Convergence to Nash in the Potential Linear Quadratic Games and Accelerated Learning in Games

by

Alireza Alian Porzani

B.Sc., Sharif University of Technology, 2019

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

October 2022

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Convergence to Nash in the Potential Linear Quadratic Games and Accelerated Learning in Games

submitted by **Alireza Alian Porzani** in partial fulfillment of the requirements for the degree of **Master of Applied Sciences** in **Electrical and Computer Engineering**

Examining Committee:

Dr. Maryam Kamgarpour, Assistant Professor, Sycamore lab, EPFL

Supervisor

Dr. Lele Wang, Assistant Professor, Electrical and Computer Engineering, UBC

Co-supervisor

Dr. Lutz Lampe, Professor, Electrical and Computer Engineering, UBC

Supervisory Committee Member

Abstract

Game theory and online optimization have a close relationship with each other. In some literature, online optimization has been employed for solving game theory problems. Many accelerated algorithms are proposed for offline optimization problems. However, to the best of our knowledge, there is not enough work done to accelerate zero-order online optimization. The goal is to propose a Nesterov accelerated online algorithm with the hope that it will converge to the Nash, with a fast convergence rate in Cournot games and Quadratic games. It is desired that this online algorithm also minimize the regret of a sequence of functions for both zero-order and first-order feedback.

In potential Linear Quadratic (LQ) games, we also study the convergence of the policy gradient algorithms, a class of conventional reinforcement learning methods. LQ games have applications in engineering. It has been shown that using policy gradient algorithms by agents does not guarantee convergence to the Nash equilibrium. However, in the LQR problem, which is essentially a one-player LQ game, the policy gradient converges to the optimum. In this work, we show that using policy gradient algorithms leads to convergence to Nash equilibrium in potential LQ games. Additionally, we identify the characteristics of potential games in both open-loop and closed-loop settings. We will demonstrate that the class of closed-loop potential games is generally trivial, and if we put restrictions on players' actions, we can have non-trivial potential games too.

Lay Summary

Optimization in a multi-agent and a single-agent environments have some similarities and differences. In a multi-agent setting, due to the presence of other agents, co-operations and competitions might help us or hold us back from getting to the optimum point. This thesis works on studying and developing methods to lead the agents to their desired stable point. In this work, we specifically narrow our focus on convex games and Linear Quadratic games.

Preface

This thesis is an original, unpublished work by the author, Alireza Alian Porzani. The basis of this thesis is unpublished work supervised by Maryam Kamgarpour. The second chapter, Accelerated Algorithms in games, was written with Mohammad Amin Roohi. The mathematical analyzes in part 2.3 and 2.4 is done by him. And the third chapter, LQ games, was written with Sara Hosseinirad. To be more precise 3.3.1 is completely done by her.

Table of Contents

Abstract . . . iii

Lay Summary . . . iv

Preface . . . v

Table of Contents . . . vi

List of Tables . . . viii

List of Figures . . . ix

Acknowledgements . . . x

1 Introduction . . . 1
- 1.1 Motivation . . . 2
- 1.2 Thesis Organization . . . 3
- 1.3 Problem Settings . . . 3
  - 1.3.1 Game and Converging to the Nash . . . 3
  - 1.3.2 LQ Games . . . 5

2 Accelerated algorithms in games . . . 8
- 2.1 Preliminaries for Online algorithms and games . . . 8
- 2.2 Algorithm for noisy gradient feedback . . . 10
- 2.3 3-stage online Nesterov method . . . 14
- 2.4 2-stage online Nesterov method . . . 20
  - 2.4.1 Quadratic Games . . . 25
- 2.5 Conclusion and Summary . . . 29
- 2.6 Next steps . . . 30

3 LQ games . . . 32
- 3.1 Review of the definitions . . . 32
  - 3.1.1 Dynamic System . . . 32

3.1.2 Quadratic Cost Function 33
3.1.3 Stochasticity in LQ games 36
3.1.4 Potential Games 36
3.2 An examples of LQ game 38
3.3 Potential games and their Conditions 39
3.3.1 Convergence of Potential LQ games 39
3.3.2 Open loop potential LQ games 40
3.3.3 Closed loop potential LQ games 42
3.3.4 Special cases of closed-loop 47
3.4 Simulations 49
3.5 Discussion 52

4 Conclusion and Future works 53

Bibliography 54

Appendices

A Appendix 57

List of Tables

2.1 We present the regret order of different online algorithms in this table. All these functions are differentiable and smooth. The best regret rate can be achieved without strong convexity $\mathcal{O}(\sqrt{T})$. 30

List of Figures

2.1 Gradient descent and accelerated solution 26
2.2 Gradient descent and accelerated solution 27
2.3 Gradient descent and accelerated solution 27
2.4 Convergence Trajectories 28
2.5 Convergence Trajectories 29

3.1 Closed loop and open loop flowchart 34
3.2 Convergence Trajectories 50
3.3 Trivial Closed Loop Potential 51
3.4 Special Potential Case 52

Acknowledgements

First of all, to my parents and my sister, who warmly supported me throughout my journey away from home. They were always available to me, even from thousands of miles away. Without their help, I would not be here the place I am.

To my supervisor, Dr. Maryam Kamgarpour, who supported me throughout this long process. Even though we were working in different time zones, and she had lots of personal and professional tasks in her life, she always helped me with feedback that shed light along my research path.

To Dr. Lele Wang, who was responsible for being my formal supervisor and her helpful reading group. She always lifted my spirit and helped me when I had a meeting with her. To my dear friend and lab mate, Mohammad Amin Roohi, who was an important part of my trip to Canada and doing this research.

To my friend and group mate, Sara Hosseinirad, who always helped me with her very helpful comments throughout this work. She has made this work a lot better with her work and feedbacks that she provided to me. To my friend and lab mate, Kai Ren, who kept following my progress and provided feedbacks during our meetings.

Chapter 1

Introduction

Game theory studies mathematical models of rational agents when they have strategic interactions [19]. It comes into use whenever we have several agents in our environment, and we want to understand the behavior they show to increase their utility or minimize their loss. We may see applications of game theory in economics where there are multiple rational agents involved and strategical scenarios such as supply and demand markets and auctions. Game theory also has many applications in engineering, where the goal is to control a system in the presence of a disturbance, which is seen as an adversary agent. Game theory can also come in handy when we wish to optimize several scalar functions at the same time (multi-objective optimization) [18].

On the other hand, online optimization is popular in computer science and is the study of optimizing an unknown time-varying function. As the target function is not constant over time, optimization might seem meaningless because we can't do optimization for every single time step. Moreover, optimization may seem unreasonable as the function is unknown, so that we cannot know the optimal point in every time step and compute it. To overcome this problem, *regret* is used as a means to see how well the online algorithm is performing. Regret is a function that evaluates the overall proximity of our selected points to the best single point one could choose in hindsight. In chapter two, we introduce the definition of regret that is used in our work.

Game theory and online optimization are strongly connected while having their own differences. In a game, each player is optimizing a function that is also a function of other players' actions, so it changes when other players decide to change what they are playing. Thus, we can see that each player is solving an online optimization problem. Moreover, in online optimization, we are interested in having a sublinear regret (which will be talked about later on) because it is reasonable to expect that our overall payoff is not much worse than the case we select the best-fixed point [16]. An online algorithm that has a sublinear regret is called a no-regret algorithm. Contrary to many optimization problems, where it is allowed to assume that

the first order or higher orders of feedback are provided to us, in games it is usually assumed that zero-order feedback is given because we can only ask for the value of the function for one extra point at each step, which is more realistic.

1.1 Motivation

As opposed to the conventional optimization problems, in games, we are interested in **converging** to the Nash equilibrium instead of finding a point that optimizes one single function. The goal is to find algorithms that the players that are participating in the game employ, and as an outcome, they reach the Nash equilibrium. In games, it is not enough that each player plays a no-regret algorithm because there are examples where if all the players employ a no-regret algorithm, the player's actions may cycle in perpetuity [15, 28] and never converge to the Nash. It is also demonstrated that no-regret algorithms lead to the min-max optimal solution for the class of zero-sum games [6], demonstrating that if all players use a no-regret algorithm, their actions will converge to the Nash equilibrium, and the convergence rate depends on the regret of the algorithm that is used, i.e., the smaller the regret is, the faster the convergence to the Nash.

Generally, to the best of our knowledge, there has been little work done to improve the convergence rate of players' actions to the Nash equilibrium [5]. Vaswani et al. [25] tried to use a Nesterov algorithm to accelerate stochastic optimization, and they were able to obtain a convergence guarantee with a proof of rate. Nesterov Algorithms were not used in games or online learning in any of the works mentioned above. The motivation is to change Nesterov Algorithms to online algorithms to have a better game convergence rate.

In addition, there is a further convergence issue that we will address in this thesis. Here we are given a family of algorithms, *Policy Gradient*, used mainly in reinforcement learning. In a single agent mode, [4] found convergence guaranteed under some conditions. However, [13] showed that in a multi-agent setting, there exist examples where using a policy gradient leads to cycling around the Nash equilibrium. We also know that the class of potential games behaves like a single-player system. Finding the class of potential LQ games and determining whether or not they have a convergence guarantee is motivating.

1.2 Thesis Organization

In the rest of this chapter, we bring the basic concepts, mathematical definitions, and the formulation of our problems into a simple mathematical form. Furthermore, we go over some of the most relevant works on the topic and their outcomes to see what has been done and what has been lacking in this field.

In Chapter 2, we focus specifically on accelerated algorithms, go through the concepts underlying them, extend them to online optimization, and see how well they work there. We present our accelerated online algorithms and analyze their performance when players of a game follow them to find their optimal actions. We are particularly interested when they are used in games because acceleration has not previously been employed in them.

Chapter 3 discusses the problems that may occur in a multi-agent LQ system and studies a subset of such games where policy gradient methods will safely ensure actions converge to Nash equilibrium. We demonstrate how potential games can show single-player behavior and identify the simpler LQ games that are potential. We also raise some conjectures about the general form of the potential LQ games. The last chapter summarizes the results and draws conclusions about what is contributed to this field, the upcoming questions, and possible future work in this area.

1.3 Problem Settings

1.3.1 Game and Converging to the Nash

In a game, there are N players, each with a utility function. For example, the utility function of player i is $f_i(x) = f_i(x_i; x\_i) : X = X_1 \times \cdots \times X_N \to \mathbb{R}$, where $x_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N)$, $x_j \in X_j$ where X_j is set of available actions for player j. The trio of (N, f, X) defines a game. If the players are rational, which we assume they are, they will aim to maximize their utility function.

Example: Quadratic games are defined as the games with the following cost function (Negative of the utility function) to the players:

$$f_i(x_i, x_{-i}) = \frac{1}{2} x_i^T \Gamma_{ii} x_i + x_i^T g_i \Gamma_{i,-i} x_{-i} + x_i^T + h_i(x_{-i}). \tag{1.1}$$

with $\Gamma_{ii} \succ 0$ and $x_i \in \mathbb{R}^{d_i}$.

As we mentioned in the example above, players may have cost functions instead of utility functions; in other words, rational players wish to mini-

mize the given function instead of maximizing it. These two problems are equivalent because we can use the old trick of having a cost function with the value of a negative utility function.

$$\arg\min_x f(x) = \arg\max_x -f(x) \tag{1.2}$$

Example: In a matrix game, the utility function can be written in matrix form. For example, consider a two-player game, where each player can play one action 1 or 2. The associated utility function to the players is shown in the following matrix:

$$F = \begin{bmatrix} (1,-1) & (3,-3) \\ (-2,2) & (-4,4) \end{bmatrix}. \tag{1.3}$$

$F_{i,j}$, where $i, j \in \{1, 2\}$ is the corresponding utility pair for our players, when the first player plays i and the second player plays j. For example if first player plays 1 and the second player plays 2, the first player will be rewarded by 3 and the second player will be rewarded by -3.

By paying attention to the utility function in 1.3, one can realize that the sum of utilities in each scenario is zero. This family of games where the sum of utilities is zero are called **zero-sum games**. Another notable note in 1.3 is that if both players play action 1, they do not have any motivation to change their actions. In other words, if only one of them single-handedly attempts to change its action, it will get a lower reward. It implies that they are in an equilibrium called *Nash Equilibrium*.

Definition 1.3.1. *Nash equilibrium is called a choice of actions* $(x_1^*, \ldots, x_N^*) \in \Pi_{i=1}^N X_i = X$ *that satisfies the following inequalities. For every* $x_i \in X_i$ *and for all* $i \in \{1, 2, \ldots, N\}$*, we have:*

$$f_i(x_1^*, \ldots, x_{i-1}^*, x_i, x_{i+1}^*, \ldots, x_N^*) \leq f_i(x_1^*, \ldots, x_{i-1}^*, x_i^*, x_{i+1}^*, \ldots, x_N^*). \tag{1.4}$$

Definition 1.3.2. *Concave (Convex) games are the games where each player's set of action are convex and also* $f_i(x_i, x_{-i})$ *is concave (convex) with respect to* x_i*.*

Examples of convex games are quadratic games are introduced earlier.

Example: *Cournot game* is defined as an N player game with payoff function of player i as $f_i(x) = p(\sum_{j=1}^N x_j)x_i - C_i(x_i)$, $x_i \geq 0$. In the aforementioned formula, $p(.)$ represents the product's market price, which is a function of the product's global supply in the entire market, whereas $C_i(.)$ represents the product's production cost for player i [24].

Under the following assumptions on $p(.)$ and $C_i(.)$, Cournot games are concave and admit unique Nash equilibrium.

1. $p(.)$ is concave, twice differentiable, $p(0) > 0$, and strictly decreasing when $p(x) > 0$.

2. C_i are convex, strictly increasing and twice differentiable with $p(0) > C_i'(0)$

$$f_i''(x) = p''(\sum x_j)x_i + 2p'(\sum x_j) - C_i''(x_i) < 0.$$

As a result, as shown above, $f_i(x)$ is concave in x_i for all i, implying that the game is concave.

Definition 1.3.3. *A game is monotone if it satisfies the following for $x, x' \in X$:*

$$\langle m(x) - m(x'), x - x' \rangle \leq 0, \tag{1.5}$$

where $m(x) = \begin{bmatrix} \nabla_1 f_1(x) \\ \vdots \\ \nabla_N f_N(x) \end{bmatrix}$ *is the game map, and the equality happens only if* $x = x'$.

There is a famous result that monotone concave games will go to their unique Nash equilibrium when players use gradient methods to optimize their actions. However, with the mentioned assumptions, we may not have a monotone concave Cournot game, so the former result does not imply anything.

Shi and Zhang [24] proved that employing no-regret algorithms by the players leads to convergence in measure to Nash equilibrium. *Convergence in measure* means that the sequence of actions will converge to optimal actions with the probability of 1.

Definition 1.3.4. *Let μ be a measure on $\mathbb{N}$, convergence in measure of a_t to a means that $\forall \epsilon > 0$, $\lim_{t\to\infty} \mu(\|a_t - a^*\| > \epsilon) = 0$ [24].*

1.3.2 LQ Games

A *Linear Quadratic (LQ) optimization problem* is defined as minimizing a loss function of $\sum_{t=0}^{T} c_t x_t . u_t = x_t^T Q x_t + u_t^T R u_t$ (quadratic cost), where x_t is the state of the system, and u_t is the input to the system at the time t.

And the dynamics of this system is linear with respect to states and inputs (actions). $x_{t+1} = Ax_t + Bu_t + w_t$, where w_t represents the noise at time t.

$$\min \sum_{t=0}^{t_f} x_t^T Q x_t + u_t^T R u_t, \tag{1.6}$$
$$\text{such that } x_{t+1} = Ax_t + Bu_t + w_t.$$

This game can have different properties depending on the length of the time horizon. Dynamic games are usually divided into these two classes:

- **Finite Horizon LQ System:** A LQ system is called a *finite horizon* if $t_f < \infty$.
- **Infinite Horizon LQ System:** A LQ system is called an *infinite horizon* if $t_f = \infty$. It is proven [2] that if A and B are such that the minimum cost is finite, the optimal play for such closed-loop games are in the form of $u_t = -Kx_t$. The coefficient K is derived from algebraic Riccati Equation. In this class of problem we are also concerned of stability, i.e. if we choose coefficients K in a careless way, as the cost function is an infinite series, we may end up with an unbounded cost function.

LQ systems can also be divided into the following categories:

- **A system with known parameters:** In this setting, we are given with the all parameters such as Q, R, A, B, and to achieve the optimal play, we can use Algebraic Riccati Equations written bellow [1]

$$P_{t_f}^* = Q_{t_f} \tag{1.7}$$
$$P_t^* = Q_t + A_t^T(P_{t+1}^* - P_{t+1}^* B_t(B_t^T Pt + 1^* B_t + R_t)^1 B_t^T P_{t+1}^*)A_t, \tag{1.8}$$
$$K_t^* = (R_t + B_t^T P_{t+1}^* B_{t+1})^{-1} B_t^T P_{t+1}^* A_t. \tag{1.9}$$

- **A system with unknown parameters:** In this setting, we want to optimize a system without knowing the parameters. This problem is also known as the model-free LQ system. When we play the input K_1 (or u_1 in an open-loop setting), we get the feedback $J_i(x, K_1)$ (the loss function). One common way to approach an optimal solution is to use *Policy Gradient* (PG) methods.

It is proven that under mild assumption, in the infinite horizon LQ systems PG will give us the global optimum point [13].

The introduced LQ settings above can be extended to a multi-agent setting and be seen as a game. A LQ game is defined as an N player dynamic game, with the following cost functions:

$$J_i(x,u) = \sum_{t=0}^{t_f} x_t^T Q_i x_t + u_{i,t}^T R_i u_{i,t}, \tag{1.10}$$

and

$$x_{t+1} = Ax_t + \sum_{i=1}^{N} B_i u_{i,t} + w_t. \tag{1.11}$$

The Nash can be calculated by the following coupled Algebraic Riccati Equation [7]:

$$P_{i,t}^* = Q_{i,t} + (K_{i,t}^*)^T R_{i,t} K_{i,t}^* + (A_t - \sum_{j=1}^{N} B_{j,t} K_{j,t}^*)^T P_{i,t+1}^* (A_t - \sum_{j=1}^{N} B_{j,t} K_{j,t}^*), \tag{1.12}$$

$$K_{i,t}^* = (R_{i,t} + B_{i,t}^T P_{i,t+1}^* B_{i,t+1})^{-1} B_{i,t}^T P_{i,t+1}^* (A_t - \sum_{j=1, j\neq i}^{N} B_{j,t} K_{j,t}). \tag{1.13}$$

Mazumdar [13] showed that, in a multi-agent LQ system, there is no guarantee of convergence to Nash using PG. However, in zero-sum LQ games [27] there is a convergence guarantee. There is a class of games called *Potential* games, which can behave like a single-player game. As we have a convergence guarantee in the single-player mode, it motivates us to identify such a class of games and study if potential LQ games converge to their Nash equilibrium.

Chapter 2

Accelerated algorithms in games

We present three new accelerated online optimization algorithms, adapted from the Nesterov Accelerated algorithms in offline optimization, by setting the parameters, and determine their regret. The goal is to speed up convergence in games that feature a convex set of actions. The online algorithms that are introduced in this chapter are inspired by non-online Nesterov algorithms that already exist but have never been used in an online algorithm. Additionally, we demonstrate through simulations that applying these algorithms speeds up convergence in quadratic and Cournot games

In games, from each individual's perspective with utility function of $u_i(x_{i,t}; x_{-i,t})$, if we restrict its domain to his controllable actions, the payoff would be a time-varying function $f_t(x_i) = u_i(x_{i,t}; x_{-i,t})$. It means that each player is performing some online optimization. In order to accelerate the convergence in the game, we try to use accelerated algorithms in online setting and hope that when the players are using the new algorithms, the convergence speed improves.

2.1 Preliminaries for Online algorithms and games

A function $f(x) : D \to \mathbb{R}$ is called L-Lipschitz continuous if:

$$|f(x) - f(y)| \leq L\|x - y\|_2 \qquad \forall x, y \in D, \tag{2.1}$$

and it is called M-smooth if the gradients $\nabla f(x)$ are M-Lipschitz continuous, i.e.,

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq M\|x - y\|_2. \tag{2.2}$$

Similarly we call a convex function $\mu-$strongly convex if:

$$\mu\|x - y\|_2 \leq \|\nabla f(x) - \nabla f(y)\|_2, \tag{2.3}$$

for every $x, y \in D$. In the above definitions $L, M, \mu > 0$. Besides M-smoothness and $\mu-$strongly convex condition for convex functions, respectively are equivalent to:

$$0 \leq f(x) - f(y) - \nabla f(x)^\top (x - y) \leq \frac{M}{2}\|x - y\|_2^2. \tag{2.4}$$

$$f(x) - f(y) \leq \nabla f(x)^\top (x - y) - \frac{\mu}{2}\|x - y\|_2^2. \tag{2.5}$$

Definition 2.1.1. *[8] Let $\mathcal{A}$ be a deterministic algorithm for online convex optimization (OCO), which at each time step t, selects action $x_t \in \mathcal{X}$, where $\mathcal{X}$ is the set of possible actions. After x_t is selected, algorithm incurs a cost of $f_t(x_t)$. For all t, f_t is unknown and chosen by an oblivious adversary. We assume all the payoff functions $f_1, \ldots, f_T \in \mathcal{F}$, where $\mathcal{F}$ is the set of bounded functions. So, after T iterations, the total cost collected by algorithm $\mathcal{A}$ is $\sum_{i=1}^{T} f_t(x_t)$. In addition, the total cost of a static feasible action x_0 is $\sum_{i=1}^{T} f_t(x_0)$. We formally define the regret of $\mathcal{A}$ after T iterations as:*

$$Reg_{\mathcal{A}}(T) = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} f_t(x). \tag{2.6}$$

Also, in case the algorithm is non-deterministic, we generalize the definition regret by taking an expectation over uncertainties:

$$Reg_{\mathcal{A}}(T) = \mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t)\right] - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} f_t(x).$$

Intuitively, an algorithm performs well if its regret is sublinear as a function of T, i.e., $Reg_{\mathcal{A}}(T) = o(T)$, since this implies that on the average the algorithm performs as well as the best fixed strategy on the hindsight. In this case, we say $\mathcal{A}$ is a no-regret algorithm.

In convex offline optimization, gradient descent is the most common way to get the optimum point. $x_{t+1} = x_t - \eta_t \nabla f(x_t)$ is the update rule. The problem with this algorithm is that, the rate of convergence is slow, i.e. $\mathcal{O}(1/t)$. That is, $f(x(t) - f(x^*)$ is decreasing proportionally to $1/t$. To address this problem, there are accelerated algorithms such as *Polyak Heavy Ball* [22] or *Nesterov Methods* [20]. The update rules for the Nesterov method are as follows:

$$\begin{cases} x_{t+1} = y_t - \eta_t \nabla f(y_t) \\ y_{t+1} = x_{t+1} + \frac{t}{t+3}(x_{t+1} - x_t) \end{cases}$$

This algorithm and its variant to an online optimization algorithm are used to accelerate game convergence.

2.2 Algorithm for noisy gradient feedback

Here we try to solve an online optimization problem with first-order feedback, with Nesterov's accelerated method 1, under the assumption of bounded variation of objective functions. In other words, we are trying to minimize functions $f_t(x)$, with the property that "they change slowly" at each step. Formally we can write:
For each t here exist a ϵ_t such that for all x we have:

$$|f_{t+1}(x) - f_t(x)| \leq \epsilon_t, \tag{2.7}$$

where ϵ_t is fairly small. Our motivation for this assumption comes from the fact that we will use such an algorithm in games, and as cost functions that we are considering in games are not changing and they are smooth, such an assumption can be realistic.

We also assume that for all f_t and all x (strong growth with additive noise):

$$\mathbb{E}_{z_t} \|\nabla f_t(x_t, z_t)\|^2 \leq \rho \|\nabla f_t(x_t)\|^2 + \sigma_t^2. \tag{2.8}$$

Our first proposed online algorithm is as follows:

Algorithm 1 Accelerated Online Nesterov

1: Starting from w_0, ν_0 and arbitrary point ζ_0
2: **for** $k = 1, 2, \ldots, T$ **do**
3: Update $w_{k+1} = \zeta_k - \eta \nabla f_k(\zeta_k, z_k)$
4: Update $\zeta_k = \alpha_k \nu_k + (1 - \alpha_k) w_k$
5: Update $\nu_{k+1} = \beta_k \nu_k + (1 - \beta_k)\zeta_k - \gamma_k \eta \nabla f_k(\zeta_k, z_k)$
6: **end for**

This is fully motivated by [25], where they used such an algorithm to improve stochastic gradient descent. We extended their algorithm for a case where the function is not constant and is time-varying. In the following section, we will see how we can reduce this algorithm to an online zero-order algorithm.

In addition to the above assumptions, we assume the followings:

- **Strong convexity and smoothness:** Here, we assumed that our functions $\mu-$strongly convex and $L-$smooth.
- **Decreasing variance:** We also assume that the running sum of vari-

ances is sublinear as well, i.e,,

$$\sum_{k=0}^{T}\sum_{j=0}^{k}\sigma_j^2 = o(T). \tag{2.9}$$

- We also assume that our functions are twice differentiable.

where σ_j^2 refers to the noise of gradient estimator at the jth step of the algorithm. The decreasing variance assumption in games is not unrealistic because, as we approach Nash, players have less motivation to deviate from their current actions, and the gradient query would be less noisy. At the end of this section, we will see that our gradient estimator's noise can be controlled by making the search area smaller, which implies that our assumption can make sense.

Theorem 2.2.1. *If the following conditions hold:*

$$\gamma_k = \frac{1}{\rho}.\left[1+\frac{\beta_k(1-\alpha_k)}{\alpha_k}\right], \tag{2.10}$$

$$\alpha_k = \frac{\gamma_k\beta_k b_{k+1}^2\eta}{\gamma_k\beta_k b_{k+1}^2\eta + a_k^2}, \tag{2.11}$$

$$\beta_k \geq 1-\gamma_k\mu\eta, \tag{2.12}$$

$$a_{k+1} = \gamma_k\sqrt{\eta\rho}b_{k+1}, \tag{2.13}$$

$$b_{k+1} \leq \frac{b_k}{\sqrt{\beta_k}}, \tag{2.14}$$

$$b_{k+1}^2\gamma_k^2\eta\rho = a_{k+1}^2, \tag{2.15}$$

$$b_{k+1}^2\gamma_k\eta - a_{k+1}^2 + a_k^2 = 0, \tag{2.16}$$

$$\eta \leq \frac{1}{\rho L}, \tag{2.17}$$

the regret bound for algorithm 1 is:

$$Reg(T) \leq \sum_{j=1}^{T}\left[\frac{1}{2b_j^2\gamma_{j-1}^2\eta\rho}\left(2a_0^2C_1 + b_0^2C_2 + \sum_{k=1}^{j-1}\left[2b_{k+1}^2\gamma_{k+1}^2\eta^2\sigma_k^2 + 4b_{k+1}^2\gamma_{k+1}^2\eta\rho\epsilon_k\right]\right)\right]. \tag{2.18}$$

where $C_1 = f_0(x_0) - f_0(x^*)$ and $C_2 = \|x_0 - x^*\|^2$ and x^* is the best action played on the hindsight. Proof in Appendix A.

Lemma 2.2.2. *: If we run algorithm 1 with parameters bellow, if* $\sum_{k=0}^{T}\sum_{j=0}^{k}\epsilon_j = o(T)$*, then we get a sublinear regret.*

$$\begin{aligned} \eta &= 1/(\rho L), & \gamma &= 1/\sqrt{\eta\mu\rho}, \\ \gamma_k &= \frac{1}{\sqrt{\mu\eta\rho}}, & \beta_k &= 1-\sqrt{\frac{\mu\eta}{\rho}}, \\ a_k &= \frac{1}{\beta_k^{k/2}}, & b_k &= \frac{\sqrt{\mu}}{\beta_k^{k/2}}. \end{aligned} \tag{2.19}$$

Proof.

$$\begin{aligned} Reg(T) &\le o(\sum_{k=1}^{T}\frac{1}{b_k^2\gamma_{k-1}^2}\left[\frac{a_0^2}{\rho\eta}C_1+\frac{b_0^2}{2\rho\eta}C_2\right]+\sum_{k}^{T}\frac{1}{b_{k+1}^2\gamma_k^2}\sum_{i=0}^{k}[\gamma_i^2 b_{[i+1]}^2(\sigma_i^2+\epsilon_i)]) \\ &= o(\sum_{k}^{T}\frac{\beta_{k+1}^{k+1}\mu\eta\rho}{\mu}\left[\frac{a_0^2}{\rho\eta}C_1+\frac{b_0^2}{2\rho\eta}C_2\right]+\sum_{k}^{T}\frac{1}{b_{k+1}^2\gamma_k^2}\sum_{i=0}^{k}[\gamma_i^2 b_{[i+1]}^2(\sigma_i^2+\epsilon_i)]) \\ &\le o(\sum_{k}^{T}\frac{\beta_{k+1}^{k/+1}\mu\eta\rho}{\mu}\left[\frac{a_0^2}{\rho\eta}C_1+\frac{b_0^2}{2\rho\eta}C_2\right]+\sum_{k}^{T}\sum_{i=0}^{k}[\sigma_i^2+\epsilon_i]) \\ &= C_3+(o(T)+o(T))=o(T) \qquad \textbf{according to 2.9}. \end{aligned}$$

Where $C_3 = \sum_k^T \frac{\beta_{k+1}^{k/+1}\mu\eta\rho}{\mu}\left[\frac{a_0^2}{\rho\eta}C_1+\frac{b_0^2}{2\rho\eta}C_2\right]$ because it is a geometric series (β_k was chosen to be constant).

Where $C_3 = \sum_k^T \frac{\beta_{k+1}^{k/+1}\mu\eta\rho}{\mu}\left[\frac{a_0^2}{\rho\eta}C_1+\frac{b_0^2}{2\rho\eta}C_2\right]$ because it is a geometric series (β_k was chosen to be constant). □

Now we discuss how our results from previous theorems are related to the zero-order online optimization. Nesterov designed a two-point estimation of gradient, which satisfies the strong growth with additive noise assumption [21].
If $f \in C^{0,0}$ (Lipschitz):

$$\mathbb{E}_u(\|g_\mu\|_*^2) \le L^2(d+4)^2, \tag{2.20}$$

If $f \in C^{1,1}$ (smoothness):

$$\mathbb{E}_u(\|g_\mu\|_*^2) \le \frac{\mu^2}{2}M^2(d+6)^3 + 2(d+4)\|\nabla f(x)\|_*^2, \tag{2.21}$$

$$\mathbb{E}_u(\|\hat{g}_\mu\|_*^2) \leq \frac{\mu^2}{8} M^2 (d+6)^3 + 2(d+4)\|\nabla f(x)\|_*^2, \tag{2.22}$$

where in above expression $g_\mu = \frac{f(x+\mu u)-f(x)}{\mu}.Bu$, $\hat{g}_\mu = \frac{f(x+\mu u)-f(x-\mu u)}{2\mu}.Bu$, $f_\mu = \mathbb{E}_u(f(x+\mu u))$, u is a random vector distributed uniformly over the unit sphere in $\mathbb{R}^d$, M is smoothness factor of $f(x)$, and L is the Lipschitz constant of $f(x)$.

For example, is smooth case g_μ, the gradient estimator gives us $\rho = 2(d+4)$ and $\sigma^2 = \frac{\mu^2}{2} L_1^2(f)(d+6)^3$ in the equation 2.23. it implies that by making μ smaller with the rate of $o(2^{-n})$, then 2.9 will be $o(1)$. However, we should decrease μ moderately because numerical calculations could be problematic if μ gets too small. In fact, at some point, we miss-estimate the gradient as 0, and our algorithm stops.

Besides, the two-point gradient estimation is not a reasonable game assumption. Because players are constantly changing their actions and when we are sampling the function at different points, it is most likely that our function is totally changed and our estimation is not valid anymore. By looking into eqs. (A.10) and (A.12) the gradient is achieved at the point ζ_k, and we assumed that we play the action w_k. So if we try to use one of the two points used in 2.21, then we need to only have one query.

As it's mentioned earlier, in games, from each player's perspective, they are solving an online optimization problem, where the functions that they are given are $f_t(x_i) = u_i(x_{i,t}; x_{-i,t})$, which comes from a constant function $u_i(.)$. Thus, with the assumption that $u_i(.)$ is smooth, if the players begin to change their actions in very small amounts at some point, we have $|f_{t+1}(x) - f_t(x)| \leq \epsilon_t$, where ϵ_t is fairly small (with continuity assumptions). With this in mind, below we explain why we had the assumption of 2.7 and we bring the intuition that we expect the gradient estimation to be less noisy as we approach the Nash.

- We assumed that $|f_{t+1}(x) - f_t(x)| \leq \epsilon_t$. Although it is a valid condition for all continuous functions defined over a compact set (note that functions $f_t(.)$ are coming from one universal continuous function), it is important to express regret regarding such differences. Because, as we argued, when we are getting close to the Nash, the continuity of gradient implies that players tend to change their actions in small amounts (as we assume that player's are using gradient based algorithms), which leads to having smaller ϵ_t's. As a result, we see it harmless to have this assumption, and more importantly, to introduce this difference bound so we can have it in the regret.

- We expect our gradient queries to be less noisy as we get close to Nash. Because same as the previous case, as we have the assumption of twice differentiability of the cost functions, when we get close to Nash (where gradient w.r.t each player's action is zero), gradient of the cost function w.r.t players' action is close to zero. It also means that players will change their actions slowly. It also means that the function $f_t(x)$ is close to $f_{t+1}(x)$, which is used to estimate the gradient so that we would have smaller noise σ_t^2.

Next we calculate the optimization error of algorithm 1 for the deterministic and stochastic cases.

Theorem 3.2: The algorithm 1 enjoys deterministic and stochastic optimization error of:

$$f(w_{T+1}) - f^* \leq \frac{1}{2a_{T+1}^2}\left(2a_0^2\phi_0 + b_0^2 r_0^2\right), \tag{2.23}$$

$$\mathbb{E}\, f(w_{T+1}) - f^* \leq \frac{1}{2a_{T+1}^2}\left(2a_0^2\phi_0 + b_0^2\,\mathbb{E}[r_0^2] + \sum_{k=1}^{T}\left[\frac{2a_{k+1}^2\sigma_k^2\eta}{\rho}\right]\right). \tag{2.24}$$

respectively, were

$$\phi_k = \mathbb{E}\, f_k(w_K) - f_k^* \tag{2.25}$$

$$r_{k+1} = \|\nu_{k+1} - w^*\| \tag{2.26}$$

The terms above will converge to 0 and $\sum_{k=1}^{T}\frac{\sigma_k^2\eta}{\rho}$ respectively.

Proof: As we had in the proof

$$\mathbb{E}\, f_{T+1}(w_{T+1}) - f_{T+1}^* \leq \frac{1}{2a_{T+1}^2}\big(2a_0^2\phi_0 + b_0^2\,\mathbb{E}[r_0^2] - b_{T+1}^2\,\mathbb{E}[r_{T+1}^2] + \sum_{k=1}^{T}\left[\frac{2a_{k+1}^2\sigma_k^2\eta}{\rho} + 4a_{k+1}^2\epsilon_k\right]\big).$$

we assumed $f_k = f$ and therefore $\epsilon_k = 0$. Replacing $\sigma^2 = 0$ for stochastic case we get the desired result. □

2.3 3-stage online Nesterov method

This thesis presents a further three-stage online Nesterov algorithm with sublinear regret. It differs from the algorithm in the preceding section as a

result of the different hyperparameters. To show that an online accelerated algorithm is a no-regret algorithm, we use the same idea as [11]. They use a 3-stage accelerated method for non-differentiable functions. We tried to show that a 3-stage Nesterov accelerated method in an online setting is a no-regret for differentiable functions. After this step, we picked the simplest version of the accelerated method in an online setting, called it the 2-stage accelerated method, and showed it is also a no-regret algorithm for some specific hyper-parameters.

This section assumes that for any $t \geq 1$, f_t, f_t is a **convex**, **differentiable** and **L-smooth** function. Also, we assume that each f_t has a bounded gradient, i.e., for all x, $\|\nabla f_t(x)\|_2 \leq G$, where $G \geq 0$. Now we want to prove that algorithm 2 is a no-regret algorithm. The idea is to bound the $f_t(x_t) - f_t(x^*)$, where x^* is the optimum point that we can choose in hindsight. Based on the properties we assumed for f_t we have, we can find an upper bound for $f_t(x_{t+1}) - f_t(x)$. Then, we try to find an upper bound for $f_t(x_t) - f_t(x_{t+1})$ based on the updated policy. We do this to find an appropriate upper bound for $f_t(x_t) - f_t(x)$. The following proofs are inspired by [11]. Lemma 2.3.1 is a famous lemma and it presents in other references. This algorithm is modified version of what is introduced in [11], and the algorithm in the next section is completely novel. In [11], we do not have any equivalent for Lemma 2.3.2 or Lemma 2.3.3, but we decomposed their proof into these easier-to-follow lemmas to help the reader.

For the following algorithms, the idea of the proof is to bound $f_t(x_t) - f_t(x)$. To make this upper bound, we introduced some lemmas to get an upper bound for $f_t(x_t) - f_t(x)$. At some steps, we got an upper bound for $f_{t-1}(x_t) - f_{t-1}(x)$, and we needed to make finite variation assumption to get our desired upper bound. This helped us to get our bounds without finite variation assumption.

Algorithm 2 3 stage-accelerated Online Optimization

Input: Sequences $\{0 < \alpha_t < 1\}$ and $\{L_t\} > L$.
Initialize $z_1 = y_1$.
for $t = 1, \ldots, T$ **do**
    $x_t = (1 - \alpha_t)y_{t-1} + \alpha_t z_{t-1}$
    $y_t = x_t - \frac{1}{L_t}\nabla f_{t-1}(x_t)$.
    $z_t = z_{t-1} - \alpha_t(x_t - y_t)$.
end for

Lemma 2.3.1. *For $t > 1$, $f_t(x)$ can quadratically bounded from below as*

$$f_{t-1}(x) \geq f_{t-1}(y_t) + \langle \nabla f_{t-1}(x_t), x - x_t \rangle + \frac{2L_t - L}{2L_t^2} \|\nabla f_{t-1}(x_t)\|^2. \quad (2.27)$$

Proof. From smoothness of $f_{t-1}(x)$ we know:

$$f_{t-1}(y_t) \leq f_{t-1}(x_t) + \langle \nabla f_{t-1}(x_t), y - x_t \rangle + \frac{L}{2} \|y_t - x_t\|^2, \quad (2.28)$$

and from the updating rule, we know that $y_t - x_t = -\frac{1}{L_t} \nabla f_{t-1}(x_t)$. As a result, we have:

$$f_{t-1}(y_t) \leq f_{t-1}(x_t) + \langle \nabla f_{t-1}(x_t), -\frac{1}{L_t} \nabla f_{t-1}(x_t) \rangle + \frac{L}{2L_t^2} \|\nabla f_{t-1}(x_t)\|^2 \quad (2.29)$$

$$\implies f_{t-1}(y_t) \leq f_{t-1}(x_t) + \frac{L - 2L_t}{2L_t^2} \|\nabla f_{t-1}(x_t)\|^2 \quad (2.30)$$

$$\implies f_{t-1}(y_t) - \frac{L - 2L_t}{2L_t^2} \|\nabla f_{t-1}(x_t)\|^2 \leq f_{t-1}(x_t). \quad (2.31)$$

In addition, from convexity of $f_{t-1}(x)$ we know:

$$f_{t-1}(x) \geq f_{t-1}(x_t) + \langle \nabla f_{t-1}(x_t), x - x_t \rangle, \quad (2.32)$$

now, we substitute $f_{t-1}(x_t)$ with the lower bound we found in equation (1.4), then we have:

$$f_{t-1}(x) \geq f_{t-1}(y_t) + \langle \nabla f_{t-1}(x_t), x - x_t \rangle + \frac{2L_t - L}{2L_t^2} \|\nabla f_{t-1}(x_t)\|^2. \quad (2.33)$$

□

Lemma 2.3.2. *From the update rule of z_t, we can say:*

$$z_t =_x V_t(x) \equiv \langle \nabla f_{t-1}(x_t), x - x_t \rangle + \frac{L_t}{2\alpha_t} \|x - z_{t-1}\|^2.$$

Proof. By setting the gradient of $V_t(x)$ equal to zero, we have:

$$\nabla f_{t-1}(x_t) + \frac{L_t}{\alpha_t}(x - z_{t-1}) = 0 \implies x = z_{t-1} - \frac{\alpha_t}{L_t} \nabla f_{t-1}(x_t), \quad (2.34)$$

and from the update rule of y_t, we have $\nabla f_{t-1}(x_t) = L_t(x_t - y_t)$, and this implies $\arg\min_x V_t(x) = z_{t-1} - \alpha_t(x_t - y_t)$ and this completes the proof. □

Lemma 2.3.3. *For any x and $t \geq 1$, we have:*

$$f_{t-1}(y_t) - f_{t-1}(x) \leq \langle \nabla f_{t-1}(x), x_t - z_t \rangle - \frac{2L_t - L}{2L_t^2} \|\nabla f_{t-1}(x_t)\| \tag{2.35}$$
$$- \frac{L_t}{2\alpha_t} \|z_t - z_{t-1}\|^2 + \frac{L_t}{2\alpha_t} \|x - z_{t-1}\|^2 - \frac{L_t}{2\alpha_t} \|x - z_t\|^2.$$

Proof. From the definition of $V_t(x)$, we can say it is a $\mu-$strongly convex function where $\mu = \frac{L_t}{\alpha_t}$. Based on Lemma 2.3.3., we know z_t in the $\arg\min_x V_t(x)$. So, we can say:

$$V_t(z_t) \leq V_t(x) - \mu \|x - z_t\|^2 = V_t(x) - \frac{L_t}{\alpha_t} \|x - z_t\|^2 \tag{2.36}$$
$$= \langle \nabla f_{t-1}(x_t), x - x_t \rangle + \frac{L_t}{2\alpha_t} \|x - z_{t-1}\|^2 - \frac{L_t}{2\alpha_t} \|x - z_t\|^2, \tag{2.37}$$

also, from Lemma 2.3.1 we know $\langle \nabla f_{t-1}(x_t), x - x_t \rangle \leq f_{t-1}(x) - f_{t-1}(y_t) - \frac{2L_t - L}{2L_t^2} \|\nabla f_{t-1}(x_t)\|^2$. So, using the later inequality and 2.37, we have:

$$V_t(z_t) \leq f_{t-1}(x) - f_{t-1}(y_t) - \frac{2L_t - L}{2L_t^2} \|\nabla f_{t-1}(x_t)\|^2 \tag{2.38}$$
$$+ \frac{L_t}{2\alpha_t} \|x - z_{t-1}\|^2 - \frac{L_t}{2\alpha_t} \|x - z_t\|^2,$$

now if we put $x = z_t$ in the definition of the $V_t(x)$, we have:

$$f_{t-1}(y_t) - f_t(x) \leq \langle \nabla f_{t-1}(x_t), x_t - z_t \rangle - \frac{L_t}{2\alpha_t} \|z_t - z_{t-1}\|^2 \tag{2.39}$$
$$- \frac{2L_t - L}{2L_t^2} \|\nabla f_{t-1}(x_t)\|^2 + \frac{L_t}{2\alpha_t} \|x - z_{t-1}\|^2 - \frac{L_t}{2\alpha_t} \|x - z_t\|^2,$$

□

Proposition 2.3.4. *Suppose $\|\nabla f_t(x)\|_* \leq g$ for any $t \geq 1$. Then for any x, we have:*

$$f_{t-1}(y_{t-1}) - f_{t-1}(x) \leq \frac{g^2}{2(1-\alpha_t)(L_t - L)} + \frac{L_t}{2\alpha_t} \|x - z_{t-1}\|^2 - \frac{L_t}{2\alpha_t} \|x - z_t\|^2$$
$$+ \frac{(1-\alpha_t)^2 L_t - \alpha_t(1-\alpha_t)L}{2} \|y_{t-1} - z_{t-1}\|^2 - \frac{L_t}{2} \|y_t - z_t\|^2.$$

Proof. At first note that we have:

$$\langle \nabla f_{t-1}(x_t), x_t - z_t \rangle - \frac{\|\nabla f_{t-1}(x_t)\|^2}{2L_t} = \langle L_t(x_t - y_t), x_t - z_t \rangle - \frac{L_t \|x_t - y_t\|^2}{2}$$
$$= \frac{L_t}{2}(x_t - y_t, 2(x_t - z_t) - \langle x_t - y_t, x_t - y_t \rangle) \tag{2.40}$$
$$\frac{L_t}{2}(\langle x_t - y_t, x_t + y_t - 2z_t \rangle) = \frac{L_t}{2}(\|z_t - x_t\|^2 - \|z_t - y_t\|^2), \tag{2.41}$$

in addition, from the update rule of x_t, we have $x_t = (1 - \alpha_t)y_t + \alpha_t z_{t-1}$. So, we can say:

$$\begin{aligned}\|z_t - x_t\|^2 = \|z_t - (1 - \alpha_t)y_t - \alpha_t z_{t-1}\|^2 &= \|(1 - \alpha_t)(z_t - y_t) + \alpha_t(z_t - z_{t-1})\|^2 \\ &\leq (1 - \alpha_t)^2 \|z_t - y_t\|^2 + \alpha_t^2 \|z_t - z_{t-1}\|^2 \\ &\leq (1 - \alpha_t)\|z_t - y_t\|^2 + \alpha_t \|z_t - z_{t-1}\|^2 \\ &\leq (1 - \alpha_t)\|z_t - y_t\|^2 + \frac{1}{\alpha_t}\|z_t - z_{t-1}\|^2, \end{aligned} \tag{2.42}$$

as a result, we have:

$$\begin{aligned}\langle \nabla f_{t-1}(x_t), x_t - z_t \rangle - \frac{\|\nabla f_{t-1}(x_t)\|^2}{2L_t} &\leq \frac{L_t}{2\alpha_t}\|z_t - z_{t-1}\|^2 \\ &+ \frac{L_t(1 - \alpha_t)}{2}\|z_{t-1} - y_{t-1}\|^2 - \frac{L_t}{2}\|z_t - y_t\|^2. \end{aligned} \tag{2.43}$$

Now, if putting the later inequality in the Lemma 2.3.3, gives us:

$$f_{t-1}(y_t) - f_{t-1}(x) \leq \frac{L_t(1 - \alpha_t)}{2}\|z_{t-1} - y_{t-1}\|^2 - \frac{L_t}{2}\|z_t - y_t\|^2 \tag{2.44}$$
$$- \frac{L_t - L}{2L_t^2}\|\nabla f_{t-1}(x_t)\|^2 + \frac{L_t}{2\alpha_t}\|x - z_{t-1}\|^2 - \frac{L_t}{2\alpha_t}\|x - z_t\|^2. \tag{2.45}$$

In addition, from the convexity of $f_{t-1}(x)$ we know:

$$f_{t-1}(y_{t-1}) - f_t(y_t) \leq \langle \nabla f_{t-1}(x_t), y_t - y_{t-1} \rangle. \tag{2.46}$$

From the Young's inequality, for every $a > 0$ we know $\langle x, y \rangle \leq \frac{\|x\|^2}{2a} + \frac{a\|y\|^2}{2}$. So, by using Young's inequality and considering $a = (1 - \alpha_t)(L_t - L)$, we

can say:

$$f_{t-1}(y_{t-1}) - f_t(y_t) \leq \frac{\|\nabla f_{t-1}(x_t)\|^2}{2(1-\alpha_t)(L_t - L)} + \frac{(1-\alpha_t)(L_t - L)}{2}\|y_{t-1} - y_t\|^2 \tag{2.47}$$

$$\leq \frac{g^2}{2(1-\alpha_t)(L_t - L)} + \frac{(1-\alpha_t)(L_t - L)}{2}\|y_{t-1} - y_t\|^2. \tag{2.48}$$

Besides, from the update rule of y_t, we can say:

$$\|y_{t-1} - y_t\|^2 = \|(y_{t-1} - x_t) + (x_t - y_t)\|^2 = \|\alpha_t(y_{t-1} - z_{t-1}) + (x_t - y_t)\|^2 \tag{2.49}$$

$$\leq \alpha^2\|y_{t-1} - z_{t-1}\|^2 + (1-\alpha_t)^2\|x_t - y_t\|^2 \leq \alpha\|y_{t-1} - z_{t-1}\|^2 \tag{2.50}$$

$$+ \frac{1}{1-\alpha_t}\|x_t - y_t\|^2$$

$$= \alpha_t\|y_{t-1} - z_{t-1}\|^2 + \frac{\|\nabla f_{t-1}(x_t)\|^2}{(1-\alpha_t)L_t^2}. \tag{2.51}$$

By using 2.51 in 2.48, we have:

$$f_{t-1}(y_{t-1}) - f_{t-1}(y_t) \leq \frac{g^2}{2(1-\alpha_t)(L_t - L)} + \frac{\alpha_t(1-\alpha_t)(L_t - L)}{2}\|y_{t-1} - z_{t-1}\|^2$$
$$+ \frac{L_t - L}{2L_t^2}\|\nabla f_{t-1}(x_t)\|^2. \tag{2.52}$$

In the end, by adding the 2.52 to 2.47 we get the inequality of proposition. □

Theorem 2.3.5. *By setting $\alpha_t = a$ where $0 < a < 1$ and $L_t = aL\sqrt{t-1}+L$, we have:*

$$\sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x^*) = \mathcal{O}(\sqrt{T}).$$

Proof. We can easily check the result by using proposition 2.4.5 and taking a summation. □

It is apparent from the Theorem 2.3.5 that algorithm 2 is no-regret. Based on [24], in a game theoretic setting, if each player follows a no-regret algorithm, we can guarantee they will converge (in measure) to the Nash equilibrium.

2.4 2-stage online Nesterov method

Again, we assume that for any $t \geq 1$, f_t is a **convex**, **differentiable** and **L-smooth** function. Also, we assume that each f_t has bounded gradient i.e., for all x, $\|\nabla f_t(x)\|_2 \leq G$, where $G \geq 0$.

The goal is to prove algorithm 2 is a no regret algorithm. The idea is to bound the $f_t(x_t) - f_t(x^*)$, where x^* is the optimum point that we can choose in hindsight. Based on properties we assumed for f_t, we will find an upper bound for $f_t(x_{t+1}) - f_t(x)$. Then, we try to find an upper bound for $f_t(x_t) - f_t(x_{t+1})$ based on the update policy. These two gives us an appropriate upper bound for $f_t(x_t) - f_t(x)$. The following proofs are inspired by [11].

Algorithm 3 2-stage Nesterov method in online setting

1: **Input:** Parameters μ_t, η_t, x_1
2: Pick $y_1 = x_1$.
3: **for** $t = 1, \ldots, T$ **do**
4: Observe the payoff function gradient $\nabla f_t(y_t)$.
5: Update $x_{t+1} = y_t - \eta_t \nabla f_t(y_t)$.
6: Update $y_{t+1} = x_{t+1} + \mu_t(x_{t+1} - x_t)$.
7: **end for**

Lemma 2.4.1. *For any $t \geq 1$,*

$$f_t(x_{t+1}) - f_t(x) \leq (\frac{L}{2} - \frac{1}{\eta_t})\|y_t - x_{t+1}\|_2^2 + \frac{(y_t - x_{t+1})^T(y_t - x)}{\eta_t}, \quad (2.53)$$

where x_{t+1} and y_t are corresponding variables in iteration t while we running algorithm 3.

Proof. This lemma can be proved just by combining the convexity and $L-$smoothness definitions. From the convexity definition, we can say:

$$f_t(x) \geq f_t(y_t) + \nabla f_t(y_t)^T(x - y_t) \implies f_t(y_t) - f_t(x) \leq \nabla f_t(y_t)^T(y_t - x), \quad (2.54)$$

and since from the update rule, we have $\nabla f_t(y_t) = \frac{y_t - x_{t+1}}{\eta_t}$, we can say:

$$f_t(y_t) - f_t(x) \leq \frac{(y_t - x_{t+1})^T(y_t - x)}{\eta_t}. \quad (2.55)$$

Furthermore, from the $L-$smoothness definition, we have:

$$f_t(y_t - \eta_t \nabla f_t(y_t)) - f_t(y_t) \leq \langle \nabla f_t(y_t), -\eta_t \nabla f_t(y_t) \rangle + \frac{L}{2} \|\eta_t \nabla f_t(y_t)\|_2^2 = (\frac{L\eta_t^2}{2} - \eta_t) \nabla f_t(y_t), \tag{2.56}$$

moreover, again from the update rule we have $\nabla f_t(y_t) = \frac{y_t - x_{t+1}}{\eta_t}$, so we can say:

$$f_t(y_t - \eta_t \nabla f_t(y_t)) - f_t(y_t) \leq (\frac{L}{2} - \frac{1}{\eta_t}) \|y_t - x_{t+1}\|_2^2. \tag{2.57}$$

Since $x_{t+1} = y_t - \eta_t \nabla f_t(y_t)$, if we add 2.57 to 2.55, we get:

$$f_t(x_{t+1}) - f(x) \leq (\frac{L}{2} - \frac{1}{\eta_t}) \|y_t - x_{t+1}\|_2^2 + \frac{(y_t - x_{t+1})^T (y_t - x)}{\eta_t}. \tag{2.58}$$

□

As we said earlier, we are aiming to bound $f_t(x_t) - f(x)$. To that end, we try to introduce an appropriate bound for $f_t(x) - f_{t+1}(x_{t+1})$. To introduce this bound, We're inspired by [8].

Lemma 2.4.2. *For any $t \geq 1$ and $\alpha \geq 0$:*

$$f_t(x_t) - f_t(x_{t+1}) \leq \frac{G^2}{2\alpha} + \frac{\alpha \mu_t}{2} \|x_{t-1} - x_t\|_2^2 + \frac{\alpha}{2(1-\mu_t)} \|y_t - x_{t+1}\|_2^2. \tag{2.59}$$

Proof. from the convexity definition, we have:

$$f_t(x_t) - f_t(x_{t+1}) \leq \langle \nabla f_t(x_{t+1}), x_t - x_{t+1} \rangle. \tag{2.60}$$

By Young's inequality, for any $\alpha > 0$, we have:

$$\langle \nabla f_t(x_{t+1}), x_t - x_{t+1} \rangle \leq \frac{\|\nabla f_t(x_{t+1})\|_2^2}{2\alpha} + \frac{\alpha}{2} \|x_t - x_{t+1}\|_2^2. \tag{2.61}$$

Now, since in 2.58 we have $\|y_t - x_{t+1}\|_2^2$, let us create a similar term here. By adding and subtracting y_t, we have:

$$\begin{aligned} \|x_t - x_{t+1}\|_2^2 &= \|(x_t - y_t) + (y_t - x_{t+1})\|_2^2 \\ &= \|\mu_t (x_t - x_{t+1}) + (y_t - x_{t+1})\|_2^2 \\ &\leq \mu_t \|x_t - x_{t+1}\|_2^2 + \frac{1}{1-\mu_t} \|y_t - x_{t+1}\|_2^2. \end{aligned} \tag{2.62}$$

Now, since $\|\nabla f_t(x)\| \leq G$ for all $t \geq 1$, if we substitute 2.62 in 2.61, we get:

$$f_t(x_t) - f_t(x_{t+1}) \leq \frac{G^2}{2\alpha} + \frac{a\mu_t}{2}\|x_{t-1} - x_t\|_2^2 + \frac{a}{2(1-\mu_t)}\|y_t - x_{t+1}\|_2^2. \quad (2.63)$$

□

The next step, is to find an upper bound for $\frac{(y_t - x_{t+1})}{\eta_t}(y_t - x)$. To do this, we use use an idea from [8]. Note that in [8], they talk about a 3-stage algorithm and basically, we have different analysis here. But the ideas are similar.

Lemma 2.4.3. *For any $t \geq 1$,*

$$f_t(x_{t+1}) - f_t(x) \leq (\frac{L}{2} - \frac{1}{2\eta_t})\|y_t - x_{t+1}\|_2^2 + \frac{1}{2\eta_t}\|x - y_t\|_2^2 - \frac{1}{2\eta_t}\|x - x_{t+1}\|_2^2. \quad (2.64)$$

Proof. Define:

$$V_t(x) = \langle \frac{y_t - x_{t+1}}{\eta_t}, x - y_t \rangle + \frac{1}{2\eta_t}\|x - y_t\|_2^2. \quad (2.65)$$

Actually, $V_t(x)$ is a $\frac{1}{\eta_t}$-strongly convex function. Also, by setting its gradient equal to zero, we see that it's minima is $x = y_t - \eta_t \nabla f_t(y_t)$. In fact, x_{t+1} is the minima of $V_t(x)$. From the definition of strong convexity, we can say:

$$V(x_{t+1}) \leq V_t(x) - \frac{1}{2\eta_t}\|x - x_{t+1}\|_2^2 \quad (2.66)$$

$$\langle \frac{y_t - x_{t+1}}{\eta_t}, x - y_t \rangle + \frac{1}{2\eta_t}\|x - y_t\|_2^2 - \frac{1}{2\eta_t}\|x - x_{t+1}\|_2^2 \quad (2.67)$$

. (2.68)

Now by using Lemma 5.1., we can say:

$$V(x_{t+1}) \leq f_t(x) - f_t(x_{t+1}) + (\frac{L}{2} - \frac{1}{\eta_t})\|y_t - x_{t+1}\|_2^2 + \frac{1}{2\eta_t}\|x - y_t\|_2^2 - \frac{1}{2\eta_t}\|x - x_{t+1}\|_2^2, \quad (2.69)$$

Besides, from the definition we have $V_t(x) = \langle \frac{y_t - x_{t+1}}{\eta_t}, x - y_t \rangle + \frac{1}{2\eta_t}\|x - y_t\|_2^2$. So, we can say:

$$f_t(x_{t+1}) - f_t(x) \leq (\frac{L}{2} - \frac{1}{2\eta_t})\|y_t - x_{t+1}\|_2^2 + \frac{1}{2\eta_t}\|x - y_t\|_2^2 - \frac{1}{2\eta_t}\|x - x_{t+1}\|_2^2. \quad (2.70)$$

□

Lemma 2.4.4. *For any $t \in \mathbb{N}$, we have:*

$$\frac{1}{\sqrt{1}} + \cdots + \frac{1}{\sqrt{t}} \leq 2\sqrt{t} - 1. \tag{2.71}$$

By induction we can prove this lemma.

Proposition 2.4.5. *For any $t \geq 1$,*

$$\begin{aligned} f_t(x_t) - f_t(x) \leq & \frac{G^2}{2(1-\mu_t)(\frac{1}{\eta_t} - \frac{L}{2})} + \frac{\mu_t(1-\mu_t)(\frac{1}{2\eta_t} - \frac{L}{2})}{2}\|x_{t-1} - x_t\|_2^2 \\ & + \frac{1}{2\eta_t}\|x - y_t\|_2^2 - \frac{1}{2\eta_t}\|x - x_{t+1}\|_2^2. \end{aligned} \tag{2.72}$$

Proof. In Lemma 2.4.2, put $\alpha = (1-\mu_t)(\frac{1}{\eta_t} - L)$. Then we have:

$$\begin{aligned} f_t(x_t) - f_t(x_{t+1}) \leq & \frac{G^2}{2(1-\mu_t)(\frac{1}{\eta_t} - L)} + \frac{\mu_t(1-\mu_t)(\frac{1}{\eta_t} - L)}{2}\|x_{t-1} \\ & - x_t\|_2^2 + \frac{(1-\mu_t)(\frac{1}{\eta_t} - L)}{2(1-\mu_t)}\|y_t - x_{t+1}\|_2^2 \qquad (2.73) \\ = & \frac{G^2}{2(1-\mu_t)(\frac{1}{\eta_t} - L)} + \frac{\mu_t(1-\mu_t)(\frac{1}{\eta_t} - L)}{2}\|x_{t-1} - x_t\|_2^2 \\ & + (\frac{1}{2\eta_t} - \frac{L}{2})\|y_t - x_{t+1}\|_2^2. \qquad (2.74) \end{aligned}$$

Now if we add 2.74 to 2.70, we get:

$$\begin{aligned} f_t(x_t) - f_t(x) \leq & \frac{G^2}{2(1-\mu_t)(\frac{1}{\eta_t} - \frac{L}{2})} + \frac{\mu_t(1-\mu_t)(\frac{1}{2\eta_t} - \frac{L}{2})}{2}\|x_{t-1} - x_t\|_2^2 \\ & + \frac{1}{2\eta_t}\|x - y_t\|_2^2 - \frac{1}{2\eta_t}\|x - x_{t+1}\|_2^2. \end{aligned} \tag{2.75}$$

□

Theorem 2.4.6. *If we assume that the diameter of the cost function's domain is at most D, by setting $\eta_t = \frac{1}{\sqrt{t}+\frac{L}{2}}$ and $\mu_t = \frac{1}{2t}$ in algorithm 3, we get a no-regret algorithm.*

Proof. From the proposition 2.4.5, we have:

$$f_t(x_t) - f_t(x) \leq \frac{G^2}{2(1-\frac{1}{2t})(\sqrt{t})} + \frac{\frac{1}{2t}(1-\frac{1}{2t})(\frac{\sqrt{t}+\frac{L}{2}}{2} - \frac{L}{2})}{2}\|x_{t-1} - x_t\|_2^2 + \frac{\sqrt{t}+\frac{L}{2}}{2}\|x - y_t\|_2^2 - \frac{\sqrt{t}+\frac{L}{2}}{2}\|x - x_{t+1}\|_2^2. \tag{2.76}$$

Note that for any $t \geq 1$, we have:

$$\frac{G^2}{2(1-\frac{1}{2t})(\sqrt{t})} \leq \frac{G^2}{\sqrt{t}}, \tag{2.77}$$

$$\frac{\frac{1}{2t}(1-\frac{1}{2t})(\frac{\sqrt{t}+\frac{L}{2}}{2} - \frac{L}{2})}{2}\|x_{t-1} - x_t\|_2^2 \leq \frac{1}{8\sqrt{t}}\|x_{t-1} - x_t\|_2^2. \tag{2.78}$$

Also, from the update rule, we have $y_t = x_t + \mu_t(x_t - x_{t-1})$. So, we can say:

$$\|x - y_t\|_2^2 = \|x - (x_t + \mu_{t-1}(x_{t-1} - x_t))\|_2^2 \tag{2.79}$$

$$= \|(x - x_t) + \mu_{t-1}(x_t - x_{t-1})\|_2^2 \leq \mu_{t-1}\|x_{t-1} - x_t\|_2^2 + \frac{1}{1-\mu_{t-1}}\|x - x_t\|_2^2 \tag{2.80}$$

$$\leq \mu_{t-1}(\|x_{t-1} - x_t\|_2^2 + \frac{1}{1-\mu_{t-1}}\|x - x_t\|_2^2) + \|x - x_t\|_2^2 \tag{2.81}$$

$$\leq \mu_{t-1}(\|x_{t-1} - x_t\|_2^2 + \frac{1}{1-\mu_0}\|x - x_t\|_2^2) + \|x - x_t\|_2^2. \tag{2.82}$$

Putting all these together, we can say (the assumption the diameter of cost function's domain is at most D, means that $\|x_{t-1} - x_t\|_2^2 \leq D^2$ and $\|x - x_t\|_2^2 \leq D^2$, and we consider $\mu_{t-1}\eta_t \leq \frac{1}{\sqrt{t}}$, note that based on value of L, there is $k > 0$ such that $\mu_{t-1}\eta_t \leq \frac{k}{\sqrt{t}}$, and the value of k is not important for the rate regret that we are looking for):

$$f_t(x_t) - f_t(x) \leq \frac{G^2}{\sqrt{t}} + \frac{1}{8\sqrt{t}}D^2 + \frac{1}{\sqrt{t}}(D^2 + \frac{1}{1-\mu_0}D^2) + (\frac{1}{2\eta_{t+1}} - \frac{1}{2\eta_t})D. \tag{2.83}$$

Finally, we can say:

$$\begin{aligned}\sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x^*) &\leq G^2(2\sqrt{T}-1) + \frac{D^2}{8}(2\sqrt{T}-1) \\ &\quad + (D^2 + \frac{1}{1-\mu_0}D^2)(2\sqrt{T}-1) + (\frac{1}{2\eta_{T+1}} - \frac{1}{2\eta_0})D \\ &= \mathcal{O}(\sqrt{T}). \end{aligned} \tag{2.84}$$

And this means we have a no-regret algorithm. Moreover, the regret it in the order of $\mathcal{O}(\sqrt{T})$. □

2.4.1 Quadratic Games

The goal is to study the behavior of gradient-based algorithms, including gradient descent and our accelerated algorithm, on different games like Quadratic games, which were introduced in [12]. We want to simplify it to a deterministic game and study its equilibrium.

With the following cost function to the players:

$$\pi_i(a_i, a_{-i}) = \frac{1}{2}a_i^T \Gamma_{ii} a_i + a_i^T \Gamma_{i,-i} a_{-i} + a_i^T g_i + h_i(a_{-i}), \tag{2.85}$$

with $\Gamma_{ii} \succ 0$ which gives:

$$\nabla_i \pi_i(a_i, a_{-i}) = \Gamma_{ii} a_i + \Gamma_{i,-i} a_{-i} + g_i.$$

We can re-write $\Gamma_{i,-i} a_{-i} = \sum_{j \neq i} \Gamma_{ij} a_j$ and define $\phi_i = \sum_j \Gamma_{ij} a_j$. Then the first order equilibrium condition gives:

$$\nabla_i \pi_i(a_i^*, a_{-i}^*) = \Gamma_{ii} a_i^* + \sum_{j \neq i} \Gamma_{ij} a_j^* + g_i = 0, \tag{2.86}$$

$$= \Gamma_{ii} a_i^* + (\phi_i - \Gamma_{ii} a_i^*) + g_i = 0. \tag{2.87}$$

Consecutively:

$$\begin{aligned} \phi_i &= -g_i \\ \sum_j \Gamma_{ij} a_j^* &= -g_i, \quad \forall i. \end{aligned} \tag{2.88}$$

Solving this set of linear equations will gives the solutions of Nash equilibria.

$$\begin{bmatrix} \Gamma_{11} & \dots & \Gamma_{1N} \\ \vdots & \ddots & \vdots \\ \Gamma_{N1} & \dots & \Gamma_{NN} \end{bmatrix} a^* = -g,$$

where a^* is the stacked vector of player's Nash actions and g is a stacked vector of g_i's. From this, we conclude that the condition for having a unique Nash is to have matrix $\begin{bmatrix} \Gamma_{11} & \dots & \Gamma_{1N} \\ \vdots & \ddots & \vdots \\ \Gamma_{N1} & \dots & \Gamma_{NN} \end{bmatrix}$ full rank.

To study the case where each player uses the gradient descent algorithm to optimize their cost functions, we have

$$a_{i,t+1} = a_{i,t} - \eta_t \nabla_i \pi_i(a_{i,t}, a_{-i,t}), \tag{2.89}$$

$$= a_{i,t} - \eta_t(\Gamma_{ii} a_{i,t} + \sum_{j \neq i} \Gamma_{ij} a_{-i,t} + g_i) \tag{2.90}$$

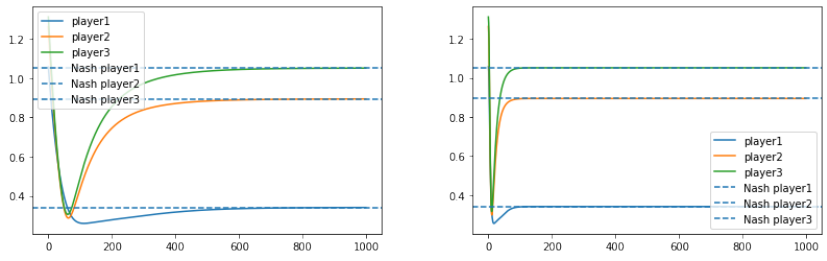
$$= a_{i,t} - \eta_t(\sum_{j} \Gamma_{ij} a_{j,t} + g_i), \tag{2.91}$$

$$a_{t+1} = (I - \eta_t \begin{bmatrix} \Gamma_{11} & \dots & \Gamma_{1N}, \\ \vdots & \ddots & \vdots, \\ \Gamma_{N1} & \dots & \Gamma_{NN} \end{bmatrix}) a_t - \eta_t g. \tag{2.92}$$

which implies that the necessary condition for this dynamic system's stability is $eig(I - \eta_t \begin{bmatrix} \Gamma_{11} & \dots & \Gamma_{1N} \\ \vdots & \ddots & \vdots \\ \Gamma_{N1} & \dots & \Gamma_{NN} \end{bmatrix})$ is located inside the unit circle.

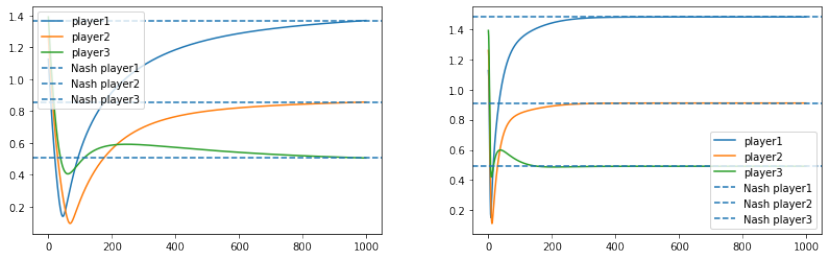
Simulations:

In this part, we perform simulations to evaluate the speed of convergence of our algorithm when it is used on a quadratic game. Also, in this part, we repeat this simulation on 3 different randomly generated quadratic games. Here we randomly select matrices Γ_{ii}'s and Γ_{ij} with $i > j$ and then set $\Gamma_{ji} = \Gamma_{ij}^T$ to make the big matrix Γ symmetric. As we see in the following graphs, the momentum method converges faster than gradient descent.



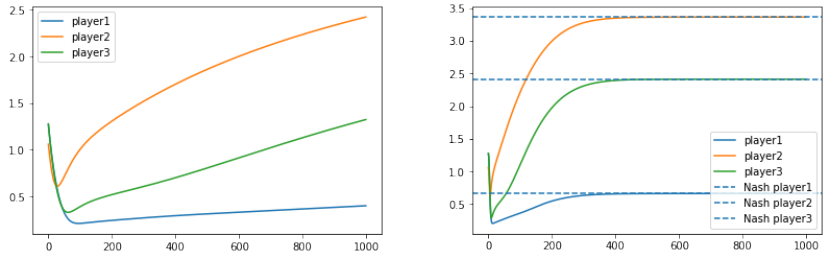
(a) without momentum (b) with momentum

Figure 2.1: Gradient descent and accelerated solution



(a) without momentum (b) with momentum

Figure 2.2: Gradient descent and accelerated solution



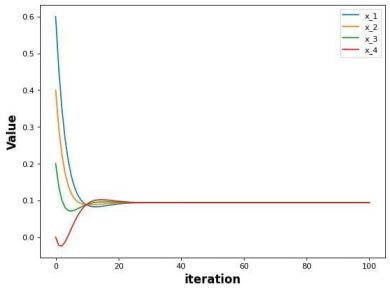
(a) without momentum (b) with momentum

Figure 2.3: Gradient descent and accelerated solution

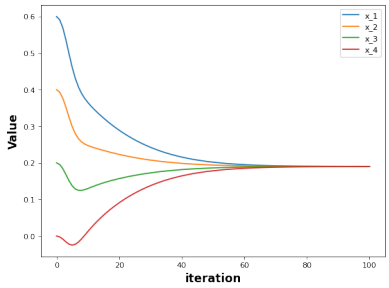
Through the simulations, we observed that accelerated algorithms enjoy better convergence in games. Let us take a look at an example of Cournot game.

Setup: We consider a four-player Cournot games with different market price and individual cost functions. Consider a Cournot game where $p(x) = 1 - (\sum_i x_i)$, and a linear production cost function is $C_i(x_i) = 0.05x_i$.

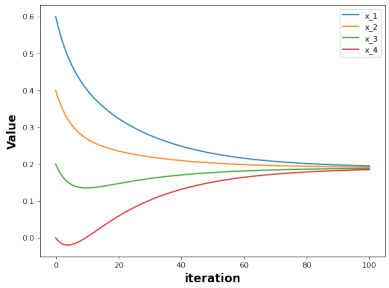
This game proceeds as follows. Every player simultaneously picks a production level at each time step, and then the market price is determined by their joint production and broadcast back to all players. Then, each player calculates his payoff, $\pi_i(x_i) = P_t x_i - C_i(x_i)$, where P_t is the market price at iteration t. The Nash equilibrium of this games is $x_1^* = x_2^* = x_3^* = x_4^* = 0.19$. Also, since the payoff function is differentiable, one can confirm this by setting the derivative of payoff equal to zero and validating this value for NE. Below, we referrer the reader an accelerated method's convergence trajectory compared to Online gradient descent. (step size in both algorithms was set to the same value)



(a) Algorithm 1



(b) Algotithm 3



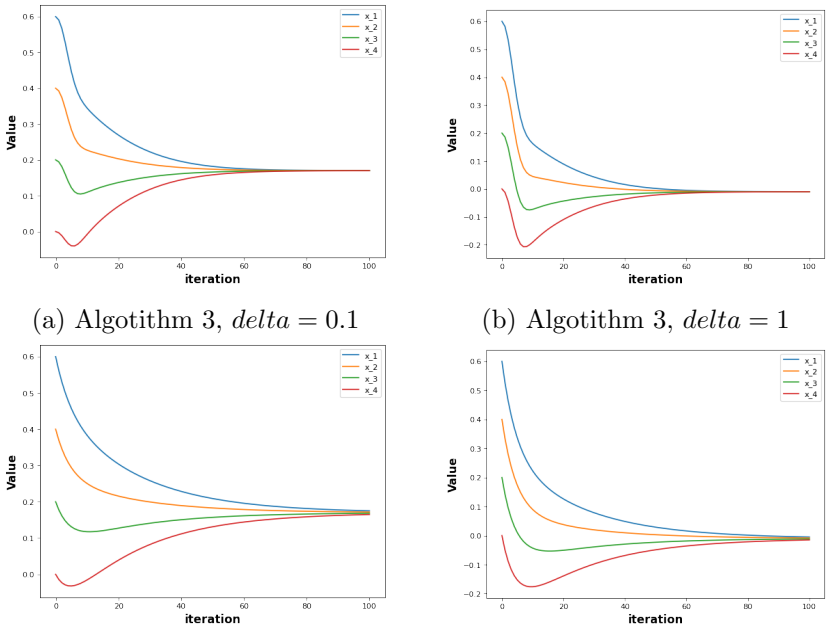
(c) Online Gradient Descent algorithm

Figure 2.4: Convergence Trajectories

Also, we repeat the same example for the case we do not have access to the gradient. We used the finite difference method for estimating the gradient. Mathematically, we considered:

$$\frac{\partial f}{\partial x_i} = \frac{f(x + \delta e_i) - f(x)}{\delta}, \tag{2.93}$$

where e_i denotes the vector with a 1 in the $i-$th coordinate, 0's elsewhere, and δ is a positive number. We chose different values for δ, and below, you can see the convergence trajectories:



(a) Algotithm 3, $delta = 0.1$

(b) Algotithm 3, $delta = 1$

(c) Online Gradient Descent algorithm, $delta = 0.1$

(d) Online Gradient Descent algorithm, $delta = 1$

Figure 2.5: Convergence Trajectories

2.5 Conclusion and Summary

Nesterov methods are well known accelerated methods. Nesterov showed in an offline setting they enjoy a fast convergence rate [20]. We can get $\mathcal{O}(\frac{1}{t^2})$ rate of convergence, which is much faster than gradient descent. We developed these algorithms in an online setting and proved that for some special hyper-parameters, they become no-regret algorithms. To the best of our knowledge, it has not been shown that the 2-stage Nesterov algorithm in an online setting is a no-regret algorithm. As for the 3-stage algorithm in the online setting, in [8], a proof presented for cases that cost functions are made of a differentiable and a non-differentiable function showed that by using sub-gradient, we can get a no-regret algorithm for some especial hyper-parameters. By employing our algorithm in Cournot games and Quadratic games, we observed faster convergence compared to Online Gradient Descent. We also extended the Nesterov accelerated algorithm for stochastic settings to online settings, in which we use zero-order estimation for the gradient. Under certain assumptions, in a game theoretic setting, if all players

Algorithm	Oracle type	Strong convexity	Finite variation	Bounded Gradient	Regret order	Optimization error
Algorithm 1	Noisy first order or zero-order	Yes	Yes	No	$\mathcal{O}(\sum_k^T \sum_{i=0}^k [\sigma_i^2 + \epsilon_i])$	$\mathcal{O}(\beta^T)$
Algorithm 2	First order	No	No	Yes	$\mathcal{O}(\sqrt{T})$	$\mathcal{O}(\frac{1}{t})$
Algorithm 3	First order	No	No	Yes	$\mathcal{O}(\sqrt{T})$	$\mathcal{O}(\frac{1}{t})$
Online Gradient Descent [8]	First order	No	No	Yes	$\mathcal{O}(\sqrt{T})$	$\mathcal{O}(\frac{1}{t})$
Online Gradient Descent [8]	First order	Yes	No	Yes	$\mathcal{O}(\log(T))$	$\mathcal{O}(\exp(-t))$

Table 2.1: We present the regret order of different online algorithms in this table. All these functions are differentiable and smooth. The best regret rate can be achieved without strong convexity $\mathcal{O}(\sqrt{T})$.

follow no-regret algorithms, they converge to the Nash equilibrium [24]. In simulations, we saw that if the player follows these accelerated methods, they converge faster to the Nash equilibrium in comparison to the case that they follow the Online mirror descent algorithm suggested by Shi [24].

2.6 Next steps

In the first section, we derived regret in terms of $o(\sum_k^T \sum_{i=0}^k [\sigma_i^2 + \epsilon_i])$. As for the next, one can find the corresponding regret in terms of T, when the algorithm is used on Cournot games defined in [24]. It is also remains to extend algorithm 2 or 3 to a zero-order algorithm and then compare it to algorithm 1 performance in practice. Moreover, a theoretical proof for

what we've seen in the simulation is needed. It can be done by finding the convergence rate of these algorithms. Furthermore, as we said earlier, one difference between algorithm one and algorithms 2 and 3 is that we did not have any assumption on the finite variation in the latter. So it is motivating to use the same ideas and relax the assumption we had in the first algorithm.

Chapter 3

LQ games

When the parameters of the *Linear Quadratic* (LQ) system are unknown to us, we use PG to calculate the optimum action to play. It has been proven that using *Policy Gradient* (PG) will get us to the optimum global feedback in LQR systems [4]. While it looks promising that it works well in the games, some counter examples indicate that PG does not have any guarantee of convergence for LQ games in a general setting [13, 14]. However, Hambly and Roudneshin [7, 23] proved that in other non-general settings, PG has a convergence guarantee. It motivates us to study other non-general LQ games, recognize them, and see if they have any PG convergence guarantee.

In *potential games*, the cost functions of players are connected to each other via one global potential function. This property of theirs makes them show one-player system characteristic when the players are using PG. In this thesis, we chose the class of potential LQ games to study. In the rest of this chapter, we go through the definitions, properties, and theorems associated with potential games. We identify the class of potential LQ games under certain assumptions and prove that using PG will converge to a Nash solution.

3.1 Review of the definitions

3.1.1 Dynamic System

A discrete dynamic system is defined as a set of states $X \subseteq \mathbb{R}^m$ and a set of actions $U \subseteq \mathbb{R}^n$ a set of evaluation functions $f_t : X \times U \to X$, where $x_{t+1} = f_t(x_t, u_t)$ for $t \in \{0, 1, 2, \dots\}$. For example the following can be a dynamic function:

$$x_{t+1} = \frac{1}{\|u_t\| + 1} x_t. \tag{3.1}$$

If functions f_t are linear with respect to x_t's and u_t's for all t, we call it a linear dynamic system, i.e., $f_t(x_t, u_t) = A_t x_t + B_t u_t$. The below function

is an example of a linear dynamic for $m = n = 2$:

$$f_t(x_t, u_t) = x_t/2 + u_t. \tag{3.2}$$

with $A_t = \frac{1}{2} I_{2\times 2}$ and $B_t = I_{2\times 2}$.

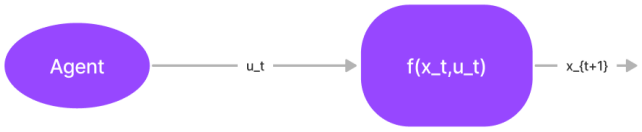
A time-invariant linear dynamic system is asymptotically stable if matrix A has a spectral radius of smaller than one [10]. In other words, if all the eigenvalues of A lie inside the unit circle, then the dynamic 3.2 is a time-invariant, stable linear dynamic. The system's stability is important to us when dealing with an infinite horizon system, especially in the closed loop setting, because, if we have an unstable system, it can easily give us an unbounded cost functions which is not desirable.

3.1.2 Quadratic Cost Function

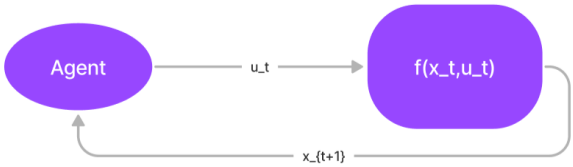
A *Linear Quadratic Regulator* (LQR) is a linear dynamic system with a quadratic cost function on states and actions. In other words, each time episode has a cost function as $c_t = x_t^T Q_t x_t + u_t^T R_t u_t$ where $Q_t \succeq 0$ and $R_t \succ 0$ for all t. Here we with to minimize the sum of $J(x, u) = \sum_{t=1}^{T} c_t$, where u and x are vectors over time horizon T, and T can be infinity.

The optimal action at time t is proven to be a linear function of the state at time t, i.e., $u_t^* = -K_t x_t$, and if the horizon is infinity, $u_t^* = -K x_t$. This encourages us to search for the optimal coefficient K_t^* rather than the optimal action u_t^*.

Whether the agent gets feedback on each time step or not, there are two settings: open-loop and closed-loop. In an open-loop LQR, the agent will be provided with no feedback. In a closed-loop LQR, however, the agent will be given with x_t at each time step, and based on that, it plays its action.



(a) Open Loop



(b) Closed Loop

Figure 3.1: Closed loop and open loop flowchart

Known and Unknown Parameters: There are two major settings in which the game is dealt with; one is where all the parameters of A_t, B_t, Q_t, and R_t are given. As opposed to this setting, in real-world problems, there is no access to any of them. All that is permitted is to query the value of the cost function $J(x, u)$ and receive a noisy or accurate value as feedback.

The optimum solution to this problem in a closed-loop setting, when there is access to all the parameters, is achieved by solving the following Algebraic Riccati equations [4]:

$$P = A^T PA + Q - A^T PB(B^T PB + R)^{-1} B_T PA \tag{3.3}$$

$$K^* = -(B^T PB + R)^{-1} B^T PA \tag{3.4}$$

for infinite horizon setting and

$$P_t = A^T P_{t+1} A - A^T P_{t+1} B(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A + Q, \tag{3.5}$$

$$K_t^* = -(B^T P_{t+1} B + R)^{-1} B^T P_{t+1} A0 \tag{3.6}$$

for finite horizon setting. These equations can be solved by Lyapunov iteration. However, if the parameters are unknown to us, none of the mentioned equations can be used to derive the value of K_t.

As a generalization of LQR systems with only one-player to play its inputs, we have LQ games where N players interact with the environment, penalizing their actions and the states they are in with a quadratic cost function. Also, all players' actions impact the state of the next time step in a linear fashion.

$$c_{i,t} = x_t^T Q_i x_t + u_{i,t}^T R_i u_{i,t} \quad \forall i, t, \tag{3.7}$$

$$x_{t+1} = Ax_t + \sum_{j=1}^{N} B_j u_{j,t} \quad \forall t. \tag{3.8}$$

Like in LQR systems, the Nash in LQ games has the Nash in the form of $u_{i,t}^* = -K_{i,t}^* x_t$. When the parameters are known, the Nash will be calculated by the following equations:

$$P_{i,t}^* = Q_{i,t} + (K_{i,t}^*)^T R_{i,t} K_{i,t}^* + (A_t - \sum_{j=1}^{N} B_{j,t} K_{j,t}^*)^T P_{i,t+1}^* (A_t - \sum_{j=1}^{N} B_{j,t} K_{j,t}^*), \tag{3.9}$$

$$K_{i,t}^* = (R_{i,t} + B_{i,t}^T P_{i,t+1}^* B_{i,t+1})^{-1} B_{i,t}^T P_{i,t+1}^* (A_t - \sum_{j=1, j\neq i}^{N} B_{j,t} K_{j,t}). \tag{3.10}$$

for finite horizon and

$$P_i^* = Q_i + (K_i^*) T R_i K_i^* + (A - \sum_{j=1}^{N} B_j K_j^*)^T P_i^* (A - \sum_{j=1}^{N} B_j K_j^*), \tag{3.11}$$

$$K_i^* = (R_i + B_i^T P_i^* B_i)^{-1} B_i^T P_i^* (A - \sum_{j=1, j\neq i}^{N} B_j K_j) 0 \tag{3.12}$$

for infinite horizon.

As shown in the previous part, if all the parameters are known, we can analytically calculate the optimum input for the LQR system or Nash for LQ games. However, as mentioned, it is not a reasonable assumption to have access to any of the parameters of the game, as the game/system is unknown to us in real-world problems. Thus, we need to have a different approach, the policy gradient. *Policy Gradient* (PG) is a popular method used in many Reinforcement Learning problems to optimize policies using estimated or actual cost functions. Here we use all the feedback we got to calculate or estimate the gradient of the cost function concerning our policy and change our policy accordingly:

$$K_{i,t}^{m+1} = K_{i,t}^m - \eta \nabla_{K_{i,t}^m} J_i. \tag{3.13}$$

The gradient of the cost function is calculated by the following equation [7]:

$$\nabla_{K_{i,t}^m} J_i = 2(R_{i,t}K_{i,t}^m - B_{i,t}^T P_{i,t+1}^m (A_t - \sum_{j=1}^{N} B_{j,t}K_{j,t}^m))\Sigma_t. \tag{3.14}$$

where

$$P_{i,t}^m = Q_{i,t} + (K_{i,t}^*)TR_{i,t}K_{i,t}^* + (A_t - \sum_{j=1}^{N} B_{j,t}K_{j,t}^*)^T P_{i,t+1}^m (A_t - \sum_{j=1}^{N} B_{j,t}K_{j,t}^*), \tag{3.15}$$

$$\Sigma_t = \mathbb{E}[x_t x_t^T]. \tag{3.16}$$

Because we do not have access to the parameters of the game, the exact gradient is unknown. However, we need to use the gradient's exact value for simulations to see how the policy gradient method is doing in different settings. One simple way to estimate gradients is to use zero-order methods, such as examples in the last chapter.

3.1.3 Stochasticity in LQ games

Like in many other games and systems around us, randomness plays an important role due to our lack of knowledge. This stochasticity can come into play when we have an LQR system or an LQ game. This randomness can be attributed to the state's random initialization, where the initial state is drawn from a distribution $\mathcal{D}$, i.e., $x_0 \sim \mathcal{D}$. Alternatively, it could be due to the presence of noise in our environment, which can affect the game's dynamics, i.e., $x_{t+1} = Ax_t + \sum_{i=1}^{N} B_i u_{i,t} + w_t$, where the noise w_t at time t can be a Gaussian random variable. It is observed that this randomness can have a meaningful impact on the convergence of the PG algorithm [7]. In this work, we mainly focus on deterministic LQ games and stochastic LQ games with random initialization.

3.1.4 Potential Games

As we said earlier, our focus is on the convergence of the policy gradient in potential LQ games because we expect them to have a convergence guarantee. So let us review the definition and properties of a static potential game.

Definition 3.1.1. *The game $J_i(u_i, u_{-i})$ is exact potential if there exists a potential function Π such that for every u and i we have:*

$$J_i(u_i, u_{-i}) - J_i(v_i, u_{-i}) = \Pi(u_i, u_{-i}) - \Pi(v_i, u_{-i}). \tag{3.17}$$

With assumptions on differentiability, we have a necessary and sufficient condition for a game to be potential. Because this potential function is not easily visible to us, the following condition assists us in determining whether a game is potential or not [9]:

$$\frac{\partial^2 J_i}{\partial u_i \partial u_j} = [\frac{\partial^2 J_j}{\partial u_i \partial u_j}]^T. \tag{3.18}$$

We might come across different definitions for *Dynamic Potential Games* (DPG). We mention a few of them:

Definition 3.1.2. *The game $J_i(x_0, u_i, u_{-i})$ is dynamic potential if there is a potential function Π such that for every x_0, u and i we have:*

$$J_i(x_0, u_i, u_{-i}) - J_i(x_0, v_i, u_{-i}) = \Pi(x_0, u_i, u_{-i}) - \Pi(x_0, v_i, u_{-i}). \tag{3.19}$$

Definition 3.1.3. *The game $J_i(x_0, u_i, u_{-i})$ is DPG if there is a family of potential functions Π_t such that for every x_0, u, t and i [26]:*

$$\sum_{t=0}^{\infty} \beta^t (c_{i,t}(x_t, u_i, u_{-i}) - c_{i,t}(x_t, v_i, u_{-i})) = \sum_{t=0}^{\infty} \beta^t (\Pi_t(x_t, u_i, u_{-i}) - \Pi_t(x_t, v_i, u_{-i})). \tag{3.20}$$

Definitions 3.17 and 3.22 are equivalent to condition 3.18 and they are easily used for the case when the game is deterministic. However for 3.20, condition 3.18 it has a sufficient condition in the form of:

$$\frac{\partial^2 c_{i,t}}{\partial u_{i,t} \partial u_{j,t}} = \frac{\partial^2 c_{j,t}}{\partial u_{j,t} \partial u_{i,t}}. \tag{3.21}$$

There are also different, but similar definitions like what is used in [17], but we do not work with them. It's worth mentioning that when we have random initialization, we can change 3.22 to:

$$\mathbb{E}_{\mathcal{D}}[J_i(x_0, u_i, u_{-i}) - J_i(x_0, v_i, u_{-i})] = \mathbb{E}_{\mathcal{D}}[\Pi(x_0, u_i, u_{-i}) - \Pi(x_0, v_i, u_{-i})], \tag{3.22}$$

remember that $\mathcal{D}$ was the distribution that we draw the initial state x_0 from. Or we can take the expectation w.r.t. any randomness we have in the game and call it the condition for potentialness of that game.

There are also two terms that we will use further down at the end of this chapter.

- **Dummy player**: A dummy player is whose utility function is independent of his/her actions, i.e., $f_i(x_i, x_{-i}) = f_i(x_{-i})$.
- **Dummy game**: A dummy game is a game that all of it's player are dummy, i.e., for every i, $f_i(x_i, x_{-i}) = f_i(x_{-i})$.

Theorem 3.1.4. *A dummy game is potential with the potential function* $\Pi = 0$.

Proof. Using differentiation rules. □

3.2 An examples of LQ game

Now we bring an example of *linear quadratic game* mentioned in [26]. Consider a dynamic in the form $x_{t+1} = Ax_t + \sum_{i\in N} B_i u_t^i$ and utility functions $c_{i,t} = (x_t - x_{t-1})^T Q(x_t - x_{t-1}) + (D^i x_t - u_t^i)^T R_i (D^i x_t - u_t^i)$, where matrices R_i and Q are negative semi-definite and definite respectively.

By defining augmented state and action vectors:

$$\tilde{x_t}^T := [x_t^T, x_{t-1}^T]^T, \qquad \tilde{u_t}^T := D^i x_t - u_t^i.$$

we can rewrite the problem as follows:

$$\max_{\{u_t^i\}\in\prod_{t=0}^{\infty}\mathcal{U}^i} \sum_{t=0}^{\infty} \beta^t (\tilde{x_t}^T \tilde{R} \tilde{x_t} + \tilde{u_t^i}^T R_i \tilde{u_t}^i),$$
$$\text{s.t. } \tilde{x}_{t+1} = A\tilde{x}_t - \sum_{i=1}^{N} \tilde{B}^i \tilde{u_t}^i,$$
$$\forall i \in \mathcal{N}.$$

where

$$\tilde{A} := \begin{bmatrix} A + \sum_{i\in N} B^i D^i & 0_{S\times S} \\ I_S & 0_{S\times S} \end{bmatrix}, \qquad \tilde{B}^i := \begin{bmatrix} B^i \\ 0_{S\times A^i} \end{bmatrix},$$
$$\tilde{Q} := \begin{bmatrix} Q & -Q \\ -Q & Q \end{bmatrix}.$$

where S and A^i are the appropriate dimensions. Now we can rewrite utilities as follows:

$$c_{i,t}(\tilde{x}_t, \tilde{u}_t) = \tilde{x}_t^T \tilde{R} \tilde{x}_t + \tilde{u}_t^{iT} R_i \tilde{u}_t^i.$$

which is a LQ game.

3.3 Potential games and their Conditions

This part aims to introduce the tools we have to identify the sub-class of potential LQ games and find a theoretical motivation for working with them. We use theorems to drive the sufficient and necessary conditions for a game to be potential. In addition, there will be a brief analysis that shows using PG in a potential game is the equivalent of using PG in a one-player system.

3.3.1 Convergence of Potential LQ games

Let us take a look at 3.17. One by dividing the sides by $\|u_i - v_i\|$ and taking the limit when $v_i \to u_i$ can get

$$\nabla_i J_i(x_0, u_i, u_{-i}) = \nabla_i \Pi(x_0, u_i, u_{-i}), \tag{3.23}$$

this means that the game map's gradient equals the gradient of the potential function. Now, if all the players use a policy gradient with an equal learning rate η, we will have:

$$u_1^+ = u_1^+ - \eta \nabla_1 J_1(x_0, u), \tag{3.24}$$

$$\vdots$$

$$u_N^+ = u_N^+ - \eta \nabla_N J_N(x_0, u). \tag{3.25}$$

by replacing $\nabla_i J_i(x_0, u)$ with $\nabla_i \Pi(x_0, u)$ stacking the above equations:

$$u^+ = u^+ - \eta \nabla \Pi(x_0, u), \tag{3.26}$$

which is the policy gradient for a single-player game with the cost function of $\Pi(.)$. The dynamic for this single player game is

$$x_{t+1} = A x_t + \tilde{B} u_t, \tag{3.27}$$

$$\tilde{B} = \begin{bmatrix} B_1 & \dots & B_n \end{bmatrix}. \tag{3.28}$$

Connecting this observation to [4] motivates us to have a potential LQ game with the quadratic potential function. Such a game should converge to the Nash because the optimum point for the potential function is a Nash equilibrium for the original game.

3.3.2 Open loop potential LQ games

In this part the objective is to derive the conditions for a finite-horizon deterministic LQ game to be potential for horizon of t_f. Here we try to write cost function and then use 3.18 to see what should be relation between parameters of the game to make that happen. The cost function for such game looks like:

$$J_i(\mathbf{u}_1, \dots, \mathbf{u}_N, x_0) = \sum_{t=0}^{t_f-1} \left(x_t^T Q_i x_t + u_{i,t}^T R_i u_{i,t} \right) + x_{t_f}^T Q_i x_{t_f}, \tag{3.29}$$

where $x_t = Ax_{t_1} + \sum_{i=1}^{N} B_i u_{i,t-1}$. So we want to replace x_t's such that 3.29 will be only a function of x_0 and $u_{i,t}$'s. We define $u_i = [u_{i,0}^T, \dots, u_{i,t_f-1}]^T$ the stacked vector of all actions played by the i'th player, then we can easily use 3.18 to see what are the conditions. Let

$$\mathcal{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_{t_f} \end{bmatrix}, \ \mathcal{S}^x = \begin{bmatrix} A \\ \vdots \\ A^{t_f} \end{bmatrix}, \ \mathcal{S}^{u_i} = \begin{bmatrix} B_i & \dots & 0 \\ AB & \dots & 0 \\ \vdots & \ddots & \vdots \\ A^{t_f-1}B_i & \dots & B_i \end{bmatrix}. \tag{3.30}$$

We can re-write $\mathcal{X}$ as [3]:

$$\mathcal{X} = \mathcal{S}^x x_0 + \sum_{i=1}^{N} \mathcal{S}^{u_i} \mathbf{u}_i, \tag{3.31}$$

and

$$J_i(x, \mathbf{u}) = x_0^T Q_i x_0 + \mathcal{X}^T \tilde{Q}_i \mathcal{X} + \mathbf{u}_i^T \tilde{R}_i \mathbf{u}_i, \tag{3.32}$$

where u and x are respectively stacked vectors of u_i's and x_t's, and $\tilde{Q}_i^{(mt_f \times mt_f)} = \text{blockdiag}\{Q_i, \dots, Q_i\}$, and $\tilde{R}_i^{(d_i t_f \times d_i t_f)} = \text{blockdiag}\{R_i, \dots, R_i\}$. To make x_t for $t > 0$ disappear, we re-write is as follow:

$$J_i(\mathbf{u}, x_0) = x_0^T (Y^i + Q_i) x_0 + 2x_0^T \sum_{j=1}^{N} F_j^i \mathbf{u}_j + \sum_{j=1}^{N} \sum_{l=1}^{N} \mathbf{u}_j^T H_{j,l}^i \mathbf{u}_l + \mathbf{u}_i^T \tilde{R}_i \mathbf{u}_i, \tag{3.33}$$

$$Y^i = \mathcal{S}^{x^T} \tilde{Q}_i \mathcal{S}^x, \ F_j^i = \mathcal{S}^{x^T} \tilde{Q}_i \mathcal{S}^{u_j}, \ H_{j,l}^i = \mathcal{S}^{u_j{}^T} \tilde{Q}_i \mathcal{S}^{u_l}. \tag{3.34}$$

Theorem 3.3.1. *The LQ game is potential if and only if $H_{j,i}^i = H_{j,i}^j$ for all players.*

Proof. Let us apply Lemma 4. Recall $\nabla_{\mathbf{u}_i} J_i$ as a linear function of actions

$$\nabla_{\mathbf{u}_i} J_i = 2{F_i^i}^T x_0 + 2\sum_{j=1}^{N} H_{i,j}^i \mathbf{u}_j + 2\tilde{R}_i \mathbf{u}_i. \tag{3.35}$$

The second derivative with respect to actions of other players is $\frac{\partial^2 J_i}{\partial \mathbf{u}_j \mathbf{u}_i} = H_{j,i}^i$. According to 3.18, $\frac{\partial^2 J_i}{\partial \mathbf{u}_j \mathbf{u}_i} = \left(\frac{\partial^2 J_j}{\partial \mathbf{u}_i \mathbf{u}_j}\right)^T$. Thus, $H_{j,i}^i = H_{j,i}^j$. □

If we open this condition up, we will get:

$$\sum_{f=t}^{t_f} \beta^f \left[2B_i^\top (A^\top)^{f-t} Q_i A^{f-t} B_j\right] = \sum_{f=t}^{t_f} \beta^f \left[2B_j^\top (A^\top)^{f-t} Q_j A^{f-t} B_i\right]^T,$$

for all t. Where $0 < \beta < 1$ is the discount factor of the game.

One simple case to satisfy such condition is to have $Q_i = Q$ for all i. To add more detail, let's re-write the cost function as:

$$\begin{aligned} J_i(\mathbf{u}_1, \ldots, \mathbf{u}_N, x_0) = &\sum_{t=0}^{t_f-1} \left(x_t^T Q_i x_t + \sum_{j=1}^{N} u_{j,t}^T R_j u_{j,t}\right) + x_{t_f}^T Q_i x_{t_f} \\ &- \sum_{t=0}^{t_f-1} \sum_{j\neq i} u_{j,t}^T R_j u_{j,t}. \end{aligned} \tag{3.36}$$

So the cost function of player i is sum of two terms $\sum_{t=0}^{t_f-1} \left(x_t^T Q_i x_t + \sum_{j=1}^{N} u_{j,t}^T R_j u_{j,t}\right) + x_{t_f}^T Q_i x_{t_f}$ and $-\sum_{t=0}^{t_f-1} \sum_{j\neq i} u_{j,t}^T R_j u_{j,t}$. The first term is the same for all the players. A game with such a cost function is called an *identical interest* game. Identical interest games are potential, with the potential function the same as each player's cost function. Now look to the second term. This term does not depend on ith player's action. This game is called a *dummy* game, which is also a potential game with a potential function of 0. It is known that a game whose players' cost functions are the sum of two other potential game cost functions is a potential game [Bring the proof/ reference]. So as a sanity check, it implies that an open-loop LQ game with $Q_i = Q$ for all i is a potential game.

However, this is not a necessary condition for a game being potential. Below, there is a game that $Q_1 \neq Q_2$ and it is open-loop potential:

$$\mathrm{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathrm{B}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, B_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, Q_1 = \begin{bmatrix} 5 & -2 \\ -2 & 1 \end{bmatrix}, Q_2 = \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix}.$$

It is easily checkable that the following identities always hold:

$$B_1^T (A^T)^{t'-t} Q_1 A^{t'-t} B_2 = B_2^T (A^T)^{t'-t} Q_2 A^{t'-t} B_1, \qquad t' > t.$$

which is the same as $\mathcal{S}^{u_2} H_{2,1}^1 \mathcal{S}^{u_1 T} = \mathcal{S}^{u_2} H_{2,1}^2 \mathcal{S}^{u_1 T}$.

3.3.3 Closed loop potential LQ games

Simple scalar case:

We consider a case where all actions and states are scalar. Here we assume that we play the same feedback coefficient at all time steps. The cost function for such a game is as follows.

$$J_i(K, x_0) = \sum_{t=0}^{T} (A + \sum_r B_r K_r)^{2t} (Q_i + K_i^2 R_i) x_0^2, \quad (3.37)$$

$$\frac{\partial^2 J_i}{\partial K_i \partial K_j} = \sum_t \Bigg((2t)(2t-1) B_i B_j (A + \sum_r B_r K_r)^{2t-2} (Q_i + K_i^2 R_i) \quad (3.38)$$
$$+ B_j (2t)(A + \sum_r B_r K_r)^{2t-1} (2K_i R_i) \Bigg) x_0^2 \qquad i \neq j,$$

$$\frac{\partial^2 J_i}{\partial K_i \partial K_j} = \sum_t \Bigg((2t)(2t-1) B_i^2 (A + \sum_r B_r K_r)^{2t-2} (Q_i + K_i^2 R_i) \quad (3.39)$$
$$+ 2(2t) B_i (A + \sum_r B_r K_r)^{2t-1} (2K_i R_i)$$
$$+ (A + \sum_r B_r K_r)^{2t} (2R_i) \Bigg) x_0^2 \qquad i = j.$$

so the necessary and sufficient condition for it to be potential is:

$$B_i B_j (Q_i + K_i^2 R_i) + B_j (A + \sum_r B_r K_r)(2K_i R_i) \quad (3.40)$$
$$= B_i B_j (Q_j + K_j^2 R_j) + B_i (A + \sum_r B_r K_r)(2K_j R_j).$$

for every i, j, and for all K. Which implies that in the two following cases the game is potential:

For every pair of i, j we have at least one of the followings cases:

- $B_i = B_j = 0$.
- $R_i = R_j = 0$, and $Q_i = Q_j$.

The first case implies that players i and j are decoupled from the game (their actions will not affect other players' actions because they are not affecting the dynamic). In contrast, the second case implies that players i and j are playing the same game of sending the state to zero.

Lemma 3.3.2. *For the first case, i.e, when $B_i = B_j = 0$ the Nash is to select $K_i = K_j = 0$.*

Proof. when our player's action $u_{i,t}$ is not affecting x_{t+1}, it means that we have no control over the term $x_t^T Q_i x_t$. However for the other term in our cost function, which is $u_{i,t}^T R_i u_{i,t}$, all we can do is to minimize it by selecting $u_{i,t} = 0$, so $K_i = 0$. More precisely, according to the equation (7a) in [1]:

$$u_i^*(x) = -R_i^{-1} B_i^T M_i \Lambda^{-1} \bar{A} x.$$

which is zero if $B_i = 0$. The solution $K = 0$ also satisfies in equation 16 in [1]. □

We performed the simulation for two cases of scalar actions, scalar states and scalar actions, 2d states, and the policy gradient method converged to the Nash equilibrium. For the second case, as we have in equation 7a in [1], we cannot compute the Nash equilibrium, because of the existence of R_i^{-1} in the equation. However, intuitively, as the only term, they have in their cost function is $x_t^T Q x_t$, they are interested in directing the states to the point that the mentioned term is close to zero.

As a conclusion, using the sufficient conditions we derived for the closed loop case to be potential, our players are partitioned into equivalency classes of decoupled players and players that are clones of one single player. For the first set of players, we will have a trivial solution of $K_i = 0$, and for the second set of players, we have to calculate the optimum with the method we were using in a single-player LQR system.

Scalar case with time horizon of 2:

In the next step, we study the case that the horizon is finite, and we play different K's at each time step. If we assume that time horizon is $T = 1$,

the cost functions will be:

$$\begin{aligned} J_1(K, x_0) =& x_0^T Q_1 x_0 + x_0^T K_1^T R_1 K_1 x_0 \\ & x_0^T \bar{A}^T Q_1 \bar{A} x_0 + x_0^T \bar{A}^T K_1^T R_1 K_1 \bar{A} x_0, \\ J_2(K, x_0) =& x_0^T Q_2 x_0 + x_0^T K_2^T R_2 K_2 x_0 \\ & x_0^T \bar{A}^T Q_1 \bar{A} x_0 + x_0^T \bar{A}^T K_2^T R_2 K_2 \bar{A} x_0. \end{aligned}$$

where $\bar{A} = A + B_1 K_1 + B_2 K_2$.

$$\begin{aligned} \frac{\partial J_1}{\partial K_1} &= 2K_1 R_1 x_0^2 + 2\bar{A} Q_1 B_1 x_0^2 + 2B_1 \bar{A} R_1 K_1^2 x_0 + 2\bar{A}^2 K_1 R_1 x_0^2, \\ \frac{\partial J_1}{\partial K_2} &= 2\bar{A} Q_1 B_2 x_0^2 + 2\bar{A} R_1 K_1^2 B_2 x_0^2, \\ \frac{\partial^2 J_1}{\partial K_2 \partial K_1} &= 2B_1 B_2 Q_1 x_0^2 + 2R_1 K_1^2 B_1 B_2 x_0^2 + 4\bar{A} R_1 K_1 B_2 x_0^2. \end{aligned}$$

similarly:

$$\begin{aligned} \frac{\partial J_2}{\partial K_2} &= 2K_2 R_2 x_0^2 + 2\bar{A} Q_2 B_2 x_0^2 + 2B_2 \bar{A} R_2 K_2^2 x_0 + 2\bar{A}^2 K_2 R_2 x_0^2, \\ \frac{\partial J_2}{\partial K_1} &= 2\bar{A} Q_2 B_1 x_0^2 + 2\bar{A} R_2 K_2^2 B_1 x_0^2, \\ \frac{\partial^2 J_2}{\partial K_2 \partial K_1} &= 2B_1 B_2 Q_2 x_0^2 + 2R_2 K_2^2 B_1 B_2 x_0^2 + 4\bar{A} R_2 K_2 B_1 x_0^2. \end{aligned}$$

Again if we put these to 3.18, we'll get same results as above. Which is that at least one of the followings holds:

- $B_1 = B_2 = 0$.
- $R_1 = R_2 = 0$ and $Q_1 = Q_2$.

The first condition indicates that the game is uncoupled, and players have no control over the state x_t. The second condition means that both players enjoy the same cost function and it's an identical interest game. As it is apparent, the necessary and sufficient conditions for the game to be potential remained the same.

2 dimensional case with time horizon 2:

For the next step, we relax our assumption to find conditions for the game, that its dimension is higher than 1. We check the case that the states are in $\mathbb{R}^2$ and our actions are in $\mathbb{R}$. To simplify our calculations keep the followings in mind:

1. In the closed-loop finite-horizon LQ game, at each step, we play $u_{i,t} = -K_{i,t}x_t$, which is a linear feedback of our current state.

2. The necessary and sufficient condition for our game be potential is that $\frac{\partial^2 J_i}{\partial u_j \partial u_i} = \frac{\partial^2 J_j}{\partial u_i \partial u_j}$, where u_i and u_j are the stacked vectors of the actions that are played by player i. As here the actions we are playing are the coefficients k, so the mentioned condition turns to: $\frac{\partial^2 J_i}{\partial k_j \partial k_i} = \frac{\partial^2 J_j}{\partial k_i \partial k_j}$, where k_i is the stacked matrix of coefficients of player i.

3. The calculation of the above condition is extremely hard to achieve, but we can break the condition down to the following equivalent conditions, $\frac{\partial^2 J_i}{\partial k_{j,\tau} \partial k_{i,\tau'}} = \frac{\partial^2 J_j}{\partial k_{i,\tau'} \partial k_{j,\tau}}$, for all $\tau, \tau' \leq T$, where T is the horizon of the game.

Let us take a look at structure of cost functions for a two player game.

$$
\begin{aligned}
J_1(k_1, k_2) =& x_0^T [Q_1 + k_{1,0}^T R_1 k_{1,0}] x_0 \\
&+ x_0^T (A - B_1 k_{1,0} - B_2 k_{2,0})^T [Q_1 + k_{1,0}^T R_1 k_{1,0}](A - B_1 k_{1,0} - B_2 k_{2,0}) x_0 \\
&\vdots \\
&+ x_0^T \bar{A}_0^T \dots \bar{A}_{t-1}^T [Q_1 + k_{1,t}^T R_1 k_{1,t}] \bar{A}_{t-1} \dots \bar{A}_0 x_0 \\
&\vdots \\
&+ x_0^T \bar{A}_0^T \dots \bar{A}_{T-1}^T [Q_1 + k_{1,T}^T R_1 k_{1,T}] \bar{A}_{T-1} \dots \bar{A}_0 x_0.
\end{aligned}
$$

where $\bar{A}_t = A - B_1 k_{1,t} - B_2 k_{2,t}$.

To simplifying the explanation let's call the term $x_0^T \bar{A}_0^T \dots \bar{A}_{t-1}^T [Q_1 + k_{1,t}^T R_1 k_{1,t}] \bar{A}_{t-1} \dots \bar{A}_0 x_0$ as $c_{1,t}$. Now note that for calculation of $\frac{\partial^2 J_i}{\partial k_{j,\tau} \partial k_{i,\tau'}} = \frac{\partial^2 J_j}{\partial k_{i,\tau'} \partial k_{j,\tau}}$ all the $c_{1,t}$ with $t < \tau$ or $t < \tau'$ has no say in the final expression, because their differentiation with respect to one of the $k_{j,\tau}$ and $k_{i,\tau'}$ will be zero. So we can re-write our condition as follows:

$$
\sum_{t' \geq \max\{\tau, \tau'\}} \frac{\partial^2 c_{i,t'}}{\partial k_{j,\tau} \partial k_{i,\tau'}} = \sum_{t' \geq \max\{\tau, \tau'\}} \frac{\partial^2 c_{j,t'}}{\partial k_{i,\tau'} \partial k_{j,\tau}} \tag{3.41}
$$

Now we only focus on $\frac{\partial^2 c_{i,t'}}{\partial k_{j,\tau} \partial k_{i,\tau'}}$ with $t' \geq \max\{\tau, \tau'\}$.

$$
c_{1,t'} = x_0^T \bar{A}_0^T \dots \bar{A}_{t'-1}^T [Q_1 + k_{1,t'}^T R_1 k_{1,t'}] \bar{A}_{t'-1} \dots \bar{A}_0 x_0.
$$

As our functions are smooth, we can change the order of differentiation if we want. Due to symmetry without loss of generality, we assume $\tau \leq \tau'$. We calculate $\frac{\partial^2 c_{1,t'}}{\partial k_{i,\tau'} \partial k_{j,\tau}}$:

$$\frac{\partial c_{1,t'}}{\partial k_{j,\tau}} = 2(\bar{A}_{\tau-1} \dots \bar{A}_0 x_0)(\bar{A}_{\tau-1} \dots \bar{A}_0 x_0)^T \Big(\Big[\bar{A}_{\tau+1}^T \dots \bar{A}_{t'-1}^T [Q_1 + k_{1,t'}^T R_1 k_{1,t'}] \bar{A}_{t'-1} \dots \bar{A}_{\tau+1} \Big] \Big) \bar{A}_\tau B_j.$$

So

$$\frac{\partial^2 c_{1,t'}}{\partial k_{i,\tau'} \partial k_{j,\tau}} = 2B_i^T (\bar{A}_{\tau-1} \dots \bar{A}_0 x_0)(\bar{A}_{\tau-1} \dots \bar{A}_0 x_0)^T \Big(\Big[\bar{A}_{\tau'+1}^T \dots \bar{A}_{t'-1}^T [Q_1 + k_{1,t'}^T R_1 k_{1,t'}] \bar{A}_{t'-1} \dots \bar{A}_{\tau'+1} \Big] \Big) \bar{A}_\tau \dots \bar{A}_{\tau'} B_j.$$

Putting all together gives us:

$$\sum_{t' \geq \max\{\tau,\tau'\}} \frac{\partial^2 c_{1,t'}}{\partial k_{j,\tau} \partial k_{i,\tau'}} = \sum_{t' \geq \max\{\tau,\tau'\}} 2B_i^T (\bar{A}_{\tau-1} \dots \bar{A}_0 x_0)(\bar{A}_{\tau-1} \dots \bar{A}_0 x_0)^T \Big(\Big[\bar{A}_{\tau'+1}^T \dots \bar{A}_{t'-1}^T [Q_1 + k_{1,t'}^T R_1 k_{1,t'}] \bar{A}_{t'-1} \dots \bar{A}_{\tau'+1} \Big] \Big) \bar{A}_\tau \dots \bar{A}_{\tau'} B_j = \sum_{t' \geq \max\{\tau,\tau'\}} 2B_j^T (\bar{A}_{\tau-1} \dots \bar{A}_0 x_0)(\bar{A}_{\tau-1} \dots \bar{A}_0 x_0)^T \Big(\Big[\bar{A}_{\tau'+1}^T \dots \bar{A}_{t'-1}^T [Q_2 + k_{2,t'}^T R_2 k_{2,t'}] \bar{A}_{t'-1} \dots \bar{A}_{\tau'+1} \Big] \Big) \bar{A}_\tau \dots \bar{A}_{\tau'} B_i = \sum_{t' \geq \max\{\tau,\tau'\}} \frac{\partial^2 c_{2,t'}}{\partial k_{i,\tau'} \partial k_{j,\tau}}.$$

The above condition should be correct for all $\tau, \tau' < T$.

In the first part of this section, we studied the case of scalar games where the actions played by each player are constant over time. Here our

goal is to compare these conditions to the case that actions and states are all scalars. If we do the same analysis for the finite horizon case, we will have the following conditions for this game's potential.

$$\sum_{t' \geq \max\{\tau,\tau'\}} x_0^2(Q_1 + k_{1,t'}^2 R_1)\frac{\bar{A}_0^2 \dots \bar{A}_{t'-1}^2}{\bar{A}_\tau \bar{A}_{\tau'}} B_1 B_2 =$$
$$\sum_{t' \geq \max\{\tau,\tau'\}} x_0^2(Q_2 + k_{2,t'}^2 R_2)\frac{\bar{A}_0^2 \dots \bar{A}_{t'-1}^2}{\bar{A}_\tau \bar{A}_{\tau'}} B_1 B_2.$$

Note that the division by $\bar{A}_\tau \bar{A}_{\tau'}$ is informal and we don't have problem with dividing anything by zero.

The conditions above for a game with time horizon of $T = 1$ will be:

$$(Q_1 + k_{1,1}^2 R_1)B_1 B_2 = (Q_2 + k_{2,1}^2 R_2)B_1 B_2 0$$

which leads the same conclusion as we had before. Now we do the same with the conditions of a $2d$ game we have had.

$$B_1^T(Q_1 + k_{1,1}^T R_1 k_{1,1})B_2 = B_2^T(Q_2 + k_{2,1}^T R_2 k_{2,1})B_1.$$

Both the right-hand and left-hand sides are symmetric (They are either scalar or the second derivative matrix of one smooth cost function.) So we can summarize it as follows:

$$B_2^T(Q_2 - Q_1 + k_{2,1}^T R_2 k_{2,1} - k_{1,1}^T R_1 k_{1,1})B_1 = 0. \tag{3.42}$$

For every $k_{1,1}$ and $k_{2,1}$. Similarly if any of B_1 or B_2 are zero then the game is uncoupled and potential. Also if $Q_1 = Q_2$ and $R_1 = R_2 = 0$ we have the same thing.

Note that as it is a finite horizon case we don't care about stability. So $k_{i,1}$ can be anything. So if B_1 and B_2 are not zero then we can make the expression non-zero. Which means that at least one of the R_i's should be zero to make the game potential, which is in contrary to our assumption R_i being non-zero.

For higher time horizons, the condition will have the same structure as 3.42, so again when we have to unite the expression to zero, we come to the same conclusion.

3.3.4 Special cases of closed-loop

As seen above, because K_1 and K_2 could be arbitrary, we had to make the coefficient zero to eliminate them. With the hope of overcoming this

problem, let us consider a case in which one of the players follows the other, so it does not have the freedom to make potential games trivial. We define trivial as matrices B_i or R_i having to be zero. We are now interested in games that are less trivial. Except for one agent, whose utility depends solely on its own action, i.e., $f_1(x_1, x_{-1}) = f_1(x_1)$, every other agent in the game is a dummy. These games are potential too, because $\frac{\partial^2 f_i}{\partial x_i \partial x_j} = 0$.

What happens if the game is not trivial? i.e., Q_i's are different R_i's and B_i's are non-zero, but $K_{i,t} = \lambda_i K_{1,t}$ (One Leader and others are Follower), λ_i can be a matrix. In this game, **if λ_i's are parameters (not variables)** player's one loss function is only a function of its actions because we can replace all other players' feedback with $\lambda_i K_1$. All other players' cost functions are only a function of player one's action. So it means the game is potential, and also, the game is not trivial.

Furthermore, this game corresponds to an LQ system with $B_1' = B_1 + \sum_{i \neq 1} \lambda_i B_i$. Thus, the policy gradient guarantees the convergence of player one to its optimum action **for the leader**. However, going to the optimum action is meaningless for the followers because their optimum action is not in the form of $K_{i,t} = \lambda_i K_{1,t}$. Generally, if all other players follow the first player's action by a function, i.e., $a_i = h_i(a_1)$, we conjecture that the same argument can be repeated and the game has non-trivial potential.

In all these cases we discussed above, other players were all supposed to play a single function of the first player's action. Now, the question is, what happens if they had some freedom to choose that function? For example, they could choose $\lambda_{i,t} = \alpha_{i,t}\lambda_i$ at each time sequence to scale K_1 differently at each step. Here are, players are not dummies anymore. Because they have a say in their cost function by choosing the $\alpha_{i,t}$, also the potential conditions will be:

$$\frac{\partial^2 f_i}{\partial K_1^2} = \frac{\partial^2 f_j}{\partial K_1^2}. \tag{3.43}$$

The above-mentioned law is derived from the chain rule when i's are scalar. It also implies that the potential condition for such a game is that the game is trivial ($Q_i = Q$, $R_i = 0$).

Another way of looking at this game can be to model it as an LQ system with a time-varying matrix $B = B_t = B_1 + \sum_{i \neq 1} \alpha_{i,t}\lambda_i B_i$.This system is an online optimization for the first player because parameters B_t are determined by other players and change over time. Here we should note that every game is a kind of online optimization, but when we restrict the actions of other players to $\lambda_i K_1 x_t$, the time-varying function is changing limited compared

to the general case. In other words, the only parameter changing is B_t, only with linear combinations of B_i's $i \neq 1$.

3.4 Simulations

Here in this section, we perform simulations on the number of games to see if open-loop LQ games have any convergence guarantee when played in a closed-loop setting. The parameters of these two games are the followings. The first one:

$$Q_1 = Q_2 = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}, B_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, B_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$
$$A = \begin{bmatrix} 0.6491 & 0.6477 \\ 0.7317 & 0.4509 \end{bmatrix}, K_{eq} = \begin{bmatrix} 0.7239 & 0.4483 \\ -0.0390 & 0.1025 \end{bmatrix}.$$

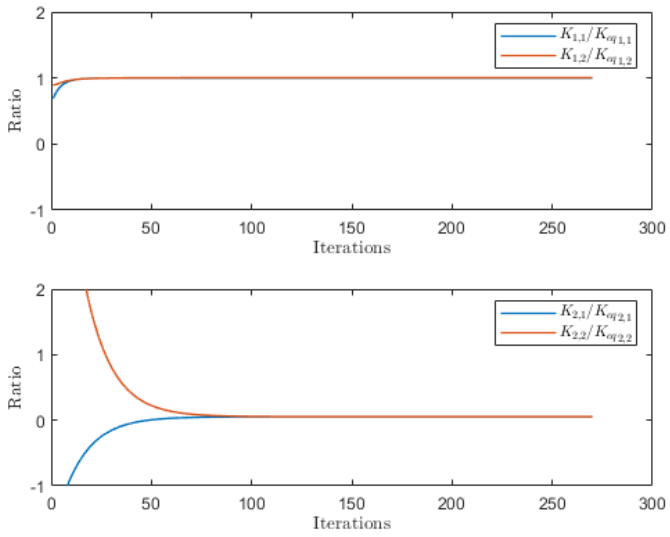
And for the second one:

$$Q_1 = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}, Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0.001 \end{bmatrix}, B_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, B_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$
$$A = \begin{bmatrix} 0.4253 & 0.1615 \\ 0.3127 & 0.1788 \end{bmatrix}, K_{eq} = \begin{bmatrix} 0.3096 & 0.1770 \\ 0.1145 & -0.0154 \end{bmatrix}.$$

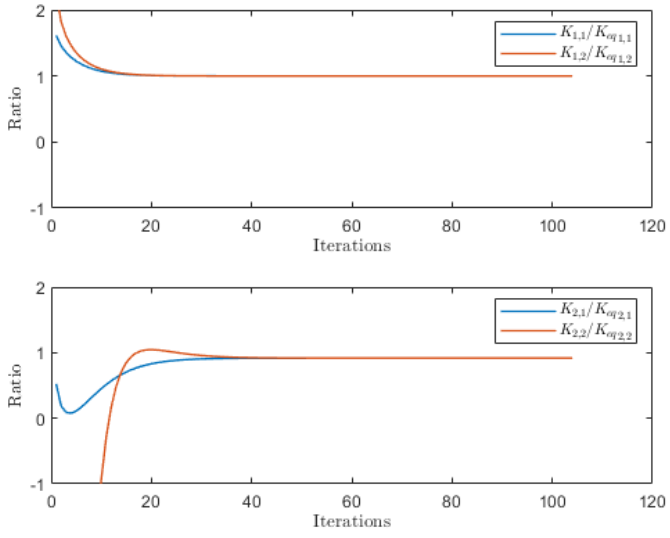
For both of them x_0 selected randomly from the same distribution.

As you see, the first game is open-loop potential because $Q_1 = Q_2$. However, as it is apparent in the figure bellow, K_2 and K_{eq2} have different values while the iterations are going forward. However, the second game is not potential and converges to the Nash equilibrium.

The takeaway from this experiment is that, although we know that potential open-loop LQ games must converge to Nash while PG is used as the learning tool, there is no guarantee when the same game is used in a closed-loop setting.



(a) PG for first game



(b) PG for the second game

Figure 3.2: Convergence Trajectories

Moreover, to sanity check, we have done another experiment for trivial potential closed-loop games and observed the convergence to the Nash. Here

are the specifications of the game:

$$Q_1 = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}, Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0.001 \end{bmatrix}, B_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, B_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$
$$A = \begin{bmatrix} 0.0542 & 0.6628 \\ 0.1771 & 0.3308 \end{bmatrix}, K_{eq} = \begin{bmatrix} 0.1741 & 0.3310 \\ 0 & 0 \end{bmatrix}.$$

You can see the results in the figure bellow:

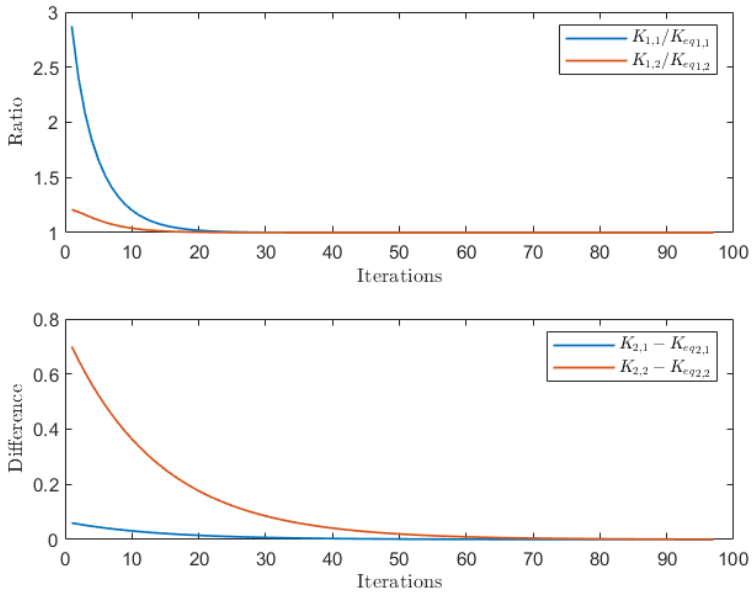


Figure 3.3: Trivial Closed Loop Potential

Note that as the optimal action by the second player is $K_{eq} = 0$, we depict the difference between the action played and the optimal action in the figure above.

As a verification for the special cases that discusses in the previous part, a simulation on the following leader-follower setting is performed.

$$Q_1 = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}, Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0.001 \end{bmatrix}, B_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, B_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$
$$A = \begin{bmatrix} 0.0542 & 0.6628 \\ 0.1771 & 0.3308 \end{bmatrix}, K_{eq} = \begin{bmatrix} 0.1753 & 0.3276 \\ -0.1200 & 0.3319 \end{bmatrix},$$
$$K_2 = 0.5K_1.$$

As expected, the leader converged to the optimum action, but the follower did not.

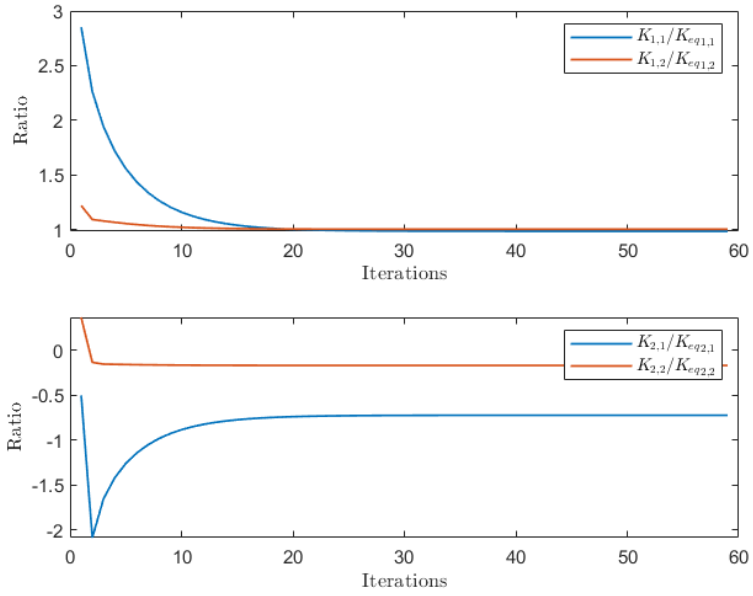


Figure 3.4: Special Potential Case

3.5 Discussion

We identified the class of deterministic potential LQ games, both for open-loop and closed-loop settings. In the last section, we tried to use an open-loop potential game in a closed-loop setting, and we observed that it didn't converge to Nash equilibrium, given by the Riccati equations. It suggests that, although potential games have the property of one-player games, they might not do so when they are used in a different setting. For the closed-loop case, there are no non-trivial potential games.

We also analyzed some imaginary leader-follow settings and proved that they are equivalent to a single-player LQ game, so potential. This result, along with the conclusion for the trivial closed-loop and potential open-loop games (when used in an open-loop setting), is our convergence guarantee in LQ games.

Chapter 4

Conclusion and Future works

To address the slow convergence to Nash in games, this work provides novel accelerated online algorithms with proofs of their regret bounds. As it is shown that their regret is sub-linear, they have a convergence guarantee when they are employed in Cournot games. These algorithms have been put into the simulations with **zero-order** feedback, and we observed that they are doing better in terms of speed of convergence compared to old algorithms that had the same regret bound, both in Cournot games and Quadratic games, but without providing a mathematical proof.

As for the next step, we would work on providing a convergence rate with a mathematical proof. Moreover, one could extend the derived regret bound for the algorithms 2 and 3 to the zero-order setting. Also, it would be helpful to calculate the regret bound of algorithm 1 in terms of T.

The unknown parameters of LQ games need PG to find the Nash solution. In general, however, PG may not converge to the Nash. To find a category of LQ games where PG has a convergence guarantee, we recognized the class of potential LQ games in open-loop. We did the same for the closed-loop settings, but under some assumptions on the dimensions of the actions and states. By performing experiments, the results are verified.

The next step for this work is to do the same for infinite horizon stochastic LQ games to identify more potential LQ games. Besides, in this work, we considered the conventional cost function of $x_t^T Q_i x_t + u_{i,t}^T R_i u_{i,t}$. However, they could be extended with new cross terms of $x_t^T S u_t$ and similar terms, as in [7]. As such, if the cost function is more general, other classes of potential games might exist that we don't know about.

Bibliography

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

[2] Dimitri P Bertsekas et al. Dynamic programming and optimal control 3rd edition, volume ii. *Belmont, MA: Athena Scientific*, 2011.

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

[4] Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International Conference on Machine Learning*, pages 1467–1476. PMLR, 2018.

[5] Simon Fischer, Harald Räcke, and Berthold Vöcking. Fast convergence to wardrop equilibria by adaptive sampling methods. *SIAM Journal on Computing*, 39(8):3700–3735, 2010.

[6] Yoav Freund and Robert E Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[7] Ben Hambly, Renyuan Xu, and Huining Yang. Policy gradient methods find the nash equilibrium in n-player general-sum linear-quadratic games. *arXiv preprint arXiv:2107.13090*, 2021.

[8] Elad Hazan. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*, 2019.

[9] João P Hespanha. *Noncooperative game theory: An introduction for engineers and computer scientists*. Princeton University Press, 2017.

[10] Joao P Hespanha. *Linear systems theory*. Princeton university press, 2018.

[11] Chonghai Hu, Weike Pan, and James Kwok. Accelerated gradient methods for stochastic optimization and online learning. *Advances in Neural Information Processing Systems*, 22:781–789, 2009.

[12] Nicolas S Lambert, Giorgio Martini, and Michael Ostrovsky. Quadratic games. Technical report, National Bureau of Economic Research, 2018.

[13] Eric Mazumdar, Lillian J Ratliff, Michael I Jordan, and S Shankar Sastry. Policy-gradient algorithms have no guarantees of convergence in linear quadratic games. *arXiv preprint arXiv:1907.03712*, 2019.

[14] Eric Mazumdar, Lillian J Ratliff, and S Shankar Sastry. On gradient-based learning in continuous games. *SIAM Journal on Mathematics of Data Science*, 2(1):103–131, 2020.

[15] Panayotis Mertikopoulos, Christos Papadimitriou, and Georgios Piliouras. Cycles in adversarial regularized learning. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2703–2717. SIAM, 2018.

[16] Panayotis Mertikopoulos and Mathias Staudigl. Convergence to nash equilibrium in continuous games with noisy first-order feedback. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5609–5614. IEEE, 2017.

[17] David H Mguni, Yutong Wu, Yali Du, Yaodong Yang, Ziyi Wang, Minne Li, Ying Wen, Joel Jennings, and Jun Wang. Learning in nonzero-sum stochastic games with potentials. In *International Conference on Machine Learning*, pages 7688–7699. PMLR, 2021.

[18] Faisal A Mohamed and Heikki N Koivo. Multiobjective optimization using modified game theory for online management of microgrid. *European Transactions on Electrical Power*, 21(1):839–854, 2011.

[19] Roger B Myerson. *Game theory: analysis of conflict*. Harvard university press, 1997.

[20] Yu E Nesterov. A method for solving the convex programming problem with convergence rate $o(\frac{1}{k^2})$. In *Dokl. Akad. Nauk SSSR,*, volume 269, pages 543–547, 1983.

[21] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[22] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[23] Masoud Roudneshin, Jalal Arabneydi, and Amir G Aghdam. Reinforcement learning in nonzero-sum linear quadratic deep structured games: Global convergence of policy optimization. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 512–517. IEEE, 2020.

[24] Yuanyuan Shi and Baosen Zhang. No-regret learning in cournot games. *arXiv preprint arXiv:1906.06612*, 2019.

[25] Sharan Vaswani, Francis Bach, and Mark Schmidt. Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1195–1204. PMLR, 2019.

[26] 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.

[27] Kaiqing Zhang, Zhuoran Yang, and Tamer Basar. Policy optimization provably converges to nash equilibria in zero-sum linear quadratic games. *Advances in Neural Information Processing Systems*, 32, 2019.

[28] Martin Zinkevich. *Theoretical guarantees for algorithms in multi-agent settings*. Carnegie Mellon University, 2004.

Appendix A

Appendix

Theorem:2.2.1 If the following conditions are hold:

$$\gamma_k = \frac{1}{\rho}.\left[1 + \frac{\beta_k(1-\alpha_k)}{\alpha_k}\right], \tag{A.1}$$

$$\alpha_k = \frac{\gamma_k\beta_k b_{k+1}^2\eta}{\gamma_k\beta_k b_{k+1}^2\eta + a_k^2}, \tag{A.2}$$

$$\beta_k \geq 1 - \gamma_k\mu\eta, \tag{A.3}$$

$$a_{k+1} = \gamma_k\sqrt{\eta\rho}b_{k+1}, \tag{A.4}$$

$$b_{k+1} \leq \frac{b_k}{\sqrt{\beta_k}}, \tag{A.5}$$

$$b_{k+1}^2\gamma_k^2\eta\rho = a_{k+1}^2, \tag{A.6}$$

$$b_{k+1}^2\gamma_k\eta - a_{k+1}^2 + a_k^2 = 0, \tag{A.7}$$

$$\eta \leq \frac{1}{\rho L}. \tag{A.8}$$

the regret bound for algorithm 1 is:

$$Reg(T) \leq \sum_{j=1}^{T}\left[\frac{1}{2b_j^2\gamma_{j-1}^2\eta\rho}\left(2a_0^2C_1 + b_0^2C_2 + \sum_{k=1}^{j-1}\left[2b_{k+1}^2\gamma_{k+1}^2\eta^2\sigma_k^2 + 4b_{k+1}^2\gamma_{k+1}^2\eta\rho\epsilon_k\right]\right)\right]. \tag{A.9}$$

where $C_1 = f_0(x_0) - f_0(x^*)$ and $C_2 = \|x_0 - x^*\|^2$ and x^* is the best action played on the hindsight.

Proof. This proof is fully inspired from [25]. Essentially all the main steps are same with [25] except in eqs. (A.13) and (A.16), where we upgraded the algorithm to online optimization.

Let w_k, ζ_k and ν_k be updated as below:

$$w_{k+1} = \zeta_k - \eta \nabla f_k(\zeta_k, z_k), \tag{A.10}$$
$$\zeta_k = \alpha_k \nu_k + (1-\alpha_k) w_k, \tag{A.11}$$
$$\nu_{k+1} = \beta_k \nu_k + (1-\beta_k)\zeta_k - \gamma_k \eta \nabla f_k(\zeta_k, z_k). \tag{A.12}$$

And let $r_{k+1} = \|\nu_{k+1} - w^*\|$, so we have:

$$r_{k+1}^2 = \|\beta_k \nu_k + (1-\beta_k)\zeta_k - w^* - \gamma_k \eta \nabla f_k(\zeta_k, z_k)\|^2,$$

$$\begin{aligned} r_{k+1}^2 =& \|\beta_k \nu_k + (1-\beta_k)\zeta_k - w^*\|^2 \\ &+ \gamma_k^2 \eta^2 \|f_k(\zeta_k, z_k)\|^2 + 2\gamma_k \eta \langle w^* - \beta_k \nu_k - (1-\beta_k)\zeta_k, \nabla f_k(\zeta_k, z_k)\rangle. \end{aligned}$$

Taking expectation w.r.t z_k,

$$\begin{aligned} \mathbb{E}[r_{k+1}^2] &= \mathbb{E}[\|\beta_k \nu_k + (1-\beta_k)\zeta_k - w^*\|^2] + \gamma_k^2 \eta^2 \, \mathbb{E}[\|f_k(\zeta_k, z_k)\|^2] \\ &+ 2\gamma_k \eta [\mathbb{E}\langle w^* - \beta_k \nu_k - (1-\beta_k)\zeta_k, \nabla f_k(\zeta_k, z_k)\rangle] \\ &\leq \|\beta_k \nu_k + (1-\beta_k)\zeta_k - w^*\|^2 + \gamma_k^2 \eta^2 \rho \|f_k(\zeta_k)\|^2 \\ &+ 2\gamma_k \eta \langle w^* - \beta_k \nu_k - (1-\beta_k)\zeta_k, \nabla f_k(\zeta_k)\rangle + \gamma_k^2 \eta^2 \sigma_k^2 \\ &= \|\beta_k(\nu_k - w^*) + (1-\beta_k)(\zeta_k - w^*)\|^2 + \gamma_k^2 \eta^2 \rho \|f_k(\zeta_k)\|^2 \\ &+ 2\gamma_k \eta \langle w^* - \beta_k \nu_k - (1-\beta_k)\zeta_k, \nabla f_k(\zeta_k)\rangle + \gamma_k^2 \eta^2 \sigma_k^2 \\ &\leq \beta_k \|\nu_k - w^*\|^2 + (1-\beta_k)\|\zeta_k - w^*\|^2 \\ &+ \gamma_k^2 \eta^2 \rho \|f_k(\zeta_k)\|^2 + 2\gamma_k \eta \langle w^* - \beta_k \nu_k - (1-\beta_k)\zeta_k, \nabla f_k(\zeta_k)\rangle \\ &+ \gamma_k^2 \eta^2 \sigma_k^2 \text{ (By convexity of norm)} \\ &= \beta_k r_k^2 + (1-\beta_k)\|\zeta_k - w^*\|^2 + \gamma_k^2 \eta^2 \rho \|f_k(\zeta_k)\|^2 \\ &+ 2\gamma_k \eta \langle w^* - \beta_k \nu_k - (1-\beta_k)\zeta_k, \nabla f_k(\zeta_k)\rangle + \gamma_k^2 \eta^2 \sigma_k^2 \\ &= \beta_k r_k^2 + (1-\beta_k)\|\zeta_k - w^*\|^2 + \gamma_k^2 \eta^2 \rho \|f_k(\zeta_k)\|^2 \\ &+ 2\gamma_k \eta [\langle \frac{\beta_k(1-\alpha_k)}{\alpha_k}(w_k - \zeta_k) + w^* - \zeta_k, \nabla f_k(\zeta_k)\rangle] + \gamma_k^2 \eta^2 \sigma_k^2 \\ &= \beta_k r_k^2 + (1-\beta_k)\|\zeta_k - w^*\|^2 + \gamma_k^2 \eta^2 \rho \|f_k(\zeta_k)\|^2 \\ &+ 2\gamma_k \eta \left[\frac{\beta_k(1-\alpha_k)}{\alpha_k} \langle (w_k - \zeta_k), \nabla f_k(\zeta_k)\rangle + \langle w^* - \zeta_k, \nabla f_k(\zeta_k)\rangle \right] + \gamma_k^2 \eta^2 \sigma_k^2 \\ &\leq \beta_k r_k^2 + (1-\beta_k)\|\zeta_k - w^*\|^2 + \gamma_k^2 \eta^2 \rho \|f_k(\zeta_k)\|^2 \\ &+ 2\gamma_k \eta \left[\frac{\beta_k(1-\alpha_k)}{\alpha_k} (f_k(w_k) - f_k(\zeta_k)) + \langle w^* - \zeta_k, \nabla f_k(\zeta_k)\rangle \right] \\ &+ \gamma_k^2 \eta^2 \sigma_k^2. \text{ (By convexity)} \end{aligned}$$

By strong convexity:

$$\begin{aligned}\mathbb{E}[r_{k+1}^2] \leq \beta_k r_K^2 + (1-\beta_k)\|\zeta_k - w^*\|^2 + \gamma_k\eta^2\rho\|\nabla f_k(\zeta_k)\|^2 \\ + 2\gamma_k\eta\left[\frac{\beta_k(1-\alpha_k)}{\alpha_k}(f_k(w_k) - f_k(\zeta_k)) + f_k^* - f_k(\zeta_k) - \frac{\mu}{2}\|\zeta_k - w^*\|^2\right] \\ + \gamma_k^2\eta^2\sigma_k^2.\end{aligned}$$

Also we have:

$$\begin{aligned}f_k(w_{k+1}) - f_k(\zeta_k) \leq \langle\nabla f_k(\zeta_k), w_{k+1} - \zeta_k\rangle + \frac{L}{2}\|w_{k+1} - \zeta_k\|^2 \\ - \eta\langle\nabla f_k(\zeta_k), \nabla f_k(\zeta_k, z_k)\rangle + \frac{L\eta^2}{2}\|\nabla f_k(\zeta_k, z_k)\|^2.\end{aligned}$$

Taking expectation of both sides w.r.t z_k (if $\eta \leq \frac{1}{\rho L}$):

$$\mathbb{E}[f_{k+1}(w_{k+1}) - f_k(\zeta_k)] \leq -\eta\|\nabla f_k(\zeta_k)\|^2 + \frac{L\rho\eta^2}{2}\|\nabla f_k(\zeta_k)\|^2 + \frac{L\eta^2\sigma_k^2}{2} + \epsilon_k \tag{A.13}$$

$$\|\nabla f_k(\zeta_k)\|^2 \leq (\frac{2}{\eta})\,\mathbb{E}[f_k(\zeta_k) - f_{k+1}(w_{k+1})] + L\eta\sigma_k^2 + \frac{2}{\eta}\epsilon_k.$$

Combining it with the previous inequality we conclude that:

$$\begin{aligned}\mathbb{E}[r_{k+1}^2] &\leq \beta_k r_k^2 + (1-\beta_k)\|\zeta_k - w^*\|^2 + 2\gamma_k^2\rho\eta\,\mathbb{E}[f_k(\zeta_k) - f_k(w_{k+1})] \\ &+ 2\gamma_k\eta\left[\frac{\beta_k(1-\alpha_k)}{\alpha_k}(f_k(w_k) - f_k(\zeta_k)) + f_k^* - f_k(\zeta_k) - \frac{\mu}{2}\|\zeta_k - w^*\|^2\right] \\ &+ \gamma_k^2\eta^2\sigma_k^2 + L\gamma_k^2\eta^3\rho\sigma_k^2 + 2\gamma_k^2\rho\eta\epsilon_k \\ &\leq \beta_k r_k^2 + \|\zeta_k - w^*\|^2[(1-\beta_k) - \gamma\mu\eta] + f_k(\zeta_k)\left[2\gamma_k^2\eta\rho - 2\gamma_k\eta.\frac{\beta_k(1-\alpha_k)}{\alpha_k} - 2\gamma_k\eta\right] \\ &- 2\gamma_k^2\eta\rho\,\mathbb{E}\,f_{k+1}(w_{k+1}) + 2\gamma_k\eta f^* + \left[2\gamma_k\eta.\frac{\beta_k(1-\alpha_k)}{\alpha_k}\right]f_k(w_k) + 2\gamma_k^2\eta^2\sigma_k^2 + 2\gamma_k^2\rho\eta\epsilon_k.\end{aligned}$$

Now multiplying both sides by b_{k+1}^2 and because of A.1 and A.3,

$$\begin{aligned}
b_{k+1}^2\,\mathbb{E}[r_{k+1}^2] &\leq b_{k+1}^2\beta_k r_k^2 + \|\zeta_k - w^*\|^2[(1-\beta_k) - \gamma\mu\eta] \\
&+ b_{k+1}^2 f_k(\zeta_k)\left[2\gamma_k^2\eta\rho - 2\gamma_k\eta.\frac{\beta_k(1-\alpha_k)}{\alpha_k} - 2\gamma_k\eta\right] \\
&- 2b_{k+1}^2\gamma_k^2\eta\rho\,\mathbb{E}\,f_{k+1}(w_{k+1}) \\
&+ 2b_{k+1}^2\gamma_k\eta f^* + b_{k+1}^2\left[2\gamma_k\eta.\frac{\beta_k(1-\alpha_k)}{\alpha_k}\right] f_k(w_k) + 2b_{k+1}^2\gamma_k^2\eta^2\sigma_k^2 \\
&+ 2b_{k+1}^2\gamma_k^2\rho\eta\epsilon_k \\
&= b_{k+1}^2\beta_k r_k^2 - 2b_{k+1}^2\gamma_k^2\eta\rho\,\mathbb{E}(f_{k+1}(w_{k+1})) + 2b_{k+1}^2\gamma_k\eta f^* \\
&+ \left[2b_{k+1}^2\gamma_k\eta.\frac{\beta_k(1-\alpha_k)}{\alpha_k}\right] f_{k+1}(w_k) + 2b_{k+1}^2\gamma_k^2\eta^2\sigma_k^2 + 2b_{k+1}^2\gamma_k^2\rho\eta\epsilon_k.
\end{aligned}$$

Since $b_{k+1}^2\beta_k \leq b_k^2$, $b_{k+1}^2\gamma_k^2\eta\rho = a_{k+1}^2$, $\frac{\gamma_k\eta\beta_k(1-\alpha_k)}{\alpha_k} = \frac{a_k^2}{b_{k+1}^2}$

$$b_{k+1}^2\,\mathbb{E}[r_{k+1}^2] \leq b_k^2 r_k^2 - 2a_{k+1}^2\,\mathbb{E}\,f_{k+1}(w_{k+1}) + 2b_{k+1}^2\gamma_k\eta f_k^* + 2a_k^2 f_k(w_k) \tag{A.14}$$

$$\begin{aligned}
&+ \frac{2a_{k+1}^2\sigma_k^2\eta}{\rho} + 2a_{k+1}^2\epsilon_k \\
&\leq b_k^2 r_k^2 - 2a_{k+1}^2[\mathbb{E}\,f_{k+1}(w_{k+1}) - f_{k+1}^*] + 2a_k^2[f_k(w_k) - f_k^*]
\end{aligned} \tag{A.15}$$

$$\begin{aligned}
&+ \frac{2a_{k+1}^2\sigma_k^2\eta}{\rho} + 4a_{k+1}^2\epsilon_k \\
&+ 2[b_{k+1}^2\gamma_k\eta - a_{k+1}^2 + a_k^2]f_k^*.
\end{aligned} \tag{A.16}$$

Since $[b_{k+1}^2\gamma_k\eta - a_{k+1}^2 + a_k^2] = 0$,

$$\begin{aligned}
b_{k+1}^2\,\mathbb{E}[r_{k+1}^2] &\leq b_k^2 r_k^2 - 2a_{k+1}^2[\mathbb{E}\,f_{k+1}(w_{k+1}) - f_{k+1}^*] + 2a_k^2[f_k(w_k) - f_k^*] \\
&+ \frac{2a_{k+1}^2\sigma_k^2\eta}{\rho} + 4a_{k+1}^2\epsilon_k
\end{aligned}$$

Denoting $\mathbb{E}\,f_k(w_K) - f_k^*$ as ϕ_k,

$$2a_{k+1}^2\phi_{k+1} + b_{k+1}^2\,\mathbb{E}[r_{k+1}^2] \leq 2a_k^2\phi_k + b_k^2\,\mathbb{E}[r_k^2] + \frac{2a_{k+1}^2\sigma_k^2\eta}{\rho} + 4a_{k+1}^2\epsilon_k.$$

By recursion,

$$\begin{aligned} 2a_{T+1}^2\phi_{T+1} + b_{T+1}^2\,\mathbb{E}[r_{T+1}^2] \leq & 2a_0^2\phi_0 + b_0^2\,\mathbb{E}[r_0^2] \\ & + \sum_{k=1}^{T}\left[\frac{2a_{k+1}^2\sigma_k^2\eta}{\rho} + 4a_{k+1}^2\epsilon_k\right], \qquad \text{(A.17)} \\ \phi_{T+1} \leq & \frac{1}{2a_{T+1}^2}\Big(2a_0^2\phi_0 + b_0^2\,\mathbb{E}[r_0^2] - b_{T+1}^2\,\mathbb{E}[r_{T+1}^2] \\ & + \sum_{k=1}^{T}\left[\frac{2a_{k+1}^2\sigma_k^2\eta}{\rho} + 4a_{k+1}^2\epsilon_k\right]\Big), \qquad \text{(A.18)} \end{aligned}$$

$$\begin{aligned} Reg(T) = \sum_{j=1}^{T}\phi_j \leq \sum_{j=1}^{T}\Bigg[\frac{1}{2a_j^2}\Bigg(& 2a_0^2\phi_0 + b_0^2\,\mathbb{E}[r_0^2] - b_j^2\,\mathbb{E}[r_j^2] \\ & + \sum_{k=1}^{j-1}\left[\frac{2a_{k+1}^2\sigma_k^2\eta}{\rho} + 4a_{k+1}^2\epsilon_k\right]\Bigg)\Bigg], \qquad \text{(A.19)} \end{aligned}$$

$$\begin{aligned} Reg(T) = \sum_{j=1}^{T}\phi_j \leq \sum_{j=1}^{T}\Bigg[\frac{1}{2b_j^2\gamma_{j-1}^2\eta\rho}\Bigg(& 2a_0^2C_1 + b_0^2C_2 \\ & + \sum_{k=1}^{j-1}\left[2b_{k+1}^2\gamma_{k+1}^2\eta^2\sigma_k^2 + 4b_{k+1}^2\gamma_{k+1}^2\eta\rho\epsilon_k\right]\Bigg)\Bigg]. \qquad \text{(A.20)} \end{aligned}$$

□