as the iteration progresses, which fits a typical linear convergence pattern.



(a) OLNE for noisy system

(b) Local FNE for noisy system

Figure 3.3: Local OLNE and ABR feedback policy for noisy system. Fig. 3.3(a) shows if the OLNE is blindly applied, the biomass is susceptible to the noise and deviates from the OLNE biomass trajectory. Fig. 3.3(b) shows the correctional effect of the local feedback policy found via ABR keeping the biomass smoother and closer to the deterministic OLNE with erratic fishing efforts.

3.6.2 Linear Quadratic Game with Convex Constraints

When a dynamic linear-quadratic(LQ) game has an convex constraint set onto which, the projection is analytically solvable, both problems in Section 3.3.2 can be analytically solved and the Douglas-Rachfor splitting method is preferred.

We demonstrate the DR splitting method on a 2-D locomotion problem with $N = 3$ players. Each player directly controls its own location. The system state $x_k \in \mathbb{R}^6$ contains N sets of 2-D coordinates of each player and action $u_{n,k} \in \mathbb{R}^2$ for each players. We use $x_{n,k}$ to denote player n's coordinate at step k, naturally we have $x_k = [x_{1,k}, x_{2,k}, x_{3,k}]$, where the variables are all column vectors. The dynamics is simply

$$x_{k+1} = I_6 x_k + I_6 u_{:,k}, k = 0, 1, ..., T \tag{3.47a}$$

where I_6 is a 6×6 identity matrix. The initial position x_0 and each players target position x_n^t are known. Each player's action is subject to a magnitude constraint $\|u_{n,k}\| \leq u_n^{\max}$. The cost of each player consists of two parts, reaching to the target and conserving its own energy.

$$c_{n,k} = \left\|x_{n,k} - x_n^t\right\|^2 + 10\left\|u_{n,k}\right\|^2,\ k = 0, 1, 2, ..., T-1 \tag{3.48}$$

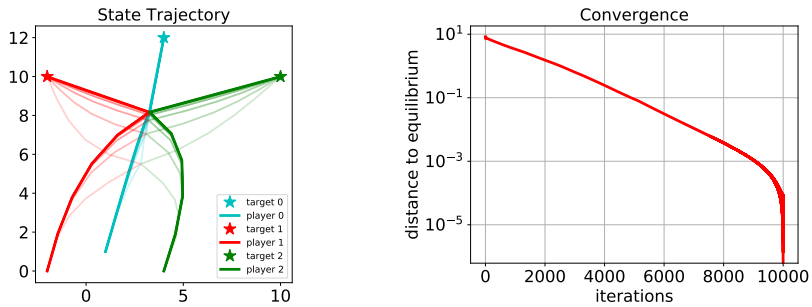
$$c_{n,T} = 1000\left\|x_{n,T} - x_n^t\right\|^2,\ k = T \tag{3.49}$$

It is also required that all three players should meet at $k = 5$, i.e., $x_{1,5} = x_{2,5} = x_{3,5}$, which makes the problem a coupled dynamic game problem rather than 3 separated optimal control problems. The particular constraint also causes the projected gradient method or Algorithm 3, if applied to the problem at hand, to require solving a constrained optimal control problem in each iteration, which needs another iterative procedure. The DR splitting on the other hands, presents two analytically solvable subproblems. In particular, the corresponding Problem 9 is solved via stagewise Newton for unconstrained game [82] and Problem 10 becomes a simple projection onto $\|u_{n,k}\| \leq u_n^{\max}$.

The parameters are chosen as

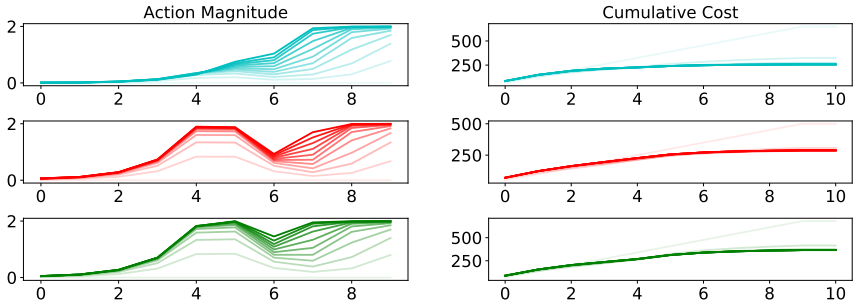
$$\begin{aligned} &T = 10, x_0 = [1, 1, -2, 0, 4, 0], \\ &x^t = [x_1^t, x_2^t, x_3^t] = [4, 12, -2, 10, 10, 10], \\ &u_1^{\max} = u_2^{\max} = u_3^{\max} = 2, \end{aligned} \tag{3.50}$$

We applied the DR splitting method with the splitting scheme in Section 3.3.2. We found $\eta = 10^{-4}$ and $\alpha = 0.5$ produced stable iteration and solution. 10^4 iterations were performed and shown in Fig. 3.4.



(a) Open-loop equilibrium trajectory (b) Convergence

Figure 3.4: Douglas-Rachford splitting for dynamic LQ game with convex constraints. 11 trajectories were sampled from 10,000 iterations and shown in Fig. 3.4(a) with more transparent curves indicating earlier trajectories in the iteration. The rendezvous point changed over iteration and all players go straight to target afterwards. Fig. 3.4(b) shows the distance to the final OLNE as the iteration progresses, which is proof that the algorithm converges.



(a) Norm of actions (b) Convergence

Figure 3.5: Magnitudes of action and cumulative cost over iterations. 11 intermediate results were sampled from 10,000 iterations and shown in Fig. 3.5 with more transparent curves indicating earlier trajectories in the iteration. The magnitude is bounded by $u_n^{\max}$ as expected in Fig. 3.5(a). Fig. 3.5(b) shows the total costs reduce for all players as the rendezvous point changes over iterations.

Chapter 4

Linear System Identification with Multiplicative noise

In this chapter, we switch gears to linear system identification problems with multiplicative noise. We will show that LSMN is a more general formulation than systems with additive noise that can capture a wider range of uncertainty of linear systems. Identifying parameters and uncertainty of linear system models can be helpful for the robust control and modeling in many industrial and economic applications. Compared to the more commonly studied linear systems with additive noise, where the effect of the uncertainty is constant in the output, the noise of output in an LSMN depends on the input as well. When the inputs have larger magnitude, the noise is more magnified in the output. Because of the different structure of the problem formulation, LSMN requires a different treatment. We start from static problems, acquire bounds on the least-squares estimates and move on to dynamic systems and robust control, applying the results from static systems.

This chapter is organized as following. Section 4.1 explains the basic linear algebraic notation and random process related definitions in this chapter. Section 4.2 introduces the basic estimation problem with multiplicative noise, its least-squares solution and the error bound. Section 4.3 solves the problem of synthesizing probabilistic bounds of first and second order moments to obtain a probabilistic box bound on a random variable. Section 4.4 moves on to identification of linear dynamic systems. Section 4.5 proposes an online robust control scheme based on our estimation method and numerical examples are shown in Section 4.6.

4.1 Notation and Definitions

We use superscripts to denote scalar elements of a matrix or vector

$$x = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^n \end{bmatrix} \in \mathbb{R}^n \quad S = \begin{bmatrix} S^{11} & S^{12} & \cdots & S^{1n} \\ S^{21} & S^{22} & \cdots & S^{2n} \\ \vdots & \vdots & \ddots & \vdots \\ S^{m1} & S^{m2} & \cdots & S^{mn} \end{bmatrix} \in \mathbb{R}^{m \times n} \tag{4.1a}$$

For a matrix, we use in the superscript, a colon to denote picking all elements in the row or column and a coma to separate indices for row and column. For example, $S^{i,:}$ is the ith row of S and $S^{:,j}$ is the jth column.

For a random variable X, we use $\mathbb{E}[X]$ or X_* to indicate its expected value and $\overline{X}$ its least-squares estimation. The bound of a least-squares estimation X is denoted $\delta(\overline{X})$.

We introduce the notation to indicate a box bound here. Given matrices M, μ_M and R_M with the same dimension and R_M being non-negative, we use the absolute value and inequalities to indicate corresponding element-wise operations, so that the box bound of M around μ_M can be denoted

$$|M - \mu_M| \leq R_M, \quad R_M \geq 0 \tag{4.2}$$

For a vector $v = [v^1, v^2, ...]^\top$, we define the diagonalization operator diag(), which creates a diagonal matrix

$$\text{diag}(v) = \begin{bmatrix} v^1 & & \\ & v^2 & \\ & & \ddots \end{bmatrix} \tag{4.3}$$

Definition 5. ***Sub-Gaussian random variables***
A vector of zero-mean random variable $X \in \mathbb{R}^d$ is $\sigma-$sub-Gaussian if and only if

$$\mathbb{E}[e^{\beta^\top X}] \leq e^{\frac{1}{2}\sigma^2 \|\beta\|^2}, \forall \beta \in \mathbb{R}^d \tag{4.4}$$

Note that a sub-Gaussian distribution is not a specific distribution but a class of distributions. Zero-mean Gaussian distributions are sub-Gaussian, and any zero-mean distribution on a bounded set is also sub-Gaussian. For example, a zero-mean, scalar random variable X such that $-a \leq X \leq a, a > 0$, according to Lemma A.1 in [85], $\mathbb{E}[e^{sX}] \leq e^{\frac{1}{2}s^2 a^2}$, therefore it is sub-Gaussian with parameter $\sigma^2 = a^2$.

Definition 6. ***Supermartingale***
An integrable random process M_t is a supermartingale w.r.t. a filtration $\mathcal{F}_t$ if and only if

$$\mathbb{E}[M_{t+1}|\mathcal{F}_t] \leq M_t \quad t = 0, 1, 2, ... \tag{4.5}$$

If $M_t \geq 0$ almost surely for all t, then M_t is a positive supermartingale.

It is obvious that a convex combination of positive supermartingales is a positive supermartingale.

The *maximal inequality* of a positive supermartingale such that M_t, is applied in this chapter

$$\mathbb{P}(\sup_t M_t \geq \frac{\mathbb{E}[M_1]}{\varepsilon}) \leq \varepsilon \tag{4.6}$$

The details of the definition, the maximal inequality, definition of a filtration and more about Martingales can be found in [60, 86].

4.2 Estimation with Multiplicative Noise

First we study the confidence bound of least-squares estimation where the underlying true mapping contains multiplicative noise with a scalar-vector output/input structure. This bound is a key building block for bounding the identification of a linear dynamic system.

Problem 20. ***Estimation with multiplicative noise***
Suppose we have T pairs of output/input measurements (z_t, g_t), where $z_t \in \mathbb{R}, g_t \in \mathbb{R}^d$ and random parameter $\theta_t \in \mathbb{R}^d$ that satisfy

$$z_t = g_t^\top \theta_t \tag{4.7}$$

where $\theta_* = \mathbb{E}[\theta_t], \forall t$, assuming constant expected value of θ_t. $\eta_t = \theta_t - \theta_*, t = 0, 1, 2, ..., T$ is a vector of conditional σ-sub-Gaussian random process such that

$$\mathcal{F}_{t-1} = \Sigma(g_0, g_1, ..., g_{t-1}, g_t, \theta_0, \theta_1, ..., \theta_{t-1}) \tag{4.8a}$$

$$\mathbb{E}[e^{\beta^\top \eta_t}|\mathcal{F}_{t-1}] \leq e^{\frac{1}{2}\sigma^2\|\beta\|^2}, \forall \beta \in \mathbb{R}^d \tag{4.8b}$$

where $\Sigma(\cdot)$ is the sigma-algebra generated by the random variables. W.L.O.G., we assume $\|g_t\| = 1$. When $\|g_t\| \neq 1$, we can re-scale both z_t and g_t as $\frac{z_t}{\|g_t\|}$ and $\frac{g_t}{\|g_t\|}$. The problem is to estimate θ_t and provide a bound on the estimation.

Later in the chapter, we also refer to this problem as scalar-vector linear estimation problem when emphasizing the output/input relation.

The problem formulation is fairly general and is associated with an inner product equation. g_t can depend on history $(\theta_0, \theta_1, ..., \theta_{t-1}, z_0, z_1, ..., z_{t-1})$, which is the case for dynamic system identification problem we study in Section 4.4. θ_t can also be specialized to be i.i.d, which is a stronger assumption than (4.8), but (4.8) is necessary. Sub-Gaussian distributions include Gaussian and all bounded distributions, therefore is a reasonable assumption since practical systems cannot have infinite uncertainty. The least-squares solution and bound to Problem 20 can be summarized in the following theorem. The proof is in Appendix C.1. A key step in the proof is to formulate a supermartingale and apply the bound on supermartingale (4.6).

Theorem 4. ***Least-squares solution to Problem 20***
The least-squares estimation $\overline{\theta_T}$ to Problem 20 with normalized g_t is

$$\overline{\theta_T} := \arg\min_{\alpha} \sum_{t=1}^{T} \left\| z_t - g_t^\top \alpha \right\|^2 + \lambda \|\alpha\|^2 = V_T^{-1} b_T \tag{4.9}$$

where

$$V_T = \lambda I + \sum_{t=0}^{T} g_t g_t^\top \quad b_T = \sum_{t=0}^{T} z_t g_t \tag{4.10}$$

*With probability at least $1-\varepsilon$, the **self-normalizing bound** on $\overline{\theta_T} - \theta_*$ holds*

$$\delta_V(\overline{\theta_T}) := \sqrt{\sigma^2 \log \frac{(\lambda+T)^d}{\varepsilon^2 \lambda^d}} + \lambda \left\| V_T^{-\frac{1}{2}} \theta_* \right\| \geq \left\| V_T^{\frac{1}{2}} (\overline{\theta_T} - \theta) \right\| \tag{4.11}$$

*and the **Euclidean norm bound** holds*

$$\delta(\overline{\theta_T}) := \sqrt{\frac{\sigma^2}{\lambda_{\min}(V_T)} \log \frac{(\lambda+T)^d}{\varepsilon^2 \lambda^d}} + \frac{\lambda}{\lambda_{\min}(V_T)} \|\theta_*\| \geq \left\| \overline{\theta_T} - \theta_* \right\| \tag{4.12}$$

where $\lambda_{\min}(V_T)$ is the minimal eigenvalue of V_T.

Note that (4.11) induces an ellipsoid bound on the error $\overline{\theta_T} - \theta_*$ and (4.12) a box bound. The ellipsoid bound is tighter in general, but for the purpose of dynamic system identification in Section 4.4, the box bound is applied with reasons explained. The sub-Gaussian parameter σ is in general unknown, however, an initial bound of uncertain parameters can be estimated based on expert knowledge of practical systems. The bound diminishes to 0 as $\lambda_{\min}(V_T)$ grows as data is accummulated.

Remark 2. *Based on theorem 5.1.1 in [87], the matrix Chernoff inequalities, $\lambda_{\min}(V_T)$ has a high probability to be bounded below linear w.r.t. T, given i.i.d. $g_t g_t^\top$ with positive definite covariance, in which case, it can be shown through simple linear algebra that the Euclidean norm bound* (4.12) *converges to zero as $T \to \infty$.*

4.3 Synthesizing Probability Bounds of Estimations

Theorem 4 finds a Euclidean norm bound, but it is still a step away from a box bound. We provide the lemma to bridge the gap in this section. We study how to synthesize the estimate and bound of first and second moments of a random variable X to get a probabilistic box bound on a sub-Gaussian random variable. The following lemma gives the synthesized bound and its proof is in Appendix C.2. The proof utilizes Markov inequality and the sub-Gaussian property. Further result assuming X is uniformly distributed is also given.

Lemma 8. ***Bound for sub-Gaussian random variables***
For a scalar random variable X such that $X - \mathbb{E}[X]$ is σ-sub-Gaussian, suppose we have estimations and bounds for its first and second moments

$$\Pr\left(\left|\overline{X} - \mathbb{E}[X]\right| \leq \delta(\overline{X})\right) \geq 1 - \varepsilon_1 \tag{4.13a}$$

$$\Pr\left(\left|\overline{X^2} - \mathbb{E}[X^2]\right| \leq \delta(\overline{X^2})\right) \geq 1 - \varepsilon_2 \tag{4.13b}$$

A box bound on X can be computed

$$\Pr(\left|X - \overline{X}\right| \leq R + \delta(\overline{X})) \geq 1 - \varepsilon_1 - 2e^{-\frac{R^2}{2\sigma^2}} \tag{4.14}$$

Furthermore, if we assume X follows a uniform distribution, the box bound can be specialized as

$$\Pr(\left|X - \overline{X}\right| \leq R + \delta(\overline{X})) \geq 1 - \sum_{i=1}^{2} \varepsilon_i \tag{4.15a}$$

$$R \leq \sqrt{3}\sqrt{\overline{X^2} + \delta(\overline{X^2}) - \min\{[\overline{X} \pm \delta(\overline{X})]^2\}} \tag{4.15b}$$

From Lemma 8, the box size R can be computed from desired probability in general. It is interesting to see that, if we have accurate estimation of the first and second moments, i.e. $\delta\overline{X} = 0, \delta\overline{X^2} = 0, \varepsilon_1 = 0, \varepsilon_2 = 0$, the bound becomes

$$\Pr(\left|X - \overline{X}\right| \leq R) \geq 1 - 2e^{-\frac{R^2}{2\sigma^2}} \tag{4.16}$$

which means we need a box size comparable to the sub-Gaussian parameter to get a high probability. For example, when $R = 3\sigma$, the probability is as high as 97.78% and when $R = 2\sigma$, the probability is as high as 72.93%. This is because higher than second moments play an important role on the shape of the density function and can stretch the distribution far away from its mean for given first and second moments of a random variable.

4.4 Identification and Bound of Linear Systems

In this section, we apply the results in Section 4.2 and 4.3 to linear dynamic systems with multiplicative noise.

4.4.1 Problem Formulation and Algorithm

Problem 21. ***Linear dynamic system estimation with uncertain system matrices***
Suppose we have the following dynamic system

$$x_{t+1} = A_t x_t + B_t u_t \tag{4.17}$$

where $x_t \in \mathbb{R}^n, u_t \in \mathbb{R}^p$, A_t and B_t are bounded joint random variables, therefore sub-Gaussian, with constant first and second moments. The problem is, given T pairs of consecutive state and input u_t, x_t data, estimated and bound first and second moments of system parameters, then provide a probabilistic box bound on A_t and B_t.

The formulation allows variations of third and higher moments and coupling of A and B. The regular assumption that A_t and B_t are i.i.d. is sufficient for fixed first and second moments. When a linear system has additive noise, it can be merged into Bu_t term as

$$x_{t+1} = A_t x_t + B_t u_t + w_t = A_t x_t + \begin{bmatrix} B_t & \operatorname{diag}(w_t) \end{bmatrix} \begin{bmatrix} u_t \\ \mathbf{1} \end{bmatrix} \tag{4.18}$$

The formulation of Problem 21 naturally covers system with additive noise.

To solve this problem, we need to estimate and bound first and second moments applying Theorem 4 to properly formulated instances of Problem 20. Then apply Lemma 8 to get a probability bound. Before we derive the solution in general, we focus on a simpler case for illustrative purpose.

4.4.2 An Example of an Autonomous System

This simple example is developed to illustrate how the dynamic estimation Problem 21 can be solved via a number of scalar-vector estimation problems as Problems 20. We show a simple case when $n = 2$ and $p = 0$, i.e., an autonomous linear system. Denote

$$A_t = \begin{bmatrix} A_t^{11} & A_t^{12} \\ A_t^{21} & A_t^{22} \end{bmatrix} \tag{4.19}$$

Based on the dynamics (4.17), we have the following dynamics for the first and second moments of A_t

$$\underbrace{\begin{bmatrix} x_{t+1}^1 \\ x_{t+1}^2 \\ (x_{t+1}^1)^2 \\ (x_{t+1}^2)^2 \\ x_{t+1}^1 x_{t+1}^2 \end{bmatrix}}_{\text{output vector}} = \underbrace{\begin{bmatrix} x_t^1 & 0 & 0 & 0 & 0 \\ x_t^2 & 0 & 0 & 0 & 0 \\ 0 & x_t^1 & 0 & 0 & 0 \\ 0 & x_t^2 & 0 & 0 & 0 \\ 0 & 0 & (x_t^1)^2 & 0 & 0 \\ 0 & 0 & (x_t^2)^2 & 0 & 0 \\ 0 & 0 & 0 & (x_t^1)^2 & 0 \\ 0 & 0 & 0 & (x_t^2)^2 & 0 \\ 0 & 0 & 2x_t^1 x_t^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (x_t^1)^2 \\ 0 & 0 & 0 & 0 & x_t^1 x_t^2 \\ 0 & 0 & 0 & 0 & x_t^1 x_t^2 \\ 0 & 0 & 0 & 0 & (x_t^2)^2 \\ 0 & 0 & 0 & 2x_t^1 x_t^2 & 0 \end{bmatrix}^\top}_{\text{input matrix}} \underbrace{\begin{bmatrix} A_t^{11} \\ A_t^{12} \\ A_t^{21} \\ A_t^{22} \\ (A_t^{11})^2 \\ (A_t^{12})^2 \\ (A_t^{21})^2 \\ (A_t^{22})^2 \\ A_t^{11} A_t^{12} \\ A_t^{11} A_t^{21} \\ A_t^{11} A_t^{22} \\ A_t^{12} A_t^{21} \\ A_t^{12} A_t^{22} \\ A_t^{21} A_t^{22} \end{bmatrix}}_{\text{parameter}} \tag{4.20}$$

Note that for the properly spacing and presenting the equation, the input matrix contains the transpose operator.

Since the system parameters A is bounded therefore sub-Gaussian, each row of (4.20) forms a scalar-vector estimation problem as Problem 20. For example, the following equation can be used to estimated $\mathbb{E}[A^{11}]$ and $\mathbb{E}[A^{12}]$

$$\underbrace{x_{t+1}^1}_{z_t} = \underbrace{\begin{bmatrix} x_t^1 & x_t^2 \end{bmatrix}}_{g_t^\top} \underbrace{\begin{bmatrix} A_t^{11} & A_t^{12} \end{bmatrix}^\top}_{\theta_t} \tag{4.21}$$

$\mathbb{E}[(A^{11})^2]$, $\mathbb{E}[(A^{12})^2], \mathbb{E}[A^{11}A^{12}]$ can be estimated with

$$\underbrace{(x_{t+1}^1)^2}_{z_t} = \underbrace{\begin{bmatrix}(x_t^1)^2 & (x_t^2)^2 & 2x_t^1x_t^2\end{bmatrix}}_{g_t^\top}\underbrace{\begin{bmatrix}(A_t^{11})^2 & (A_t^{12})^2 & A_t^{11}A_t^{12}\end{bmatrix}^\top}_{\theta_t} \tag{4.22}$$

where matching to the variables of Problem 20 is indicated.

It is worth noting that, the input matrix has only one non-zero element in each column (mind the transpose), therefore the least-squares problems corresponding to different columns are not coupled. This enables us to estimate different elements of moments of A by solving different instances of Problem 20 separately instead of solving a whole vector-matrix structured data estimation problem.

As in this simple example, we can estimate all first moments of A and the quadratic second moments $\mathbb{E}[(A^{ij})^2], i, j \in \{1,2\}$. For the second cross-moments, in this example, we can estimate all the other four except for $\mathbb{E}[A^{11}A^{22}]$ and $\mathbb{E}[A^{12}A^{21}]$, because they share the same coefficients $x_t^1x_t^2$ in the estimation problem associated with the last row, i.e.,

$$\underbrace{x_{t+1}^1x_{t+1}^2}_{z_t} = \underbrace{\begin{bmatrix}(x_t^1)^2 & x_t^1x_t^2 & x_t^1x_t^2 & (x_t^2)^2\end{bmatrix}}_{g_t^\top}\underbrace{\begin{bmatrix}A_t^{11}A_t^{21} & A_t^{11}A_t^{22} & A_t^{12}A_t^{21} & A_t^{12}A_t^{22}\end{bmatrix}^\top}_{\theta_t} \tag{4.23}$$

Due to this inability to estimate some second cross-moments, a full ellipsoid bound on A can not be obtained. However, we can synthesize the estimation and bound on $\mathbb{E}[A^{ij}]$ and $\mathbb{E}[(A^{ij})^2]$ to get a box bound. This phenomena generalizes to arbitrary n and p, which are the dimensions of states and inputs, as we show in Section 4.4.3.

4.4.3 General Dynamic Problem

The same idea of the estimation problem from Section 4.4.2 can be generalized to arbitrary n and p. Specifically, the first and second moments of the parameters are decoupled, and that some second cross-moments cannot be distinguished. We develop the procedure in detail in this section.

Define

$$P_t = \begin{bmatrix}A_t & B_t\end{bmatrix} \in \mathbb{R}^n \times \mathbb{R}^{n+p} \quad s_t = \begin{bmatrix}x_t \\ u_t\end{bmatrix} \in \mathbb{R}^{n+p} \tag{4.24}$$

We use P_t to indicate the random variable.

The dynamics of each x_t^i, $(x_t^i)^2$ and $x_t^i x_t^j$ terms are

$$x_{t+1}^i = \sum_{j=1}^{n+p} P_t^{ij} s_t^j = s_t^\top P_t^{i,:}, \quad i = 1,2,...,n \tag{4.25a}$$

$$(x_{t+1}^i)^2 = \sum_{j=1}^{n+p} (P_t^{ij})^2 (s_t^j)^2 + 2 \sum_{1 \le j < k \le n+p} P_t^{ij} P_t^{ik} s_t^j s_t^k, \quad i = 1,2,...,n \tag{4.25b}$$

$$x_{t+1}^i x_{t+1}^j = \sum_{k=1}^{n+p} \sum_{l=1}^{n+p} P_t^{ik} P_t^{jl} s_t^k s_t^l \quad i \neq j \tag{4.25c}$$

Equation (4.25) is a generalization of (4.20).

Each term that we are interested in only shows in one of the equations (4.25). We propose to follow (4.25a) and (4.25b) to estimate $\mathbb{E}[P^{ij}]$, $\mathbb{E}[(P^{ij})^2]$, bound the estimation of first and second moments, and synthesize a box bound on P^{ij} with high probability based on Lemma 8.

Note that each equation in (4.25a) forms an estimation problem the same as Problem 20, so we can apply Theorem 4 and obtain the estimation $\overline{P^{i,:}}$ and bound $\delta_E\left(\overline{P^{i,:}}\right)$ such that

$$\left\| \overline{P^{i,:}} - \mathbb{E}[P^{i,:}] \right\| \le \delta_E\left(\overline{P^{i,:}}\right), \quad i = 1,2,...,n \tag{4.26}$$

holds with high probability at least $1 - \varepsilon_1^i$. Each individual term is also bounded

$$\left\| \overline{P^{ij}} - \mathbb{E}[P^{ij}] \right\| \le \delta_E\left(\overline{P^{i,:}}\right), \quad \forall i, \forall j \tag{4.27}$$

We define Q^i to contain the variables to be estimated in each (4.25b), i.e., Q^i is a vector formed by all elements in the set

$$\{P^{ij} P^{ik} | 1 \le j \le k \le n+p\} \tag{4.28}$$

Q^i is defined for notational convenience to express the bound when Theorem 4 is applied to (4.25b). The order of the elements is irrelevant for this purpose.

Now we can express the estimation of the second moments. With probability more than $1 - \varepsilon_2^i$, the bound we get from Theorem 4 on (4.25b) is

$$\left\| \sum_{j=1}^{n+p} \overline{(P^{ij})^2} - \mathbb{E}[(P^{ij})^2] + \sum_{1 \le j < k \le n+p} \overline{P^{ij} P^{ik}} - \mathbb{E}[P^{ij} P^{ik}] \right\| \le \delta_E(\overline{Q^i}) \tag{4.29}$$

The bound can also be applied to each individual terms

$$\left\|\overline{(P^{ij})^2} - \mathbb{E}[(P^{ij})^2]\right\| \leq \delta_E(\overline{Q^i}), j = 1,2,...,n+p \tag{4.30}$$

Applying Lemma 8, based on the estimation and bound of first and second moments of P^{ij} in (4.27) and (4.30), we can get a probability bound on each P^{ij}

$$\Pr(\left|P^{ij} - \overline{P^{ij}}\right| \leq R^{ij}) \geq 1 - \varepsilon_P^{ij} \tag{4.31}$$

where R^{ij} and ε_P^{ij} come from (4.14) or (4.15a).

We can assemble all R^{ij} and ε_P^{ij}, following the notation for box bound we defined in (4.2), the overall probability box bound on P can be formulated

$$\Pr(\left|P - \overline{P}\right| \leq R) \geq 1 - \sum_{i,j} \varepsilon_P^{ij} \tag{4.32}$$

The box bound on P can be split into bounds for A and B as

$$\left|A - \overline{A}\right| \leq R_A \quad \left|B - \overline{B}\right| \leq R_B \tag{4.33a}$$

$$\overline{P} = \begin{bmatrix}\overline{A} & \overline{B}\end{bmatrix} \quad R_P = \begin{bmatrix}R_A & R_B\end{bmatrix} \tag{4.33b}$$

The procedure is summarized in Algorithm 5.

Algorithm 5 Identifying probability box bound

Collect state-input data x_t, u_t for $t = 1,2,...,T$

Apply theorem 4 to (4.25a) and (4.25b), obtaining estimation $\overline{P^{ij}}, \overline{(P^{ij})^2}$ and bound $\delta_E(\overline{P^{i,:}}), \delta_E(\overline{Q^i})$ of P^{ij} as in (4.27) and (4.30), $\forall i, j$

Apply Lemma 8, get a box bounds for $P^{ij}, \forall i, j$ with probability at least $1 - \varepsilon$

Form overall probability box bound as in (4.32) and (4.33)

4.4.4 Discussion

Ideally, we would like to estimate all the first moments $\mathbb{E}(P^{ij})$ and second moments $\mathbb{E}(P^{ik}P^{jl})$ based on (4.25) and given data s_t, obtaining a tighter ellipsoid bound (4.11). However, as we illustrated with (4.23) in Section 4.4.2, we cannot estimate all interested moments as desired in general. In (4.25c), terms $P_t^{ik}P_t^{jl}$ and $P_t^{il}P_t^{jk}$ share the same coefficient $z_t^k z_t^l$, therefore $\mathbb{E}[P^{ik}P^{jl}]$ and $\mathbb{E}[P^{ik}P^{jl}]$ cannot be separately identified and we cannot achieve a complete ellipsoid bound

on P. Additionally, there exists robust control methods for boxed uncertainty of parameters for linear systems. Therefore, we chose a probability box bound as our goal of estimation (4.32).

It would be interesting to find a sufficient condition for the norm bound to converge to zero when $T \to \infty$ specifically for the dynamic identification problem as we did in Remark 2. However, the condition requiring that $g_t g_t^\top$ be i.i.d. in Remark 2 is unlikely to be true for dynamic problems. A dedicated sufficient condition requires proving $\lambda_{\min}(V_T)$ increases linearly w.r.t. T for each of the least-squares estimation problems (4.25). The structure of the dynamic problem makes it hard to find an a priori sufficient condition. Nevertheless, the intended application is to run Algorithm 5 online and compute a series of box bounds. Regardless of predetermined convergence, the box bounds are always true and is valuable information combined with estimate of the mean of the parameters.

4.5 Online Robust Control and Identification of LSMN

Section 4.4 solves the problem of bounding parameters with given data for linear systems with multiplicative noise. We move one step forward, utilizing the bound for robust control, which also provides us with an online scheme for control/identification of linear systems. We address a few issues before presenting the final procedure.

For a real system, the sub-Gaussian parameters σ are not known. A sufficiently large bound can be chosen based on expert knowledge of the subject system, and the sub-Gaussian parameter chosen based on the bound. It is reasonable to assume such bounds can be achieved via expert knowledge of the underlying system.

The box bound would not converge to zero if we apply a pure linear feedback policy $u_t = Kx_t$. For example, when $n = 2$ and $p = 1$, the feedback policy would be

$$u_t = \begin{bmatrix} K^1 & K^2 \end{bmatrix} \begin{bmatrix} x_t^1 \\ x_t^2 \end{bmatrix} \tag{4.34}$$

and the first estimation problem associated with (4.25a) becomes

$$\underbrace{x_{t+1}^1}_{z_t} = \underbrace{\begin{bmatrix} x_t^1 & x_t^2 & K^1 x_t^1 + K^2 x_t^2 \end{bmatrix}}_{g_t^\top} \underbrace{\begin{bmatrix} A_t^{11} & A_t^{12} & B_t^{11} \end{bmatrix}^\top}_{\theta_t} \tag{4.35}$$

For this estimation problem, because of the structure of g_t, the self-normalizing matrix $V_T = \lambda I + \sum_t g_t g_t^\top$ would have a minimal eigenvalue of λ regardless of T. However, (4.12) requires

$\lambda_{\min}(V_T)$ to increase faster than $\log(T)$ rate for the bound to converge to zero. The same issue exists for all estimation problems associated with (4.25). To over come this issue, we add zero-mean exploratory Gaussian noise v_t to the linear feedback policy as $u_t = Kx_t + v_t$. The system remains stable.

With the discussion above and following Section 4.4.3, we can obtain estimation and bounds of first and second moments of system parameters, then synthesize a probabilistic box bound with them.

Obtaining a stabilizing gain in face of boxed uncertainty of parameters has been studied in robust control literature [88]. Typically, a stabilizing gain is found via calculating a Lyapunov function that decreases for all possible system parameters, which usually requires finding a feasible solution to a semidefinite constraint. More specifically, We adapted Chapter 4.7.2 in [88] for a method computing a stable gain for boxed uncertainty. Note that a robust stabilizing gain is not unique.

Our procedure runs the system continuously and updates the feedback control gain periodically with an updated box bound from the most recent data. Exploratory noise is added in the control policy. At the early stages of the process, one may not get a sufficiently small box bound such that a robust feedback gain can be computed. In that case, a nominal policy needs to be applied to gather more data. The scheme is summarized in Algorithm 6.

Algorithm 6 LSMN identification with robust control

Choose a period T for updating robust control law

loop $t = 1, 2, ...,$

if Robust stabilizing gain not found **then**

Apply input u_t from a nominal control policy

else

Apply input with the robust stabilizing feedback gain and exploratory noise

$$u_t = Kx_t + v_t, \quad v_t \sim \mathcal{N}(0, I) \tag{4.36}$$

end if

Run the system with u_t, measure x_t

Apply Algorithm 5 with data $[x_t, u_t], \forall t$, obtain a box bound of the parameters with probability $1 - \varepsilon$

$$|A - \mu_A| \leq R_A \quad |B - \mu_B| \leq R_B \tag{4.37}$$

if t is a multiple of T **then**

Compute a robust stabilizing gain K for (4.37)

end if

end loop

It is worth noting that, the general box bound given by Lemma 8 does not converge to the true range even if accurate mean and second moments are known. This is because the lack of knowledge of the distribution of the system parameters. When the true distribution is assumed or known, the exact range can be reached, as is the case in Lemma 8 when we specialize down to uniform distribution. Otherwise, there needs to be a larger set of parameters that is robustly stabilizable and contains the true support of the random parameters. This is further illustrated with a numerical example in Section 4.6.

4.6 Numerical Examples

4.6.1 A Scalar Linear Dynamic System

We simulated a scalar dynamic system with $p = 1$, $n = 1$ so that the true system and estimations can be easily illustrated on a 2-D plot. The parameters are uniformly distributed between

$$-1.388 \leq A \leq -1.212 \quad -4.264 \leq B \leq -4.215 \tag{4.38}$$

Our estimation assumes uniform distribution of system parameters. We simulated 10^6 steps applying Algorithm 6, used 5 times the true values of the sub-Gaussian parameters σ and the true norm $\|\theta_*\|$ in (4.12) as the estimated upper bound on these values. It is reasonable to assume such estimation can be achieved based on expert knowledge of the physical system. We used exploratory input noise $v_t \sim \mathcal{N}(0, 10^2)$. Our estimated box approaches the true box and a robust gain $K = -0.2985$ was computed that can stabilize the system.

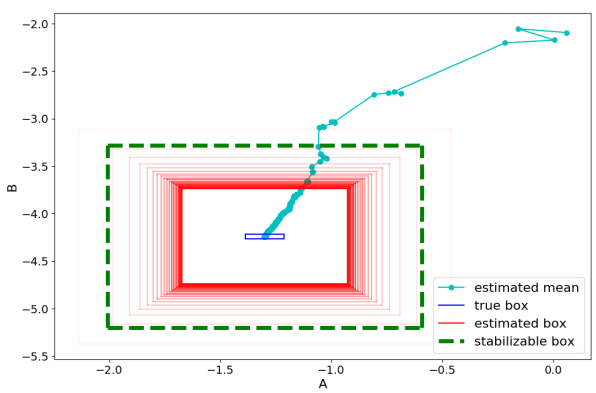


Figure 4.1: Estimation and box bound over iteration. The green box indicates a range of system parameters that can be robustly stabilized. As can be seen, the true box is well-within the robustly stable range. Our estimation falls in the robust stable box quickly approaching the true box.

This reveals a requirement for Algorithm 6. Either there exists a larger stabilizable box outside of the true box, or we have knowledge of the distribution so our synthesis can achieve the true box, then we can find a robust gain with our estimated box bound. On the other hand, if

the true system is barely robustly stabilizable, and we have little knowledge of the distribution that allows us to get more accurate estimates, we cannot find a robust gain via Algorithm 6.

4.6.2 A Vector Linear Dynamic System

Following the compact notation of P (4.24), we simulate a system with uniformly distributed system parameters upper and lower bounded by

$$P_{UB} = \begin{bmatrix} -0.1305 & -0.5410 & -0.6195 \\ -0.05983 & 1.289 & 5.773 \end{bmatrix} \tag{4.39a}$$

$$P_{LB} = \begin{bmatrix} -0.1603 & -0.5810 & -0.6595 \\ -0.0998 & 1.249 & 5.733 \end{bmatrix} \tag{4.39b}$$

We simulated 10^6 steps applying Algorithm 6. Other details such as choice of sub-Gaussian parameters, exploratory noise and choice of small probabilities, etc. are the same as in Section 4.6.1. The mean value, second moments and synthesized bound can be estimated as shown in Fig. 4.5, 4.6 and 4.7.

For the bound on first and second moments, we chose probability $\varepsilon_1 = \varepsilon_2 = 0.01$ as in (4.12). For the synthesized bound, we assumed uniform distribution and followed (4.15b), and the probability is less than 0.02 for the true value to fall out of the box.

When synthesizing the first and second moments bound applying (4.15b), $\overline{X^2} + \delta(\overline{X^2}) - \min\{[\overline{X} \pm \delta(\overline{X})]^2\}$ might be negative, which is contradicting a basic probability theory property $\mathbb{E}[X^2] - \mathbb{E}[X]^2 = \mathrm{Var}[X] \geq 0$. This happens because at early stage of estimation, the bounds on first and second moments $\delta(\overline{X})$ and $\delta(\overline{X^2})$ are very crude. In this case, we synthesize with the sub-Gaussian assumption (4.14), which gives a more conservative bound.

At early episodes when the estimated box is too large for a robust controller, we shrunk the box centered around the estimated mean until a robust controller can be found. The robust control gain is updated every 10^5 steps. Starting at 8×10^5 steps, the estimated box can be robustly stabilized. The final robust control gain found is $K = [0.01669, -0.2126]$.

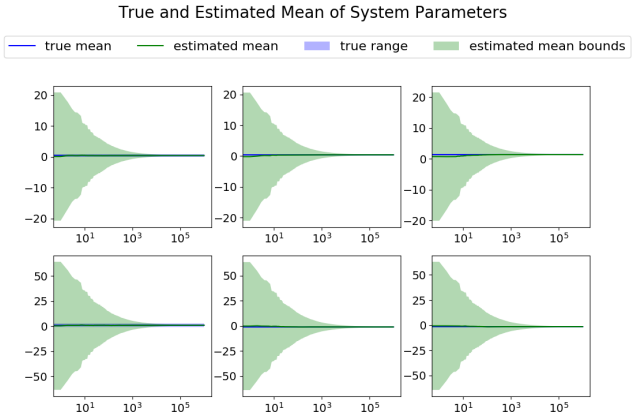


Figure 4.2: Estimation and bound of the mean value of system parameters. The estimated bound shrinks as the number of samples increases and covers the true value.

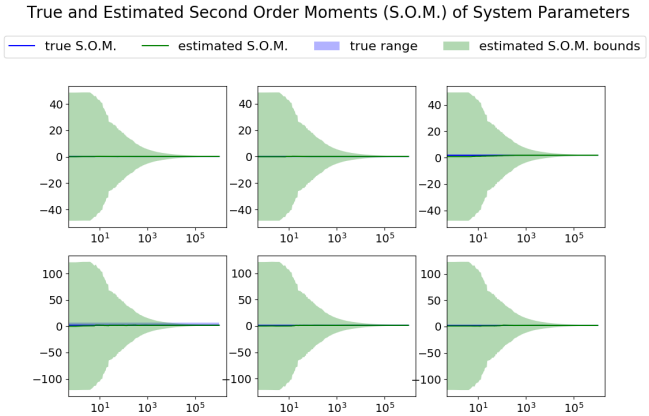


Figure 4.3: Estimation and bound of second moments of system parameters. The estimated bound shrinks as the number of samples increases and covers the true value.

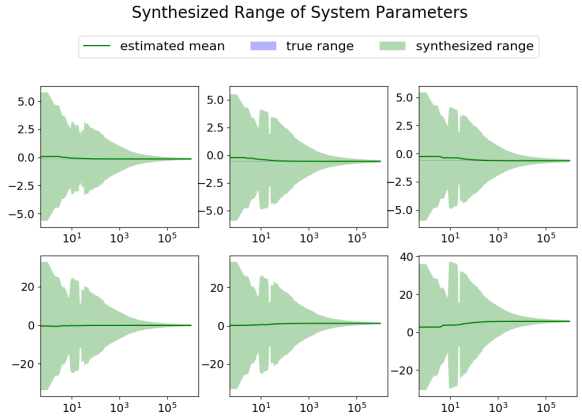


Figure 4.4: Synthesized bound. The synthesized bound shrinks more slowly.

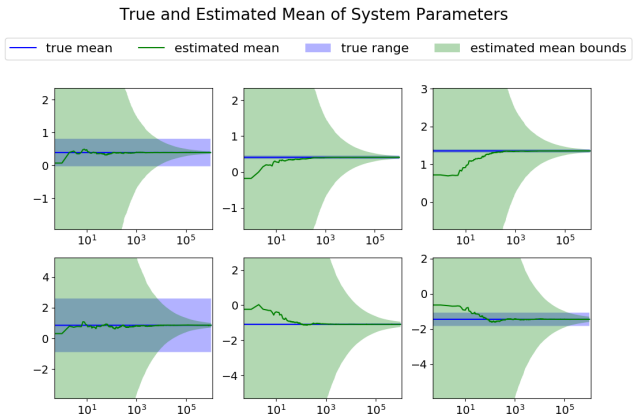


Figure 4.5: Estimation and bound of the mean value of system parameters zoomed in. Accurate estimated on means are achieved after approximately 10^3 points of data.

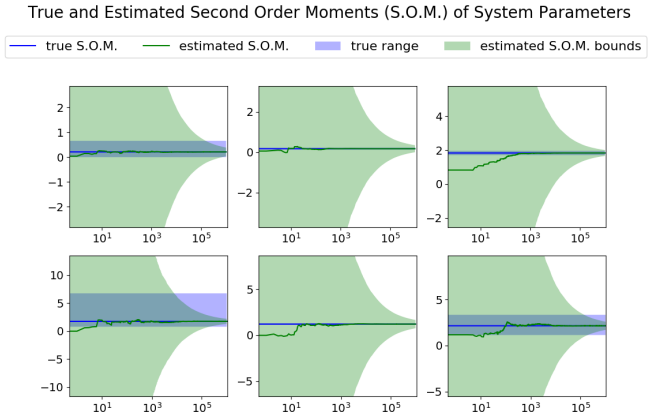


Figure 4.6: Estimation and bound of second moments of system parameters zoomed in. Accurate estimated on second moments are achieved after approximately 10^4 points of data.

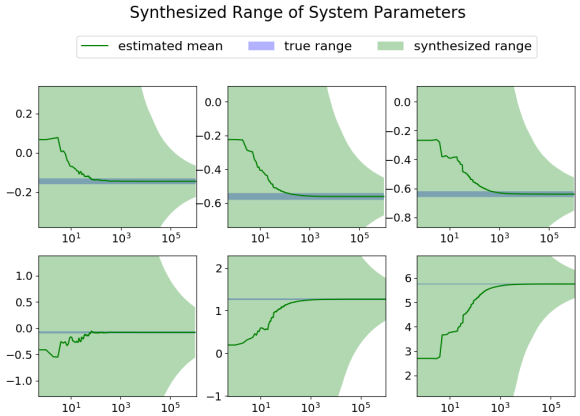


Figure 4.7: Synthesized bound. Synthesized bound getting closer to the true bound and a robustly stabilizing gain can be found for all system within the bound, including the true system, after 8×10^5 steps.

Chapter 5

Conclusion and Discussion

We extended some classic methods from optimization, optimal control and variational inequalities to constrained and unconstrained full-information non-zero-sum dynamic game, including Newton's method, Bellman recursion, differential dynamic programming, gradient descent and Douglas-Rachford splitting. Different methods apply to different combinations of constrained/unconstrained feedback/open-loop Nash equilibria. For open-loop Nash equilibria, we get local quadratic convergence for unconstrained dynamic games and local linear convergence for constrained dynamic games under reasonable assumptions. Our methods are numerically efficient w.r.t. the length of horizon because we utilize the temporal structure of dynamic games. While for the feedback methods, we can solve local Nash equilibria around stationary trajectories, but the convergence needs further studies.

As our study suggests, open-loop Nash equilibrium and feedback Nash equilibrium have very different mathematical structures and hence different methods. The open-loop game Nash equilibria and their algorithms are better understood in terms of their existence, solvability and convergence. Many further studies are worthwhile, including Nash equilibria of different information structures and partial information.

At the front of robust control and identification of linear dynamic system with multiplicative noise, we extended the standard confidence bounds results for least-squares estimation from additive noise linear equation to the more general multiplicative noise case. We further reformulated LSMN as batches of the basic least-squares estimation problems with multiplicative noise and proposed an online system identification algorithm with robust control. Our methods apply to more general systems than the existing method [72].

References

[1] Tamer Basar, Alain Haurie, and Georges Zaccour. Nonzero-sum differential games, July 2018.

[2] Suresh P Sethi. Differential games. In *Optimal Control Theory*, pages 385–407. Springer, 2019.

[3] Dario Bauso. *Game theory with engineering applications*, volume 30. Siam, 2016.

[4] Alberto Bressan. Noncooperative differential games. *Milan Journal of Mathematics*, 79(2):357–427, 2011.

[5] Ilan Rusnak. The lady, the bandits, and the bodyguards–a two team dynamic game. In *Proceedings of the 16th world IFAC congress*, pages 934–939, 2005.

[6] Oleg Prokopov and Tal Shima. Linear quadratic optimal cooperative strategies for active aircraft protection. *Journal of Guidance, Control, and Dynamics*, 36(3):753–764, 2013.

[7] Eloy Garcia, David W Casbeer, Khanh Pham, and Meir Pachter. Cooperative aircraft defense from an attacking missile. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 2926–2931. IEEE, 2014.

[8] Fouad El Ouardighi, Steffen Jørgensen, and Federico Pasin. A dynamic game with monopolist manufacturer and price-competing duopolist retailers. *OR spectrum*, 35(4):1059–1084, 2013.

[9] Quanyan Zhu, Zhu Han, and Tamer Başar. A differential game approach to distributed demand side management in smart grid. In *Communications (ICC), 2012 IEEE International Conference on*, pages 3345–3350. IEEE, 2012.

[10] Tamer Basar and Geert Jan Olsder. *Dynamic noncooperative game theory*, volume 23. Siam, 1999.

[11] Jacek B Krawczyk and Vladimir Petkov. Multistage games. *Handbook of Dynamic Game Theory*, pages 157–213, 2018.

[12] J Ben Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica: Journal of the Econometric Society*, pages 520–534, 1965.

[13] James W Friedman. On entry preventing behavior and limit price models of entry. In *Applied game theory*, pages 236–253. Springer, 1979.

[14] Chaim Fershtman and Eitan Muller. Capital accumulation games of infinite duration. *Journal of Economic Theory*, 33(2):322–339, 1984.

[15] Simone Cacace, Emiliano Cristiani, and Maurizio Falcone. Numerical approximation of nash equilibria for a class of non-cooperative differential games. *arXiv preprint arXiv:1109.3569*, 2011.

[16] Francisco Facchinei and Christian Kanzow. Generalized nash equilibrium problems. *4OR*, 5(3):173–210, 2007.

[17] Francisco Facchinei, Andreas Fischer, and Veronica Piccialli. Generalized nash equilibrium problems and newton methods. *Mathematical Programming*, 117(1-2):163–194, 2009.

[18] Steffan Berridge and Jacek B Krawczyk. Relaxation algorithms in finding nash equilibria. *Available at SSRN 66448*, 1997.

[19] Jacek B Krawczyk and Stanislav Uryasev. Relaxation algorithms to find nash equilibria with economic applications. *Environmental Modeling & Assessment*, 5(1):63–73, 2000.

[20] Jacek Krawczyk. Numerical solutions to coupled-constraint (or generalised Nash) equilibrium problems. *Computational Management Science*, 4(2):183–204, 2007.

[21] Javier Contreras, Matthias Klusch, and Jacek B Krawczyk. Numerical solutions to nash-cournot equilibria in coupled constraint electricity markets. *IEEE Transactions on Power Systems*, 19(1):195–206, 2004.

[22] Jacek B Krawczyk. Coupled constraint nash equilibria in environmental games. *Resource and Energy Economics*, 27(2):157–181, 2005.

[23] Paul Frihauf, Miroslav Krstic, and Tamer Basar. Nash equilibrium seeking in noncooperative games. *IEEE Transactions on Automatic Control*, 57(5):1192–1207, 2012.

[24] Paul Frihauf, Miroslav Krstic, and Tamer Başar. Finite-horizon lq control for unknown discrete-time linear systems via extremum seeking. *European Journal of Control*, 19(5):399–407, 2013.

[25] Paul Frihauf, Miroslav Krstic, and Tamer Başar. Nash equilibrium seeking for dynamic systems with non-quadratic payoffs. In *Advances in Dynamic Games*, pages 179–198. Springer, 2013.

[26] Alberto Bressan and Khai T Nguyen. Stability of feedback solutions for infinite horizon noncooperative differential games. *Dynamic Games and Applications*, 8(1):42–78, 2018.

[27] Jacob Engwerda. A numerical algorithm to calculate the unique feedback nash equilibrium in a large scalar lq differential game. *Dynamic Games and Applications*, 7(4):635–656, 2017.

[28] Hamed Kebriaei and Luigi Iannelli. Discrete-time robust hierarchical linear-quadratic dynamic games. *IEEE Transactions on Automatic Control*, 63(3):902–909, 2018.

[29] JC Engwerda et al. Feedback nash equilibria for linear quadratic descriptor differential games. *Automatica*, 48(4):625–631, 2012.

[30] N Krikelis and Z Rekasius. On the solution of the optimal linear control problems under conflict of interest. *IEEE Transactions on Automatic Control*, 16(2):140–147, 1971.

[31] Tyrone E Duncan and Bozenna Pasik-Duncan. Some stochastic differential games with state dependent noise. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 3773–3777. IEEE, 2015.

[32] David González-Sánchez and Onésimo Hernández-Lerma. A survey of static and dynamic potential games. *Science China Mathematics*, 59(11):2075–2102, 2016.

[33] David González-Sánchez and Onésimo Hernández-Lerma. *Discrete–time stochastic control and dynamic potential games: the Euler–Equation approach*. Springer Science & Business Media, 2013.

[34] David González-Sánchez and Onésimo Hernández-Lerma. Dynamic potential games: The discrete-time stochastic case. *Dynamic Games and Applications*, 4(3):309–328, 2014.

[35] Vladimir Viktorovich Mazalov, Anna Nikolaevna Rettieva, and Konstantin Evgen'evich Avrachenkov. Linear-quadratic discrete-time dynamic potential games. *Automation and Remote Control*, 78(8):1537–1544, 2017.

[36] Santiago Zazo, Sergio Valcarcel, S Matilde, Javier Zazo, et al. A new framework for solving dynamic scheduling games. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 2071–2075. IEEE, 2015.

[37] Santiago Zazo, Sergio Valcarcel Macua, Matilde Sánchez-Fernández, and Javier Zazo. Dynamic potential games in communications: Fundamentals and applications. *arXiv preprint arXiv:1509.01313*, 2015.

[38] Wei Sun, Evangelos A Theodorou, and Panagiotis Tsiotras. Game theoretic continuous time differential dynamic programming. In *American Control Conference (ACC), 2015*, pages 5593–5598. IEEE, 2015.

[39] Wei Sun, Evangelos A Theodorou, and Panagiotis Tsiotras. Stochastic game theoretic trajectory optimization in continuous time. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 6167–6172. IEEE, 2016.

[40] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2nd edition, 2006.

[41] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007.

[42] Pontus Giselsson and Stephen Boyd. Linear convergence and metric selection for douglas-rachford splitting and admm. *IEEE Transactions on Automatic Control*, 62(2):532–544, 2016.

[43] Brendan O'Donoghue, Giorgos Stathopoulos, and Stephen Boyd. A splitting method for optimal control. *IEEE Transactions on Control Systems Technology*, 21(6):2432–2442, 2013.

[44] Giorgos Stathopoulos, Harsh Shukla, Alexander Szucs, Ye Pu, Colin N Jones, et al. Operator splitting methods in control. *Foundations and Trends® in Systems and Control*, 3(3):249–362, 2016.

[45] Tamer Basar and Georges Zaccour. Handbook of dynamic game theory, 2018.

[46] Giancarlo Bigi, Marco Castellani, Massimo Pappalardo, and Mauro Passacantando. Nonlinear programming techniques for equilibria. 2018.

[47] Yaodong Pan and U Ozguner. Sliding mode extremum seeking control for linear quadratic dynamic game. In *Proceedings of the 2004 American Control Conference*, volume 1, pages 614–619. IEEE, 2004.

[48] T Basar. On the uniqueness of the nash solution in linear-quadratic differential games. *International Journal of Game Theory*, 5(2-3):65–90, 1976.

[49] Jacob Engwerda. Feedback nash equilibria in the scalar infinite horizon lq-game. *Automatica*, 36(1):135–139, 2000.

[50] Wei Lin. Differential games for multi-agent systems under distributed information. 2013.

[51] Alain Haurie, Jacek B Krawczyk, and Georges Zaccour. *Games and dynamic games*, volume 1. World Scientific Publishing Company, 2012.

[52] W Davis Dechert. Non cooperative dynamic games: a control theoretic approach. *Unpublished. Available on request to the author*, 1997.

[53] Santiago Zazo, Sergio Valcarcel Macua, Matilde Sánchez-Fernández, and Javier Zazo. Dynamic potential games with constraints: fundamentals and applications in communications. *IEEE Transactions on Signal Processing*, 64(14):3806–3821, 2016.

[54] Rufus Isaacs. *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1999.

[55] Ioannis Exarchos, Evangelos Theodorou, and Panagiotis Tsiotras. Stochastic differential games: A sampling approach via fbsdes. *Dynamic Games and Applications*, pages 1–20, 2018.

[56] Harold J Kushner. Numerical approximations for stochastic differential games. *SIAM journal on control and optimization*, 41(2):457–486, 2002.

[57] Moritz Diehl and Jakob Bjornberg. Robust dynamic programming for min-max model predictive control of constrained uncertain systems. *IEEE Transactions on Automatic Control*, 49(12):2253–2257, 2004.

[58] Dean Carlson, Alain Haurie, and Georges Zaccour. Infinite horizon concave games with coupled constraints. *Handbook of Dynamic Game Theory*, pages 1–44, 2016.

[59] Pontus Giselsson. Tight global linear convergence rate bounds for douglas–rachford splitting. *Journal of Fixed Point Theory and Applications*, 19(4):2241–2270, 2017.

[60] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[61] W Murray Wonham. Optimal stationary control of a linear system with state-dependent noise. *SIAM Journal on Control*, 5(3):486–500, 1967.

[62] Laurent El Ghaoui. State-feedback control of systems with multiplicative noise via linear matrix inequalities. *Systems & Control Letters*, 24(3):223–228, 1995.

[63] Y Phillis. Controller design of systems with multiplicative noise. *IEEE Transactions on Automatic Control*, 30(10):1017–1019, 1985.

[64] Benjamin Gravell, Peyman Mohajerin Esfahani, and Tyler Summers. Learning robust control for lqr systems with multiplicative noise via policy gradient. *arXiv preprint arXiv:1905.13547*, 2019.

[65] Noel E Du Toit and Joel W Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 28(1):101–115, 2011.

[66] Panos Antsaklis and John Baillieul. Special issue on technology of networked control systems. *Proceedings of the IEEE*, 95(1):5–8, 2007.

[67] Joo P Hespanha, Payam Naghshtabrizi, and Yonggang Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, 2007.

[68] Yi Guo and Tyler H Summers. A performance and stability analysis of low-inertia power grids with stochastic system inertia. In *2019 American Control Conference (ACC)*, pages 1965–1970. IEEE, 2019.

[69] Thomas B Schön, Adrian Wills, and Brett Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

[70] Nikolas Kantas, Arnaud Doucet, Sumeetpal S Singh, Jan Maciejowski, Nicolas Chopin, et al. On particle methods for parameter estimation in state-space models. *Statistical science*, 30(3):328–351, 2015.

[71] Thomas B Schön, Fredrik Lindsten, Johan Dahlin, Johan Wragberg, Christian A Naesseth, Andreas Svensson, and Liang Dai. Sequential monte carlo methods for system identification. *IFAC-PapersOnLine*, 48(28):775–786, 2015.

[72] Yu Xing, Ben Gravell, Xingkang He, Karl Henrik Johansson, and Tyler Summers. Linear system identification under multiplicative noise from multiple trajectory data. *arXiv preprint arXiv:2002.06613*, 2020.

[73] Francisco Facchinei, Andreas Fischer, and Veronica Piccialli. On generalized nash games and variational inequalities. *Operations Research Letters*, 35(2):159–164, 2007.

[74] Christophe Dutang. A survey of gne computation methods: theory and algorithms. 2013.

[75] DM Murray and SJ Yakowitz. Differential dynamic programming and newton's method for discrete optimal control problems. *Journal of Optimization Theory and Applications*, 43(3):395–414, 1984.

[76] Joseph C Dunn and Dimitri P Bertsekas. Efficient dynamic programming implementations of newton's method for unconstrained optimal control problems. *Journal of Optimization Theory and Applications*, 63(1):23–38, 1989.

[77] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian

Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[78] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[79] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.

[80] L-Z Liao and Christine A Shoemaker. Convergence in unconstrained discrete-time differential dynamic programming. *IEEE Transactions on Automatic Control*, 36(6):692–706, 1991.

[81] Yuval Tassa. *Theory and Implementation of Biomimetic Motor Controllers*. Hebrew University of Jerusalem, 2011.

[82] Bolei Di and Andrew Lamperski. Newton's Method and Differential Dynamic Programming for Unconstrained Nonlinear Dynamic Games. *arXiv e-prints*, page arXiv:1906.09097, Jun 2019, 1906.09097.

[83] James Blake Rawlings, David Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2019.

[84] Moritz Diehl and Sebastien Gros. *Numerical Optimal Control*. –, expected to be published in 2018.

[85] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[86] Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[87] Joel A Tropp. An introduction to matrix concentration inequalities. *arXiv preprint arXiv:1501.01571*, 2015.

[88] Robert E Skelton, Tetsuya Iwasaki, and Dimitri E Grigoriadis. *A unified algebraic approach to control design*. CRC Press, 1997.

[89] Asen L Dontchev and R Tyrrell Rockafellar. Implicit functions and solution mappings. *Springer Monographs in Mathematics. Springer*, 2014.

[90] Shu Lu and Stephen M Robinson. Variational inequalities over perturbed polyhedral convex sets. *Mathematics of Operations Research*, 33(3):689–711, 2008.

[91] Stephen M Robinson. Normal maps induced by linear transformations. *Mathematics of Operations Research*, 17(3):691–714, 1992.

Appendix A

Auxiliary Proofs for Unconstrained Dynamic Games

A.1 Proof of Lemma 2

First, we prove that the dynamics constraints (2.11d) and (2.11e) are inductive definitions of (2.12a) and (2.12b). Note that x_0 is fixed so that (2.12a) and (2.12b) hold at $k = 0$. Now we handle each of the terms inductively. For δx_{k+1}, we have

$$\begin{aligned}
\delta x_{k+1} &= \sum_{i=0}^{T} \frac{\partial x_{k+1}}{\partial u_{:,i}}\Big|_{\bar{x},\bar{u}} \delta u_{:,i} \\
&= \sum_{i=0}^{T} \frac{\partial f_k(x_k, u_{:,k})}{\partial u_{:,i}}\Big|_{\bar{x},\bar{u}} \delta u_{:,i} \\
&= \frac{\partial f_k(x_k, u_{:,k})}{\partial x_k}\Big|_{\bar{x},\bar{u}} \sum_{i=0}^{T} \frac{\partial x_k}{\partial u_{:,k}} \delta u_{:,i} + \frac{\partial f_k(x_k, u_{:,k})}{\partial u_{:,k}}\Big|_{\bar{x},\bar{u}} \delta u_{:,k} \\
&= A_k \delta x_k + B_k \delta u_{:,k} \qquad \text{(A.1)}
\end{aligned}$$

We used the fact that $\frac{\partial f(x_k, u_{:,k})}{\partial u_{:,i}}$ is zero unless $i = k$.

For Δx_{k+1}, row l is given by:

$$\begin{aligned}
\Delta x_{k+1}^{l} &= \sum_{i=0}^{T} \sum_{j=0}^{T} \delta u_{:,i}^{\top} \frac{\partial^2 f_k^l(x_k, u_{:,k})}{\partial u_{:,i} \partial u_{:,j}}\Big|_{\bar{u}} \delta u_{:,j} \qquad \text{(A.2a)} \\
&= \sum_{i=0}^{T} \sum_{j=0}^{T} \delta u_{:,i}^{\top} \left(\frac{\partial^2 f_k^l}{\partial u_{:,i} \partial u_{:,j}} + (\frac{\partial x_k}{\partial u_{:,i}})^{\top} \frac{\partial^2 f_k^l}{\partial x_k^2} \frac{\partial x_k}{\partial u_{:,j}} \right)\Big|_{\bar{u}} \delta u_{:,j}
\end{aligned}$$

$$+\sum_{i=0}^{T}\sum_{j=0}^{T}\delta u_{:,i}^{\top}\left((\frac{x_k}{\partial u_{:,i}})^{\top}\frac{\partial^2 f_k^l}{\partial x_k \partial u_{:,j}}+\frac{\partial^2 f_k^l}{\partial u_{:,i}\partial x_k}\frac{x_k}{\partial u_{:,j}}\right)\Big|_{\bar{u}}\delta u_{:,j}$$

$$+\sum_{i=0}^{T}\sum_{j=0}^{T}\delta u_{:,i}^{\top}\left(\sum_{p=1}^{n_x}\frac{\partial f_k^l}{\partial x_k^p}\frac{\partial^2 x_k^p}{\partial u_{:,i}\partial u_{:,j}}\right)\Big|_{\bar{u}}\delta u_{:,j} \tag{A.2b}$$

$$=\delta u_{:,k}^{\top}\frac{\partial^2 f_k^l}{\partial u_{:,k}^2}\delta u_{:,k}+\delta x_k^{\top}\frac{\partial^2 f_k^l}{\partial x_k^2}\delta x_k+\delta x_k^{\top}\frac{\partial^2 f_k^l}{\partial x_k \partial u_{:,k}}\delta u_{:,k}$$

$$+\delta u_{:,k}^{\top}\frac{\partial^2 f_k^l}{\partial u_{:,k}\partial x_k}\delta x_k+\sum_{p=1}^{n_x}\frac{\partial f_k^l}{\partial x_k^p}\sum_{i=0}^{T}\sum_{j=0}^{T}\delta u_{:,i}^{\top}\frac{\partial^2 x_k^p}{\partial u_{:,i}\partial u_{:,j}}\delta u_{:,j} \tag{A.2c}$$

$$=\begin{bmatrix}\delta x_k\\ \delta u_{:,k}\end{bmatrix}^{\top}G_k^l\begin{bmatrix}\delta x_k\\ \delta u_{:,k}\end{bmatrix}+\sum_{p=1}^{n_x}A_k^{lp}\Delta x_k^p \tag{A.2d}$$

To get to each terms in (A.2c), we used the fact that

$$\frac{\partial^2 f_k^l}{\partial u_{:,i}\partial u_{:,j}}=0,\ \text{for } i\neq k \text{ or } j\neq k \tag{A.3a}$$

$$\delta x_k=\sum_{i=0}^{T}\frac{\partial x_k}{\partial u_{:,i}}\delta u_{:,i}=\sum_{i=0}^{k-1}\frac{\partial x_k}{\partial u_{:,i}}\delta u_{:,i} \tag{A.3b}$$

To get to (A.2d), we used the fact

$$\frac{\partial f_k^l}{\partial x_k^p}=A_k^{lp} \tag{A.4a}$$

$$\sum_{i=0}^{T}\sum_{j=0}^{T}\delta u_{:,i}^{\top}\frac{\partial^2 x_k^p}{\partial u_{:,i}\partial u_{:,j}}\Big|_{\bar{u}}\delta u_{:,j}=\Delta x_k^p \tag{A.4b}$$

$$\begin{bmatrix}\frac{\partial^2 f_k^l}{\partial x_k^2} & \frac{\partial^2 f_k^l}{\partial x_k \partial u_{:,k}}\\ \frac{\partial^2 f_k^l}{\partial x_k \partial u_{:,k}} & \frac{\partial^2 f_k^l}{\partial u_{:,k}^2}\end{bmatrix}\Bigg|_{\bar{x},\bar{u}}=G_k^l \tag{A.4c}$$

Both l and p are used to pick out the corresponding element for a vector or matrix. A_k^{lp} means the lth row and pth column of matrix A_k. Equation (A.2d) actually describes each element in (2.12b), so we have proven that both are true.

Next we prove (2.11a) is the quadratic approximation of $J_n(u)$, i.e.

$$\text{quad}(J_n(u))_{\bar{u}}=J_n(\bar{u})+\frac{\partial J_n(\bar{u})}{\partial u}\delta u+\frac{1}{2}\delta u^{\top}\frac{\partial^2 J_n(\bar{u})}{\partial u^2}\delta u$$

$$=\frac{1}{2}\sum_{k=0}^{T}\left(\begin{bmatrix}1\\ \delta x_k\\ \delta u_{:,k}\end{bmatrix}^{\top}M_{n,k}\begin{bmatrix}1\\ \delta x_k\\ \delta u_{:,k}\end{bmatrix}+M_{n,k}^{1k}\Delta x_k\right) \tag{A.5}$$

We need the explicit expressions for the associated derivatives.

$$\frac{\partial J_n(u)}{\partial u_{:,i}} = \sum_{k=0}^{T} \left(\frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial u_{:,i}} + \frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial x_k} \frac{\partial x_k}{\partial u_{:,i}} \right) \tag{A.6a}$$

$$\begin{aligned}
&\frac{\partial^2 J_n(u)}{\partial u_{:,i} \partial u_{:,j}} \\
&= \sum_{k=0}^{T} \left(\frac{\partial^2 c_{n,k}(x_k, u_{:,k})}{\partial u_{:,i} \partial u_{:,j}} + \frac{\partial x_k}{\partial u_{:,i}}^\top \frac{\partial^2 c_{n,k}(x_k, u_{:,k})}{\partial x_k^2} \frac{\partial x_k}{\partial u_{:,j}} \right) \\
&+ \sum_{k=0}^{T} \left(\frac{\partial x_k}{\partial u_{:,i}}^\top \frac{\partial^2 c_{n,k}(x_k, u_{:,k})}{\partial x_k \partial u_{:,j}} + \frac{\partial^2 c_{n,k}(x_k, u_{:,k})}{\partial u_{:,i} \partial x_k} \frac{\partial x_k}{\partial u_{:,j}} \right) \\
&+ \sum_{k=0}^{T} \sum_{l=1}^{n_x} \frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial x_k^l} \frac{\partial^2 x_k^l}{\partial u_{:,i} \partial u_{:,i}}
\end{aligned} \tag{A.6b}$$

We break down each term in (A.5). First the second order term.

$$\delta u^\top \frac{\partial^2 J_n(\bar{u})}{\partial u^2} \delta u = \sum_{i,j=0}^{T} \delta u_{:,i}^T \frac{\partial^2 J_n(u)}{\partial u_{:,i} \partial u_{:,j}} \bigg|_{\bar{u}} \delta u_{:,j} \tag{A.7a}$$

$$\begin{aligned}
&= \sum_{i,j,k=0}^{T} \delta u_{:,i}^\top \left(\frac{\partial^2 c_{n,k}(x_k, u_{:,k})}{\partial u_{:,i} \partial u_{:,j}} \bigg|_{\bar{u}} \right) \delta u_{:,j} \\
&+ \sum_{i,j,k=0}^{T} \delta u_{:,i}^\top \left(\frac{\partial x_k}{\partial u_{:,i}}^\top \frac{\partial^2 c_{n,k}(x_k, u_{:,k})}{\partial x_k^2} \bigg|_{\bar{u}} \frac{\partial x_k}{\partial u_{:,j}} \right) \delta u_{:,j} \\
&+ \sum_{i,j,k=0}^{T} \delta u_{:,i}^\top \left(\frac{\partial x_k}{\partial u_{:,i}}^\top \frac{\partial^2 c_{n,k}(x_k, u_{:,k})}{\partial x_k \partial u_{:,j}} \bigg|_{\bar{u}} \right) \delta u_{:,j} \\
&+ \sum_{i,j,k=0}^{T} \delta u_{:,i}^\top \left(\frac{\partial^2 c_{n,k}(x_k, u_{:,k})}{\partial u_{:,i} \partial x_k} \bigg|_{\bar{u}} \frac{\partial x_k}{\partial u_{:,j}} \right) \delta u_{:,j} \\
&+ \sum_{k=0}^{T} \sum_{p=1}^{n} \frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial x_k^p} \bigg|_{\bar{u}} \sum_{i,j=0}^{T} \delta u_{:,i}^\top \frac{\partial^2 x_k^p}{\partial u_{:,i} \partial u_{:,i}} \delta u_{:,j}
\end{aligned} \tag{A.7b}$$

$$\begin{aligned}
&= \sum_{k=0}^{T} \left(\delta u_{:,k}^\top \frac{\partial^2 c_{n,k}}{\partial u_{:,k}^2} \bigg|_{\bar{u}} \delta u_{:,k} + \delta x_k^\top \frac{\partial^2 c_{n,k}}{\partial x_k^2} \bigg|_{\bar{u}} \delta x_k \right) + \\
&\sum_{k=0}^{T} \left(\delta x_k^\top \frac{\partial^2 c_{n,k}}{\partial x_k \partial u_{:,k}} |_{\bar{u}} \delta u_{:,k} + \delta u_{:,k}^\top \frac{\partial^2 c_{n,k}}{\partial u_{:,k} \partial x_k} \bigg|_{\bar{u}} \delta x_k \right) + \\
&\sum_{k=0}^{T} \sum_{p=1}^{n} \frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial x_k^p} \Delta x_k^p
\end{aligned} \tag{A.7c}$$

$$= \sum_{k=0}^{T} \left(\begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top \begin{bmatrix} \frac{\partial^2 c_{n,k}}{\partial x_k^2} & \frac{\partial^2 c_{n,k}}{\partial x_k \partial u_{:,k}} \\ \frac{\partial^2 c_{n,k}}{\partial u_{:,k} \partial x_k} & \frac{\partial^2 c_{n,k}}{\partial u_{:,k}^2} \end{bmatrix} \Bigg|_{\bar{u}} \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix} \right)$$

$$+ \left(\frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial x_k} \Delta x_k \right) \tag{A.7d}$$

The first term in (A.7b) to (A.7c) holds because $c_{n,k}(x_k, u_{:,k})$ only depends directly on $u_{:,i}$ and $u_{:,j}$ when $i = j = k$. The others hold because x_k only depends on $u_{:,i}$ and $u_{:,j}$ when $i, j < k$. The last term uses the definition of Δx_k in (2.12b).

The first order term

$$\frac{\partial J_n(\bar{u})}{\partial u} \delta u = \sum_{i=0}^{T} \frac{\partial J_n(u)}{\partial u_{:,i}} \delta u_{:,i} \tag{A.8a}$$

$$= \sum_{i,k=0}^{T} \left(\frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial u_{:,i}} + \frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial x_k} \frac{\partial x_k}{\partial u_{:,i}} \right) \delta u_{:,i} \tag{A.8b}$$

$$= \sum_{k=0}^{T} \left(\frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial u_{:,k}} \delta u_{:,k} + \frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial x_k} \sum_{i=0}^{T} \frac{\partial x_k}{\partial u_{:,i}} \right) \tag{A.8c}$$

$$= \sum_{k=0}^{T} \left(\frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial u_{:,k}} \delta u_{:,k} + \frac{\partial c_{n,k}(x_k, u_{:,k})}{\partial x_k} \delta x_k \right) \tag{A.8d}$$

And constant term

$$J_n(\bar{u}) = \sum_{k=0}^{T} c_{n,k}(\bar{x}_k, \bar{u}_{:,k}) \tag{A.9}$$

From (A.7), (A.8), and (A.9) it follows that (A.5) is true.

A.2 Equivalency of Reformulated Game for OLNE

Proof. All that is needed is to prove

$$\sum_{k=0}^{T} M_{n,k}^{1x} \Delta x_k = \sum_{k=0}^{T} \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top D_{n,k} \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix} \tag{A.10}$$

Because the cost term $M_{n,k}^{1x} \Delta x$ is linear w.r.t. Δx, the propagation $\Delta x_{k+1} = A_k \Delta x_k + R_k(\delta x_k, \delta u_k)$ is linear in Δx_k and quadratic in δx_k and δu_k, we know that the L.H.S. is the summation of quadratic terms of δx_k and δu_k without constant terms, therefore it can be written as

$$\sum_{k=0}^{T} M_{n,k}^{1x} \Delta x_k = \sum_{k=0}^{T} \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top \hat{D}_{n,k} \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix} \tag{A.11}$$

Each additional term is the contribution of a pair δx_k and δu_k to the total cost. We examine the contribution of δx_l and δu_l, $0 \leq l \leq T$, i.e.,

$$\begin{aligned}\begin{bmatrix}\delta x_l \\ \delta u_{:,l}\end{bmatrix}^\top \hat{D}_{n,l} \begin{bmatrix}\delta x_l \\ \delta u_{:,l}\end{bmatrix} =& M^{1x}_{n,T} A_{T-1} A_{T-2} \cdots A_{l+1} R_l(\delta x_l, \delta u_l) + \\ & M^{1x}_{n,T-1} A_{T-2} A_{T-3} \cdots A_{l+1} R_l(\delta x_l, \delta u_l) + \\ & \vdots \\ & M^{1x}_{n,l+2} A_{l+1} R_l(\delta x_l, \delta u_l) + \\ & M^{1x}_{n,l+1} R_l(\delta x_l, \delta u_l)\end{aligned} \tag{A.12a}$$

where each term in the R.H.S. is how $R_l(\delta x_l, \delta u_l)$ contributes to each $M^{1x}_{n,k} \Delta x_k$. The coefficients in front of $R_l(\delta x_l, \delta u_l)$ can be merged together and it is easy to further verify that

$$\begin{bmatrix}\delta x_l \\ \delta u_{:,l}\end{bmatrix}^\top \hat{D}_{n,l} \begin{bmatrix}\delta x_l \\ \delta u_{:,l}\end{bmatrix} = \Omega_{n,l+1} R_l(\delta x_l, \delta u_l) = \begin{bmatrix}\delta x_l \\ \delta u_{:,l}\end{bmatrix}^\top \sum_{i=1}^{n_x} \Omega^i_{n,l+1} G^i_l \begin{bmatrix}\delta x_l \\ \delta u_{:,l}\end{bmatrix} \tag{A.12b}$$

$$= \begin{bmatrix}\delta x_l \\ \delta u_{:,l}\end{bmatrix}^\top D_{n,l} \begin{bmatrix}\delta x_l \\ \delta u_{:,l}\end{bmatrix} \tag{A.12c}$$

Therefore, $D_{n,k} = \hat{D}_{n,k}$ and (A.10) holds. □

A.3 Derive the Gradient of the Open-Loop Game and Solve OLNE

We solve the open-loop Nash equilibrium of (2.15) in this section, which is also the Newton step (2.4). The following expression of the states $\delta \hat{x}_k$ depending only on the actions $\delta u_{:,:k-1}$ is useful.

$$\delta \hat{x}_k = \Pi_{i=0}^{k-1} \hat{A}_i \delta \hat{x}_0 + \sum_{i=0}^{k-1} \Pi_{j=i+1}^{k-1} \hat{A}_j \hat{B}_i u_{:,i} \tag{A.13a}$$

where we use the Π to indicate consecutive matrix multiplication on the left, i.e.,

$$\Pi_{i=0}^{k-1} \hat{A}_i = \hat{A}_{k-1} \cdots \hat{A}_1 \hat{A}_0 \tag{A.14}$$

In case when the superscript is less than the subscript for the Π operator, we define it to be the identity of the proper dimension, i.e.

$$\Pi_{i=k}^{k-1} \hat{A}_i = I \tag{A.15}$$

Deriving the Gradient of the Game

We begin with rearranging (2.15) in a matrix form, eliminating the explicit dynamics and collecting each player's action over time together, so it is easier to compute the gradient of the game.

$$\begin{aligned}\min_{\delta u_{n,:}} \quad \delta J_n(\delta\hat{x},\delta u) &= \frac{1}{2}\begin{bmatrix}\delta\hat{x}\\ \delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix}^\top H_n \begin{bmatrix}\delta\hat{x}\\ \delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix}\\ &= \frac{1}{2}\begin{bmatrix}\delta\hat{x}\\ \delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix}^\top \begin{bmatrix} H_n^{xx} & H_n^{xu_1} & H_n^{xu_2} & \cdots & H_n^{xu_N}\\ H_n^{u_1x} & H_n^{u_1u_1} & H_n^{u_1u_2} & \cdots & H_n^{u_1u_N}\\ H_n^{u_2x} & H_n^{u_2u_1} & H_n^{u_2u_2} & \cdots & H_n^{u_2u_N}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ H_n^{u_Nx} & H_n^{u_Nu_1} & H_n^{u_Nu_2} & \cdots & H_n^{u_Nu_N}\end{bmatrix}\begin{bmatrix}\delta\hat{x}\\ \delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix}\end{aligned} \tag{A.16a}$$

$$\text{subject to} \quad \delta\hat{x} = \begin{bmatrix}\delta\hat{x}_0\\ \delta\hat{x}_1\\ \vdots\\ \delta\hat{x}_T\end{bmatrix} = \begin{bmatrix}F_1 & F_2 & \cdots & F_N\end{bmatrix}\begin{bmatrix}\delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix} + E\,\delta\hat{x}_0 \tag{A.16b}$$

where

$$H_n^{xx} = \begin{bmatrix}\hat{M}_{n,0}^{xx} & & & \\ & \hat{M}_{n,1}^{xx} & & \\ & & \ddots & \\ & & & \hat{M}_{n,T}^{xx}\end{bmatrix} \quad (H_n^{u_ix})^\top = H_n^{xu_i} = \begin{bmatrix}\hat{M}_{n,0}^{xu_i} & & & \\ & \hat{M}_{n,1}^{xu_i} & & \\ & & \ddots & \\ & & & \hat{M}_{n,T}^{xu_i}\end{bmatrix} \tag{A.17a}$$

$$(H_n^{u_ju_i})^\top = H_n^{u_iu_j} = \begin{bmatrix}\hat{M}_{n,0}^{u_iu_j} & & & \\ & \hat{M}_{n,1}^{u_iu_j} & & \\ & & \ddots & \\ & & & \hat{M}_{n,T}^{u_iu_j}\end{bmatrix} \quad E = \begin{bmatrix}0\\ \hat{A}_0\\ \hat{A}_1A_0\\ \vdots\\ \Pi_{i=0}^{T-1}\hat{A}_i\end{bmatrix} \tag{A.17b}$$

$$F_n = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ \hat{B}_{n,0} & 0 & 0 & \cdots & 0 & 0 \\ \hat{A}_1\hat{B}_{n,0} & \hat{B}_{n,1} & 0 & \cdots & 0 & 0 \\ \hat{A}_2\hat{A}_1\hat{B}_{n,0} & \hat{A}_2\hat{B}_{n,1} & \hat{B}_{n,2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \Pi_{k=1}^{T-1}\hat{A}_k\hat{B}_{n,0} & \Pi_{k=2}^{T-1}\hat{A}_k\hat{B}_{n,1} & \cdots & \hat{A}_{T-1}\hat{B}_{n,T-2} & \hat{B}_{n,T-1} & 0 \end{bmatrix} \tag{A.17c}$$

Note that F_n is strictly lower block triangular and it propagates player n's action to all the states. Similarly, E propagates the initial condition to all the rest of the states. While the previous formulation (2.15) has an explicit time dependency, this formulation collects players' action together and then over time. The current formulation is especially convenient for computing the gradient of the game.

We substitute (A.16b) into (A.16a) to get rid of the dependency on $\delta\hat{x}$. The terms related to $\delta\hat{x}$ in (A.16a) are

$$\frac{1}{2}\delta\hat{x}^\top H_n^{xx}\delta\hat{x} + \delta\hat{x}^\top \begin{bmatrix} H_n^{xu_1} & H_n^{xu_2} & \cdots & H_n^{xu_N} \end{bmatrix} \begin{bmatrix} \delta u_{1,:} \\ \delta u_{2,:} \\ \vdots \\ \delta u_{N,:} \end{bmatrix} \tag{A.18a}$$

$$\begin{aligned} =&\frac{1}{2}\left(\begin{bmatrix} F_1 & F_2 & \cdots & F_N \end{bmatrix} \begin{bmatrix} \delta u_{1,:} \\ \delta u_{2,:} \\ \vdots \\ \delta u_{N,:} \end{bmatrix} + E\delta\hat{x}_0 \right)^\top H_n^{xx} \left(\begin{bmatrix} F_1 & F_2 & \cdots & F_N \end{bmatrix} \begin{bmatrix} \delta u_{1,:} \\ \delta u_{2,:} \\ \vdots \\ \delta u_{N,:} \end{bmatrix} + E\delta\hat{x}_0 \right) \\ &+ \left(\begin{bmatrix} F_1 & F_2 & \cdots & F_N \end{bmatrix} \begin{bmatrix} \delta u_{1,:} \\ \delta u_{2,:} \\ \vdots \\ \delta u_{N,:} \end{bmatrix} + E\delta\hat{x}_0 \right)^\top \begin{bmatrix} H_n^{xu_1} & H_n^{xu_2} & \cdots & H_n^{xu_N} \end{bmatrix} \begin{bmatrix} \delta u_{1,:} \\ \delta u_{2,:} \\ \vdots \\ \delta u_{N,:} \end{bmatrix} \end{aligned} \tag{A.18b}$$

$$=\frac{1}{2}\begin{bmatrix} \delta u_{1,:} \\ \delta u_{2,:} \\ \vdots \\ \delta u_{N,:} \end{bmatrix}^\top \begin{bmatrix} F_1^\top H_n^{xx} F_1 & F_1^\top H_n^{xx} F_2 & \cdots & F_1^\top H_n^{xx} F_N \\ F_2^\top H_n^{xx} F_1 & F_2^\top H_n^{xx} F_2 & \cdots & F_2^\top H_n^{xx} F_N \\ \vdots & \vdots & \ddots & \vdots \\ F_N^\top H_n^{xx} F_1 & F_N^\top H_n^{xx} F_2 & \cdots & F_N^\top H_n^{xx} F_N \end{bmatrix} \begin{bmatrix} \delta u_{1,:} \\ \delta u_{2,:} \\ \vdots \\ \delta u_{N,:} \end{bmatrix} + \frac{1}{2}\delta\hat{x}_0^\top E^\top H_n^{xx} E\delta\hat{x}_0$$

$$+\begin{bmatrix}\delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix}^\top \begin{bmatrix} F_1^\top H_n^{xu_1} & F_1^\top H_n^{xu_2} & \cdots & F_1^\top H_n^{xu_N}\\ F_2^\top H_n^{xu_1} & F_2^\top H_n^{xu_2} & \cdots & F_2^\top H_n^{xu_N}\\ \vdots & \vdots & \ddots & \vdots\\ F_N^\top H_n^{xu_1} & F_N^\top H_n^{xu_2} & \cdots & F_N^\top H_n^{xu_N}\end{bmatrix}\begin{bmatrix}\delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix}$$

$$+\delta\hat{x}_0^\top E^\top \begin{bmatrix} H_n^{xx}F_1+H_n^{xu_1} & H_n^{xx}F_2+H_n^{xu_2} & \cdots & H_n^{xx}F_N+H_n^{xu_N}\end{bmatrix}\begin{bmatrix}\delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix} \tag{A.18c}$$

Symmetrize the quadratic terms, merge them into (A.16a) and ignore the constant term $\frac{1}{2}\delta\hat{x}_0^\top E^\top H_n^{xx} E\delta\hat{x}_0$, we have

$$\delta J_n(\delta\hat{x}_0,\delta u) \tag{A.19}$$

$$=\frac{1}{2}\begin{bmatrix}\delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix}^\top \begin{bmatrix} G_n^{u_1u_1} & G_n^{u_1u_2} & \cdots & G_n^{u_1u_N}\\ G_n^{u_2u_1} & G_n^{u_2u_2} & \cdots & G_n^{u_2u_N}\\ \vdots & \vdots & \ddots & \vdots\\ G_n^{u_Nu_1} & G_n^{u_Nu_2} & \cdots & G_n^{u_Nu_N}\end{bmatrix}\begin{bmatrix}\delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix}+\delta\hat{x}_0^\top E^\top \begin{bmatrix}F_1^\top H_n^{xx}+H_n^{u_1x}\\ F_2^\top H_n^{xx}+H_n^{u_2x}\\ \vdots\\ F_N^\top H_n^{xx}+H_n^{u_Nx}\end{bmatrix}^\top \begin{bmatrix}\delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix} \tag{A.20}$$

where

$$G_n^{u_iu_j}=H_n^{u_iu_j}+F_i^\top H_n^{xx}F_j+H_n^{u_ix}F_j+F_i^\top H_n^{xu_j} \tag{A.21}$$

With this notation, the gradient of the game for player n can be expressed as

$$\frac{\partial\delta J_n(\delta\hat{x}_0,\delta u)}{\partial\delta u_{n,:}}=\sum_{i=1}^{N}G_n^{u_nu_i}u_{i,:}+(F_n^\top H_n^{xx}+H_n^{u_nx})E\delta\hat{x}_0 \tag{A.22}$$

Stacking all players' gradient, the gradient of the game becomes

$$\delta\mathcal{J}(\delta\hat{x}_0,\delta u)=\begin{bmatrix} G_1^{u_1u_1} & G_1^{u_1u_2} & \cdots & G_1^{u_1u_N}\\ G_2^{u_2u_1} & G_2^{u_2u_2} & \cdots & G_2^{u_2u_N}\\ \vdots & \vdots & \ddots & \vdots\\ G_N^{u_Nu_1} & G_N^{u_Nu_2} & \cdots & G_N^{u_Nu_N}\end{bmatrix}\begin{bmatrix}\delta u_{1,:}\\ \delta u_{2,:}\\ \vdots\\ \delta u_{N,:}\end{bmatrix}+\begin{bmatrix}F_1^\top H_1^{xx}+H_n^{u_1x}\\ F_2^\top H_2^{xx}+H_2^{u_2x}\\ \vdots\\ F_N^\top H_N^{xx}+H_N^{u_Nx}\end{bmatrix}E\delta\hat{x}_0 \tag{A.23}$$

After taking the derivative and obtaining the gradient of the game, we are motivated to plug $\delta\hat{x}$ back to the gradient expression, because the backward recursion to solve the gradient equals

zero should be expressed in terms of the states as well. The playerwise gradient (A.22) can be manipulated as

$$\frac{\partial \delta J_n(\delta\hat{x}_0, \delta u)}{\partial u_{n,:}} = \sum_{i=1}^{N} (H_n^{u_n u_i} + F_n^\top H_n^{xx} F_i + H_n^{u_n x} F_i + F_n^\top H_n^{x u_i}) u_{i,:} + (F_n^\top H_n^{xx} + H_n^{u_n x}) E \delta\hat{x}_0 \tag{A.24a}$$

$$= \sum_{i=1}^{N} (H_n^{u_n u_i} + F_n^\top H_n^{x u_i}) u_{i,:} + (F_n^\top H_n^{xx} + H_n^{u_n x})(\sum_{i=1}^{N} F_i u_{i,:} + E\delta\hat{x}_0) \tag{A.24b}$$

$$= \sum_{i=1}^{N} (H_n^{u_n u_i} + F_n^\top H_n^{x u_i}) u_{i,:} + (F_n^\top H_n^{xx} + H_n^{u_n x}) \delta\hat{x} \tag{A.24c}$$

$$= \begin{bmatrix} F_n^\top & \underbrace{\mathbf{0} \quad \cdots \quad \mathbf{0}}_{n-1} & I & \underbrace{\mathbf{0} \quad \cdots \quad \mathbf{0}}_{N-n} \end{bmatrix} \begin{bmatrix} H_n^{xx} & H_n^{x u_1} & H_n^{x u_2} & \cdots & H_n^{x u_N} \\ H_n^{u_1 x} & H_n^{u_1 u_1} & H_n^{u_1 u_2} & \cdots & H_n^{u_1 u_N} \\ H_n^{u_2 x} & H_n^{u_2 u_1} & H_n^{u_2 u_2} & \cdots & H_n^{u_2 u_N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ H_n^{u_N x} & H_n^{u_N u_1} & H_n^{u_N u_2} & \cdots & H_n^{u_N u_N} \end{bmatrix} \begin{bmatrix} \delta\hat{x} \\ \delta u_{1,:} \\ \delta u_{2,:} \\ \vdots \\ \delta u_{N,:} \end{bmatrix} \tag{A.24d}$$

$$= \begin{bmatrix} F_n^\top & \underbrace{\mathbf{0} \quad \cdots \quad \mathbf{0}}_{n-1} & I & \underbrace{\mathbf{0} \quad \cdots \quad \mathbf{0}}_{N-n} \end{bmatrix} H_n \begin{bmatrix} \delta\hat{x} \\ \delta u_{1,:} \\ \delta u_{2,:} \\ \vdots \\ \delta u_{N,:} \end{bmatrix} \tag{A.24e}$$

Solving the Gradient Condition

It remains to solve the players' actions that set (A.24) equal to zero for all players $n = 1, 2, ..., N$. We are motivated to find a backward recursive form that has linear complexity over the number of steps T.

First we study the structure of the last two multiplicative terms in (A.24)

$$H_n \begin{bmatrix} \delta\hat{x} \\ \delta u_{1,:} \\ \delta u_{2,:} \\ \vdots \\ \delta u_{N,:} \end{bmatrix} = \begin{bmatrix} H_n^{xx} & H_n^{xu_1} & H_n^{xu_2} & \cdots & H_n^{xu_N} \\ H_n^{u_1x} & H_n^{u_1u_1} & H_n^{u_1u_2} & \cdots & H_n^{u_1u_N} \\ H_n^{u_2x} & H_n^{u_2u_1} & H_n^{u_2u_2} & \cdots & H_n^{u_2u_N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ H_n^{u_Nx} & H_n^{u_Nu_1} & H_n^{u_Nu_2} & \cdots & H_n^{u_Nu_N} \end{bmatrix} \begin{bmatrix} \delta\hat{x} \\ \delta u_{1,:} \\ \delta u_{2,:} \\ \vdots \\ \delta u_{N,:} \end{bmatrix} \tag{A.25a}$$

$$= \begin{bmatrix} H_n^{xx}\delta\hat{x} + H_n^{xu_1}\delta u_{1,:} + H_n^{xu_2}\delta u_{2,:} + \cdots + H_n^{xu_N}\delta u_{N,:} \\ \hline H_n^{u_1x}\delta\hat{x} + H_n^{u_1u_1}\delta u_{1,:} + H_n^{u_1u_2}\delta u_{2,:} + \cdots + H_n^{u_1u_N}\delta u_{N,:} \\ \hline H_n^{u_2x}\delta\hat{x} + H_n^{u_2u_1}\delta u_{1,:} + H_n^{u_2u_2}\delta u_{2,:} + \cdots + H_n^{u_2u_N}\delta u_{N,:} \\ \hline \vdots \\ \hline H_n^{u_Nx}\delta\hat{x} + H_n^{u_Nu_1}\delta u_{1,:} + H_n^{u_Nu_2}\delta u_{2,:} + \cdots + H_n^{u_Nu_N}\delta u_{N,:} \end{bmatrix} \tag{A.25b}$$

$$
= \left[\begin{array}{c}
\hat{M}^{xx}_{n,0}\delta\hat{x}_0 + \hat{M}^{xu_1}_{n,0}\delta u_{1,0} + \hat{M}^{xu_2}_{n,0}\delta u_{2,0} + \cdots + \hat{M}^{xu_N}_{n,0}\delta u_{N,0} \\
\hat{M}^{xx}_{n,1}\delta\hat{x}_1 + \hat{M}^{xu_1}_{n,1}\delta u_{1,1} + \hat{M}^{xu_2}_{n,1}\delta u_{2,1} + \cdots + \hat{M}^{xu_N}_{n,1}\delta u_{N,1} \\
\vdots \\
\hat{M}^{xx}_{n,T}\delta\hat{x}_T + \hat{M}^{xu_1}_{n,T}\delta u_{1,T} + \hat{M}^{xu_2}_{n,T}\delta u_{2,T} + \cdots + \hat{M}^{xu_N}_{n,T}\delta u_{N,T} \\
\hline
\hat{M}^{u_1x}_{n,0}\delta\hat{x}_0 + \hat{M}^{u_1u_1}_{n,0}\delta u_{1,0} + \hat{M}^{u_1u_2}_{n,0}\delta u_{2,0} + \cdots + \hat{M}^{u_1u_N}_{n,0}\delta u_{N,0} \\
\hat{M}^{u_1x}_{n,1}\delta\hat{x}_1 + \hat{M}^{u_1u_1}_{n,1}\delta u_{1,1} + \hat{M}^{u_1u_2}_{n,1}\delta u_{2,1} + \cdots + \hat{M}^{u_1u_N}_{n,1}\delta u_{N,1} \\
\vdots \\
\hat{M}^{u_1x}_{n,T}\delta\hat{x}_T + \hat{M}^{u_1u_1}_{n,T}\delta u_{1,T} + \hat{M}^{u_1u_2}_{n,T}\delta u_{2,T} + \cdots + \hat{M}^{u_1u_N}_{n,T}\delta u_{N,T} \\
\hline
\hat{M}^{u_2x}_{n,0}\delta\hat{x}_0 + \hat{M}^{u_2u_1}_{n,0}\delta u_{1,0} + \hat{M}^{u_2u_2}_{n,0}\delta u_{2,0} + \cdots + \hat{M}^{u_2u_N}_{n,0}\delta u_{N,0} \\
\hat{M}^{u_2x}_{n,1}\delta\hat{x}_1 + \hat{M}^{u_2u_1}_{n,1}\delta u_{1,1} + \hat{M}^{u_2u_2}_{n,1}\delta u_{2,1} + \cdots + \hat{M}^{u_2u_N}_{n,1}\delta u_{N,1} \\
\vdots \\
\hat{M}^{u_2x}_{n,T}\delta\hat{x}_T + \hat{M}^{u_2u_1}_{n,T}\delta u_{1,T} + \hat{M}^{u_2u_2}_{n,T}\delta u_{2,T} + \cdots + \hat{M}^{u_2u_N}_{n,T}\delta u_{N,T} \\
\hline
\vdots \\
\hline
\hat{M}^{u_Nx}_{n,0}\delta\hat{x}_0 + \hat{M}^{u_Nu_1}_{n,0}\delta u_{1,0} + \hat{M}^{u_Nu_2}_{n,0}\delta u_{2,0} + \cdots + \hat{M}^{u_Nu_N}_{n,0}\delta u_{N,0} \\
\hat{M}^{u_Nx}_{n,1}\delta\hat{x}_1 + \hat{M}^{u_Nu_1}_{n,1}\delta u_{1,1} + \hat{M}^{u_Nu_2}_{n,1}\delta u_{2,1} + \cdots + \hat{M}^{u_Nu_N}_{n,1}\delta u_{N,1} \\
\vdots \\
\hat{M}^{u_Nx}_{n,T}\delta\hat{x}_T + \hat{M}^{u_Nu_1}_{n,T}\delta u_{1,T} + \hat{M}^{u_Nu_2}_{n,T}\delta u_{2,T} + \cdots + \hat{M}^{u_Nu_N}_{n,T}\delta u_{N,T}
\end{array}\right] \tag{A.25c}
$$

Therefore the playerwise gradient would be

$$
\frac{\partial \delta J_n(\delta\hat{x}_0, \delta u)}{\partial \delta u_{n,:}} = F_n^\top \begin{bmatrix}
\hat{M}^{xx}_{n,0}\delta\hat{x}_0 + \hat{M}^{xu_1}_{n,0}\delta u_{1,0} + \hat{M}^{xu_2}_{n,0}\delta u_{2,0} + \cdots + \hat{M}^{xu_N}_{n,0}\delta u_{N,0} \\
\hat{M}^{xx}_{n,1}\delta\hat{x}_1 + \hat{M}^{xu_1}_{n,1}\delta u_{1,1} + \hat{M}^{xu_2}_{n,1}\delta u_{2,1} + \cdots + \hat{M}^{xu_N}_{n,1}\delta u_{N,1} \\
\vdots \\
\hat{M}^{xx}_{n,T}\delta\hat{x}_T + \hat{M}^{xu_1}_{n,T}\delta u_{1,T} + \hat{M}^{xu_2}_{n,T}\delta u_{2,T} + \cdots + \hat{M}^{xu_N}_{n,T}\delta u_{N,T}
\end{bmatrix} +
$$

$$
\begin{bmatrix}
\hat{M}^{u_nx}_{n,0}\delta\hat{x}_0 + \hat{M}^{u_nu_1}_{n,0}\delta u_{1,0} + \hat{M}^{u_nu_2}_{n,0}\delta u_{2,0} + \cdots + \hat{M}^{u_nu_N}_{n,0}\delta u_{N,0} \\
\hat{M}^{u_nx}_{n,1}\delta\hat{x}_1 + \hat{M}^{u_nu_1}_{n,1}\delta u_{1,1} + \hat{M}^{u_nu_2}_{n,1}\delta u_{2,1} + \cdots + \hat{M}^{u_nu_N}_{n,1}\delta u_{N,1} \\
\vdots \\
\hat{M}^{u_nx}_{n,T}\delta\hat{x}_T + \hat{M}^{u_nu_1}_{n,T}\delta u_{1,T} + \hat{M}^{u_nu_2}_{n,T}\delta u_{2,T} + \cdots + \hat{M}^{u_nu_N}_{n,T}\delta u_{N,T}
\end{bmatrix} \tag{A.26a}
$$

$$
\begin{aligned}
&= F_n^\top \begin{bmatrix} \hat{M}_{n,0}^{xx}\delta\hat{x}_0 + \hat{M}_{n,0}^{xu}\delta u_{:,0} \\ \hat{M}_{n,1}^{xx}\delta\hat{x}_1 + \hat{M}_{n,0}^{xu}\delta u_{:,1} \\ \vdots \\ \hat{M}_{n,N}^{xx}\delta\hat{x}_T + \hat{M}_{n,N}^{xu}\delta u_{:,T} \end{bmatrix} + \begin{bmatrix} \hat{M}_{n,0}^{u_n x}\delta\hat{x}_0 + \hat{M}_{n,0}^{u_n u}\delta u_{:,0} \\ \hat{M}_{n,1}^{u_n x}\delta\hat{x}_1 + \hat{M}_{n,1}^{u_n u}\delta u_{:,1} \\ \vdots \\ \hat{M}_{n,N}^{u_n x}\delta\hat{x}_T + \hat{M}_{n,N}^{u_n u}\delta u_{:,T} \end{bmatrix} && \text{(A.26b)} \\
&= F_n^\top \hat{\mathcal{M}}_n^x + \hat{\mathcal{M}}_n^{u_n} && \text{(A.26c)}
\end{aligned}
$$

where we define the shorthand notation $\hat{\mathcal{M}}_n^x, \hat{\mathcal{M}}_n^{u_n}$ for the two bigger matrices, because we will need to refer to them repeatedly later.

According to the definition of F_n (A.17c), we can write its transpose in detail.

$$
F_n^\top = \begin{bmatrix}
0 & \hat{B}_{n,0}^\top & \hat{B}_{n,0}^\top\hat{A}_1^\top & \hat{B}_{n,0}^\top\hat{A}_1^\top\hat{A}_2^\top & \cdots & \hat{B}_{n,0}^\top(\Pi_{k=1}^{T-2}\hat{A}_k)^\top & \hat{B}_{n,0}^\top(\Pi_{k=1}^{T-1}\hat{A}_k)^\top \\
0 & 0 & \hat{B}_{n,1}^\top & \hat{B}_{n,1}^\top\hat{A}_2^\top & \cdots & \hat{B}_{n,1}^\top(\Pi_{k=2}^{T-2}\hat{A}_k)^\top & \hat{B}_{n,1}^\top(\Pi_{k=2}^{T-1}\hat{A}_k)^\top \\
0 & 0 & 0 & \hat{B}_{n,2}^\top & \cdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \hat{B}_{n,T-3}^\top\hat{A}_{T-2}^\top & \hat{B}_{n,T-3}^\top\hat{A}_{T-2}^\top\hat{A}_{T-1}^\top \\
0 & 0 & 0 & 0 & \ddots & \hat{B}_{n,T-2}^\top & \hat{B}_{n,T-2}^\top\hat{A}_{T-1}^\top \\
0 & 0 & 0 & 0 & \cdots & 0 & \hat{B}_{n,T-1}^\top \\
0 & 0 & 0 & 0 & \cdots & 0 & 0
\end{bmatrix} \tag{A.27}
$$

We give the first term a shorthand name because it will be used extensively.

We solve the OLNE $u_{:,k}^\star$ backward in time for $k = T, T-1, ..., 0$. Stacking the last row of (A.26) for all players and set it to zero, we have

$$
\begin{bmatrix}
\hat{M}_{1,T}^{u_1u_1} & \hat{M}_{1,T}^{u_1u_2} & \cdots & \hat{M}_{1,T}^{u_1u_N} \\
\hat{M}_{2,T}^{u_2u_1} & \hat{M}_{2,T}^{u_2u_2} & \cdots & \hat{M}_{2,T}^{u_2u_N} \\
\vdots & \vdots & \ddots & \vdots \\
\hat{M}_{N,T}^{u_Nu_1} & \hat{M}_{N,T}^{u_Nu_2} & \cdots & \hat{M}_{N,T}^{u_Nu_N}
\end{bmatrix}
\begin{bmatrix} \delta u_{1,T} \\ \delta u_{2,T} \\ \vdots \\ \delta u_{N,T} \end{bmatrix}
+ \begin{bmatrix} \hat{M}_{1,T}^{u_1x} \\ \hat{M}_{2,T}^{u_2x} \\ \vdots \\ \hat{M}_{N,T}^{u_Nx} \end{bmatrix} \delta\hat{x}_T = 0 \tag{A.28}
$$

So the last OLNE action in terms of time would be

$$
\delta u_{:,T} = \hat{K}_T\delta\hat{x}_T = \begin{bmatrix} \hat{K}_{1,T} \\ \hat{K}_{2,T} \\ \vdots \\ \hat{K}_{N,T} \end{bmatrix} \delta\hat{x}_T = -\begin{bmatrix}
\hat{M}_{1,T}^{u_1u_1} & \hat{M}_{1,T}^{u_1u_2} & \cdots & \hat{M}_{1,T}^{u_1u_N} \\
\hat{M}_{2,T}^{u_2u_1} & \hat{M}_{2,T}^{u_2u_2} & \cdots & \hat{M}_{2,T}^{u_2u_N} \\
\vdots & \vdots & \ddots & \vdots \\
\hat{M}_{N,T}^{u_Nu_1} & \hat{M}_{N,T}^{u_Nu_2} & \cdots & \hat{M}_{N,T}^{u_Nu_N}
\end{bmatrix}^{-1}
\begin{bmatrix} \hat{M}_{1,T}^{u_1x} \\ \hat{M}_{2,T}^{u_2x} \\ \vdots \\ \hat{M}_{N,T}^{u_Nx} \end{bmatrix} \delta\hat{x}_T \tag{A.29}
$$

Now we substitute $\delta u_{:,T} = \hat{K}_T\delta\hat{x}_T$ to the second to the last row of each (A.26).

$$
\hat{B}_{n,T-1}^\top(\hat{M}_{n,T}^{xx}\delta\hat{x}_T + \hat{M}_{n,T}^{xu_1}\delta u_{1,T} + \hat{M}_{n,T}^{xu_2}\delta u_{2,T} + \cdots + \hat{M}_{n,T}^{xu_N}\delta u_{N,T}) +
$$

$$\hat{M}^{u_n x}_{n,T-1}\delta\hat{x}_{T-1}+\hat{M}^{u_n u_1}_{n,T-1}\delta u_{1,T-1}+\hat{M}^{u_n u_2}_{n,T-1}\delta u_{2,T-1}+\cdots+\hat{M}^{u_n u_N}_{n,T-1}\delta u_{N,T-1} \tag{A.30a}$$

$$=\hat{B}^{\top}_{n,T-1}(\hat{M}^{xx}_{n,T}+\hat{M}^{xu}_{n,T}\hat{K}_T)(\hat{A}_{T-1}\delta\hat{x}_{T-1}+\hat{B}_{T-1}\delta u_{:,T-1})+$$

$$\hat{M}^{u_n x}_{n,T-1}\delta\hat{x}_{T-1}+\hat{M}^{u_n u_1}_{n,T-1}\delta u_{1,T-1}+\hat{M}^{u_n u_2}_{n,T-1}\delta u_{2,T-1}+\cdots+\hat{M}^{u_n u_N}_{n,T-1}\delta u_{N,T-1} \tag{A.30b}$$

$$=\hat{B}^{\top}_{n,T-1}\hat{S}_{n,T}(\hat{A}_{T-1}\delta\hat{x}_{T-1}+\hat{B}_{T-1}\delta u_{:,T-1})+$$

$$\hat{M}^{u_n x}_{n,T-1}\delta\hat{x}_{T-1}+\hat{M}^{u_n u_1}_{n,T-1}\delta u_{1,T-1}+\hat{M}^{u_n u_2}_{n,T-1}\delta u_{2,T-1}+\cdots+\hat{M}^{u_n u_N}_{n,T-1}\delta u_{N,T-1} \tag{A.30c}$$

where we let $\hat{S}_{n,T}=\hat{M}^{xx}_{n,T}+\hat{M}^{xu}_{n,T}\hat{K}_T$ and

$$\hat{M}^{xu}_{n,k}=\begin{bmatrix}\hat{M}^{xu_1}_{n,k} & \hat{M}^{xu_2}_{n,k} & \cdots & \hat{M}^{xu_N}_{n,k}\end{bmatrix},\quad k=0,1,...,T \tag{A.31}$$

We can stack all players' gradient.

$$\left(\hat{S}^{BB}_T+\begin{bmatrix}\hat{M}^{u_1u_1}_{1,T-1} & \hat{M}^{u_1u_2}_{1,T-1} & \cdots & \hat{M}^{u_1u_N}_{1,T-1}\\ \hat{M}^{u_2u_1}_{2,T-1} & \hat{M}^{u_2u_2}_{2,T-1} & \cdots & \hat{M}^{u_2u_N}_{2,T-1}\\ \vdots & \vdots & \ddots & \vdots\\ \hat{M}^{u_Nu_1}_{N,T-1} & \hat{M}^{u_Nu_2}_{N,T-1} & \cdots & \hat{M}^{u_Nu_N}_{N,T-1}\end{bmatrix}\right)\begin{bmatrix}\delta u_{1,T-1}\\ \delta u_{2,T-1}\\ \vdots\\ \delta u_{N,T-1}\end{bmatrix}+$$

$$\left(\hat{S}^{BA}_T+\begin{bmatrix}\hat{M}^{u_1x}_{1,T-1}\\ \hat{M}^{u_2x}_{2,T-1}\\ \vdots\\ \hat{M}^{u_Nx}_{N,T-1}\end{bmatrix}\right)\delta\hat{x}_{T-1}=0 \tag{A.32}$$

where $\hat{S}^{BB}_T$ and $\hat{S}^{BA}_T$ collect the terms related to $\hat{S}_{n,T}$ for all players and the superscript indicates the left and right multiplication by either $\hat{A}_{T-1}$ or $\hat{B}_{T-1}$.

$$\hat{S}^{BB}_T=\begin{bmatrix}\hat{B}^{\top}_{1,T-1}\hat{S}_{1,T}\hat{B}_{1,T-1} & \hat{B}^{\top}_{1,T-1}\hat{S}_{1,T}\hat{B}_{2,T-1} & \cdots & \hat{B}^{\top}_{1,T-1}\hat{S}_{1,T}\hat{B}_{N,T-1}\\ \hat{B}^{\top}_{2,T-1}\hat{S}_{2,T}\hat{B}_{1,T-1} & \hat{B}^{\top}_{2,T-1}\hat{S}_{2,T}\hat{B}_{2,T-1} & \cdots & \hat{B}^{\top}_{2,T-1}\hat{S}_{2,T}\hat{B}_{N,T-1}\\ \vdots & \vdots & \ddots & \vdots\\ \hat{B}^{\top}_{N,T-1}\hat{S}_{N,T}\hat{B}_{1,T-1} & \hat{B}^{\top}_{N,T-1}\hat{S}_{N,T}\hat{B}_{2,T-1} & \cdots & \hat{B}^{\top}_{N,T-1}\hat{S}_{N,T}\hat{B}_{N,T-1}\end{bmatrix} \tag{A.33a}$$

$$\hat{S}^{BA}_T=\begin{bmatrix}\hat{B}^{\top}_{1,T-1}\hat{S}_{1,T}\hat{A}_{T-1}\\ \hat{B}^{\top}_{2,T-1}\hat{S}_{2,T}\hat{A}_{T-1}\\ \vdots\\ \hat{B}^{\top}_{N,T-1}\hat{S}_{N,T}\hat{A}_{T-1}\end{bmatrix} \tag{A.33b}$$

So the second to last action would be

$$\delta u^{\star}_{:,T-1}=\hat{K}_{T-1}\delta x_{T-1} \tag{A.34a}$$

$$\hat{K}_{T-1} = -\left(\hat{S}_T^{BB} + \begin{bmatrix} \hat{M}_{1,T-1}^{u_1u_1} & \hat{M}_{1,T-1}^{u_1u_2} & \cdots & \hat{M}_{1,T-1}^{u_1u_N} \\ \hat{M}_{2,T-1}^{u_2u_1} & \hat{M}_{2,T-1}^{u_2u_2} & \cdots & \hat{M}_{2,T-1}^{u_2u_N} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{M}_{N,T-1}^{u_Nu_1} & \hat{M}_{N,T-1}^{u_Nu_2} & \cdots & \hat{M}_{N,T-1}^{u_Nu_N} \end{bmatrix}\right)^{-1} \left(\hat{S}_T^{BA} + \begin{bmatrix} \hat{M}_{1,T-1}^{u_1x} \\ \hat{M}_{2,T-1}^{u_2x} \\ \vdots \\ \hat{M}_{N,T-1}^{u_Nx} \end{bmatrix}\right) \tag{A.34b}$$

Following a similar pattern, we walk back one more step. The third to the last row of (A.26) is

$$\begin{aligned}
&\hat{B}_{n,T-2}^\top(\hat{M}_{n,T-1}^{xx}\delta\hat{x}_{T-1} + \hat{M}_{n,T-1}^{xu_1}\delta u_{1,T-1} + \hat{M}_{n,T-1}^{xu_2}\delta u_{2,T-1} + \cdots + \hat{M}_{n,T-1}^{xu_N}\delta u_{N,T-1}) + \\
&\hat{B}_{n,T-2}^\top\hat{A}_{T-1}^\top(\hat{M}_{n,T}^{xx}\delta\hat{x}_T + \hat{M}_{n,T}^{xu_1}\delta u_{1,T} + \hat{M}_{n,T}^{xu_2}\delta u_{2,T} + \cdots + \hat{M}_{n,T}^{xu_N}\delta u_{N,T}) + \\
&\hat{M}_{n,T-2}^{u_nx}\delta\hat{x}_{T-2} + \hat{M}_{n,T-2}^{u_nu_1}\delta u_{1,T-2} + \hat{M}_{n,T-2}^{u_nu_2}\delta u_{2,T-2} + \cdots + \hat{M}_{n,T-2}^{u_nu_N}\delta u_{N,T-2} && \text{(A.35a)} \\
=&\hat{B}_{n,T-2}^\top(\hat{M}_{n,T-1}^{xx} + \hat{M}_{n,T-1}^{xu}\hat{K}_{T-1})\delta\hat{x}_{T-1} + \hat{B}_{n,T-2}^\top\hat{A}_{T-1}^\top\hat{S}_T(\hat{A}_{T-1}\delta\hat{x}_{T-1} + \hat{B}_{T-1}\delta u_{:,T-1}) + \\
&\hat{M}_{n,T-2}^{u_nx}\delta\hat{x}_{T-2} + \hat{M}_{n,T-2}^{u_nu_1}\delta u_{1,T-2} + \hat{M}_{n,T-2}^{u_nu_2}\delta u_{2,T-2} + \cdots + \hat{M}_{n,T-2}^{u_nu_N}\delta u_{N,T-2} && \text{(A.35b)} \\
=&\hat{B}_{n,T-2}^\top[\hat{M}_{n,T-1}^{xx} + \hat{M}_{n,T-1}^{xu}\hat{K}_{T-1} + \hat{A}_{T-1}^\top\hat{S}_T(\hat{A}_{T-1} + \hat{B}_{T-1}\hat{K}_{T-1})]\delta\hat{x}_{T-1} + \\
&\hat{M}_{n,T-2}^{u_nx}\delta\hat{x}_{T-2} + \hat{M}_{n,T-2}^{u_nu_1}\delta u_{1,T-2} + \hat{M}_{n,T-2}^{u_nu_2}\delta u_{2,T-2} + \cdots + \hat{M}_{n,T-2}^{u_nu_N}\delta u_{N,T-2} && \text{(A.35c)} \\
=&\hat{B}_{n,T-2}^\top\hat{S}_{n,T-1}(\hat{A}_{T-2}\delta\hat{x}_{T-2} + \hat{B}_{T-2}\delta\hat{u}_{:,T-2}) + \\
&\hat{M}_{n,T-2}^{u_nx}\delta\hat{x}_{T-2} + \hat{M}_{n,T-2}^{u_nu_1}\delta u_{1,T-2} + \hat{M}_{n,T-2}^{u_nu_2}\delta u_{2,T-2} + \cdots + \hat{M}_{n,T-2}^{u_nu_N}\delta u_{N,T-2} && \text{(A.35d)}
\end{aligned}$$

where we let $\hat{S}_{n,T-1} = \hat{M}_{n,T-1}^{xx} + \hat{M}_{n,T-1}^{xu}\hat{K}_{T-1} + \hat{A}_{T-1}^\top\hat{S}_T(\hat{A}_{T-1} + \hat{B}_{T-1}\hat{K}_{T-1})$ and we have the same form as (A.30). Stacking all of gradients for $n = 1, 2, ..., N$ and setting it to zero, we can solve for $u_{:,T-2}^\star$.

$$\begin{aligned}
&\left(\hat{S}_{T-1}^{BB} + \begin{bmatrix} \hat{M}_{1,T-2}^{u_1u_1} & \hat{M}_{1,T-2}^{u_1u_2} & \cdots & \hat{M}_{1,T-2}^{u_1u_N} \\ \hat{M}_{2,T-2}^{u_2u_1} & \hat{M}_{2,T-2}^{u_2u_2} & \cdots & \hat{M}_{2,T-2}^{u_2u_N} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{M}_{N,T-2}^{u_Nu_1} & \hat{M}_{N,T-2}^{u_Nu_2} & \cdots & \hat{M}_{N,T-2}^{u_Nu_N} \end{bmatrix}\right) \begin{bmatrix} \delta u_{1,T-2} \\ \delta u_{2,T-2} \\ \vdots \\ \delta u_{N,T-2} \end{bmatrix} + \\
&\left(\hat{S}_{T-1}^{BA} + \begin{bmatrix} \hat{M}_{1,T-2}^{u_1x} \\ \hat{M}_{2,T-2}^{u_2x} \\ \vdots \\ \hat{M}_{N,T-2}^{u_Nx} \end{bmatrix}\right)\delta\hat{x}_{T-2} = 0 \qquad \text{(A.36)}
\end{aligned}$$

where

$$\hat{S}^{BB}_{T-1} = \begin{bmatrix} \hat{B}^\top_{1,T-2}\hat{S}_{1,T-1}\hat{B}_{1,T-2} & \hat{B}^\top_{1,T-2}\hat{S}_{1,T-1}\hat{B}_{2,T-2} & \cdots & \hat{B}^\top_{1,T-2}\hat{S}_{1,T-1}\hat{B}_{N,T-2} \\ \hat{B}^\top_{2,T-2}\hat{S}_{2,T-1}\hat{B}_{1,T-2} & \hat{B}^\top_{2,T-2}\hat{S}_{2,T-1}\hat{B}_{2,T-2} & \cdots & \hat{B}^\top_{2,T-2}\hat{S}_{2,T-1}\hat{B}_{N,T-2} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{B}^\top_{N,T-2}\hat{S}_{N,T-1}\hat{B}_{1,T-2} & \hat{B}^\top_{N,T-2}\hat{S}_{N,T-1}\hat{B}_{2,T-2} & \cdots & \hat{B}^\top_{N,T-2}\hat{S}_{N,T-1}\hat{B}_{N,T-2} \end{bmatrix} \tag{A.37a}$$

$$\hat{S}^{BA}_{T-1} = \begin{bmatrix} \hat{B}^\top_{1,T-2}\hat{S}_{1,T-1}\hat{A}_{T-2} \\ \hat{B}^\top_{2,T-2}\hat{S}_{2,T-1}\hat{A}_{T-2} \\ \vdots \\ \hat{B}^\top_{N,T-2}\hat{S}_{N,T-1}\hat{A}_{T-2} \end{bmatrix} \tag{A.37b}$$

which has the same form as (A.32) and we can solve for $\hat{K}_{T-2}$ in the same way. The same pattern can be found for all $k = T, T-1, ..., 0$. which completes the recursion.

We proceed to show that the same procedure can be performed backward for all $k = T, T-1, ..., 0$. It boils down to showing that the kth row of (A.26) is

$$\{F_n^\top \hat{\mathcal{M}}^x_n + \hat{\mathcal{M}}^{u_n}_n\}^k = \hat{B}^\top_{n,k-1}\hat{S}_{n,k}(\hat{A}_{k-1}\delta\hat{x}_{k-1} + \hat{B}_{k-1}\delta u_{:,k-1}) + \hat{M}^{u_n x}_{n,k-1}\delta\hat{x}_{k-1} + \hat{M}^{u_n u}_{n,k-1}\delta u_{:,k-1} \tag{A.38}$$

given the backward recursion of $\hat{S}_{n,k}$ and gain

$$\hat{S}_{n,k} = \hat{M}^{xx}_{n,k} + \hat{M}^{xu}_{n,k}\hat{K}_k + \hat{A}^\top_k \hat{S}_{n,k+1}(\hat{A}_k + \hat{B}_k\hat{K}_k), \quad k = T-1, T-2, ..., 0 \tag{A.39a}$$

$$\delta u_{:,k} = \hat{K}_k \delta\hat{x}_{:,k} \tag{A.39b}$$

We prove this by induction. (A.38) is clearly true for $k = T-1$ and $T-2$ as we have shown. Because $\{\hat{\mathcal{M}}^{u_n}_n\}^k = \hat{M}^{u_n x}_{n,k-1}\delta\hat{x}_{k-1} + \hat{M}^{u_n u_1}_{n,k-1}\delta u_{1,k-1} + \hat{M}^{u_n u_2}_{n,k-1}\delta u_{2,k-1} + \cdots + \hat{M}^{u_n u_N}_{n,k-1}\delta u_{N,k-1}$, which is true for all k, if suffices to show the first part of (A.38)

$$\{F_n^\top \hat{\mathcal{M}}^x_n\}^k = \hat{B}^\top_{n,k-1}\hat{S}_{n,k}(\hat{A}_{k-1}\delta\hat{x}_{k-1} + \hat{B}_{k-1}\delta u_{:,k-1}) \tag{A.40}$$

Assume that (A.40) is true for $k+1$, i.e.,

$$\{F_n^\top \hat{\mathcal{M}}^x_n\}^{k+1} = \hat{B}^\top_{n,k}\hat{S}_{n,k+1}(\hat{A}_k\delta\hat{x}_k + \hat{B}_k\delta u_{:,k}) \tag{A.41}$$

Based on (A.41), we would like to show the kth row of $F_n^\top \hat{\mathcal{M}}^x_n$ in (A.26) is

$$\{F_n^\top \hat{\mathcal{M}}^x_n\}^k = \hat{B}^\top_{n,k-1}\hat{S}_{n,k}(\hat{A}_{k-1}\delta\hat{x}_{k-1} + \hat{B}_{k-1}\delta u_{:,k-1}) \tag{A.42}$$

We layout the kth row of $F_n^\top \hat{\mathcal{M}}_n^x$ and compare it to the $(k+1)$th.

$$\{F_n^\top \hat{\mathcal{M}}_n^x\}^k = \hat{B}_{n,k-1}^\top \hat{A}_k \sum_{j=k}^{T-1} (\Pi_{l=k+1}^j \hat{A}_l)^\top (\hat{M}_{n,j+1}^{xx} \delta\hat{x}_{j+1} + \hat{M}_{n,j+1}^{xu} \delta u_{:,j+1})$$
$$+ \hat{B}_{n,k-1}^\top (\hat{M}_{n,k}^{xx} \delta\hat{x}_k + \hat{M}_{n,k}^{xu} \delta u_{:,k}) \tag{A.43a}$$
$$\{F_n^\top \hat{\mathcal{M}}_n^x\}^{k+1} = \hat{B}_{n,k}^\top \sum_{j=k}^{T-1} (\Pi_{l=k+1}^j \hat{A}_l)^\top (\hat{M}_{n,j+1}^{xx} \delta\hat{x}_{j+1} + \hat{M}_{n,j+1}^{xu} \delta u_{:,j+1}) \tag{A.43b}$$
$$= \hat{B}_{n,k}^\top \hat{S}_{n,k+1} (\hat{A}_k \delta\hat{x}_k + \hat{B}_k \delta u_{:,k}) \tag{A.43c}$$

We can see the summation that appears in both rows is equal to

$$\sum_{j=k}^{T-1} (\Pi_{l=k+1}^j \hat{A}_l)^\top (\hat{M}_{n,j+1}^{xx} \delta\hat{x}_{j+1} + \hat{M}_{n,j+1}^{xu} \delta u_{:,j+1}) = \hat{S}_{n,k+1} (\hat{A}_k \delta\hat{x}_k + \hat{B}_k \delta u_{:,k}) \tag{A.44}$$

Therefore the kth row is

$$\{F_n^\top \hat{\mathcal{M}}_n^x\}^k = \hat{B}_{n,k-1}^\top \hat{A}_k^\top \hat{S}_{n,k+1} (\hat{A}_k \delta\hat{x}_k + \hat{B}_k \delta u_{:,k}) + \hat{B}_{n,k-1}^\top (\hat{M}_{n,k}^{xx} \delta\hat{x}_k + \hat{M}_{n,k}^{xu} \delta u_{:,k}) \tag{A.45a}$$
$$= \hat{B}_{n,k-1}^\top \hat{A}_k^\top \hat{S}_{n,k+1} (\hat{A}_k + \hat{B}_k \hat{K}_k) \delta\hat{x}_k + \hat{B}_{n,k-1}^\top (\hat{M}_{n,k}^{xx} + \hat{M}_{n,k}^{xu} \hat{K}_k) \delta\hat{x}_k \tag{A.45b}$$
$$= \hat{B}_{n,k-1}^\top [\hat{A}_k^\top \hat{S}_{n,k+1} (\hat{A}_k + \hat{B}_k \hat{K}_k) + \hat{M}_{n,k}^{xx} + \hat{M}_{n,k}^{xu} \hat{K}_k] \delta\hat{x}_k \tag{A.45c}$$
$$= \hat{B}_{n,k-1}^\top \hat{S}_{n,k} (\hat{A}_{k-1} \delta\hat{x}_{k-1} + \hat{B}_{k-1} \delta u_{:,k-1}) \tag{A.45d}$$

which matches that predicted by (A.38) for k. Therefore, given the backward recursion and gain (A.39a), (A.38) is true, and all the OLNE actions $u_{:,k}^\star$ can be computed in a fashion same as (A.34).

A.4 Proof of Lemma 3

Substituting the approximated dynamic game (2.21) to the Bellman equation (2.6) leads us to,

$$\hat{V}_{n,T+1}^{\bar{u}} (\delta x_{T+1}, \Delta x_{T+1}) = 0 \tag{A.46a}$$
$$\hat{Q}_{n,k}^{\bar{u}} (\delta x_k, \Delta x_k, \delta u_{:,k}) =$$
$$\frac{1}{2} \left(\begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top M_{n,k} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix} + M_{n,k}^{1x} \Delta x_k \right)$$

$$+\hat{V}^{\bar{u}}_{n,k+1}(A_k\delta x_k + B_k\delta u_{:,k}, A_k\Delta x_k + R_k(\delta x_k, \delta u_{:,k})) \tag{A.46b}$$

$$\hat{V}^{\bar{u}}_{n,k}(\delta x_k, \Delta x_k) = \min_{\delta u_{n,k}} \hat{Q}^{\bar{u}}_{n,k}(\delta x_k, \Delta x_k, \delta u_{:,k}). \tag{A.46c}$$

Note that (A.46c) defines a static quadratic game and $\hat{V}^{\bar{u}}_{n,k}(\delta x_k, \Delta x_k)$ is found by solving the game and substituting the solution back to $\hat{Q}^{\bar{u}}_{n,k}(\delta x_k, \Delta x_k, \delta u_{:,k})$.

Solving the equilibrium strategy and $\hat{V}^{\bar{u}}_{n,k}(\delta x_k, \Delta x_k)$ based on $\hat{Q}^{\bar{u}}_{n,k}(\delta x_k, \Delta x_k, \delta u_{:,k})$ is the same as how we arrived at (2.35) and (2.34g), since the extra terms of Δx_k are not coupled with $\delta u_{:,k}$ and other terms are of the exact same form. The stepping back in time of $\hat{Q}^{\bar{u}}_{k}(\delta x_k, \Delta x_k, \delta u_{:,k})$ is acheived by substituting (2.11d) and (2.11e) into (2.23a), which is slightly different because of the extra terms related to Δx_k.

$$\hat{Q}^{\bar{u}}_{n,k}(\delta x_k, \Delta x_k, \delta u_{:,k}) \tag{A.47a}$$

$$= \frac{1}{2}\left(\begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top M_{n,k} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix} + M^{1k}_{n,k}\Delta x_k\right)$$

$$+\hat{V}^{\bar{u}}_{n,k+1}(A_k\delta x_k + B_k\delta u_k, A_k\Delta x_k + R_k(\delta x_k, \delta u_{:,k})) \tag{A.47b}$$

$$= \frac{1}{2}\left(\begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top M_{n,k} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}\right) +$$

$$\frac{1}{2}\begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top \begin{bmatrix} S^{11}_{n,k+1} & S^{1x}_{n,k+1}A_k & S^{1x}_{n,k+1}B_k \\ A_k^\top S^{x1}_{n,k+1} & A_k^\top S^{xx}_{n,k+1}A_k & A_k^\top S^{xx}_{n,k+1}B_k \\ B_k^\top S^{x1}_{n,k+1} & B_k^\top S^{xx}_{n,k+1}A_k & B_k^\top S^{xx}_{n,k+1}B_k \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_{:,k} \end{bmatrix}$$

$$+\frac{1}{2}\left((M^{1k}_{n,k} + \Omega_{n,k+1}A_k)\Delta x_k\right)$$

$$+\frac{1}{2}\left(\begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix}^\top D_{n,k} \begin{bmatrix} \delta x_k \\ \delta u_{:,k} \end{bmatrix}\right) \tag{A.47c}$$

So (2.25a), (2.25b) and (2.25c) are true.

The proof of (2.24) can be found in the proof of Lemma 4.

A.5 Approximated Parametric Optimization Lemma

Lemma 9. *Given an unconstrained optimization problem with a differentiable objective*

$$\min_x f(x,p) \tag{A.48}$$

According to the implicit function theorem, there exists a feedback policy $x = \phi(p)$ that solves the necessary condition of the optimization problem

$$\frac{\partial f(x,p)}{\partial x} = 0 \tag{A.49}$$

Furthermore, there exists a feedback policy $\delta x = \bar{x} + K\delta p$ in the neighborhood of $\bar{p}$ where $\varepsilon = \|\delta p\|$ is small that solves

$$\min_x \ \text{quad}(f(x,p))|_{\bar{x},\bar{p}} \tag{A.50}$$

where $\bar{x} = \phi(\bar{p})$ and $\frac{\partial f(\bar{x},\bar{p})}{\partial x} = 0$. The feedback policy approximates the true solution locally well in the sense

$$f(\bar{x} + K\delta p, \bar{p} + \delta p) = f(\phi(p), p) + O(\varepsilon^2) \tag{A.51}$$

Proof. We consider x and p values in the neighborhood of $\bar{x}$ and $\bar{p}$, define $x = \bar{x} + \delta x$, $p = \bar{p} + \delta p$ and $\|\delta p\| = \varepsilon$. All derivatives in this proof are evaluated at $\bar{x}$ and $\bar{p}$. Expand $x = \phi(p)$

$$\bar{x} + \delta x = \phi(\bar{p}) + \frac{\partial \phi}{\partial p}\delta p + O(\varepsilon^2) \tag{A.52a}$$

$$\delta x = \frac{\partial \phi}{\partial p}\delta p + O(\varepsilon^2) = O(\varepsilon) \tag{A.52b}$$

The quadratic approximation is expanded as

$$\begin{aligned} &\text{quad}(f(x,p))_{\bar{x},\bar{p}} = f(\bar{x},\bar{p}) + \\ &\begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial p} \end{bmatrix}\Big|_{\bar{x},\bar{p}} \begin{bmatrix} \delta x \\ \delta p \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \delta x \\ \delta p \end{bmatrix}^\top \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial p} \\ \frac{\partial^2 f}{\partial p \partial x} & \frac{\partial^2 f}{\partial p^2} \end{bmatrix}\Bigg|_{\bar{x},\bar{p}} \begin{bmatrix} \delta x \\ \delta p \end{bmatrix} \end{aligned} \tag{A.53}$$

It can be seen that the δx minimizes the approximation is

$$\delta x = -\left(\frac{\partial^2 f}{\partial x^2}\right)^{-1} \frac{\partial^2 f}{\partial x \partial p}\delta p - \left(\frac{\partial^2 f}{\partial x^2}\right)^{-1} \frac{\partial f}{\partial x} = K\delta p + s \tag{A.54}$$

where all derivatives are evaluated at $\bar{x}$, $\bar{p}$. Because $\frac{\partial f(\bar{x},\bar{p})}{\partial x} = 0$, the constant term s is zero. Therefore $\delta x = K\delta p$ minimizes the quadratic approximation.

Two inequalities come naturally from $\phi(p)$ and $K\delta p$ are minimizers of (A.48)(A.53).

$$f(\phi(p),p) \leq f(\bar{x}+K\delta p, \bar{p}+\delta p) \tag{A.55a}$$

$$\text{quad}\,(f(\bar{x}+K\delta p, \bar{p}+\delta p))|_{\bar{x},\bar{p}} \leq \text{quad}\,(f(\phi(p),p))|_{\bar{x},\bar{p}} \tag{A.55b}$$

Taylor series expansion

$$f(\phi(p),p) = f(\phi(\bar{p}) + \frac{\partial \phi}{\partial p}\delta p + O(\varepsilon^2), \bar{p}+\delta p + O(\varepsilon^2)) \tag{A.56a}$$

$$= f(\bar{x},\bar{p}) + \frac{\partial f}{\partial x}\frac{\partial \phi}{\partial p}\delta p + \frac{\partial f}{\partial p}\delta p + O(\varepsilon^2) \tag{A.56b}$$

$$= f(\bar{x},\bar{p}) + \frac{\partial f}{\partial p}\delta p + O(\varepsilon^2) \tag{A.56c}$$

Compare (A.56) and (A.53), and similar comparison can be done for $\text{quad}\,(f(\bar{x}+K\delta p, \bar{p}+\delta p))|_{\bar{x},\bar{p}}$ and $f(\bar{x}+K\delta p, \bar{p}+\delta p)$, we get the closeness results

$$\text{quad}\,(f(\phi(p),p))|_{\bar{x},\bar{p}} = f(\phi(p),p) + O(\varepsilon^2) \tag{A.57a}$$

$$\text{quad}\,(f(\bar{x}+K\delta p, \bar{p}+\delta p))|_{\bar{x},\bar{p}} = f(\bar{x}+K\delta p, \bar{p}+\delta p) + O(\varepsilon^2) \tag{A.57b}$$

Based on (A.57) and (A.55), we have the following inequalities

$$f(\phi(p),p) \leq f(\bar{x}+K\delta p, \bar{p}+\delta p) \tag{A.58a}$$

$$= \text{quad}\,(f(\bar{x}+K\delta p, \bar{p}+\delta p))|_{\bar{x},\bar{p}} + O(\varepsilon^2) \tag{A.58b}$$

$$\leq \text{quad}\,(f(\phi(p),p))|_{\bar{x},\bar{p}} = f(\phi(p),p) + O(\varepsilon^2) \tag{A.58c}$$

$$= f(\phi(p),p) + O(\varepsilon^2) \tag{A.58d}$$

Therefore, $f(\bar{x}+K\delta p, \bar{p}+\delta p) = f(\phi(p),p) + O(\varepsilon^2)$. □

A.6 Approximated Parametric Static Game Lemma

Lemma 10. *Given a static N-player game problem,*

$$\min_{x_n} f_n(x,p) \tag{A.59}$$

where p is a parameter, $x = [x_1^\top, x_2^\top, \cdots, x_N^\top]^\top$, x_n is each player's action and each objective f_n is differentiable. Suppose $x = \phi(p)$ solves the necessary condition for Nash equilibrium of the parametric game, i.e.,

$$\frac{\partial f_n([x_n, x_{-n}], p)}{\partial x_n} = 0 \tag{A.60}$$

Furthermore, there exists a feedback policy $\delta x = \bar{x} + K\delta p$ in the neighborhood of $\bar{p}$ where $\varepsilon = \|\delta p\|$ is small that solves the quadratic game

$$\min_{x_n} \ \mathrm{quad}(f_n(x,p))|_{\bar{x},\bar{p}} \tag{A.61}$$

where $\bar{x} = \phi(\bar{p})$ and $\frac{\partial f_n(\bar{x},\bar{p})}{\partial x_n} = 0$. The feedback policy approximates the true solution locally well in the sense

$$f_n(\bar{x} + K\delta p, \bar{p} + \delta p) = f_n(\phi(p), p) + O(\varepsilon^2), \forall n \tag{A.62}$$

Proof. We analyze the necessary condition around $\bar{p}$. All derivatives are evaluated at $\bar{x}, \bar{p}$.

$$\mathbf{0} = \frac{\partial f_n}{\partial x_n}(\phi(p), p) \tag{A.63a}$$

$$= \frac{\partial f_n}{\partial x_n}(\bar{x} + \frac{\partial \phi}{\partial p}\delta p, \bar{p} + \delta p) \tag{A.63b}$$

$$= \frac{\partial f_n}{\partial x_n}(\bar{x}, \bar{p}) + \frac{\partial^2 f_n}{\partial x_n \partial x}\frac{\partial \phi}{\partial p}\delta p + \frac{\partial^2 f_n}{\partial x_n \partial p}\delta p + O(\varepsilon^2) \tag{A.63c}$$

For (A.63) to be zero regardless of δp, the following must be true

$$\frac{\partial f_n}{\partial x_n}(\bar{x}, \bar{p}) = \mathbf{0} \tag{A.64a}$$

$$\frac{\partial^2 f_n}{\partial x_n \partial x}\frac{\partial \phi}{\partial p} + \frac{\partial^2 f_n}{\partial x_n \partial p} = \mathbf{0} \tag{A.64b}$$

We re-arrange the later for all players

$$\begin{bmatrix} \frac{\partial^2 f_1}{\partial x_1 \partial x} \\ \frac{\partial^2 f_2}{\partial x_2 \partial x} \\ \vdots \\ \frac{\partial^2 f_N}{\partial x_N \partial x} \end{bmatrix} \frac{\partial \phi}{\partial p} + \begin{bmatrix} \frac{\partial^2 f_1}{\partial x_1 \partial p} \\ \frac{\partial^2 f_2}{\partial x_2 \partial p} \\ \vdots \\ \frac{\partial^2 f_N}{\partial x_N \partial p} \end{bmatrix} + O(\varepsilon^2) = 0 \tag{A.65}$$

Compared to the solution to (A.61).

$$K = -\begin{bmatrix} \frac{\partial^2 f_1}{\partial x_1 \partial x} \\ \frac{\partial^2 f_2}{\partial x_2 \partial x} \\ \vdots \\ \frac{\partial^2 f_N}{\partial x_N \partial x} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial^2 f_1}{\partial x_1 \partial p} \\ \frac{\partial^2 f_2}{\partial x_2 \partial p} \\ \vdots \\ \frac{\partial^2 f_N}{\partial x_N \partial p} \end{bmatrix} \tag{A.66}$$

Therefore, the true parametric equilibrium, around $\bar{x}, \bar{p}$

$$\frac{\partial \phi}{\partial p} = K + O(\varepsilon^2) \tag{A.67a}$$

$$\phi(p) = \bar{x} + K\delta p + O(\varepsilon^2) \tag{A.67b}$$

The approximation becomes

$$f_n(\bar{x} + K\delta p, \bar{p} + \delta p) = f_n(\phi(p) + O(\varepsilon^2), \bar{p} + \delta p) \tag{A.68a}$$

$$= f_n(\phi(p), p) + O(\varepsilon^2) \tag{A.68b}$$

which is (A.62). □

A.7 Background Results

We derive a few results that facilitate the convergence proof.

Lemma 11. *The following holds*

$$\Omega_{n,k} = \frac{\partial}{\partial x_k} \sum_{i=k}^{T} c_{n,i}(x_i, u_{:,i})\Big|_{\bar{x},\bar{u}} \tag{A.69}$$

Proof. $\Omega_{n,k}$ is constructed according to (2.25a). Equation (A.69) is true for $k = T$ by construction. We proof by induction and assume that (A.69) holds for $k+1$, i.e.

$$\Omega_{n,k+1} = \frac{\partial}{\partial x_{k+1}} \sum_{i=k+1}^{T} c_{n,i}(x_i, u_{:,i})\Big|_{\bar{x},\bar{u}} \tag{A.70}$$

, then

$$\Omega_{n,k} = M_{n,k}^{1x} + \Omega_{n,k+1} A_k \tag{A.71a}$$

$$= \frac{\partial c_{n,k}(x_k, u_{:,k})}{x_k}\Big|_{\bar{x},\bar{u}} + \left(\frac{\partial}{\partial x_{k+1}} \sum_{i=k+1}^{T} c_{n,i}(x_i, u_{:,i})\right) \frac{\partial x_{k+1}}{\partial x_k}\Big|_{\bar{x},\bar{u}} \tag{A.71b}$$

$$= \frac{\partial c_{n,k}(x_k, u_{:,k})}{x_k}\Big|_{\bar{x},\bar{u}} + \left(\frac{\partial}{\partial x_k}\Big|_{\bar{x},\bar{u}} \sum_{i=k+1}^{T} c_{n,i}(x_i, u_{:,i})\right) \tag{A.71c}$$

$$= \frac{\partial}{\partial x_k} \sum_{i=k}^{T} c_{n,i}(x_i, u_{:,i})\Big|_{\bar{x},\bar{u}} \tag{A.71d}$$

Therefore, (A.69) holds for k. And by induction, all $k = 0, 1, \ldots, T$.

□

Lemma 12. *The following is true*

$$M_{n,k}^{1u} + \Omega_{n,k+1} B_k = \frac{\partial J_n(u)}{\partial u_{:,k}}\Big|_{\bar{u}} = O(\varepsilon) \tag{A.72}$$

Proof. $\frac{\partial J_n(u)}{\partial u_{:,k}}\Big|_{\bar{u}} = O(\varepsilon)$ is true because $J_n(u)$ is twice differentiable hence Lipschitz, i.e.

$$\left|\left|\frac{J_n(u)}{\partial u_{:,k}}\Big|_{\bar{u}} - \frac{J_n(u)}{\partial u_{:,k}}\Big|_{u^\star}\right|\right| \leq \text{constant} \cdot \|\bar{u} - u^\star\| = O(\varepsilon) \tag{A.73}$$

The first equality holds because

$$\frac{\partial J(u)}{\partial u_{:,k}}\Big|_{\bar{u}} = \frac{\partial}{\partial u_{:,k}}\Big|_{\bar{x},\bar{u}} \sum_{i=0}^{T} c_{n,i}(x_i, u_{:,i}) = \frac{\partial}{\partial u_{:,k}}\Big|_{\bar{x},\bar{u}} \sum_{i=k}^{T} c_{n,i}(x_i, u_{:,i}) \tag{A.74a}$$

$$= \frac{\partial}{\partial u_{:,k}}\Big|_{\bar{x},\bar{u}} c_{n,k}(x_k, u_{:,k}) + \frac{\partial}{\partial u_{:,k}}\Big|_{\bar{x},\bar{u}} \sum_{i=k+1}^{T} c_{n,i}(x_i, u_{:,i}) \tag{A.74b}$$

$$= M_{n,k}^{1u} + \frac{\partial}{\partial x_{k+1}}\Big|_{\bar{x},\bar{u}} \sum_{i=k+1}^{T} c_{n,i}(x_i, u_{:,i}) \frac{\partial x_{k+1}}{\partial u_{:,k}}\Big|_{\bar{x},\bar{u}} \tag{A.74c}$$

$$= M_{n,k}^{1u} + \Omega_{n,k+1} B_k \tag{A.74d}$$

□

These results are used implicitly in the later proofs of lemmas.

A.8 Closeness Lemmas

This section contains lemmas that prove the updates generated by approximated Bellman recursion method and DDP are close. The first lemma shows that the matrices used in the backward recursions are close.

Lemma 13. *The matrices from the backwards recursions of DDP and ABR method are close in the following sense:*

$$\tilde{D}_{n,k} = D_{n,k} + O(\varepsilon) \tag{A.75a}$$

$$S^{1x}_{n,k} = \Omega_{n,k} + O(\varepsilon) \tag{A.75b}$$

$$\tilde{S}^{1x}_{n,k} = \Omega_{n,k} + O(\varepsilon) \tag{A.75c}$$

$$\tilde{S}^{1x}_{n,k} = S^{1x}_{n,k} + O(\varepsilon^2) \tag{A.75d}$$

$$\tilde{S}^{11}_{n,k} = S^{11}_{n,k} + O(\varepsilon^2) \tag{A.75e}$$

$$\tilde{S}_{n,k} = S_{n,k} + O(\varepsilon) \tag{A.75f}$$

$$\begin{bmatrix} \tilde{\Gamma}^{1x}_{n,k} & \tilde{\Gamma}^{1u}_{n,k} \end{bmatrix} = \begin{bmatrix} \Gamma^{1x}_{n,k} & \Gamma^{1u}_{n,k} \end{bmatrix} + O(\varepsilon^2) \tag{A.75g}$$

$$\tilde{\Gamma}^{11}_{n,k} = \Gamma^{11}_{n,k} + O(\varepsilon^2) \tag{A.75h}$$

$$\tilde{\Gamma}_{n,k} = \Gamma_{n,k} + O(\varepsilon) \tag{A.75i}$$

$$\Gamma^{1u}_{n,k} = O(\varepsilon) \tag{A.75j}$$

$$\tilde{\Gamma}^{1u}_{n,k} = O(\varepsilon) \tag{A.75k}$$

and that

$$\tilde{F}_k = F_k + O(\varepsilon) \tag{A.76a}$$

$$\tilde{P}_k = P_k + O(\varepsilon) \tag{A.76b}$$

$$\tilde{H}_k = H_k + O(\varepsilon^2) \tag{A.76c}$$

$$\tilde{H}_k = O(\varepsilon) \tag{A.76d}$$

$$H_k = O(\varepsilon) \tag{A.76e}$$

$$\tilde{s}_k = s_k + O(\varepsilon^2) \tag{A.76f}$$

$$\tilde{s}_k = O(\varepsilon) \tag{A.76g}$$

$$s_k = O(\varepsilon) \tag{A.76h}$$

$$\tilde{K}_k = K_k + O(\varepsilon), \tag{A.76i}$$

Proof. We give a proof by induction. For $k = T$, because of the way these variables are constructed, they are identical, i.e.

$$\Gamma_{n,T} = \tilde{\Gamma}_{n,T} \tag{A.77a}$$

$$F_T = \tilde{F}_T \tag{A.77b}$$

$$P_T = \tilde{P}_T \tag{A.77c}$$

$$H_T = \tilde{H}_T \tag{A.77d}$$

$$s_T = \tilde{s}_T \tag{A.77e}$$

$$K_T = \tilde{K}_T \tag{A.77f}$$

$$S_{n,T} = \tilde{S}_{n,T} \tag{A.77g}$$

so we have (A.75d) (A.75f) (A.75g) (A.75i) (A.75h) (A.76a) (A.76b) (A.76c) (A.76f) (A.76i) hold for $k = T$.

We also know that

$$M^{1u}_{n,T} = \frac{\partial c_{n,T}}{\partial u_T} = \frac{\partial J_n(u)}{\partial u_T} = O(\varepsilon) \tag{A.78}$$

where the first equality is by construction, the second is true because u_T only appears in $J_n(u)$ in $c_{n,T}$. By construction, $\Gamma^{1u}_{n,T} = \tilde{\Gamma}^{1u}_{n,T} = M^{1u}_{n,T} = O(\varepsilon)$, so (A.75j)(A.75k) are true for $k = T$. Similarly, H_T and $\tilde{H}_T$ are constructed from $\Gamma^{u1}_{n,T}$ and $\tilde{\Gamma}^{u1}_{n,T}$, so (A.76d)(A.76e) are true for $k = T$.

Because F_k^{-1} is bounded above, $s_T = -F_T^{-1} H_T = -F_T^{-1} O(\varepsilon) = O(\varepsilon)$. Similarly, $\tilde{s}_T = O(\varepsilon)$. Equations (A.76g)(A.76h) are true for $k = T$.

From (2.25h) and $\Gamma_{n,T} = M_{n,T}$, we can get

$$S^{1x}_{n,T} = M^{1x}_{n,T} + M^{1u}_{n,T} K_T + s_T^\top (M^{ux}_{n,T} + M^{uu}_{n,T} K_T) \tag{A.79a}$$

$$= \Omega_{n,T} + O(\varepsilon) K_T + O(\varepsilon)(M^{ux}_{n,T} + M^{uu}_{n,T} K_T) \tag{A.79b}$$

$$= \Omega_{n,T} + O(\varepsilon) \tag{A.79c}$$

because $M_{n,T}$ is bounded. Hence (A.75b) is true. Further, (A.75c) is also true.

The time indices for $D_{n,T-1}$ and $\tilde{D}_{n,T-1}$ go to a maximum of $T-1$, so to prove things inductively, we need (A.75a) to hold for $k = T-1$. The difference between constructions of $D_{n,T-1}$ and $\tilde{D}_{n,T-1}$ is in that the former uses $\Omega_{n,T}$ and the later uses $\tilde{S}^{1x}_{n,T}$. But since we have proved that $\Omega_{n,T} = \tilde{S}^{1x}_{n,T} + O(\varepsilon)$, and G_{T-1} is bounded, we can also conclude $D_{n,T-1} = \tilde{D}_{n,T-1} + O(\varepsilon)$. Therefore (A.75a) is true for $k = T-1$.

So far, we have proved that for the last step, either $k = T$ or $k = T-1$, (A.75)(A.76) are true. Assuming except for (A.75a), (A.75)(A.76) are true for $k+1$ and (A.75a) is true for k. If we can prove all equations hold one step back, our proof by induction would be done.

Assume (A.75a) holds for k and other equations in (A.75)(A.76) hold for $k+1$. Readers be aware that we use these assumptions implicitly in the derivations following.

From (2.25c) we can get

$$\begin{aligned}\Gamma^{1u}_{n,k} =& M^{1u}_{k,T} + S^{1x}_{n,k+1} B_k && \text{(A.80a)}\\ =& M^{1u}_{k,T} + \Omega_{n,k+1} B_k + O(\varepsilon) B_k && \text{(A.80b)}\\ =& O(\varepsilon) && \text{(A.80c)}\end{aligned}$$

Here we used (A.72). Similarly, we can prove $\tilde{\Gamma}^{1u}_{n,k} = O(\varepsilon)$. So (A.75j) and (A.75k) hold for k.

From (2.25c) and (2.34b) we can compute the difference between $\tilde{\Gamma}_{n,k}$ and $\Gamma_{n,k}$ as

$$\begin{aligned}&\tilde{\Gamma}_{n,k} - \Gamma_{n,k}\\ =& \begin{bmatrix} \tilde{S}^{11}_{n,k+1} - S^{11}_{n,k+1} & \mathbf{0} & \mathbf{0} \\ A_k^\top (\tilde{S}^{x1}_{n,k+1} - S^{x1}_{n,k+1}) & \mathbf{0} & \mathbf{0} \\ B_k^\top (\tilde{S}^{x1}_{n,k+1} - S^{x1}_{n,k+1}) & \mathbf{0} & \mathbf{0} \end{bmatrix}\\ &+ \begin{bmatrix} \mathbf{0} & (\tilde{S}^{1x}_{n,k+1} - S^{1x}_{n,k+1}) A_k & \mathbf{0} \\ \mathbf{0} & A_k^\top (\tilde{S}^{xx}_{n,k+1} - S^{xx}_{n,k+1}) A_k + (\tilde{D}^{xx}_k - D^{xx}_k) & \mathbf{0} \\ \mathbf{0} & B_k^\top (\tilde{S}^{xx}_{n,k+1} - S^{xx}_{n,k+1}) A_k + (\tilde{D}^{ux}_k - D^{ux}_k) & \mathbf{0} \end{bmatrix}\\ &+ \begin{bmatrix} \mathbf{0} & \mathbf{0} & (\tilde{S}^{1x}_{n,k+1} - S^{1x}_{n,k+1}) B_k \\ \mathbf{0} & \mathbf{0} & A_k^\top (\tilde{S}^{xx}_{n,k+1} - S^{xx}_{n,k+1}) B_k + (\tilde{D}^{xu}_k - D^{xu}_k) \\ \mathbf{0} & \mathbf{0} & B_k^\top (\tilde{S}^{xx}_{n,k+1} - S^{xx}_{n,k+1}) B_k + (\tilde{D}^{uu}_k - D^{uu}_k) \end{bmatrix} && \text{(A.81a)}\\ =& \begin{bmatrix} O(\varepsilon^2) & A_k O(\varepsilon^2) & B_k O(\varepsilon^2) \\ A_k^\top O(\varepsilon^2) & A_k^\top O(\varepsilon) A_k + O(\varepsilon) & A_k^\top O(\varepsilon) B_k + O(\varepsilon) \\ B_k^\top O(\varepsilon^2) & B_k^\top O(\varepsilon) A_k + O(\varepsilon) & B_k^\top O(\varepsilon) B_k + O(\varepsilon) \end{bmatrix} && \text{(A.81b)}\\ =& \begin{bmatrix} O(\varepsilon^2) & O(\varepsilon^2) & O(\varepsilon^2) \\ O(\varepsilon^2) & O(\varepsilon) & O(\varepsilon) \\ O(\varepsilon^2) & O(\varepsilon) & O(\varepsilon) \end{bmatrix} && \text{(A.81c)}\end{aligned}$$

from which we can see that (A.75h)(A.75g) and (A.75i) are true. Once we proved the closeness between $\tilde{\Gamma}_{n,k}$ and $\Gamma_{n,k}$ and the specific terms are $O(\varepsilon)$, i.e. (A.75g) to (A.75k), because of they way they are constructed from $\tilde{\Gamma}_{n,k}$ and $\Gamma_{n,k}$, it is safe to say

$$\tilde{F}_k = F_k + O(\varepsilon) \qquad \text{(A.82a)}$$

$$\tilde{P}_k = P_k + O(\varepsilon) \tag{A.82b}$$

$$\tilde{H}_k = H_k + O(\varepsilon^2) \tag{A.82c}$$

$$\tilde{H}_k = O(\varepsilon) \tag{A.82d}$$

$$H_k = O(\varepsilon) \tag{A.82e}$$

Therefore, (A.76a), (A.76b), (A.76c), (A.76d) and (A.76e) are true for k.

Now that we have the results with F_k, $\tilde{F}_k$, H_k, $\tilde{H}_k$, P_k and $\tilde{P}_k$, we can move to what are immediately following, i.e. s_k, $\tilde{s}_k$, K_k and $\tilde{K}_k$.

$$s_k = -F_k^{-1}H_k = -F_k^{-1}O(\varepsilon) = O(\varepsilon) \tag{A.83}$$

which is true because F_k^{-1} is bounded above. Similarly, we have $\tilde{s}_k = O(\varepsilon)$. Equations (A.76g) and (A.76h) are true.

$$\tilde{s}_k = -\tilde{F}_k^{-1}\tilde{H}_k = -(F_k + O(\varepsilon))^{-1}(H_k + O(\varepsilon^2)) \tag{A.84a}$$

$$= -(F_k^{-1} + O(\varepsilon))(H_k + O(\varepsilon^2)) \tag{A.84b}$$

$$= -F_k^{-1}H_k + F_k^{-1}O(\varepsilon^2) + H_k O(\varepsilon) + O(\varepsilon^2) \tag{A.84c}$$

$$= s_k + O(\varepsilon^2) \tag{A.84d}$$

$$\tilde{K}_k = -\tilde{F}_k^{-1}\tilde{P}_k = -(F_k + O(\varepsilon))^{-1}(P_k + O(\varepsilon)) \tag{A.84e}$$

$$= -(F_k^{-1} + O(\varepsilon))(P_k + O(\varepsilon)) \tag{A.84f}$$

$$= -F_k^{-1}P_k + (F_k^{-1} + P_k)O(\varepsilon) + O(\varepsilon^2) \tag{A.84g}$$

$$= K_k + O(\varepsilon) \tag{A.84h}$$

Equations (A.76f) and (A.76i) are true for k.

Now we are equipped to get closeness/small results for $S_{n,k}$ and $\tilde{S}_{n,k}$.

$$\begin{aligned}
&\tilde{S}_{n,k} - S_{n,k} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & I & \tilde{K}_k^\top \end{bmatrix} \tilde{\Gamma}_{n,k} \begin{bmatrix} 1 & 0 \\ 0 & I \\ 0 & \tilde{K}_k \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & I & K_k^\top \end{bmatrix} \Gamma_{n,k} \begin{bmatrix} 1 & 0 \\ 0 & I \\ 0 & K_k \end{bmatrix} \\
&+ \begin{bmatrix} \tilde{s}_k^\top \tilde{\Gamma}_{n,k}^{uu} \tilde{s}_k + 2\tilde{s}_k^\top \tilde{\Gamma}_{n,k}^{u1} - s_k^\top \Gamma_{n,k}^{uu} s_k - 2s_k^\top \Gamma_{n,k}^{u1} & \mathbf{0} \\ (\tilde{\Gamma}_{n,k}^{xu} + \tilde{\Gamma}_{n,k}^{uu}\tilde{K}_k)\tilde{s}_k - (\Gamma_{n,k}^{ux} + \Gamma_{n,k}^{uu}K_k)^\top s_k & \mathbf{0} \end{bmatrix}
\end{aligned}$$

$$+\begin{bmatrix}\mathbf{0} & \tilde{s}_k^\top(\tilde{\Gamma}^{ux}_{n,k}+\tilde{\Gamma}^{uu}_{n,k}\tilde{K}_k)-s_k^\top(\Gamma^{ux}_{n,k}+\Gamma^{uu}_{n,k}K_k)\\ \mathbf{0} & \mathbf{0}\end{bmatrix} \tag{A.85a}$$

$$=\begin{bmatrix}\tilde{\Gamma}^{11}_{n,k} & \tilde{\Gamma}^{1x}_{n,k}+\tilde{\Gamma}^{1u}_{n,k}\tilde{K}\\ \tilde{\Gamma}^{x1}_{n,k}+\tilde{K}^\top\tilde{\Gamma}^{u1}_{n,k} & \tilde{\Gamma}^{xx}_{n,k}+2\tilde{\Gamma}^{xu}_{n,k}\tilde{K}+\tilde{K}^\top\tilde{\Gamma}^{uu}_{n,k}\tilde{K}\end{bmatrix}$$

$$-\begin{bmatrix}\Gamma^{11}_{n,k} & \Gamma^{1x}_{n,k}+\Gamma^{1u}_{n,k}K\\ \Gamma^{x1}_{n,k}+K^\top\Gamma^{u1}_{n,k} & \Gamma^{xx}_{n,k}+2\Gamma^{xu}_{n,k}K+K^\top\Gamma^{uu}_{n,k}K\end{bmatrix}$$

$$+\begin{bmatrix}O(\varepsilon^2) & O(\varepsilon^2)\\ O(\varepsilon^2) & 0\end{bmatrix} \tag{A.85b}$$

$$=\begin{bmatrix}O(\varepsilon^2) & O(\varepsilon^2)\\ O(\varepsilon^2) & O(\varepsilon)\end{bmatrix} \tag{A.85c}$$

$$\tag{A.85d}$$

So that (A.75e), (A.75d) and (A.75f) are true for k.

$$S^{1x}_{n,k}=M^{1x}_{n,k}+S^{1x}_{n,k+1}A_k+\Gamma^{1u}_{n,k}K_k+s_k^\top(\Gamma^{ux}_{n,k}+\Gamma^{uu}_{n,k}K_k) \tag{A.86a}$$

$$=M^{1x}_{n,k}+\Omega_{n,k+1}A_k+A_kO(\varepsilon)$$

$$+K_kO(\varepsilon)+(\Gamma^{ux}_{n,k}+\Gamma^{uu}_{n,k}K_k)O(\varepsilon) \tag{A.86b}$$

$$=\Omega_{n,k}+O(\varepsilon) \tag{A.86c}$$

Therefore, (A.75b) holds and then naturally (A.75c) holds.

We continue to prove that $\tilde{D}_{n,k-1}$ and $D_{n,k-1}$ are close, which is true because

$$\tilde{D}_{n,k-1}=\sum_{l=1}^{n_x}\tilde{S}^{1x^l}_{n,k}G^l_k \tag{A.87a}$$

$$=\sum_{l=1}^{n_x}(\Omega^l_{n,k}+O(\varepsilon))G^l_k \tag{A.87b}$$

$$=\sum_{l=1}^{n_x}\Omega^l_{n,k}G^l_k+O(\varepsilon) \tag{A.87c}$$

$$=D_{n,k-1}+O(\varepsilon) \tag{A.87d}$$

So (A.75a) is true.

□

The following lemma shows that the states and actions computed in the update steps of both algorithms are close.

Lemma 14. *The updates by two algorithms are small and close*

$$\delta x_{k+1}^D = A_k \delta x_k^D + B_k \delta u_k^D + O(\varepsilon^2) \tag{A.88a}$$

$$\delta x_{k+1}^A = A_k \delta x_k^A + B_k \delta u_k^A + O(\varepsilon^2) \tag{A.88b}$$

$$\delta u_k^A = O(\varepsilon) \tag{A.88c}$$

$$\delta u_k^D = O(\varepsilon) \tag{A.88d}$$

$$\delta x_k^A = O(\varepsilon) \tag{A.88e}$$

$$\delta x_k^D = O(\varepsilon) \tag{A.88f}$$

$$\delta u_k^A - \delta u_k^D = O(\varepsilon^2) \tag{A.88g}$$

$$\delta x_k^A - \delta x_k^D = O(\varepsilon^2) \tag{A.88h}$$

$$\delta u^A - \delta u^D = O(\varepsilon^2) \tag{A.88i}$$

$$\|\bar{u} + \delta u^A - u^\star\| = O(\varepsilon^2). \tag{A.88j}$$

$$\|\bar{u} + \delta u^D - u^\star\| = O(\varepsilon^2). \tag{A.88k}$$

Proof. Equation (A.88a) comes directly from the Taylor series expansion of (2.2) and (A.88b) from (2.11d).

We prove (A.88c) to (A.88h) by induction. For $k = 0$, $\delta x_0^A = \delta x_0^D = 0$ and $\delta u_0^A = s_0,\ \delta x_0^D = \tilde{s}_0$. We know from the proof of lemma 13 that $s_0 = \tilde{s}_0 + O(\varepsilon^2)$, $s_0 = O(\varepsilon)$ and $\tilde{s}_0 = O(\varepsilon)$, so (A.88c) to (A.88h) hold for $k = 0$. Assume (A.88c) to (A.88h) hold for k, then

$$\delta u_{k+1}^A = K_{k+1} \delta x_k^A + s_{k+1} = O(\varepsilon) \tag{A.89a}$$

$$\delta u_{k+1}^D = \tilde{K}_{k+1} \delta x_k^A + \tilde{s}_{k+1} = O(\varepsilon) \tag{A.89b}$$

$$\delta x_{k+1}^A = A_k \delta x_k^A + B_k \delta u_k^A + O(\varepsilon^2) = O(\varepsilon) \tag{A.89c}$$

$$\delta x_{k+1}^D = A_k \delta x_k^D + B_k \delta u_k^D + O(\varepsilon^2) = O(\varepsilon) \tag{A.89d}$$

$$\delta u_{k+1}^A - \delta u_{k+1}^D = K_k \delta x_k^A - \tilde{K}_k \delta x_k^D + s_k - \tilde{s}_k \tag{A.89e}$$

$$= K_k \delta x_k^A - (K_k + O(\varepsilon))(\delta x_k^A + O(\varepsilon^2)) + O(\varepsilon^2) \tag{A.89f}$$

$$= O(\varepsilon) \delta x_k^A + O(\varepsilon^2) \tag{A.89g}$$

$$=O(\varepsilon^2) \tag{A.89h}$$

$$\delta x_{k+1}^A - \delta x_{k+1}^D = A_k(\delta x_k^A - \delta x_k^D) + B_k(\delta u_k^A - \delta u_k^D) + O(\varepsilon^2)$$

$$=O(\varepsilon^2) \tag{A.89i}$$

So (A.88c) to (A.88h) hold for $k+1$ and the proof by induction is done. Equation (A.88i) comes directly as a result. Equation (A.88j) is classic convergence analysis for Newton's method [40]. Equation (A.88k) follows directly from (A.88i) and (A.88j).

□

Appendix B

Constrained Dynamic Games Technicalities

B.1 Open-loop Nash Equilibrium Related

This section contains proofs of OLNE.

B.1.1 Problem 7 is Equivalent Problem 8

Because the normal cone of intersections of convex sets is equal to the sum of normal cones of the convex sets [89], the inclusion problem Problem 8 is equivalent to

$$-\eta \mathcal{J}_{xu}([x^\star, u^\star]) \in \mathcal{N}_{\mathcal{D}\cap\mathcal{G}}([x^\star, u^\star]) \tag{B.1}$$

According to the definition of normal cone

$$\begin{aligned} &\mathcal{N}_{\mathcal{D}\cap\mathcal{G}}([x^\star, u^\star]) := \\ &\quad \{[y,z] | -[y,z]^\top([x,y]-[x^\star,u^\star]),\ \forall\ [x,y] \in \mathcal{D}\cap\mathcal{G}\} \end{aligned} \tag{B.2}$$

(B.1) is further equivalent to

$$\eta \mathcal{J}_{xu}([x^\star, u^\star])^\top([x,u]-[x^\star,u^\star]) \geq 0,\ \forall [x,y] \in \mathcal{D}\cap\mathcal{G} \tag{B.3}$$

which is the VI problem Problem 7.

When $\mathcal{J}_{xu}$ is strongly monotone, $\mathcal{D}$ and $\mathcal{G}$ are convex, $\mathcal{J}_{xu}$, $\mathcal{N}_{\mathcal{G}}$ and $\mathcal{N}_{\mathcal{D}}$ are all maximally monotone operators. Because adding operators preserve maximal monotonicity, so we can apply DR splitting algorithm to Problem 8.

B.1.2 Resolvents of Operators in Problem 8

The resolvent of a set-valued/single-valued monotone map Φ is defined by

$$r_{\Phi} := (I+\Phi)^{-1} \tag{B.4}$$

which is single-valued and non-expansive [41].

Resolvent of the normal cone of a convex set

Suppose we have a convex set $\mathcal{X}$. Following the definition, the resolvent of $\mathcal{N}_{\mathcal{X}}(x)$

$$y = r_{\mathcal{N}_G}(x) = (I+\mathcal{N}_{\mathcal{X}})^{-1}(x) \tag{B.5a}$$

$$\tag{B.5b}$$

which is equivalent to

$$x - y \in \mathcal{N}_{\mathcal{X}}(y) \tag{B.6}$$

$$(x-y)^{\top}(z-y) \leq 0,\ \forall z \in \mathcal{X} \tag{B.7}$$

This VI problem is equivalent to the optimization of a projection of x onto $\mathcal{X}$

$$\min_{z} \quad y = \|x-z\| \tag{B.8a}$$

$$\text{s.t.} \quad z \in \mathcal{X} \tag{B.8b}$$

Therefore the resolvent of a normal cone of a convex set is the projection onto the convex set. This justifies Problem 10 and 12.

Resolvent of a gradient vector

We study the resolvent of the gradient operator $\eta\mathcal{J}(u)$ of a game as in (3.7). Suppose $u = r_{\eta\mathcal{J}}(x)$, equivalently we have

$$u + \eta\mathcal{J}(u) = x \tag{B.9a}$$

Consider a static game problem

$$\min_{u_n} \quad J_n(u) + \frac{1}{2\eta} \|u - x\|^2 \tag{B.10a}$$

whose Nash equilibrium is equivalent to the solution of

$$\eta \mathcal{J}(u) + u - x = 0 \tag{B.11}$$

which is equivalent to (B.9a). Therefore, the resolvent is equivalent to the solution of a static game.

Based on the arguement of Section B.1.2 and B.1.2, the subproblems of Section 3.3 can be justified.

B.1.3 Generative Cone Condition to Linear Inequalities

Lemma 15. *A generative cone condition, where S has full row rank*

$$x \in cone\{S^\top\} \tag{B.12}$$

can be equivalently expressed as

$$Lx \leq 0 \tag{B.13}$$

where

$$L = \begin{bmatrix} -(SS^\top)^{-1}S \\ I - S^\top(SS^\top)^{-1}S \\ -I + S^\top(SS^\top)^{-1}S \end{bmatrix} \tag{B.14}$$

Proof. The generative condition is equivalent to

$$x = S^\top \lambda \tag{B.15a}$$

$$\lambda \geq 0 \tag{B.15b}$$

which is equivalent to

$$\exists M \text{ s.t. } Mx = \lambda \geq 0 \tag{B.16}$$

$$(I - S^\top M)x = 0 \tag{B.17}$$

Choose $M = (SS^\top)^{-1}S$, the equivalent conditions can be easily checked

$$Mx = (SS^\top)^{-1}Sx = (SS^\top)^{-1}SS^\top\lambda = \lambda \geq 0 \tag{B.18a}$$

Given the SVD decomposition of S

$$S = U\begin{bmatrix}\Sigma & 0\end{bmatrix}\begin{bmatrix}V_1^\top \\ V_2^\top\end{bmatrix} \tag{B.19}$$

It is easy to check that

$$(I - S^\top M) = I - S^\top(SS^\top)^{-1}S = V_2V_2^\top \tag{B.20}$$

Since $V_2^\top x = 0$, (B.16) is true. □

B.1.4 Proof of Theorem 3

First we show that the policy produces feasible trajectories.

Recall that $[\bar{x}, \bar{u}]$ is a stationary trajectory for the ABR method, so that the feedback policy has the form:

$$u_{:,k} = \phi_{:,k}(x_k) = \bar{u}_k + K_k(x_k - \bar{x}_k). \tag{B.21}$$

By construction, (3.36f) ensures that active constraints for $[\bar{x}, \bar{u}]$ remain active for $[x, u]$.

Now we analyze the inactive constraints. Note that $[\bar{x}, \bar{u}]$ satisfies

$$W_k^{\bar{i}}\bar{x}_k + S_k^{\bar{i}}\bar{u}_{:,k} + p_k^{\bar{i}} \leq -\gamma_k^{\bar{i}} < 0. \tag{B.22}$$

Furthermore, if $\|x_t - \bar{x}_t\| \leq \varepsilon$ the affine dynamics and polyhedral constraints imply that we must have that $\|u_{:,k} - \bar{u}_{:,k}\| = O(\varepsilon)$ and $\|x_k - \bar{x}_k\| = O(\varepsilon)$ for all $k \geq t$. It follows that for sufficiently small ε, the constraint (3.41d) holds. Thus, the feedback policy produces a feasible trajectory.

Finally, we show that the approximate feedback equilibrium condition, (3.42), holds. Note that left inequality always holds by construction, so we only need to prove the right inequality.

Fix a player n and a time $t \geq 0$. Assume that the other players are using the strategy profile $\phi_{-n,t:T}$. Then, with the strategies of the other players fixed, the policy, $\psi_{n,t:}$, that minimizes $J_{n,t}(x_t, [\psi_{n,t:}, \phi_{-n,t:}])$ can be computed from the following optimal control problem:

$$\min_{u_{n,t:T}} \quad \sum_{k=t}^{T} c_{n,k}(x_k, (u_{n,k}, \phi_{-n,k}(x_k))) \tag{B.23a}$$

$$\text{s.t.} \quad u_{-n,k} = \phi_{-n,k}(x_k) \tag{B.23b}$$

$$x_{k+1} = A_k x_k + B_k u_{:,k} + b_k \tag{B.23c}$$

$$W_k^{\bar{a}} x_k + S_k^{\bar{a}} u_{:,k} + p_k^{\bar{a}} \leq -\gamma_k^{\bar{a}} \tag{B.23d}$$

$$W_k^{\bar{i}} x_k + S_k^{\bar{i}} u_{:,k} + p_k^{\bar{i}} \leq 0 \tag{B.23e}$$

$$x_t \text{ is fixed.} \tag{B.23f}$$

The quadraticization around $[\bar{x}, \bar{u}]$ of the optimal control problem (B.23) is exactly the same as that of player n in Problem 19. Thus, when all other players follow the ABR strategy, the minimizer $\psi_{n,t:}$ should be the same as $\phi_{n,t:}$, since player n will have no incentive to change its strategy on the quadraticized problem. Then, to show that the strategy is an approximate feedback equilibrium, it suffices to show that $\phi_{n,t:}$ is approximately optimal. The bound on optimality follows from the general result on parameterized optimization from Lemma 17 below.

B.1.5 Lemmas on Optimization Approximation

In this section, we present Lemma 17 which is used to prove Theorem 3, along with supporting results.

Let $f(x,u)$ be a strongly convex with respect to u in a neighborhood of $(\bar{x}, \bar{u})$. Assume that $\bar{u}$ minimizes $f(\bar{x}, u)$ with respect to u. Let $\hat{f}(x,u)$ be its quadratic approximation around the nominal point $(\bar{x}, \bar{u})$. Let $C(x) = \{u | Wx + Su + p \leq 0\}$. Define the policies by

$$\psi(x) = \underset{u \in C(x)}{\arg\min}\, f(x,u) \tag{B.24a}$$

$$\rho(x) = \underset{u \in C(x)}{\arg\min}\, \hat{f}(x,u) \tag{B.24b}$$

Note that $\rho(x)$ has the form

$$u = \rho(x) = K(\mathscr{A})x + h(\mathscr{A}), \tag{B.25}$$

each pair $(K(\mathscr{A}), h(\mathscr{A}))$ corresponds to the active indices, $\mathscr{A}$, of (x,u).

Let $\phi(x) = K(\bar{\mathscr{A}})x + h(\bar{\mathscr{A}})$, where $\bar{\mathscr{A}}$ is the active set of $(\bar{u}, \bar{x})$. Note that in the game context, this strategy corresponds precisely to the individual players' approximate strategy computed by ABR method.

The eventual goal is to show that ϕ and ψ give similar costs. As an intermediate result, we show that ψ and ρ give similar costs.

Lemma 16. *Let $\|x-\bar{x}\| = \|\delta x\| = \varepsilon$. Say that $f(x,u)$ has Lipschitz second derivatives and is strongly convex with respect to u in a neighborhood of (x,u). Further assume that $C(x)$ is non-empty for all x in this neighborhood. Then the following bounds hold*

$$f(x,\psi(x)) \leq f(x,\rho(x)) \leq f(x,\psi(x)) + O(\varepsilon^2) \tag{B.26}$$

Proof. Note that the first inequality of (B.26) is immediate from the definition of ψ. Thus, we focus on the second inequality.

We will show that $\psi(x) - \rho(x) = O(\varepsilon)$. To do this, we will show that both functions are Lipschitz and that $\phi(\bar{x}) = \rho(\bar{x}) = \bar{u}$.

Note that $\psi(x)$ is the solution to the following VI

$$\nabla_u f(x,u)^\top (v-u) \quad \forall v \in C(x). \tag{B.27}$$

By strong convexity and differentiability, $\nabla_u^2 f(x,u)$ is positive definite. This implies that for any matrix, B with full column rank, we have that $B^\top \nabla_u^2 f(x,u) B$ is also positive definite, and thus has positive determinant. Then general results on perturbed VIs show that $\psi(x)$ must be Lipschitz. See [90]. Furthermore, by construction $\psi(\bar{x}) = \bar{u}$.

By the same reasoning, we must also have that $\rho(x)$ is Lipschitz and again by construction we have that $\rho(\bar{x}) = \bar{u}$. It follows that $\rho(x) = \bar{u} + O(\varepsilon)$. Thus

$$\rho(x) - \psi(x) = \rho(x) - \bar{u} + \bar{u} - \rho(x) = O(\varepsilon).$$

Now we will prove the upper bound. For compact notation, let $\tilde{u} = \psi(x)$ and let $u = \rho(x)$. Then we have the approximation:

$$f(x,u) = f(x,\tilde{u}) + \nabla_u f(x,\tilde{u})^\top (u-\tilde{u}) + O(\varepsilon^2).$$

By optimality of $\tilde{u}$ and feasibility of u, we have that

$$0 \leq \nabla_u f(x,\tilde{u})^\top (u-\tilde{u}).$$

Thus, the proof will be completed if we can bound this term above by $O(\varepsilon^2)$.

Let $\mathcal{J} = \nabla_u f$. Since $\mathcal{J}$ is differentiable and $u - \tilde{u} = O(\varepsilon)$, we have that

$$\begin{aligned}\nabla_u f(x,\tilde{u}) &= \nabla_u f(x,u) + O(\varepsilon) \\ &= \mathcal{J}(\bar{x},\bar{u}) + \frac{\partial \mathcal{J}}{\partial x}\delta x + \frac{\partial \mathcal{J}}{\partial u}\delta u + O(\varepsilon),\end{aligned}$$

where the partial derivatives are evaluated at $(\bar{x},\bar{u})$.

Thus, the desired bound is given by

$$\begin{aligned}&\nabla_u f(x,\tilde{u})^\top (u-\tilde{u}) \\ &= \left(\mathcal{J}(\bar{x},\bar{u}) + \frac{\partial \mathcal{J}}{\partial x}\delta x + \frac{\partial \mathcal{J}}{\partial u}\delta u\right)^\top (u-\tilde{u}) + O(\varepsilon^2) \\ &\leq O(\varepsilon^2),\end{aligned}$$

where the second inequality due to optimality of u for the corresponding affine VI.

□

Now we present the optimization approximation result required for Theorem 3.

Lemma 17. *Assume that $\phi(x) \in C(x)$ and let $\|x - \bar{x}\| \leq \varepsilon$. Then the following bounds hold.*

$$f(x,\psi(x)) \leq f(x,\phi(x)) \leq f(x,\psi(x)) + O(\varepsilon^2). \tag{B.28}$$

Proof. Proof . The bound on the left holds automatically since $\phi(x)$ is feasible and $\psi(x)$ is the corresponding optimal solution.

Specializing Lemma 16 to the case of optimization implies that $f(x,\psi(x)) = f(x,\rho(x)) + O(\varepsilon^2)$. Thus, it suffices to show that

$$f(x,\phi(x)) = f(x,\rho(x)) + O(\varepsilon^2). \tag{B.29}$$

Let $x(\theta) = (1-\theta)\bar{x} + \theta x$, let $u(\theta) = \rho(x(\theta))$, let $\mathscr{A}(\theta)$ be the active set for $(x(\theta),u(\theta))$, and let $\mathscr{I}(\theta)$ be the inactive set. We will show that the active sets only switch a finite number of times.

We claim that for each subset $\hat{\mathscr{A}} \subset \{1,\ldots,n_c\}$, the set $\Theta(\hat{\mathscr{A}}) = \{\theta | \mathscr{A}(\theta) = \hat{\mathscr{A}}\}$ is convex. In particular, $\Theta(\hat{\mathscr{A}})$ must be an interval.

Proposition 7.10 of [83] implies that the set of x such that $(x,\rho(x))$ has active set $\hat{\mathscr{A}}$ is a polyhedron. Since each $\Theta(\hat{\mathscr{A}})$ is the projection of the intersection of this set with a line, we must have that $\Theta(\hat{\mathscr{A}})$ is convex.

Since there are at most 2^{n_c} active sets, it follows that there are sets $\mathscr{A}_1, \mathscr{A}_2, \ldots, \mathscr{A}_k$ with $k \leq 2^{n_c}$ such that $\Theta(\mathscr{A}_i)$ are intervals that partition $[0,1]$. These sets can be arranged so that $\theta_i \leq \theta_{i+1}$ for $\theta_i \in \Theta(\mathscr{A}_i)$ and $\theta_{i+1} \in \Theta(\mathscr{A}_{i+1})$.

Let ρ^i be the affine strategy corresponding to $\mathscr{A}_i$. Then by construction, we have

$$f(x,\phi(x)) - f(x,\rho(x)) = \sum_{i=1}^{k-1} (f(x,\rho^i(x)) - f(x,\rho^{i+1}(x))). \tag{B.30}$$

These active sets have the property that either $\mathscr{A}_i \subset \mathscr{A}_{i+1}$ or $\mathscr{A}_i \supset \mathscr{A}_{i+1}$. Either $\mathscr{A}_i$ must be open on the right or $\mathscr{A}_{i+1}$ must be open on the left. Consider the case that $\mathscr{A}_i$ is open on the right. (The case of $\mathscr{A}_{i+1}$ open on the left is similar.) Then the boundary between $\mathscr{A}_i$ and $\mathscr{A}_{i+1}$ is a point $\hat{\theta} \in \mathscr{A}_{i+1}$. By continuity, all of the constraints in $\mathscr{A}_i$ must be tight at $\hat{\theta}$. Thus, we must have that $\mathscr{A}_i \subset \mathscr{A}_{i+1}$.

Consider the case that $\mathscr{A}_i \subset \mathscr{A}_{i+1}$. (The other case is similar.) In particular, the constraints from $\mathscr{A}_i$ must be active for both corresponding VI solutions. Thus, if $S^+_{\mathscr{A}_i}$ is the Moore-Penrose pseudoinverse of $S_{\mathscr{A}_i}$ and $R_{\mathscr{A}_i}$ is a matrix with full column rank such that $\mathscr{R}(R_{\mathscr{A}_i}) = \mathscr{N}(S_{\mathscr{A}_i})$, we must have that

$$\rho^j(x) = -S^+_{\mathscr{A}_i}(W_{\mathscr{A}_i}x + p_{\mathscr{A}_i}) + R_{\mathscr{A}_i}z_j,$$

for $j = i, i+1$, and some vectors z_j. It follows that

$$\rho^{i+1}(x) - \rho^i(x) = R_{\mathscr{A}_i}(z_{i+1} - z_i) \tag{B.31}$$

Let $\theta_i \in \Theta(\mathscr{A}_i)$ be some value for which $\mathscr{A}_i$ is the active set for $x(\theta_i) =: x^i$, and let u^i be the corresponding optimizer. Note that $\|\bar{x} - x(\theta_i)\| \leq \varepsilon$. It follows that

$$\nabla_u f(x,\rho^i(x)) = \nabla_u \hat{f}(x^i,u^i) + O(\varepsilon).$$

Since u^i minimizes $\hat{f}(x^i,u)$, with $\mathscr{A}_i$ active, we must have that

$$\nabla_u \hat{f}(x^i,u^i)^\top (u^i + R_{\mathscr{A}_i}z - u^i) \geq 0,$$

for all sufficiently small z. It follows that $\nabla_u \hat{f}(x^i,u^i)^\top R_{\mathscr{A}_i} = 0$.

Thus, we can get the bound:

$$\begin{aligned} &f(x,\rho^{i+1}(x)) \\ &\quad = f(x,\rho^i(x)) \end{aligned}$$

$$
\begin{aligned}
&+\nabla_u f(x,\rho^i(x))^\top(\rho^{i+1}(x)-\rho^i(x))+O(\varepsilon^2)\\
&=f(x,\rho^i(x))+\\
&\nabla_u \hat{f}(x^i,u^i)^\top R_{\mathscr{A}_i}(z_{i+1}-z_i)+O(\varepsilon^2)\\
&=f(x,\rho^i(x))+O(\varepsilon^2). \qquad \text{(B.32)}
\end{aligned}
$$

An analogous argument argument in the case of $\mathscr{A}_i \supset \mathscr{A}_{i+1}$ shows that the same bound holds.

Plugging (B.32) into (B.30) completes the proof. □

B.2 Parametric Games and Feedback Equilibrium in Details

This section contains the detailed development for Section 3.4. Some problem definitions and lemmas are restated so this section is self-contained for ease of reading. We study in this section some basic parametric quadratic games with polyhedral constraints to gain insight of the feedback Nash equilibrium of dynamic games in general. We assume each player's cost function $J_n(x,u)$ is continuous and strictly convex throughout this section.

B.2.1 Linear Equality Constrained Quadratic Parametric Game

Problem 22 is a basic form that is encountered when approximating a constrained dynamic game, where $u=[u_1^\top,u_2^\top,...,u_N^\top,]^\top$ collects all players' action and x is a vector parameter. An FNE policy $u=\phi(x)$ is desired. This game has an explicit analytical solution with simple assumptions.

Problem 22. ***Linear equality constrained quadratic parametric game***

$$
\min_{u_n} \quad J_n(x,u)=\frac{1}{2}\begin{bmatrix}1\\x\\u\end{bmatrix}^\top\begin{bmatrix}\Gamma_n^{11} & \Gamma_n^{1x} & \Gamma_n^{1u}\\ \Gamma_n^{x1} & \Gamma_n^{xx} & \Gamma_n^{xu}\\ \Gamma_n^{u1} & \Gamma_n^{ux} & \Gamma_n^{uu}\end{bmatrix}\begin{bmatrix}1\\x\\u\end{bmatrix} \qquad \text{(B.33a)}
$$

$$
s.t. \quad Wx+Su+p=0 \qquad \text{(B.33b)}
$$

The following lemma describes its solution.

Lemma 18. *Given playerwise convexity of objective functions, i.e.,* Γ_n^{uu} *are positive definite, Problem 22 has a unique affine feedback Nash equilibrium as in* (B.34a) *if* F *is invertible and* S *has full row rank.*

$$u^\star = Kx + s \tag{B.34a}$$

$$\lambda^\star = (SF^{-1}S^\top)^{-1}(W - SF^{-1}P)x + \tag{B.34b}$$

$$(SF^{-1}S^\top)^{-1}(p - SF^{-1}H) \tag{B.34c}$$

where

$$K = -F^{-1}S^\top(SF^{-1}S^\top)^{-1}(W - SF^{-1}P) - F^{-1}P \tag{B.35a}$$

$$s = -F^{-1}S^\top(SF^{-1}S^\top)^{-1}(p - SF^{-1}H) - F^{-1}H \tag{B.35b}$$

$$F = \begin{bmatrix} \Gamma_1^{u_1u_1} & \Gamma_1^{u_1u_2} & \dots & \Gamma_1^{u_1u_N} \\ \Gamma_2^{u_2u_1} & \Gamma_2^{u_2u_2} & \dots & \Gamma_2^{u_2u_N} \\ \vdots & \vdots & \ddots & \vdots \\ \Gamma_N^{u_Nu_1} & \Gamma_N^{u_Nu_2} & \dots & \Gamma_N^{u_Nu_N} \end{bmatrix} \tag{B.35c}$$

$$P = \begin{bmatrix} \Gamma_1^{u_1x} \\ \Gamma_2^{u_2x} \\ \vdots \\ \Gamma_N^{u_Nx} \end{bmatrix} \quad H = \begin{bmatrix} \Gamma_n^{u_11} \\ \Gamma_n^{u_21} \\ \vdots \\ \Gamma_n^{u_N1} \end{bmatrix} \tag{B.35d}$$

Proof. The feedback Nash equilibrium can be solved via solving the KKT conditions of all players [73]. The Lagrangians can be formulated as

$$J_n(x,u) = \frac{1}{2}\begin{bmatrix} 1 \\ x \\ u \end{bmatrix}^\top \begin{bmatrix} \Gamma_n^{11} & \Gamma_n^{1x} & \Gamma_n^{1u} \\ \Gamma_n^{x1} & \Gamma_n^{xx} & \Gamma_n^{xu} \\ \Gamma_n^{u1} & \Gamma_n^{ux} & \Gamma_n^{uu} \end{bmatrix} \begin{bmatrix} 1 \\ x \\ u \end{bmatrix} + \tag{B.36a}$$

$$\lambda^\top(Wx + Su + p) \tag{B.36b}$$

Therefore the KKT conditions are

$$\frac{\partial}{u_n} L_n(x,u) = \Gamma_n^{u_nx}x + \sum_j^N \Gamma_n^{u_nu_j}u_j + \Gamma_n^{u_n1} + (S^{n_cu_n})^\top\lambda \tag{B.37a}$$

$$n = 1, 2, ..., N \tag{B.37b}$$

$$0 = Wx + Su + p \tag{B.37c}$$

Thus, for any λ, the unique solution (B.37a) for u is given by $u = -F^{-1}(Px + H + S^\top \lambda)$. Plugging this result into (B.37c) and solving for λ gives

$$\lambda^\star = (SF^{-1}S^\top)^{-1}(W - SF^{-1}P)x + (SF^{-1}S^\top)^{-1}(p - SF^{-1}H) \tag{B.38}$$

Plugging this back into the expression for u gives the result.

□

B.2.2 Linearly Constrained Quadratic Parametric Game

The more generalized problem with inequality constraints are studied in this section. Related problems are studied and the piecewise affine solution was recognized in the variational inequality literature [90, 91]. Our analysis focuses on games which in addition, recognizes the piecewise quadratic value functions. We inherit the notation of u, x, Γ, F, P, H and ϕ from Section B.2.1. Such problems serve as a backbone for analyzing FNE for dynamic games when we solve the static game (3.6c) formed by the state-action value function at a stage.

Problem 23. ***Linearly constrained quadratic parametric game***

$$\min_{u_n} \quad J_n(x,u) = \frac{1}{2}\begin{bmatrix}1\\x\\u\end{bmatrix}^\top \begin{bmatrix}\Gamma_n^{11} & \Gamma_n^{1x} & \Gamma_n^{1u}\\ \Gamma_n^{x1} & \Gamma_n^{xx} & \Gamma_n^{xu}\\ \Gamma_n^{u1} & \Gamma_n^{ux} & \Gamma_n^{uu}\end{bmatrix}\begin{bmatrix}1\\x\\u\end{bmatrix} \tag{B.39a}$$

$$s.t. \quad Wx + Su + p \leq 0 \tag{B.39b}$$

We use $\mathcal{X}$ and $\mathcal{U}$ to indicate the feasible sets of x and u, i.e.,

$$\mathcal{X} = \{x \mid \exists\, u \text{ s.t. } Wx + Su + p \leq 0\} \tag{B.40a}$$

$$\mathcal{U} = \{u \mid \exists\, x \text{ s.t. } Wx + Su + p \leq 0\} \tag{B.40b}$$

Note that we do not lose generality without explicit linear equality constraints, since a linear equality constraint can be equivalently formulated with two inequality constraints. The following lemma and proof offers a descriptive solution to this problem.

Lemma 19. *Given playerwise convexity of objective functions, i.e., Γ_n^{uu} are positive definite, Problem 23 has a piecewise affine feedback Nash equilibrium $u = \phi^\star(x)$ on a finite polyhedral partition of $\mathcal{X}$ if and only if F is invertible.*

Proof. Assume S has n_c rows and we use E^{n_c} to indicate the power set of $\{1,2,...,n_c\}$. For a set of indices $a \in E^{n_c}$, we use W_a, S_a and p_a to denote picking the corresponding rows. The solution to Problem 23 can be found via the following procedure.

Pick one element $a \in E^{n_c}$, which we suppose to be the indices of active constraints, and solve a linear equality constrained quadratic parametric game with the method in Section B.2.1 as following

$$\min_{u_n} \quad J_n(x,u) = \frac{1}{2}\begin{bmatrix}1\\x\\u\end{bmatrix}^\top \begin{bmatrix}\Gamma_n^{11} & \Gamma_n^{1x} & \Gamma_n^{1u}\\ \Gamma_n^{x1} & \Gamma_n^{xx} & \Gamma_n^{xu}\\ \Gamma_n^{u1} & \Gamma_n^{ux} & \Gamma_n^{uu}\end{bmatrix}\begin{bmatrix}1\\x\\u\end{bmatrix} \tag{B.41a}$$

$$\text{s.t.} \quad W_a x + S_a u + p_a = 0 \tag{B.41b}$$

We obtain an affine policy $u = K_a x + s_a$ and the matrices F, P, H following Lemma 18. Note that we are adding subscribes $_a$ to indicate values found associated with the active constraints set of a.

We apply the optimality conditions of VI problem [41] to find the set in which the policy is an equilibrium in terms of x

$$\begin{cases} Wx + S(K_a x + s_a) + p \leq 0 \\ -(F(K_a x + s_a) + Px + H) \in \text{cone}\{S_a^\top\} \end{cases} \tag{B.42}$$

where cone$\{S_a^\top\}$ means the cone generated by the rows of S_a. Condition (B.42) can be reformulated as a polyhedral constraint $L_a x + l_a \leq 0$, since the second condition is equivalent to an linear inequality as shown in Appendix B.1.3, where

$$L_a = \begin{bmatrix} W + SK_a \\ (S_a S_a^\top)^{-1} S_a (FK_a + P) \\ [I - S_a^\top (S_a S_a^\top)^{-1} S_a](FK_a + P) \\ [-I + S_a^\top (S_a S_a^\top)^{-1} S_a](FK_a + P) \end{bmatrix} \tag{B.43a}$$

$$l_a = \begin{bmatrix} Ss_a + p \\ (S_a S_a^\top)^{-1} S_a (F s_a + H) \\ [I - S_a^\top (S_a S_a^\top)^{-1} S_a](F s_a + H) \\ [-I + S_a^\top (S_a S_a^\top)^{-1} S_a](F s_a + H) \end{bmatrix} \tag{B.43b}$$

Hence we have found an affine policy $u = K_a x + s_a$ on a polyhedral region $\mathcal{X}_a := \{x \mid L_a x + l_a \leq 0\}$ in $\mathcal{X}$. In the case of $\mathcal{X}_a = \emptyset$, the policy is not an equilibrium.

The procedure can be repeated for all combinations of active constraints, i.e., $\forall a \in E^{n_c}$. For any $x \in \mathcal{X}$, there exists an equilibrium $u^\star(x)$ and a set of active constraints, therefore x must fall in one of the $\mathcal{X}_a$. Because there are a finite number of combinations of active constraints, a finite number of partitions of $\mathcal{X}$ with $\mathcal{X}_a$ and corresponding equilibria policy in each partition can be found.

□

B.2.3 Linearly Constrained Piecewise Quadratic Parametric Game

We focus on static game problems where each player's cost function $J_n(x,u)$ is continuous, strictly convex and piecewise quadratic on a polyhedral partition $\mathcal{P}$. Due to the complexity of the problem, we refrain from obtaining explicit solutions, but study the properties of the solution in this section.

Problem 24. ***Linearly Constrained Piecewise Quadratic Parametric Game***

$$\min_{u_n} \quad J_n(x,u) \tag{B.44a}$$

$$\text{s.t.} \quad z = [x,u] \in \mathcal{P} = \{\mathcal{Z}_i | i \in \mathcal{I}\} \tag{B.44b}$$

$$\mathcal{Z} = \{[x,y] | W_i x + S_i u + p_i \leq 0\} \tag{B.44c}$$

$$\mathcal{I} \text{ is an index set} \tag{B.44d}$$

Assume all players share the same polyhedral partition W.L.O.G and that $\mathcal{J}(u)$ is strongly monotone for any given x to guarantee the existence and uniqueness of a solution. In each partition $\mathcal{Z}_i$, the problem reduces to a linearly constrained quadratic parametric game as Problem 23, and F_i, P_i, H_i are inherited from Section B.2.1, where the subscript $_i$ denotes the values in $\mathcal{Z}_i$. Suppose there are n_P polyhedral partitions, i.e., $\mathcal{I} = 1, 2, ..., n_P$. The exact quadratic expression is suppressed. We define an auxiliary problem for each $i \in \mathcal{I}$.

Problem 25. ***Auxiliary problem*** $\mathbb{P}_i$

$$\min_{u_n} \quad J_n(x,u) \tag{B.45a}$$

$$\text{s.t.} \quad z \in \mathcal{Z} = \{[x,y] \mid W_i x + S_i u + p_i \leq 0\} \tag{B.45b}$$

Each auxiliary problem can be solved as in Section B.2.2, obtaining a piecewise affine policy and piecewise quadratic value functions. Since Problems 24 and 23 are closely connected, it is sensible to study the connection between the solutions of these two problems.

We define the concept of *active polyhedrons* $A_p([x,u])$ for $[x,u]$ in $\mathcal{P}$ as the set of indices $i \in \mathcal{I}$ such that $\mathcal{Z}_i$ contains $[x,u]$, i.e., $A([x,u]) = \{i \mid W_i x + S_i u + p_i \leq 0\}$. The next lemma explains how to recover the solution to Problem 24 from the auxiliary problems. We use the $\mathcal{U}(x)$ to indicate the set of feasible u with given x and define $\mathcal{U}_i(x) := \{u \mid [x,u] \in \mathcal{Z}_i\}$.

The following lemma answers the question of how to solve the equilibrium of Problem 24 for a given x with the help of auxiliary problems Problem 25.

Lemma 20. *$u^\star(x)$ is the open-loop Nash equilibrium to Problem 24 given parameter x if and only if it is the solution to $\mathbb{P}_i, \forall i \in A([x,u^\star(x)])$.*

Proof. Because for any given x there is a unique solution for Problem 24, a local solution is also the global solution. The equivalent VI problem for finding the local solution to Problem 24 is

$$\text{find } u^\star \text{ s.t. } \mathcal{J}(u^\star)^\top (u - u^\star) \geq 0 \tag{B.46a}$$

$$\forall u \text{ in a neighborhood of } u^\star(x) \tag{B.46b}$$

The condition (B.46b) can be expanded to $\forall u \in \cup\, \mathcal{U}_i(x), \forall i \in A_p([x,u^\star(x)])$. Due to the same argument that local and global solution are equivalent, this expansion does not change the solution found. Therefore, (B.46) is equivalent to $u^\star(x)$ being the solution to $\mathbb{P}_i$ simultaneously. We can find the solution of Problem 24 via the solutions of $\mathbb{P}_i$.

□

Next we focus on finding the feedback Nash equilibrium in terms of x. Suppose there are a

total of n_c inequalities. The inequalities defining all partitions can be collected together as

$$W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_{n_c} \end{bmatrix} \quad S = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n_c} \end{bmatrix} \quad p = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n_c} \end{bmatrix} \tag{B.47}$$

Therefore we can create a one-to-one mapping from an index set $j \in \{1, 2, ..., n_c\}$ to the inequality $W_j x + S_j + p_j \leq 0$. For any pair of (x, u), we can check the partition(s) it is living in and collect the indices in $A_p([x, u])$, and check the active constraints and collect them in $A_c([x, u])$. We call $A_c([x, u])$ *acitve inequalities*. Note that though $A_p([x, u])$ and $A_c([x, u])$ are tied to pairs of $[x, u]$, there could only be a finite number of different $A_p([x, u])$ and $A_c([x, u])$ since there are only a finite number of partitions and inequality constraints. In other words, we can find a finite number of representative pairs of $[x_k, u_k], k = 1, 2, ..., n_s$, such that $\mathcal{S}_p = \{A_p([x_k, u_k]) \mid k \in \{1, 2, ..., n_s\}\}$ and $\mathcal{S}_c = \{A_c([x_k, u_k]) \mid k \in \{1, 2, ..., n_s\}\}$ contain all possible active polyhedrons and active constraints combinations. We can get rid of the dependency on (x, u), use the index k to indicate different feasible active polyhedrons $A_p(k)$ and feasible active constraints $A_c(k)$. Finding the sets $\mathcal{S}_p$ and $\mathcal{S}_c$ requires studying the structure of the polyhedral partition $\mathcal{P}$, which is not within the scope of this paper.

The next lemma describes the feedback Nash equilibrium of Problem 24.

Lemma 21. *Given playerwise convexity of objective functions, i.e., Γ_n^{uu} are positive definite, Problem 24 has a piecewise affine feedback Nash equilibrium $u^\star = \phi^\star(x)$ on a finite polyhedral partition of $\mathcal{X}$ if and only if F_i is invertible $\forall i \in \mathcal{I}$.*

Proof. Similar to the proof for Lemma 19, we describe a procedure for finding affine policies and their corresponding region of x where they are feedback Nash equilibrium. Then we conclude by arguing all $x \in \mathcal{X}$ are included in the procedure.

We iterate through all n_s pairs of $A_p(k)$ and $A_c(k)$. For each k, we can solve for an affine policy $u^\star = K_k x + s_k$ with the feasible active inequalities $A_c(k)$ and the objective functions of all players as they constitute an instance of Problem 15. To find the region where this policy is indeed an FNE, we need to solve the corresponding optimality conditions

$$\begin{cases} W_i x + S_i(K_k x + s_k) + p_i \leq 0, \forall i \in A_p(k) \\ -(F_i(K_k x + s_k) + P_i x + H_i) \in \text{cone}\{S_{i,a}^\top\}, \forall i \in A_p(k) \end{cases} \tag{B.48}$$

where $S_{i,k}$ means picking the rows of active inequalities in S_i that are also in $A_p(k)$. The second condition (B.48) can be expressed with linear inequalities as in Appendix B.1.3, therefore the feasible region $\mathcal{X}_k$, if non-empty, is polyhedral, and $u^\star = K_k x + s_k$ is the FNE in this region. We omit the explicit expression here.

This procedure can be repeated $\forall k \in \{1, 2, ..., n_s\}$. For any x in $\mathcal{X}$, we can find $u^\star(x)$ via a solver for monotone VI problems, find the active polyhedrons and active inequalities, therefore the k. The solution $[x, u^\star(x)]$ is covered by the condition (B.48) with k because they share the same local optimality condition. Note that such procedure may require exponentially many steps w.r.t. the number of polyhedrons and inequality constraints.

□

B.2.4 Linearly Constrained Quadratic Dynamic Games

Now that we are equipped with some basic results, we can move on to dynamic games. A linearly constrained quadratic dynamic game is formulated as following. It is one of the most complicated form of dynamic games of which we can acquire analytical solution in theory. We briefly discuss the dynamic programming solution to such problems.

Problem 26. ***Linearly Constrained Quadratic Dynamic Games***

$$\min_{u_n} \quad J_n(x,u) = \sum_{k=0}^{T} \frac{1}{2} \begin{bmatrix} 1 \\ x_k \\ u_{:,k} \end{bmatrix}^\top \begin{bmatrix} \Gamma^{11}_{n,k} & \Gamma^{1x}_{n,k} & \Gamma^{1u}_{n,k} \\ \Gamma^{x1}_{n,k} & \Gamma^{xx}_{n,k} & \Gamma^{xu}_{n,k} \\ \Gamma^{u1}_{n,k} & \Gamma^{ux}_{n,k} & \Gamma^{uu}_{n,k} \end{bmatrix} \begin{bmatrix} 1 \\ x_k \\ u_{:,k} \end{bmatrix} \tag{B.49a}$$

$$s.t. \quad x_{k+1} = A_k x_k + B_k u_{:,k} + b_k \tag{B.49b}$$

$$W_{n,k} x + S_{n,k} u + p_{n,k} \leq 0 \tag{B.49c}$$

The first static problem required to be solved by dynamic programming is a piecewise quadratic, resulting in a piecewise affine policy and piecewise quadratic value functions on $\mathcal{X}_T$, which is the space of x_T. Based on the linear dynamics and constraints at step $T-1$, we can form a polyhedral partition for $[x_{T-1}, u_{T-1}]$, resulting in the next static game to be a linearly constrained piecewise quadratic parametric game. As we have seen in Section B.2.2, the solution remains to be a piecewise affine policy and quadratic value functions. Therefore, the backward pass by dynamic programming can be done obtaining a series of piecewise quadratic value functions with linear inequality constraints. However, the number of partitions on each

space $\mathcal{X}_k$ can grow exponentially causing the procedure to be computationally prohibiting. It is reasonable to believe that the feedback Nash equilibrium of general constrained dynamic games can be more complex. This fact drives us to seek local FNE.

Appendix C

Linear System Identification

C.1 Proof of Theorem 4

Note that s_t is overloaded and was used to indicate a different variable in Section 4.4.3.

Proof. Given (4.7) and $\eta_t = \theta_t - \theta_*$,

$$b_T = \sum_{t=0}^{T} g_t (g_t^\top \theta_* + g_t^\top \eta_t) \tag{C.1a}$$

$$= \sum_{t=0}^{T} g_t g_t^\top \theta_* + \sum_{t=0}^{T} g_t g_t^\top \eta_t \tag{C.1b}$$

$$= (V_T - \lambda I)\theta_* + \sum_{t=0}^{T} g_t g_t^\top \eta_t \tag{C.1c}$$

$$= (V_T - \lambda I)\theta_* + s_T \tag{C.1d}$$

where

$$s_T = \sum_{t=0}^{T} g_t g_t^\top \eta_t \tag{C.2}$$

We would like to check the estimation error

$$\overline{\theta_T} - \theta_* = V_T^{-1} b_T \tag{C.3a}$$

$$= V_T^{-1}(V_T - \lambda I)\theta_* + V_T^{-1} s_T - \theta_* \tag{C.3b}$$

$$= V_T^{-1} s_T - \lambda V_T^{-1} \theta_* \tag{C.3c}$$

where we used (C.1d). Therefore we need to bound s_T. The basic idea is to create a supermartingale and apply (4.6). We begin with the following construction with an arbitrary vector $\beta \in \mathbb{R}^d$.

$$\mathbb{E}_\eta[e^{\beta^\top s_T}|\mathcal{F}_{T-1}] = \mathbb{E}_\eta[e^{\beta^\top s_{T-1}}|\mathcal{F}_{T-1}]\mathbb{E}_\eta[e^{\beta^\top g_T g_T^\top \eta_T}|\mathcal{F}_{T-1}] \tag{C.4a}$$

$$= e^{\beta^\top s_{T-1}}\mathbb{E}_\eta[e^{\beta^\top g_T g_T^\top \eta_T}|\mathcal{F}_{T-1}] \tag{C.4b}$$

$$\leq e^{\beta^\top s_{T-1}} e^{\frac{1}{2}\sigma^2 \|g_T g_T^\top \beta\|^2} \tag{C.4c}$$

$$= e^{\beta^\top s_{T-1}} e^{\frac{1}{2}\sigma^2 \beta^\top g_T g_T^\top g_T g_T^\top \beta} \tag{C.4d}$$

$$= e^{\beta^\top s_{T-1}} e^{\frac{1}{2}\sigma^2 \beta^\top g_T g_T^\top \beta} \tag{C.4e}$$

where we applied (4.8b), the fact that s_{T-1}, g_T are determined given $\mathcal{F}_{T-1}$ and $g_T^\top g_T = 1$. We construct the following random process $M_T(\beta)$, aiming at creating a positive supermartingale, apply (4.6) and obtain a probabilistic bound

$$M_T(\beta) := e^{\beta^\top s_T - \frac{1}{2}\sigma^2 \beta^\top (V_T - \lambda I)\beta} \tag{C.5a}$$

$$= e^{\beta^\top s_T - \frac{1}{2}\sigma^2 \beta^\top \sum_{t=0}^{T} g_t g_t^\top \beta} > 0 \tag{C.5b}$$

$$\mathbb{E}_\eta[M_T(\beta)|\mathcal{F}_{T-1}] = \mathbb{E}_\eta[e^{\beta^\top s_T - \frac{1}{2}\sigma^2 \beta^\top \sum_{t=0}^{T} g_t g_t^\top \beta}|\mathcal{F}_{T-1}] \tag{C.5c}$$

$$= \mathbb{E}_\eta[e^{\beta^\top s_T}|\mathcal{F}_{T-1}] e^{-\frac{1}{2}\sigma^2 \beta^\top \sum_{t=0}^{T} g_t g_t^\top \beta} \tag{C.5d}$$

$$\leq e^{\beta^\top s_{T-1} - \frac{1}{2}\sigma^2 \beta^\top \sum_{t=0}^{T-1} g_t g_t^\top \beta} \tag{C.5e}$$

$$= M_{T-1}(\beta) \tag{C.5f}$$

where (C.4) is applied. Equation (C.5) proves $M_T(\beta)$ is indeed a positive supermartingale w.r.t. the filtration $\mathcal{F}_T$, for $T = 1, 2, \ldots$.

Directly following (C.5), choose $\beta \sim \mathcal{N}(0, (\sigma^2\lambda)^{-1} I)$ and independent of other random variables, then $\mathbb{E}_\beta[M_T(\beta)]$ is still a positive supermartingale

$$\bar{M}_T = \mathbb{E}_\beta[M_T(\beta)] = \frac{|(\sigma^2\lambda)^{-1} I|^{-\frac{1}{2}}}{(2\pi)^{\frac{d}{2}}}$$

$$\int e^{\beta^\top s_T - \frac{1}{2}\sigma^2 \beta^\top (V_T - \lambda I)\beta} e^{-\frac{1}{2}\beta^\top ((\sigma^2\lambda)^{-1} I)^{-1}\beta} \mathrm{d}\beta \tag{C.6a}$$

$$= \left(\frac{\sigma^2\lambda}{2\pi}\right)^{\frac{d}{2}} \int e^{\beta^\top s_T - \frac{\sigma^2}{2}\beta^\top V_T \beta} \mathrm{d}\beta \tag{C.6b}$$

$$= \left(\frac{\sigma^2\lambda}{2\pi}\right)^{\frac{d}{2}} e^{\frac{1}{2} s_T^\top (\sigma^2 V_T)^{-1} s_T} \tag{C.6c}$$

$$\int e^{-\frac{1}{2}(\beta-(\sigma^2 V_T)^{-1}s_T)^\top(\sigma^2 V_T)(\beta-(\sigma^2 V_T)^{-1}s_T)}\mathrm{d}\beta \tag{C.6d}$$

$$=\left(\frac{\sigma^2\lambda}{2\pi}\right)^{\frac{d}{2}} e^{\frac{1}{2}s_T^\top(\sigma^2 V_T)^{-1}s_T}\frac{(2\pi)^{\frac{2}{d}}}{|(\sigma^2 V_T)^{-1}|^{-\frac{1}{2}}} \tag{C.6e}$$

$$=\frac{\lambda^{\frac{d}{2}}}{|V_T|^{\frac{1}{2}}} e^{\frac{1}{2}s_T^\top(\sigma^2 V_T)^{-1}s_T} \tag{C.6f}$$

We check the initial condition

$$\bar{M}_0 = \mathbb{E}_\eta[M_0(\beta)] = e^{\beta^\top s_0 - \frac{1}{2}\sigma^2\beta^\top g_0 g_0^\top \beta} \tag{C.7a}$$

$$\leq e^{\frac{1}{2}\sigma^2\beta^\top g_0 g_0^\top\beta - \frac{1}{2}\sigma^2\beta^\top g_0 g_0^\top\beta} = 1 \tag{C.7b}$$

where (C.4) is applied for $T=0$. Therefore $\mathbb{E}_\beta[\bar{M}_0]\leq 1$. Apply the maximal inequality (4.6) to $\bar{M}_T$,

$$\mathbb{P}(\sup_T \log \bar{M}_T \geq \log\frac{\mathbb{E}_\beta[\bar{M}_0]}{\varepsilon}) \leq \varepsilon \tag{C.8a}$$

$$\mathbb{P}(\sup_T \log \bar{M}_T \geq \log\frac{1}{\varepsilon}) \leq \varepsilon \tag{C.8b}$$

Therefore, with probability at least $1-\varepsilon$, we have the following self-normalizing bound for s_T

$$\log \bar{M}_T \leq \log\frac{1}{\varepsilon} \tag{C.9a}$$

$$\frac{1}{2}s_T^\top(\sigma^2 V_T)^{-1}s_T + \log\frac{\lambda^{\frac{d}{2}}}{|V_T|^{\frac{1}{2}}} \leq \log\frac{1}{\varepsilon} \tag{C.9b}$$

$$\left\|V_T^{-\frac{1}{2}}s_T\right\|^2 = s_T^\top V_T^{-1}s_T \leq \sigma^2\log\frac{|V_T|}{\varepsilon^2\lambda^d} \tag{C.9c}$$

Based on the estimation error of θ (C.3)

$$V_T^{\frac{1}{2}}(\theta_T-\theta_*) = V_T^{-\frac{1}{2}}s_T - \lambda V_T^{-\frac{1}{2}}\theta_* \tag{C.10}$$

The self-normalizing bound for $\theta_T-\theta_*$ becomes

$$\left\|V_T^{\frac{1}{2}}(\overline{\theta_T}-\theta_*)\right\| \leq \left\|V_T^{\frac{1}{2}}s_T\right\| + \lambda\left\|V_T^{-\frac{1}{2}}\theta_*\right\| \tag{C.11a}$$

$$\leq \sqrt{\sigma^2\log\frac{|V_T|}{\varepsilon^2\lambda^d}} + \lambda\left\|V_T^{-\frac{1}{2}}\theta_*\right\| \tag{C.11b}$$

$$\leq \sqrt{\sigma^2\log\frac{(\lambda+T)^d}{\varepsilon^2\lambda^d}} + \lambda\left\|V_T^{-\frac{1}{2}}\theta_*\right\| \tag{C.11c}$$

where we used the fact that g_t is normalized, so the maximal eigenvalue of V_T is bounded above by $\lambda + T$ and V_T is positive semi-definite. Let $\lambda_{\min}(V_T)$ be the minimal eigenvalue of V_T, then (C.11c) can be relaxed as

$$\left\|\overline{\theta_T} - \theta_*\right\| \leq \sqrt{\frac{\sigma^2}{\lambda_{\min}(V_T)} \log \frac{(\lambda+T)^d}{\varepsilon^2 \lambda^d}} + \frac{\lambda}{\sqrt{\lambda_{\min}(V_T)}} \left\| V_T^{-\frac{1}{2}} \theta_* \right\| \tag{C.12a}$$

$$\leq \sqrt{\frac{\sigma^2}{\lambda_{\min}(V_T)} \log \frac{(\lambda+T)^d}{\varepsilon^2 \lambda^d}} + \frac{\lambda}{\lambda_{\min}(V_T)} \|\theta_*\| \tag{C.12b}$$

□

C.2 Proof of Lemma 8

Proof. From (4.13) we know, the following inequalities hold with probability at least $1-\varepsilon_1$

$$\overline{X} - \delta(\overline{X}) \leq \mathbb{E}[X] \leq \overline{X} + \delta(\overline{X}) \tag{C.13a}$$

and the following holds with probability $1-\varepsilon_2$

$$\overline{X^2} - \delta(\overline{X^2}) \leq \mathbb{E}[X^2] \leq \overline{X^2} + \delta(\overline{X^2}) \tag{C.13b}$$

From here on we assume both the bounds hold, which is true with at least probability of $1-\varepsilon_1-\varepsilon_2$.

It is also assumed through out the paper we are dealing with $\sigma-$sub-Gaussian random variables

$$\mathbb{E}[e^{\beta(X-\mathbb{E}[X])}] \leq \frac{1}{2} e^{\sigma^2\beta^2}, \forall \beta \in \mathbb{R} \tag{C.14}$$

The triangle inequality suggests

$$\left|X - \overline{X}\right| \leq |X - \mathbb{E}[X]| + \left|\mathbb{E}[X] - \overline{X}\right| \tag{C.15a}$$

$$\leq |X - \mathbb{E}[X]| + \delta(\overline{X}) \tag{C.15b}$$

We then focus on bounding $|X - \mathbb{E}[X]|$.

$$\Pr[|X - \mathbb{E}[X]| > R] = \Pr[X - \mathbb{E}[X] > R] + \Pr[X - \mathbb{E}[X] < -R] \tag{C.16a}$$

where R is the box radius for the bound.

$$\Pr[X - \mathbb{E}[X] > R] = \Pr[e^{\gamma(X-\mathbb{E}[X])} > e^{\gamma R}] \tag{C.17a}$$
$$\leq e^{-\gamma R}\mathbb{E}[e^{\gamma(X-\mathbb{E}[X])}] \tag{C.17b}$$
$$\leq e^{\frac{1}{2}\sigma^2\gamma^2 - R\gamma} \tag{C.17c}$$

where $\gamma > 0$ is a free parameter, (C.17b) and (C.17c) applied the Markov's inequality and the sub-Gaussian property, respectively. Similarly, we can bound the other term

$$\Pr[X - \mathbb{E}[X] < -R] = \Pr[-(X - \mathbb{E}[X]) > R] \tag{C.18a}$$
$$= \Pr[e^{-\gamma(X-\mathbb{E}[X])} > e^{\gamma R}] \tag{C.18b}$$
$$\leq e^{-\gamma R}\mathbb{E}[e^{-\gamma(X-\mathbb{E}[X])}] \tag{C.18c}$$
$$\leq e^{\frac{1}{2}\sigma^2\gamma^2 - R\gamma} \tag{C.18d}$$

To minimize the probability of X falling out of the box, we minimize $\frac{1}{2}\sigma^2\gamma^2 - R\gamma$ with choice of $\gamma = \frac{R}{\sigma^2}$, so that

$$\Pr[|X - \mathbb{E}[X]| > R] \leq 2e^{\frac{1}{2}\sigma^2\gamma^2 - R\gamma} \tag{C.19a}$$
$$\Pr[|X - \mathbb{E}[X]| > R] \leq 2e^{-\frac{R^2}{2\sigma^2}} \tag{C.19b}$$

Combining (C.15) and (C.19b), we have the box bound with probability at least $1 - \varepsilon_1 - 2e^{-\frac{R^2}{2\sigma^2}}$

$$\left|X - \overline{X}\right| \leq R + \delta(\overline{X}) \tag{C.20}$$

If we further assume that X follows a uniform distribution $\mathcal{U}(\mathbb{E}[X] - R, \mathbb{E}[X] + R)$ whose variance is $\mathrm{Var}[X] = \frac{1}{3}R^2$. Based on the basic property of variance

$$\mathbb{E}[X^2] - \mathbb{E}[X]^2 = \mathrm{Var}[X] \geq 0 \tag{C.21}$$

and (C.13), we have an upper bound for the variance

$$\mathrm{Var}(X) = \frac{1}{3}R^2 \leq \overline{X^2} + \delta(\overline{X^2}) - \min\{[\overline{X} \pm \delta(\overline{X})]^2\} \tag{C.22}$$

therefore an upper bound on the range R

$$R \leq \sqrt{3}\sqrt{\overline{X^2} + \delta(\overline{X^2}) - \min\{[\overline{X} \pm \delta(\overline{X})]^2\}} \tag{C.23}$$

This bound does not require any additional assumptions, so it holds with probability at least $1 - \varepsilon_1 - \varepsilon_2$. Applying the triangle inequality, we can get the same expression as in (C.20). □