Title
Quasi-Newton Algorithms for Non-smooth Online Strongly Convex Optimization

Author
Godwin, Mark Franklin

Publication Date
2011

Peer reviewed|Thesis/dissertation

Quasi-Newton Algorithms for Non-smooth Online Strongly Convex Optimization

by

Mark Franklin Godwin

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Francesco Borrelli, Chair
Professor J. Karl Hedrick
Professor Raja Sengupta

Fall 2011

Quasi-Newton Algorithms for Non-smooth Online Strongly Convex Optimization

Abstract

Quasi-Newton Algorithms for Non-smooth Online Strongly Convex Optimization

by

Mark Franklin Godwin

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Francesco Borrelli, Chair

The growing prevalence of networked systems with local sensing and computational capability will result in an increasing array of online and large scale optimization problems. We adopt the framework of online convex optimization that allows for the optimization of an arbitrary sequence of convex functions subject to a single convex constraint set. We identify quasi-Newton algorithms as a promising class of algorithms to solve online strongly convex optimization problems. We first examine the relationships between several known algorithms for convex functions that satisfy a α-exp-concave assumption. Next we present two new quasi-Newton algorithms for non-smooth strongly convex functions and show how these algorithms can be parallelized given a summed objective function. Our new algorithms require fewer parameters and have provably tighter bounds then similar known algorithms. Also, our bounds are not a function of the number of iterations, but instead a function of the sequence of strongly convex parameters that correspond to a sequence of strongly convex functions. We then extend these algorithms to use a block diagonal hessian approximation. An algorithm with a fully diagonal hessian approximation results in a large scale quasi-Newton algorithm for online convex optimization. Our results can be translated to convergence bounds and optimization algorithms that solve non-smooth strongly convex functions. We perform numerical experiments on test functions of different dimension and compare our algorithms to similar known algorithms. These experiments show our algorithms perform well in the majority of test cases we consider. We apply our algorithms to online portfolio optimization with a ℓ_2-norm regularized constant-rebalanced portfolio model and compare our algorithm to known methods. In addition, a heuristic algorithm for online vehicle routing is presented. Although online vehicle routing does not fit within the framework of online convex optimization, the work provided significant insight into online optimization and provides a future source of ideas and motivation. Finally, we provide conclusions and discuss future research directions.

To my wonderful family and friends.

For all the love and support over the years.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to acknowledge the support of my advisors Professor Francesco Borrelli, Professor J. Karl Hedrick and Professor Raja Sengupta. I would also like to acknowledge the financial support of the Office of Naval Research and the Air Force Office of Scientific Research. Finally, I would like to acknowledge the many past and current members of the Center for Collaborative Control of Unmanned Vehicles, the Vehicle Dynamics Laboratory and the Model Predictive Control Lab for many fruitful discussions, the hard work at Camp Roberts and for making UC Berkeley a wonderful place to work.

Chapter 1

Introduction

This dissertation is motivated by the online optimization of large scale systems. This chapter introduces our research topic and discusses our motivation. We introduce online convex optimization, discuss related work, review our contributions and outline the chapters to follow. We assume an understanding of the mathematical background found in Appendix A.

1.1 Motivation

The future will see an increasing number of large systems that consist of many agents with local sensing and computing capability connected through a communication network. Systems of this class include computer systems connected through communication networks, but also physical systems with embedded processors connected through communications networks. There exists a need for algorithms that optimize computer controlled power systems, transportation systems and networked unmanned vehicle systems [1]. Many of these applications can be posed as a structured optimization problem with a summed objective function.

This dissertation focuses on structured optimization problems with a summed objective function that can be solved using parallelized algorithms [2]. We consider three forms of summed objective functions with varying degrees of coupling. As the objective function becomes more decoupled, a "more parallel" algorithmic solution is possible. A optimization problem can be trivially parallelized if it is entirely decoupled, as in the following problem

$$\min_{\mathbf{x}^r \in \mathcal{X}^r, 1 \leq r \leq R} \sum_{r=1}^{R} f^r(\mathbf{x}^r) \tag{1.1}$$

where R is the number of computational platforms. This thesis focuses on optimization problems with coupling and a summed objective function. Consider the following optimization

problem that has coupling through its constraints

$$\min_{[\mathbf{x}^{1\top},\ldots,\mathbf{x}^{R\top}]^\top \in \mathcal{X}} \sum_{r=1}^{R} f^r(\mathbf{x}^r) \tag{1.2}$$

where $\mathcal{X}$ represents a set coupling the decision variables $\{\mathbf{x}^1, \ldots, \mathbf{x}^R\}$. Online vehicle routing has this problem structure and is discussed further in Chapter 7. This thesis also considers optimization problems with coupling both in the objective function and in the constraints. This problem is represented as

$$\min_{\mathbf{x}\in\mathcal{X}} \sum_{r=1}^{R} f^r(\mathbf{x}) \tag{1.3}$$

Although this problem is more coupled, the problem structure can still be exploited by a gradient or subgradient based algorithm. Given a well known property of subgradients, presented in Appendix A, we have

$$\partial f(\mathbf{x}) = \sum_{r=1}^{R} \partial f^r(\mathbf{x}) \tag{1.4}$$

where $\partial f(\mathbf{x})$ denotes the subdifferential of f. The subdifferential is the set of subgradients associated with a given point $\mathbf{x}$. A subgradient is a generalized gradient for non-smooth functions. Given (1.4) the subgradient calculation can be parallelized. If the calculation of a subgradient is computationally difficult, or if R is large, then the parallelization of these calculations can result in increased performance. Therefore, to exploit the structure of problems with a summed objective functions we will consider subgradient based methods.

In this dissertation we assume our application requires "good" feasible solutions at regular intervals. In the control theory community this is often referred to as real-time control. Real-time control of an aircraft's trajectory may require updates every few milliseconds while real-time control of a transportation network may require updates every few seconds or minutes. We consider applications where the required update rate is on the order of seconds or minutes. Therefore although subgradient methods are not known to be particularly fast they are "fast enough" for the applications we consider. Furthermore, the computational requirements of subgradient methods per iteration typically scale well with problem dimension. Therefore for problems with many variables subgradient methods may be the only feasible algorithmic solution. The computational complexity of subgradient based algorithms is discussed further in Section 1.2. As previously discussed subgradient based algorithms can use parallelism to solve structured optimization problems, have computational requirements that scale well with problem size and are fast enough for applications that require an update on the order of seconds or minutes. For these reasons subgradient based algorithms are a good class of algorithm for structured large scale convex optimization problems.

1.2 Online Convex Optimization

Online convex optimization minimizes the regret of an online algorithm [3]. Regret is the total cost of an online algorithm's decisions made in sequence, given only the past information, minus the best decision made in hindsight. This formalism allows for the optimization of an arbitrary sequence of convex functions subject to a single convex constraint set. Subgradient based algorithms can solve an online convex optimization problem. An online convex optimization problem is formally introduced in Chapter 2. An algorithm that can solve an online convex optimization problem can be converted into an algorithm that minimizes a single convex function as discussed in Section 2.1.1.

An algorithm for online convex optimization is Online Gradient Descent. It was adapted from offline optimization and achieves $\mathcal{O}(\sqrt{T})$ regret where T is the number of iterations [3]. This implies that the worst case error between the online algorithm's decisions made in sequence, given only the past information, minus the best decision made in hindsight grows with $\mathcal{O}(\sqrt{T})$.

The novel idea of using a strongly convex assumption to achieve $\mathcal{O}(\log T)$ regret with Online Gradient Descent was first proposed by Hazan et al. [4]. Intuitively, strongly convex functions have a minimum curvature at every point. Strongly convex functions have been identified as an important function class that enables efficient algorithm development with logarithmic regret bounds. Many practical optimization problems are strongly convex or can be made strongly convex using strongly convex regularizers [5, 6, 7].

Given that strongly convex functions have a minimum curvature, we consider the construction of an approximate hessian from subgradient information to improve algorithm performance. If a multidimensional function is twice differentiable then the matrix of second partial derivatives is referred to as the hessian. This type of algorithm is referred to as a quasi-Newton algorithm. In summary, this dissertation focuses on quasi-Newton algorithms for non-smooth online strongly convex optimization problems.

1.2.1 Related Work

The following provides a brief review of adaptive algorithms for non-smooth convex and strongly convex functions. Adaptive algorithms use information obtained through execution to adapt algorithmic parameters and improve performance. Adaptive algorithms include quasi-Newton algorithms are also referred to as variable metric algorithms [8].

Adaptive gradient methods have drawn from research on the Follow the Leader method starting in 1950s game theory literature. In this literature the Follow the Leader method is also known as the *sequential compound decision problem*, an approach first put forward by Robbins [9]. Hannan [10] proposed a randomized variant of the Follow the Leader algorithm called the Perturbed Follow the Leader algorithm for linear functions subject to a simplex that attains $\mathcal{O}(\sqrt{T})$ regret.

Merhav and Feder [11] extended the Follow the Leader approach to strongly convex

functions over a simplex and show that the approach can attain $\mathcal{O}(\log T)$ regret. Similar results were obtained by Cesa-Bianchi and Lugosi [12] and Gaivoronski and Stella [13]. This work led to the work by Hazan, Agarwal and Kale [14] on the Follow the Approximate Leader algorithm for α-exp-concave convex functions subject to a convex constraint set.

In the 1970s and 80s an array of adaptive algorithms were developed including the Broyden-Fletcher-Goldfarb-Shanno (BFGS) family of quasi-Newton algorithms [15], Shor's R-Algorithm for minimization of a non-smooth nonlinear functions [16] and the ellipsoidal method developed by Yudin and Nemirovski [17].

Although not designed to optimize non-smooth functions, the BFGS algorithm has been shown to perform well on non-smooth problems when the correct line search is used [18]. The limited memory BFGS method has also been shown to perform well on non-smooth problems when the correct line search is used [19]. In addition, recent work has extended the BFGS family of quasi-Newton algorithms to non-smooth optimization [20]. Also, the work of Bordes et al. [21] presents a quasi-Newton stochastic gradient descent procedure but assumes a smooth objective function.

The Levenberg-Marquardt algorithm is known for solving non-linear least squares problems and can be interpreted as an adaptive algorithm that combines a Gauss-Newton algorithm with gradient descent [22]. This algorithm approximates function curvature with the outer product of gradients, a technique used by other adaptive algorithms [23, 14].

In recent years significant progress has been made on both non-adaptive and adaptive subgradient methods. An extension of Nesterov's Dual Averaging method [24] for non-smooth convex functions was developed that exploits the regularization structure in an online setting [25]. Nesterov has also proposed a smoothing technique for non-smooth functions with an explicit max-structure that improves the worst case converage rate [26]. The best worst case bound for optimizing non-smooth strongly convex functions with bounded gradients is known to be $\mathcal{O}(1/T)$ [27]. Several algorithms have been developed that achieve this best worst case bound [28, 29]. It has also been shown that online convex optimization is strictly more difficult and the best known regret bound for non-smooth strongly convex cost functions with bounded gradients is $\mathcal{O}(\log T)$ [29].

An adaptive algorithm for non-smooth convex functions that considers adaptive parameter selection as an online (meta) learning problem has been developed by Duchi et al. [23]. This adaptive research has drawn from work on forward-backward splitting [30] and composite mirror-descent generalizations [31] that includes special cases of projected gradient descent [3] and mirror descent [27, 32]. These algorithms use a strongly convex proxy function multiplied by a time-dependent scalar while the new adaptive algorithm of Duchi et al. [23] adapts the strongly convex proxy function. Algorithms that use a similar approach have been developed in parallel [33].

Other algorithms for online convex optimization that exploit strong convexity have been developed. An adaptive algorithm that interpolates between the work of Zinkevich [3] and Hazan et al. [4] to obtain a regret bound between $\sqrt{T}$ and $\log(T)$ has been developed by Bartlett et al. [34]. An online primal-dual algorithm for strongly convex functions was

proposed and analyzed by Kakade et al. [5]. Building on the work of Hazan et al. [4] Online Gradient Descent was extended to a more general notion of strongly convex functions [35, 36]. Online convex optimization algorithms are typically bounded by a function of an algorithm's iterations. However, a regret bound has been derived that is a function of the observed variations of α-exp-concave functions and this analysis has been applied to online portfolio management [37, 38].

1.3 Contributions and Outline

The following chapters are outlined below along with our contributions.

- Chapter 2: We formally introduce online convex optimization, regret and state our assumptions. We show how a regret bound can be translated into a convergence bound. We present known algorithms for strongly convex and α-exp-concave convex functions and the relation between these algorithms is discussed. We extend the Online Newton Step (ONS) and Follow the Approximate Leader (FTAL) algorithms of Hazan et al. [14] to non-smooth α-exp-concave convex functions. We also prove that ONS and FTAL are the same algorithm under appropriate assumptions.

- Chapter 3: We present new ONS and FTAL algorithms for strongly convex functions and present two choices of initialization parameters. These new algorithms require fewer parameters and have a provably tighter bound than the original algorithms. Also, our bounds are a function of a strongly convex parameter sequence instead of the algorithm's iterations.

- Chapter 4: We extend ONS and FTAL for strongly convex functions to used block diagonal hessian approximations. This reduces the computational requirements such that these algorithms can be used on large scale problems. For a fully diagonal hessian approximation the algorithm's update step scales with $\mathcal{O}(n)$, where n is the problem dimension, and hence matches that of Online Gradient Descent.

- Chapter 5: We apply the new FTAL Algorithm for strongly convex functions to a typical online convex optimization application, online portfolio optimization. We propose a strongly convex ℓ_2-norm regularized constant-rebalanced portfolio model and compare our algorithm and model with a known algorithm and model for online portfolio optimization.

- Chapter 6: In an offline optimization framework we perform numerical experiments on a set of non-smooth strongly convex test functions and compare the performance of our algorithms to a similar class of known algorithms. Numerical experiments show that our new algorithms, in the majority of cases considered, outperform known algorithms.

- Chapter 7: The online vehicle routing problem is defined as a sequence of NP-hard optimization problems. We present a heuristic algorithm for online vehicle routing. Although the problem and algorithm does not strictly fit in an online convex optimization framework, the application has motivated our research and provides further insight to advance the state of the art in online optimization.

- Chapter 8: We conclude and discuss two potential future research directions. First, we discuss several possible approaches to decentralize these algorithms over a communication network with delays. Second, we discuss the potential application of online convex optimization to classical control theory.

Chapter 2

Online Algorithms

This chapter presents the mathematical framework and notation used throughout this dissertation. We assume an understanding of the mathematical background found in Appendix A. This chapter formally presents online convex optimization and known algorithms for strongly convex and α-exp-concave convex functions. The presentation of known algorithms is used to give a better understanding of how subgradient algorithms work and how they are analyzed. We first present the Online Gradient Descent (OGD) algorithm for non-smooth strongly convex functions that achieves logarithmic regret $\mathcal{O}(\log T)$ [4]. Next we present two quasi-Newton algorithms for α-exp-concave convex functions known as the Online Newton Step (ONS) and Follow the Approximate Leader (FTAL) algorithm. We extend the analysis of ONS and FTAL to non-smooth α-exp-concave convex functions. We show how FTAL is related to ONS. Finally, the efficient implementation of these algorithms is discussed [14].

2.1 Online Convex Optimization

We now formally introduce online convex optimization [3]. In contrast to minimizing a single convex function, the goal of online convex optimization is to minimize regret. Regret is the total cost of an online algorithm's decisions made in sequence, given only past information, minus the best decision made in hindsight. An online convex optimization problem is formally defined as follows.

Definition 1 (Online Convex Optimization Problem)**.** *An online convex optimization problem consists of a feasible set $\mathcal{X} \subseteq \mathbb{R}^n$ and an infinite sequence $\{f_1, f_2, \dots\}$ of convex functions. An algorithm $\mathcal{A}$ at each iteration t must choose a $\mathbf{x}_t \in \mathcal{X}$, given only the vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_{t-1}\}$ and the information obtained from the functions $\{f_1, f_2, \dots, f_{t-1}\}$, such that*

the regret is minimized. The regret of algorithm $\mathcal{A}$ up until time T is

$$\sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T} f_t(\mathbf{x}) = R_T(\mathcal{A}) \tag{2.1}$$

For example, consider the online convex optimization problem $(\mathcal{X}, \{f_1, f_2, \dots\})$ where we are given the subgradients $\{\mathbf{y}_1, \mathbf{y}_2, \dots\}$ drawn from the sequence of functions $\{f_1, f_2, \dots\}$. For some large, possibly infinite, T we would like to solve

$$x^* = \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T} f_t(\mathbf{x}) \tag{2.2}$$

However, we require a prediction of $\mathbf{x}^*$ at each iteration $t \in \{1, \dots, T\}$ using only the knowledge of past feasible points $\{\mathbf{x}_1, \dots, \mathbf{x}_{t-1}\}$ and past subgradients $\{\mathbf{y}_1, \dots, \mathbf{y}_{t-1}\}$. That is, we desire an algorithm $\mathcal{A}$ such that $\mathbf{x}_t = \mathcal{A}(\{\mathbf{x}_1, \dots, \mathbf{x}_{t-1}\}, \{\mathbf{y}_1, \dots, \mathbf{y}_{t-1}\})$ where regret (2.1) is minimized. The sequence of functions are all a function of the same variable and therefore knowledge of the past can help predict the minimum argument (2.2). Consider the following example application of online convex optimization.

Example 1 (Sequence of Quadratic Functions)**.** Consider the online convex optimization problem defined by the set $\mathcal{X} = \mathbb{R}$ the sequence of functions $\{f_1, f_2, \dots\}$ where $f_t : \mathcal{X} \to \mathbb{R}$ and

$$f_t(x) = \begin{cases} \frac{1}{2}x^2 + x, & \text{if t odd} \\ \frac{1}{2}x^2 - x, & \text{if t even} \end{cases} \tag{2.3}$$

Each gradient is calculated as follows

$$\nabla f_t(x_t) = \begin{cases} x_t + 1, & \text{if t odd} \\ x_t - 1, & \text{if t even} \end{cases} \tag{2.4}$$

For some large T we would like to solve (2.2) where

$$x^* = \begin{cases} \frac{-1}{T}, & \text{if T odd} \\ 0, & \text{if T even} \end{cases} \tag{2.5}$$

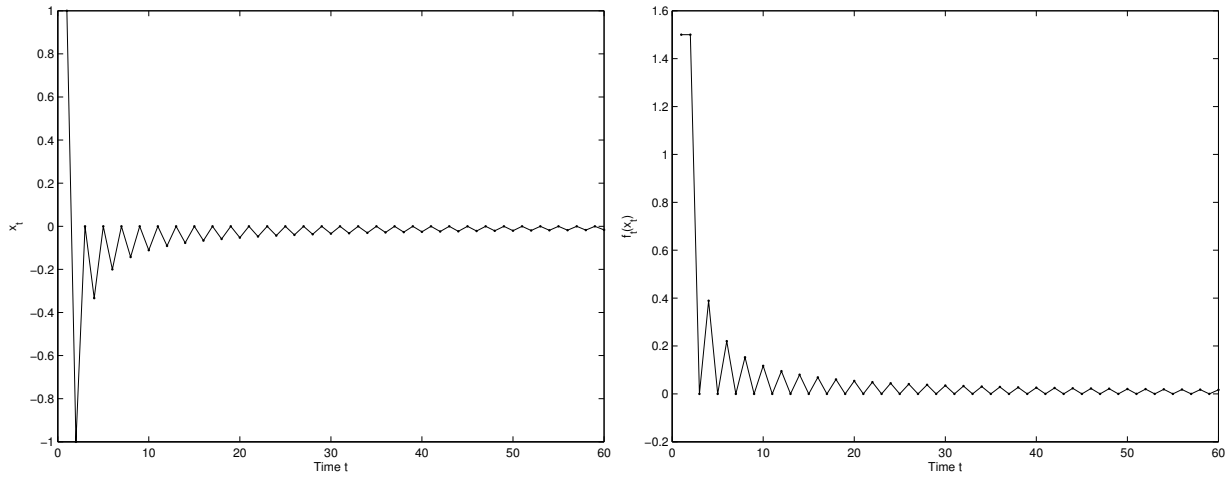
However, we need a prediction of the optimal x^* at each iteration t using only the vectors $\{x_1, \dots, x_{t-1}\}$ and the gradients $\{\nabla f_1(x_1), \dots, \nabla f_{t-1}(x_{t-1})\}$ of the functions $\{f_1, f_2, \dots, f_{t-1}\}$. That is, we need an algorithm such that $x_t = \mathcal{A}(\{x_1, \dots, x_{t-1}\}, \{\nabla f_1(x_1), \dots, \nabla f_{t-1}(x_{t-1})\})$. Therefore, we propose Algorithm 1 with the goal of minimizing regret. At $t = 1$ we guess $x_1 = 1$ is the minimum of (2.2).

Algorithm 1 Proposed algorithm

Require: $x_1 = 1$

1: **for** $t = 1$ to T **do**
2: $y_t = \nabla f_t(x_t)$
3: $x_{t+1} = x_t - \frac{1}{t} y_t$
4: **end for**

Starting from the initial guess, $x_1 = 1$, Figure 2.1 plots the iterations of Algorithm 1 for $T = 60$ iterations. Notice that the algorithm's iterations $\{x_1, x_2, \dots\}$ converge to the minimum argument (2.5) of problem (2.2).



(a) Prediction over time

(b) Cost of predictions over time

Figure 2.1: Results for one dimensional example

△

We assume our algorithms can calculate the subgradient of a function for any feasible argument. We also assume a bound on the subgradients, parameterized by L, and a bound on the convex set $\mathcal{X}$, parameterized by D.

Assumption 1. *Unless otherwise stated the following assumptions are made throughout the remainder of this work.*

(a) The set $\mathcal{X}$ is convex.

(b) The set $\mathcal{X}$ is bounded. There exists a $D \in \mathbb{R}$ such that

$$||\mathbf{x} - \mathbf{z}||_2 \leq D$$

for all $\mathbf{x}, \mathbf{z} \in \mathcal{X}$.

(c) The set $\mathcal{X}$ is closed. For all sequences $\{\mathbf{x}_1, \mathbf{x}_2, \dots\}$ where $\mathbf{x}_t \in \mathcal{X}$ for all t, if there exists a $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x} = \lim_{t\to\infty} \mathbf{x}_t$, then $\mathbf{x} \in \mathcal{X}$.

(d) The set $\mathcal{X}$ is nonempty.

(e) Let $f : \mathcal{X} \to \mathbb{R}$ be a convex function. For any $\mathbf{x} \in \mathcal{X}$ we can obtain a subgradient $\mathbf{y} \in \partial f(\mathbf{x})$ and the function value $f(\mathbf{x})$.

(f) Let $f : \mathcal{X} \to \mathbb{R}$ be a convex function. There exists a $L \in \mathbb{R}$ such that $||\mathbf{y}||_2 \leq L$ for any $\mathbf{x} \in \mathcal{X}$ and any subgradient $\mathbf{y} \in \partial f(\mathbf{x})$.

We now examine how a bound on regret relates to a convergence bound for a single convex function.

2.1.1 Convergence Bound for Non-smooth Convex Functions

Online convex optimization is a generalization of optimization for a single non-smooth convex function. Lemma 1 shows that if an online convex optimization algorithm achieves asymptotically sub-linear regret and it is applied to a sequence of identical convex functions then it is an optimization algorithm that obtains the minimum of the single convex function.

Lemma 1. *Let $f : \mathcal{X} \to \mathbb{R}$ be a convex function where $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$. Now consider the online convex optimization problem $(\mathcal{X}, \{f_1, f_2, \dots\})$ where $f_t = f$ for all $t \in \{1, \dots, T\}$. If algorithm $\mathcal{A}$ is applied to the online convex optimization problem $(\mathcal{X}, \{f_1, f_2, \dots\})$ and achieves asymptotically sub-linear regret such that $R_T(\mathcal{A}) = o(T)$ then*

$$f(\hat{\mathbf{x}}_T) - f(\mathbf{x}^*) \leq \frac{R_T(\mathcal{A})}{T} \tag{2.6}$$

where $\hat{\mathbf{x}}_T = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}_t$ and $\lim_{T\to\infty} \frac{R_T(\mathcal{A})}{T} = 0$

Proof (Lemma 1)**.** Given $f = f_t$ for all $t \in \{1, \dots, T\}$ we have

$$\sum_{t=1}^{T} [f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] = R_T(\mathcal{A}) \tag{2.7}$$

$$\sum_{t=1}^{T} f(\mathbf{x}_t) - T f(\mathbf{x}^*) = R_T(\mathcal{A}) \tag{2.8}$$

dividing through by T and using Jenson's inequality we have

$$f(\hat{\mathbf{x}}_T) - f(\mathbf{x}^*) \leq \frac{R_T(\mathcal{A})}{T} \tag{2.9}$$

where $\hat{\mathbf{x}}_T = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}_t$. By definition of $R_T(\mathcal{A}) = o(T)$

$$\lim_{T\to\infty} \frac{R_T(\mathcal{A})}{T} = 0 \tag{2.10}$$

□

2.2 Online Gradient Descent

We present the Online Gradient Descent algorithm for strongly convex functions as Algorithm 2. This analysis was introduced by Hazan, et al. [4] who used strong convexity to achieve logarithmic regret. The proof for Lemma 2 using Definition 7 can be found in Shalev-Shwartz et al. [36].

Lemma 2 (Strongly Convex)**.** *The function $f : \mathcal{X} \to \mathbb{R}$ is H-strongly convex with respect to a norm $||\cdot||$ if and only if for all $\mathbf{z}, \mathbf{x} \in \mathcal{X}$*

$$f(\mathbf{z}) \geq f(\mathbf{x}) + \mathbf{y}^\top(\mathbf{z} - \mathbf{x}) + \frac{H}{2}||\mathbf{z} - \mathbf{x}||^2 \tag{2.11}$$

for all $\mathbf{y} \in \partial f(\mathbf{x})$.

Algorithm 2 Online Gradient Descent

Require: $\mathbf{x}_1 \in \mathcal{X} \subseteq \mathbb{R}^n$ and $\{\eta_t\}_{t=1}^T$
1: **for** $t = 1$ to T **do**
2: $\quad \mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$
3: $\quad \mathbf{z}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{y}_t$
4: $\quad \mathbf{x}_{t+1} = \Pi_{\mathcal{X}}(\mathbf{z}_{t+1}) = \arg\min_{x \in \mathcal{X}} ||\mathbf{z}_{t+1} - \mathbf{x}||_2$
5: **end for**

We will now derive the logarithmic regret bound for Online Gradient Descent. This proof is instructive by showing how the strongly convex assumption can be used to reduce regret.

Theorem 1. *If $f_t : \mathcal{X} \to \mathbb{R}$ is a H-strongly convex function with respect to $||\cdot||_2$ for all $t \in \{1, \ldots, T\}$ where $H > 0$ and subject to $\mathcal{X} \subseteq \mathbb{R}^n$ then Algorithm $\mathcal{A}_2$, where $\eta_t = \frac{1}{Ht}$ for $t \in \{1, \ldots, T\}$, has the following regret bound*

$$R_T(\mathcal{A}_2) \leq \frac{L^2}{2H}(\log(T) + 1) \tag{2.12}$$

Proof (Theorem 1)**.** Let $\mathbf{x}^* \in \arg\min_{\mathbf{x} \in \mathcal{X}} \sum_{t=1}^T f_t(\mathbf{x})$. Given $f_t : \mathcal{X} \to \mathbb{R}$ is H-strongly convex with respect to $||\cdot||_2$ then by Lemma 2 we have

$$f_t(\mathbf{x}^*) \geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{x}^* - \mathbf{x}_t) + \frac{H}{2}||\mathbf{x}^* - \mathbf{x}_t||_2^2 \tag{2.13}$$

where $\mathbf{y}_t \in \partial f(\mathbf{x}_t)$. This can be written as the upper bound

$$f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*) \leq \mathbf{y}_t^\top(\mathbf{x}_t - \mathbf{x}^*) - \frac{H}{2}||\mathbf{x}_t - \mathbf{x}^*||_2^2 \tag{2.14}$$

Given the non-expansive property of projection as in Lemma 19 we have

$$||\mathbf{x}_{t+1} - \mathbf{x}^*||_2^2 = ||\Pi_{\mathcal{X}}(\mathbf{x}_t - \eta_t \mathbf{y}_t) - \mathbf{x}^*||_2^2 \leq ||\mathbf{x}_t - \eta_t \mathbf{y}_t - \mathbf{x}^*||_2^2 \tag{2.15}$$

and therefore

$$\begin{aligned} ||\mathbf{x}_{t+1} - \mathbf{x}^*||_2^2 &\leq ||\mathbf{x}_t - \eta_t \mathbf{y}_t - \mathbf{x}^*||_2^2 & (2.16) \\ &= ||\mathbf{x}_t - \mathbf{x}^*||_2^2 - 2\eta_t \mathbf{y}_t^\top (\mathbf{x}_t - \mathbf{x}^*) + \eta_t^2 ||\mathbf{y}_t||_2^2 & (2.17) \end{aligned}$$

by rearranging terms we obtain

$$2\mathbf{y}_t^\top (\mathbf{x}_t - \mathbf{x}^*) \leq \frac{||\mathbf{x}_t - \mathbf{x}^*||_2^2 - ||\mathbf{x}_{t+1} - \mathbf{x}^*||_2^2}{\eta_t} + \eta_t ||\mathbf{y}_t||_2^2 \tag{2.18}$$

Combining (2.18) with (2.14) then

$$\begin{aligned} 2\sum_{t=1}^{T} [f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] \leq\ & (\frac{1}{\eta_1} - H)||\mathbf{x}_1 - \mathbf{x}^*||_2^2 - \frac{1}{\eta_T}||\mathbf{x}_{T+1} - \mathbf{x}^*||_2^2 + \\ & \sum_{t=2}^{T} ||\mathbf{x}_t - \mathbf{x}^*||_2^2 (\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} - H) + \sum_{t=1}^{T} \eta_t ||\mathbf{y}_t||_2^2 \end{aligned} \tag{2.19}$$

then given $\eta_t = \frac{1}{Ht}$ the first three terms can be dropped and

$$\sum_{t=1}^{T} [f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] \leq \sum_{t=1}^{T} \frac{1}{2Ht} ||\mathbf{y}_t||_2^2 \tag{2.20}$$

then using Assumption 1(f) and an upper bound on the harmonic series we have

$$\begin{aligned} \sum_{t=1}^{T} [f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] &\leq \frac{L^2}{2H} \sum_{t=1}^{T} \frac{1}{t} & (2.21) \\ &\leq \frac{L^2}{2H} (\log(T) + 1) & (2.22) \end{aligned}$$

Giving the result

$$R_T(\mathcal{A}_2) \leq \frac{L^2}{2H} (\log(T) + 1) \tag{2.23}$$

□

2.3 Online Newton Step

This section presents a quasi-Newton algorithm for α-exp-concave convex functions. This class of functions is a more general class than strongly convex functions. We extend the Online Newton Step (ONS) algorithm to non-smooth α-exp-concave convex functions. A α-exp-concave convex function is formally defined as follows.

Definition 2 (Exponential Concavity)**.** *A function convex $f : \mathcal{X} \to \mathbb{R}$ is a α-exp-concave convex function if $\exp(-\alpha f(\mathbf{x}))$ is concave for all $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$.*

The Online Newton Step (ONS) is similar to the Broyden-Fletcher-Goldfarb-Shanno (BFGS) family of quasi-Newton algorithms in that it descends the objective function from its current primal iterate using the gradient or, in this case, the subgradient [15]. Unlike the BFGS family of quasi-newton algorithms ONS uses a α-exp-concave assumption to approximate the hessian. The ONS algorithm is as follows.

Algorithm 3 Online Newton Step

Require: $\mathbf{x}_1 \in \mathcal{X} \subseteq \mathbb{R}^n$, $\beta = \frac{1}{2}\min\{\frac{1}{4LD}, \alpha\}$, $\mathbf{Q}_0 = \mathbf{I}_n\epsilon$ and $\epsilon = \frac{1}{\beta^2 D^2}$

1: **for** $t = 1$ to T **do**
2: $\quad \mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$
3: $\quad \mathbf{Q}_t = \mathbf{y}_t\mathbf{y}_t^\top + \mathbf{Q}_{t-1}$
4: $\quad \mathbf{z}_{t+1} = \mathbf{x}_t - \frac{1}{\beta}\mathbf{Q}_t^{-1}\mathbf{y}_t$
5: $\quad \mathbf{x}_{t+1} = \Pi_{\mathcal{X}}^{\mathbf{Q}_t}(\mathbf{z}_{t+1}) = \arg\min_{\mathbf{x}\in\mathcal{X}}(\mathbf{z}_{t+1} - \mathbf{x})^\top \mathbf{Q}_t(\mathbf{z}_{t+1} - \mathbf{x})$
6: **end for**

Before we can present the regret bound for this algorithm we need to extend a result obtained by Hazan et al. [14] to non-smooth α-exp-concave convex functions.

Lemma 3. *Consider a function $f : \mathcal{X} \to \mathbb{R}$ where $\exp(-\alpha f(\mathbf{x}))$ is concave for all $\mathbf{x} \in \mathcal{X}$ then the following holds for $\beta \leq \frac{1}{2}\min\{\frac{1}{4LD}, \alpha\}$:*

$$\forall \mathbf{x}, \mathbf{z} \in \mathcal{X} \qquad f(\mathbf{z}) \geq f(\mathbf{x}) + \mathbf{y}^\top(\mathbf{z} - \mathbf{x}) + \frac{\beta}{2}(\mathbf{z} - \mathbf{y})^\top \mathbf{y}\mathbf{y}^\top(\mathbf{z} - \mathbf{y}) \tag{2.24}$$

for all $\mathbf{y} \in \partial f(\mathbf{x})$.

Proof (Lemma 3)**.** Since $\exp(-\alpha f(\mathbf{x}))$ is concave and $2\beta \leq \alpha$ the function $h(\mathbf{x}) \equiv \exp(-2\beta f(\mathbf{x}))$ is also concave. Given that the exponential function is nondecreasing, the convexity of f and $\mathbf{y} \in \partial f(\mathbf{x})$ then

$$\begin{aligned} h(\mathbf{z}) &= \exp(-2\beta f(\mathbf{z})) && (2.25)\\ &\leq \exp(-2\beta[f(\mathbf{x}) + \mathbf{y}^\top(\mathbf{z} - \mathbf{x})]) && (2.26)\\ &= \exp(-2\beta f(\mathbf{x}) - 2\beta\mathbf{y}^\top(\mathbf{z} - \mathbf{x})) && (2.27)\\ &\leq \exp(-2\beta f(\mathbf{x})) - \exp(-2\beta f(\mathbf{x}))2\beta\mathbf{y}^\top(\mathbf{z} - \mathbf{x}) && (2.28) \end{aligned}$$

The last statement follows from: If $g : \mathbb{R} \to \mathbb{R}$ is concave then $g(a+b) \leq g(a) + g'(a)b$ for all $a, b \in \mathbb{R}$. Now consider

$$\exp(-2\beta f(\mathbf{z})) \quad \leq \quad \exp(-2\beta f(\mathbf{x}))[1 - 2\beta \mathbf{y}^\top (\mathbf{z} - \mathbf{x})] \tag{2.29}$$

Rearranging terms then

$$f(\mathbf{z}) \quad \geq \quad f(\mathbf{x}) - \frac{1}{2\beta} \log(1 - 2\beta \mathbf{y}^\top (\mathbf{z} - \mathbf{x})) \tag{2.30}$$

Using bounds on the set $\mathcal{X}$ and subgradients, Assumptions 1(b) and 1(f), then

$$\begin{aligned} |2\beta \mathbf{y}^\top (\mathbf{z} - \mathbf{x})| &\leq 2\beta ||\mathbf{y}||_2 ||\mathbf{z} - \mathbf{x}||_2 && (2.31) \\ &\leq 2\beta LD && (2.32) \\ &\leq \frac{1}{4} && (2.33) \end{aligned}$$

For $|r| \leq \frac{1}{4}$ then $-\log(1-r) \geq r + \frac{1}{4}r^2$. Applying this inequality for $r = 2\beta \mathbf{y}^\top (\mathbf{z} - \mathbf{x})$ we obtain

$$\begin{aligned} f(\mathbf{z}) &\geq f(\mathbf{x}) - \frac{1}{2\beta} \log(1 - 2\beta \mathbf{y}^\top (z - \mathbf{x})) && (2.34) \\ &\geq f(\mathbf{x}) + \frac{1}{2\beta} [2\beta \mathbf{y}^\top (z - \mathbf{x}) + \beta^2 (\mathbf{z} - \mathbf{x}) \mathbf{y}\mathbf{y}^\top (\mathbf{z} - \mathbf{x})] && (2.35) \\ &= f(\mathbf{x}) + \mathbf{y}^\top (\mathbf{z} - \mathbf{x}) + \frac{\beta}{2} (\mathbf{z} - \mathbf{x}) \mathbf{y}\mathbf{y}^\top (\mathbf{z} - \mathbf{x}) && (2.36) \end{aligned}$$

□

The proof of the following regret bound for the Online Newton Step is found in Hazan, et al. [14]. However, our Lemma 3 must be used in place of Lemma 3 from Hazan, et al. [14] to extend the result to non-smooth α-exp-concave convex functions.

Theorem 2. *If $f_t : \mathcal{X} \to \mathbb{R}$ is a α-exp-concave convex function for all $t \in \{1, \ldots, T\}$ subject to the convex set $\mathcal{X} \subseteq \mathbb{R}^n$ then Algorithm $\mathcal{A}_3$ has the following regret bound*

$$R_T(\mathcal{A}_3) \leq 5(\frac{1}{\alpha} + LD) n \log(T) \tag{2.37}$$

As stated previously the class of α-exp-concave convex functions is more general than the class of H-strongly convex functions. A strongly convex function with subgradients upper bounded by L and a parameter $H > 0$ are α-exp-concave for any $\alpha > 0$ such that $\alpha \leq \frac{H}{L^2}$ [14]. By assuming strong convexity, $\alpha = \frac{H}{L^2}$ and using Theorem 2 we obtain the following regret bound

$$R_T(\mathcal{A}_3) \leq 5(\frac{L^2}{H} + LD) n \log(T) \tag{2.38}$$

2.4 Follow the Approximate Leader

In this section we present the Follow the Approximate Leader (FTAL) algorithm [14] that was developed using intuition from the Follow the Leader algorithm [10, 39]. In Section 2.5 we discuss the relation between ONS and FTAL. The Follow the Approximate Leader (FTAL) algorithm is as follows.

Algorithm 4 Follow the Approximate Leader Algorithm (Version 1)

Require: $\mathbf{x}_1 \in \mathcal{X} \subseteq \mathbb{R}^n$ and $\beta = \frac{1}{2}\min\{\frac{1}{4LD}, \alpha\}$
1: **for** $t = 1$ to T **do**
2: $\quad \mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$
3: $\quad \mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{i=1}^{t} f_i(\mathbf{x}_i) + \mathbf{y}_i^\top(\mathbf{x} - \mathbf{x}_i) + \frac{\beta}{2}(\mathbf{x} - \mathbf{x}_i)^\top \mathbf{y}_i \mathbf{y}_i^\top (\mathbf{x} - \mathbf{x}_i)$
4: **end for**

The proof of the regret bound for FTAL is found in Hazan et al. [14]. However, our Lemma 3 must be used in place of Lemma 3 from Hazan et al. [14] to extend the result to non-smooth α-exp-concave convex functions.

Theorem 3. *If $f_t : \mathcal{X} \to \mathbb{R}$ is a α-exp-concave convex function for all $t \in \{1, \ldots, T\}$ and subject to the convex set $\mathcal{X} \subseteq \mathbb{R}^n$ then Algorithm $\mathcal{A}_4$ has the following regret bound*

$$R_T(\mathcal{A}_4) \leq 64(\frac{L^2}{H} + LD)n(\log(T) + 1) \tag{2.39}$$

To show how FTAL is related to a quasi-Newton algorithm we present an alternative form of Algorithm 4. Given that $\mathbf{Q}_t^\dagger$ denotes the Moore-Penrose pseudoinverse of $\mathbf{Q}_t$ consider the following algorithm.

Algorithm 5 Follow the Approximate Leader Algorithm (Version 2)

Require: $\mathbf{x}_1 \in \mathcal{X} \subseteq \mathbb{R}^n$, $\beta = \frac{1}{2}\min\{\frac{1}{4LD}, \alpha\}$, $\mathbf{Q}_0 = 0$ and $\mathbf{b}_0 = 0$
1: **for** $t = 1$ to T **do**
2: $\quad \mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$
3: $\quad \mathbf{Q}_t = \mathbf{y}_t \mathbf{y}_t^\top + \mathbf{Q}_{t-1}$
4: $\quad \mathbf{b}_t = \mathbf{y}_t \mathbf{y}_t^\top \mathbf{x}_t - \frac{1}{\beta}\mathbf{y}_t + \mathbf{b}_{t-1}$
5: $\quad \mathbf{z}_{t+1} = \mathbf{Q}_t^\dagger \mathbf{b}_t$
6: $\quad \mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{X}} (\mathbf{z}_{t+1} - \mathbf{x})^\top \mathbf{Q}_t (\mathbf{z}_{t+1} - \mathbf{x})$
7: **end for**

At first glance these two algorithms appear to be different. However, following Hazan et al. [14] we show with Lemma 4 that Algorithm 4 and Algorithm 5 are essentially equivalent.

Lemma 4. *Given $\mathbf{Q}_t$ is invertible then Algorithm 4 and Algorithm 5 are equivalent.*

Proof (Lemma 4)**.** Consider the update from Algorithm 4

$$\begin{aligned}
\mathbf{x}_{t+1} &= \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{i=1}^{t} f_i(\mathbf{x}_i) + \mathbf{y}_i^\top(\mathbf{x}-\mathbf{x}_i) + \frac{\beta}{2}(\mathbf{x}-\mathbf{x}_i)^\top \mathbf{y}_i\mathbf{y}_i^\top(\mathbf{x}-\mathbf{x}_i) && (2.40)\\
&= \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{i=1}^{t} -(\beta\mathbf{x}_i^\top\mathbf{y}_i\mathbf{y}_i^\top - \mathbf{y}_i^\top)\mathbf{x} + \frac{\beta}{2}\mathbf{x}^\top\mathbf{y}_i\mathbf{y}_i^\top\mathbf{x} && (2.41)\\
&= \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{i=1}^{t} \frac{1}{2}\mathbf{x}^\top\mathbf{y}_i\mathbf{y}_i^\top\mathbf{x} - (\mathbf{y}_i\mathbf{y}_i^\top\mathbf{x}_i - \frac{1}{\beta}\mathbf{y}_i)^\top\mathbf{x} && (2.42)\\
&= \arg\min_{\mathbf{x}\in\mathcal{X}} \frac{1}{2}\mathbf{x}^\top\mathbf{Q}_t\mathbf{x} - \mathbf{b}_t^\top\mathbf{x} && (2.43)\\
&= \arg\min_{\mathbf{x}\in\mathcal{X}} \mathbf{x}^\top\mathbf{Q}_t\mathbf{x} - 2\mathbf{b}_t^\top\mathbf{x} && (2.44)\\
&= \arg\min_{\mathbf{x}\in\mathcal{X}} (\mathbf{x}-\mathbf{Q}_t^{-1}\mathbf{b}_t)^\top\mathbf{Q}_t(\mathbf{x}-\mathbf{Q}_t^{-1}\mathbf{b}_t) - \mathbf{b}_t^\top\mathbf{Q}_t^{-1}\mathbf{b}_t && (2.45)
\end{aligned}$$

which is the update step for Algorithm 5.

□

Lemma 4 shows that Algorithm 4 and Algorithm 5 are essentially equivalent. This result provides insight into how FTAL works and how it is related to a quasi-Newton algorithm. Section 2.3 also presented ONS as a quasi-Newton algorithm. In the next section we will explore how ONS and FTAL are related.

2.5 Relation of ONS and FTAL Algorithms

Previous work states that ONS, presented as Algorithm 3, was derived using intuition from FTAL, presented as Algorithm 5 [14]. However, it is unclear exactly how these two algorithms are related. Their relationship is highlighted by Lemma 5.

Lemma 5. *Given Algorithms 3 and 5 let $\mathbf{z}_t \in \mathcal{X}$ for all $t \in \{1,\dots,T\}$ and $\mathbf{x}_1 = \mathbf{0}$. If Algorithm 5 is such that $\mathbf{Q}_0 = \epsilon\mathbf{I}_n$ where $\epsilon > 0$ then the resulting algorithm is equivalent to Algorithm 3.*

Proof (Lemma 5)**.** Given $\mathbf{z}_t \in \mathcal{X}$ for all $t \in \{1,\dots,T\}$ then $\mathbf{x}_t = \mathbf{z}_t$ for all $t \in \{1,\dots,T\}$. Given our assumptions we will inductively show that the algorithms are equivalent. For $t = 1$ the update of Algorithm 3 is

$$\begin{aligned}
\mathbf{x}_2 &= \mathbf{x}_1 - \frac{1}{\beta}\mathbf{Q}_1^{-1}\mathbf{y}_1 && (2.46)\\
\mathbf{x}_2 &= -\frac{1}{\beta}\mathbf{Q}_1^{-1}\mathbf{y}_1 && (2.47)
\end{aligned}$$

The update for Algorithm 5 where $\mathbf{Q}_0 = \epsilon \mathbf{I}_n$ and $\epsilon > 0$ is

$$\mathbf{x}_{t+1} = \mathbf{Q}_t^{-1}(\sum_{i=1}^{t} \mathbf{y}_i \mathbf{y}_i^\top \mathbf{x}_i - \frac{1}{\beta}\mathbf{y}_i) \tag{2.48}$$

where $\mathbf{Q}_t = \sum_{i=1}^{t} \mathbf{y}_i \mathbf{y}_i^\top + \epsilon \mathbf{I}_n$. At $t = 1$ the update is as follows

$$\mathbf{x}_2 = \mathbf{Q}_1^{-1}(\mathbf{y}_1 \mathbf{y}_1^\top \mathbf{x}_1 - \frac{1}{\beta}\mathbf{y}_1) \tag{2.49}$$

$$\mathbf{x}_2 = -\frac{1}{\beta}\mathbf{Q}_1^{-1}\mathbf{y}_1 \tag{2.50}$$

showing that the update is equivalent at $t = 1$. Given the update is equivalent at t we show that the update is also equivalent at $t + 1$. Consider again the update of Algorithm 5.

$$\mathbf{x}_{t+1} = \mathbf{Q}_t^{-1}(\sum_{i=1}^{t} \mathbf{y}_i \mathbf{y}_i^\top \mathbf{x}_i - \frac{1}{\beta}\sum_{i=1}^{t-1} \mathbf{y}_i) - \frac{1}{\beta}\mathbf{Q}_t^{-1}\mathbf{y}_t \tag{2.51}$$

$$\mathbf{x}_{t+1} = \mathbf{p}_t - \frac{1}{\beta}\mathbf{Q}_t^{-1}\mathbf{y}_t \tag{2.52}$$

where $\mathbf{p}_t = \mathbf{Q}_t^{-1}(\sum_{i=1}^{t} \mathbf{y}_i \mathbf{y}_i^\top \mathbf{x}_i - \frac{1}{\beta}\sum_{i=1}^{t-1} \mathbf{y}_i)$. Now consider

$$\mathbf{x}_t = (\sum_{i=1}^{t-1} \mathbf{y}_i \mathbf{y}_i^\top + \mathbf{Q}_0)^{-1}(\sum_{i=1}^{t-1} \mathbf{y}_i \mathbf{y}_i^\top \mathbf{x}_i - \frac{1}{\beta}\mathbf{y}_i) \tag{2.53}$$

$$(\sum_{i=1}^{t-1} \mathbf{y}_i \mathbf{y}_i^\top + \mathbf{Q}_0)\mathbf{x}_t = \sum_{i=1}^{t-1} \mathbf{y}_i \mathbf{y}_i^\top \mathbf{x}_i - \frac{1}{\beta}\sum_{i=1}^{t-1} \mathbf{y}_i \tag{2.54}$$

$$(\sum_{i=1}^{t-1} \mathbf{y}_i \mathbf{y}_i^\top + \mathbf{Q}_0)\mathbf{x}_t + \mathbf{y}_t \mathbf{y}_t^\top \mathbf{x}_t = \sum_{i=1}^{t-1} \mathbf{y}_i \mathbf{y}_i^\top \mathbf{x}_i - \frac{1}{\beta}\sum_{i=1}^{t-1} \mathbf{y}_i + \mathbf{y}_t \mathbf{y}_t^\top \mathbf{x}_t \tag{2.55}$$

$$(\sum_{i=1}^{t} \mathbf{y}_i \mathbf{y}_i^\top + \mathbf{Q}_0)\mathbf{x}_t = \sum_{i=1}^{t} \mathbf{y}_i \mathbf{y}_i^\top \mathbf{x}_i - \frac{1}{\beta}\sum_{i=1}^{t-1} \mathbf{y}_i \tag{2.56}$$

$$\mathbf{Q}_t \mathbf{x}_t = \sum_{i=1}^{t} \mathbf{y}_i \mathbf{y}_i^\top \mathbf{x}_i - \frac{1}{\beta}\sum_{i=1}^{t-1} \mathbf{y}_i \tag{2.57}$$

$$\mathbf{x}_t = \mathbf{Q}_t^{-1}(\sum_{i=1}^{t} \mathbf{y}_i \mathbf{y}_i^\top \mathbf{x}_i - \frac{1}{\beta}\sum_{i=1}^{t-1} \mathbf{y}_i) \tag{2.58}$$

$$\mathbf{x}_t = \mathbf{p}_t \tag{2.59}$$

and therefore

$$\mathbf{x}_{t+1} = \mathbf{p}_t - \frac{1}{\beta}\mathbf{Q}_t^{-1}\mathbf{y}_t \tag{2.60}$$

$$= \mathbf{x}_t - \frac{1}{\beta}\mathbf{Q}_t^{-1}\mathbf{y}_t \tag{2.61}$$

showing that given our assumptions Algorithm 5 is equivalent to Algorithm 3.

□

As proven by Lemma 5 the only difference between ONS, Algorithm 3, and FTAL, Algorithm 5, is an initialization of the hessian approximation and a projection step onto a convex set $\mathcal{X}$. The projection step of FTAL incorporates all primal and dual points in calculating the descent direction represented by $\mathbf{b}_t$ in Algorithm 5. In contrast, the projection step of ONS uses only the most up-to-date subgradient $-\mathbf{y}_t$ as the descent direction in Algorithm 3. This indicates that ONS is more like a classic quasi-Newton algorithm [40] while FTAL creates an ever improving quadratic approximation of the objective function. At each iteration FTAL minimizes the approximate objective function subject to the constraints.

2.6 Implementation

A naive implementation of ONS, Algorithm 3, requires that a matrix inverse be calculated at each iteration. However, a well known formula can be used to update the inverse of a matrix with rank one updates [41].

$$(\mathbf{Q} + \mathbf{y}\mathbf{y}^\top)^{-1} = \mathbf{Q}^{-1} - \frac{\mathbf{Q}^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{Q}^{-1}}{1 + \mathbf{y}^\top\mathbf{Q}^{-1}\mathbf{y}} \tag{2.62}$$

Therefore, updates of ONS can be calculated in $\mathcal{O}(n^2)$ time with matrix-vector and vector-vector products, not including the projection step. Similarly, $\mathcal{O}(n^2)$ space is required to store the matrix at each iteration. Therefore Algorithm 3 can be implemented in $\mathcal{O}(n^2)$ time and space without the projection step.

The projection step can be solved as a convex optimization problem with space and time requirements depending on the constraint set. For many common and simple convex sets, such as the unit cube, efficient algorithms exist.

Chapter 3

New Quasi-Newton Algorithms

This chapter presents two new algorithms for strongly convex functions. Section 3.1 presents a new Online Newton Step (ONS) algorithm for strongly convex functions. In Section 3.2, using the intuition from Section 2.5, a new FTAL algorithm for strongly convex functions is presented. The regret bounds for these algorithms are not a function of the number of iterations but instead depend on a sequence of strongly convex parameters $\{H_1, \ldots, H_T\}$. In Chapter 4, Section 4.3, we discuss how these quasi-Newton algorithms can be parallelized when given a summed objective function.

3.1 ONS for Strongly Convex Functions

We present a new Online Newton Step (ONS) algorithm for strongly convex functions and the corresponding regret bound. Although our analysis requires that the convex set and the subgradients are bounded, unlike the algorithms of Chapter 2 our algorithms do not depend on the parameters D or L as defined by Assumption 1. Also, we obtain a provably tighter regret bound than Algorithm 3. The choice of the initialization parameter ϵ, and the resulting bound, is addressed in Section 3.1.2. The new ONS algorithm is presented as Algorithm 6.

Algorithm 6 Online Newton Step for Strongly Convex Functions

Require: $\mathbf{x}_1 \in \mathcal{X} \subseteq \mathbb{R}^n$, $\{H_t\}_{t=1}^T$, $\mathbf{Q}_0 = \mathbf{I}_n \epsilon$ and $\epsilon > 0$

1: **for** $t = 1$ to T **do**
2: $\quad \mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$
3: $\quad \mathbf{v}_t = \frac{\sqrt{H_t}}{||\mathbf{y}_t||_2} \mathbf{y}_t$
4: $\quad \mathbf{Q}_t = \mathbf{v}_t \mathbf{v}_t^\top + \mathbf{Q}_{t-1}$
5: $\quad \mathbf{z}_{t+1} = \mathbf{x}_t - \mathbf{Q}_t^{-1} \mathbf{y}_t$
6: $\quad \mathbf{x}_{t+1} = \Pi_{\mathcal{X}}^{\mathbf{Q}_t}(\mathbf{z}_{t+1}) = \arg\min_{\mathbf{x} \in \mathcal{X}} (\mathbf{z}_{t+1} - \mathbf{x})^\top \mathbf{Q}_t (\mathbf{z}_{t+1} - \mathbf{x})$
7: **end for**

3.1.1 Analysis

We now present our results with proofs to follow. Lemma 6 exploits strong convexity to obtain a lower bound on the objective function.

Lemma 6. *Given a H_t-strongly convex function with respect to $||\cdot||_p$ where $p \in \{1,2\}$ then for all $\mathbf{z}, \mathbf{x}_t \in \mathcal{X}$ and any $\mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$ where $||\mathbf{y}_t||_2 \leq L$*

$$f_t(\mathbf{z}) \geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z} - \mathbf{x}_t) + \frac{1}{2}[\mathbf{v}_t^\top(\mathbf{z} - \mathbf{x}_t)]^2 \tag{3.1}$$

where $\mathbf{v}_t = \alpha_t \mathbf{y}_t$ and $\alpha_t = \frac{\sqrt{H_t}}{||\mathbf{y}_t||_2}$. Also, given $\beta = \frac{H_{\min}}{L^2}$ where $H_{\min} = \min_t H_t$ then $\frac{\alpha_t^2}{\beta} \geq 1$.

Next we present the main theorem using Lemma 6 and Lemmas 18 and 19 of Appendix B. The regret bound is presented with an unknown $\epsilon > 0$. The choice of ϵ is addressed in Section 3.1.2. Notice that the strongly convex constant is allowed to change with each iteration.

Theorem 4. *If $f_t : \mathcal{X} \to \mathbb{R}$ is a H_t-strongly convex function with respect to $||\cdot||_p$ for all $t \in \{1, \ldots, T\}$ where $p \in \{1,2\}$, $H_t > 0$ and $H_{\min} = \min_t H_t$ subject to the convex set $\mathcal{X} \subseteq \mathbb{R}^n$ then Algorithm $\mathcal{A}_6$ has the following regret bound*

$$R_T(\mathcal{A}_6) \leq \frac{nL^2}{2H_{\min}} \log(\frac{1}{\epsilon}\sum_{t=1}^{T} H_t + 1) + \frac{\epsilon D^2}{2} \tag{3.2}$$

where $\epsilon > 0$ is an initialization parameter.

The proofs for the above results are as follows.

Proof (Lemma 6)**.** Consider Theorem 7.7.3 of Horn and Johnson [42]. Given $\mathbf{A} \in \mathbb{S}_{++}^n$ and $\mathbf{B} \in \mathbb{S}_{+}^n$ then $\mathbf{A} \succeq \mathbf{B}$ if and only if $\lambda_{\max}(\mathbf{B}\mathbf{A}^{-1}) \leq 1$. Let $\mathbf{A} = \mathbf{I}_n$ and let $\mathbf{B} = \frac{1}{||\mathbf{y}_t||_2^2}\mathbf{y}_t\mathbf{y}_t^\top$. Therefore if $\lambda_{\max}(\mathbf{B}) \leq 1$ then $\mathbf{I}_n \succeq \mathbf{B}$.

$$\begin{aligned} \text{trace}(\mathbf{B}) &= \frac{1}{||\mathbf{y}_t||_2^2}\sum_{i=1}^{n}(y_t(i))^2 && (3.3)\\ &= \frac{1}{||\mathbf{y}_t||_2^2}||\mathbf{y}_t||_2^2 && (3.4)\\ &= 1 && (3.5)\end{aligned}$$

We have $\text{trace}(\mathbf{B}) = \sum_{i=1}^n \lambda_i = 1$ and given $\mathbf{B} \succeq 0$ then $\lambda_i \geq 0$ for all $i \in \{1, \ldots, n\}$. Therefore $\lambda_i \leq 1$ for all $i \in \{1, \ldots, n\}$ and therefore $\lambda_{max}(\mathbf{B}) \leq 1$. This implies that $\mathbf{I}_n \succeq \mathbf{B}$. Using strong convexity, Lemma 2 and given $||\cdot||_1^2 \geq ||\cdot||_2^2$ we have for some $p \in \{1,2\}$ that

$$\begin{aligned} f_t(\mathbf{z}) &\geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z} - \mathbf{x}_t) + \frac{H_t}{2}||\mathbf{z} - \mathbf{x}_t||_p^2 && (3.6)\\ &\geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z} - \mathbf{x}_t) + \frac{H_t}{2}||\mathbf{z} - \mathbf{x}_t||_2^2 && (3.7)\end{aligned}$$

Now consider

$$\begin{aligned} f_t(\mathbf{z}) &\geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z}-\mathbf{x}_t) + \frac{H_t}{2}(\mathbf{z}-\mathbf{x}_t)^\top\mathbf{B}(\mathbf{z}-\mathbf{x}_t) && (3.8)\\ &= f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z}-\mathbf{x}_t) + \frac{1}{2}(\mathbf{z}-\mathbf{x}_t)^\top\mathbf{v}_t\mathbf{v}_t^\top(\mathbf{z}-\mathbf{x}_t) && (3.9)\\ &= f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z}-\mathbf{x}_t) + \frac{1}{2}[\mathbf{v}_t^\top(\mathbf{z}-\mathbf{x}_t)]^2 && (3.10)\end{aligned}$$

where $\mathbf{v}_t = \alpha_t\mathbf{y}_t$ and $\alpha_t = \frac{\sqrt{H_t}}{||\mathbf{y}_t||_2}$. Now consider that $\alpha_t^2 = \frac{H_t}{||\mathbf{y}_t||_2^2}$ and $\beta = \frac{H_{\min}}{L^2}$ we then have

$$\begin{aligned} {\alpha_t}^2 &= \frac{H_t}{||\mathbf{y}_t||_2^2} && (3.11)\\ &\geq \frac{H_{\min}}{L^2} && (3.12)\\ &= \beta && (3.13)\end{aligned}$$

and therefore $\frac{\alpha_t^2}{\beta} \geq 1$.

□

Proof (Theorem 4)**.** Starting with the definition of $\mathbf{z}_{t+1}$ we have

$$\begin{aligned} \mathbf{z}_{t+1} - \mathbf{x}^* &= \mathbf{x}_t - \mathbf{x}^* - \mathbf{Q}_t^{-1}\mathbf{y}_t && (3.14)\\ \mathbf{Q}_t(\mathbf{z}_{t+1} - \mathbf{x}^*) &= \mathbf{Q}_t(\mathbf{x}_t - \mathbf{x}^*) - \mathbf{y}_t && (3.15)\end{aligned}$$

multiplying on the left by (3.14) we obtain

$$\begin{aligned} (\mathbf{z}_{t+1}-\mathbf{x}^*)^\top\mathbf{Q}_t(\mathbf{z}_{t+1}-\mathbf{x}^*) &= (\mathbf{x}_t - \mathbf{x}^* - \mathbf{Q}_t^{-1}\mathbf{y}_t)^\top\mathbf{Q}_t(\mathbf{x}_t-\mathbf{x}^*) - \\ &\quad (\mathbf{x}_t - \mathbf{x}^* - \mathbf{Q}_t^{-1}\mathbf{y}_t)^\top\mathbf{y}_t && (3.16)\\ (\mathbf{z}_{t+1}-\mathbf{x}^*)^\top\mathbf{Q}_t(\mathbf{z}_{t+1}-\mathbf{x}^*) &= (\mathbf{x}_t-\mathbf{x}^*)^\top\mathbf{Q}_t(\mathbf{x}_t-\mathbf{x}^*) - 2\mathbf{y}_t^\top(\mathbf{x}_t-\mathbf{x}^*) + \\ &\quad \mathbf{y}_t^\top\mathbf{Q}_t^{-1}\mathbf{y}_t && (3.17)\end{aligned}$$

rearranging terms we have

$$2\mathbf{y}_t^\top(\mathbf{x}_t-\mathbf{x}^*) = ||\mathbf{x}_t-\mathbf{x}^*||_{\mathbf{Q}_t}^2 - ||\mathbf{z}_{t+1}-\mathbf{x}^*||_{\mathbf{Q}_t}^2 + \mathbf{y}_t^\top\mathbf{Q}_t^{-1}\mathbf{y}_t \quad (3.18)$$

Given $\mathbf{Q}_t^{-1}$ is positive definite then using $\frac{\alpha_t^2}{\beta} \geq 1$ of Lemma 6 and let $\mathbf{v}_t \equiv \alpha_t\mathbf{y}_t$ we have

$$\begin{aligned} 2\mathbf{y}_t^\top(\mathbf{x}_t-\mathbf{x}^*) &\leq ||\mathbf{x}_t-\mathbf{x}^*||_{\mathbf{Q}_t}^2 - ||\mathbf{z}_{t+1}-\mathbf{x}^*||_{\mathbf{Q}_t}^2 + \frac{\alpha_t^2}{\beta}\mathbf{y}_t^\top\mathbf{Q}_t^{-1}\mathbf{y}_t && (3.19)\\ &= ||\mathbf{x}_t-\mathbf{x}^*||_{\mathbf{Q}_t}^2 - ||\mathbf{z}_{t+1}-\mathbf{x}^*||_{\mathbf{Q}_t}^2 + \frac{1}{\beta}\mathbf{v}_t^\top\mathbf{Q}_t^{-1}\mathbf{v}_t && (3.20)\end{aligned}$$

Using Lemma 19 we have

$$||\mathbf{z}_{t+1} - \mathbf{x}^*||^2_{\mathbf{Q}_t} \geq ||\mathbf{x}_{t+1} - \mathbf{x}^*||^2_{\mathbf{Q}_t} \tag{3.21}$$

and then

$$2\mathbf{y}_t^\top(\mathbf{x}_t - \mathbf{x}^*) \leq ||\mathbf{x}_t - \mathbf{x}^*||^2_{\mathbf{Q}_t} - ||\mathbf{x}_{t+1} - \mathbf{x}^*||^2_{\mathbf{Q}_t} + \frac{1}{\beta}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \tag{3.22}$$

summing together all T inequalities

$$\begin{aligned} 2\sum_{t=1}^{T}\mathbf{y}_t^\top(\mathbf{x}_t - \mathbf{x}^*) &\leq ||\mathbf{x}_1 - \mathbf{x}^*||^2_{\mathbf{Q}_1} + \sum_{t=2}^{T}(\mathbf{x}_t - \mathbf{x}^*)^\top(\mathbf{Q}_t - \mathbf{Q}_{t-1})(\mathbf{x}_t - \mathbf{x}^*) - \\ &\quad ||\mathbf{x}_{T+1} - \mathbf{x}^*||^2_{\mathbf{Q}_T} + \frac{1}{\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \end{aligned} \tag{3.23}$$

given $\mathbf{Q}_t - \mathbf{Q}_{t-1} = \mathbf{v}_t\mathbf{v}_t^\top$ then

$$\begin{aligned} 2\sum_{t=1}^{T}\mathbf{y}_t^\top(\mathbf{x}_t - \mathbf{x}^*) &\leq (\mathbf{x}_1 - \mathbf{x}^*)^\top[\mathbf{Q}_1 - \mathbf{v}_1\mathbf{v}_1^\top](\mathbf{x}_1 - \mathbf{x}^*) + \sum_{t=1}^{T}[\mathbf{v}_t^\top(\mathbf{x}_t - \mathbf{x}^*)]^2 - \\ &\quad ||\mathbf{x}_{T+1} - \mathbf{x}^*||^2_{\mathbf{Q}_T} + \frac{1}{\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \end{aligned} \tag{3.24}$$

$$\begin{aligned} 2\sum_{t=1}^{T}\mathbf{y}_t^\top(\mathbf{x}_t - \mathbf{x}^*) &\leq (\mathbf{x}_1 - \mathbf{x}^*)^\top\mathbf{Q}_0(\mathbf{x}_1 - \mathbf{x}^*) + \sum_{t=1}^{T}[\mathbf{v}_t^\top(\mathbf{x}_t - \mathbf{x}^*)]^2 - \\ &\quad ||\mathbf{x}_{T+1} - \mathbf{x}^*||^2_{\mathbf{Q}_T} + \frac{1}{\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \end{aligned} \tag{3.25}$$

given Assumption 1(b) of Section 2.1, $\mathbf{Q}_0 = I_n\epsilon$ and dropping negative terms we obtain

$$2\sum_{t=1}^{T}\mathbf{y}_t^\top(\mathbf{x}_t - \mathbf{x}^*) \leq \epsilon D^2 + \sum_{t=1}^{T}[\mathbf{v}_t^\top(\mathbf{x}_t - \mathbf{x}^*)]^2 + \frac{1}{\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \tag{3.26}$$

using the lower bound of Lemma 6 then

$$\begin{aligned} \sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] &\leq \frac{\epsilon D^2}{2} + \frac{1}{2}\sum_{t=1}^{T}[\mathbf{v}_t^\top(\mathbf{x}_t - \mathbf{x}^*)]^2 - \frac{1}{2}\sum_{t=1}^{T}[\mathbf{v}_t^\top(\mathbf{x}_t - \mathbf{x}^*)]^2 + \\ &\quad \frac{1}{2\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \end{aligned} \tag{3.27}$$

$$\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] \leq \frac{1}{2\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t + \frac{\epsilon D^2}{2} \tag{3.28}$$

using Lemma 18 and $||\mathbf{v}_t||_2 \leq \sqrt{H_t}$ then

$$\sum_{t=1}^{T} \mathbf{v}_t^\top \mathbf{Q}_t^{-1} \mathbf{v}_t \leq n \log(\frac{1}{\epsilon} \sum_{t=1}^{T} H_t + 1) \tag{3.29}$$

finally given $\beta = \frac{H_{\min}}{L^2}$ we have the result

$$R_T(\mathcal{A}_6) \leq \frac{nL^2}{2H_{\min}} \log(\frac{1}{\epsilon} \sum_{t=1}^{T} H_t + 1) + \frac{\epsilon D^2}{2} \tag{3.30}$$

□

3.1.2 Initial Conditions

We now present two different choices for ϵ to use in Algorithm 6 and the resulting bound. One candidate initial parameters ϵ is determined by solving a simple optimization problem and other is determined by inspection. We assume that all strongly convex constants are equal to H since it does not effect the choice of ϵ.

Initial ϵ by Optimization

Given the bound from Theorem 4 then

$$\begin{aligned} R_T(\mathcal{A}_6) &\leq \frac{L^2}{2H} n \log(\frac{HT}{\epsilon} + 1) + \frac{\epsilon D^2}{2} && (3.31) \\ &= \frac{L^2}{2H} n \log(\frac{HT}{\epsilon}) + \frac{L^2}{2H} n \log(\frac{\frac{HT}{\epsilon} + 1}{\frac{HT}{\epsilon}}) + \frac{\epsilon D^2}{2} && (3.32) \end{aligned}$$

Given $H, \epsilon > 0$ then for large enough T we have $1 \leq \frac{HT}{\epsilon}$ and

$$R_T(\mathcal{A}_6) \leq \frac{L^2}{2H} n \log(\frac{HT}{\epsilon}) + \frac{L^2}{2H} n \log(2) + \frac{\epsilon D^2}{2} \tag{3.33}$$

let $g : \mathbb{R}_+ \to \mathbb{R}$ and

$$\begin{aligned} g(\epsilon) &= \frac{L^2}{2H} n \log(\frac{HT}{\epsilon}) + \frac{L^2}{2H} n \log(2) + \frac{\epsilon D^2}{2} && (3.34) \\ &= \frac{L^2}{2H} n \log(HT) - \frac{L^2}{2H} n \log(\epsilon) + \frac{L^2}{2H} n \log(2) + \frac{\epsilon D^2}{2} && (3.35) \end{aligned}$$

By inspection, $g(\epsilon)$ is convex. Now, solving

$$\min_{\epsilon \in \mathbb{R}} g(\epsilon)$$

$$g'(\epsilon) = -\frac{nL^2}{2H}\frac{1}{\epsilon} + \frac{D^2}{2} = 0$$

and then

$$\epsilon = \frac{nL^2}{HD^2}$$

It is tempting to use this optimized parameter in practice. However, intuition indicates the n parameter is an artifact of analysis. Therefore, in the numerical experiments of Chapter 6 we use $\epsilon = \frac{L^2}{HD^2}$. Again using the bound from Theorem 4 and letting $\epsilon = \frac{L^2}{HD^2}$

$$R_T(\mathcal{A}_6) \leq \frac{L^2}{2H} n \log(\frac{D^2H^2T}{L^2} + 1) + \frac{L^2}{2H} \tag{3.36}$$

Now let $\kappa = \frac{L^2}{H}$ be the effective condition number of the function sequence $\{f_1, f_2, \dots\}$ then

$$R_T(\mathcal{A}_6) \leq \frac{n\kappa}{2} \log(\frac{D^2H}{\kappa}T + 1) + \frac{\kappa}{2} \tag{3.37}$$

Initial ϵ by Inspection

In addition, by inspection we can obtain another bound that is better than bound (2.38). Also, the algorithm using the following choice of ϵ does not dependent on the knowledge of the parameters D or L. Consider

$$R_T(\mathcal{A}_6) \leq \frac{L^2}{2H} n \log(\frac{HT}{\epsilon} + 1) + \frac{\epsilon D^2}{2} \tag{3.38}$$

let $\epsilon = H$ then

$$\begin{aligned}
R_T(\mathcal{A}_6) &\leq \frac{1}{2}\frac{L^2}{H} n \log(T+1) + \frac{HD^2}{2} && (3.39)\\
&= \frac{1}{2}\frac{L^2}{H} n \log(T) + \frac{1}{2}\frac{L^2}{H} n \log(T+1) - \frac{1}{2}\frac{L^2}{H} n \log(T) + \frac{HD^2}{2} && (3.40)\\
&= \frac{1}{2}\frac{L^2}{H} n \log(T) + \frac{1}{2}\frac{L^2}{H} n \log(\frac{T+1}{T}) + \frac{HD^2}{2} && (3.41)\\
&\leq \frac{1}{2}\frac{L^2}{H} n \log(T) + \frac{1}{2}\frac{L^2}{H} n \log(2) + \frac{HD^2}{2} && (3.42)
\end{aligned}$$

Now compare this to (2.38), restated below for convenience.

$$R_T(\mathcal{A}_3) \leq 5\frac{L^2}{H} n \log(T) + 5\frac{LD}{2} n \log(T)$$

It is clear that the first term dependent on $\log(T)$ is larger than the first term of our bound. The second term is also dependent on $\log(T)$ but our term is simply a constant. It is therefore clear for some T our bound is tighter than (2.38). Furthermore, our result shows that the effect of choosing an initial argument far from the optimal, or having a large constraint set $\mathcal{X}$, does not grow with the number of iterations T.

3.2 FTAL for Strongly Convex Functions

This section presents a FTAL algorithm for strongly convex functions. Algorithm 7 is similar to Algorithm 5 but has an initial approximate hessian, a strongly convex assumption and does not depend on the parameters D or L as defined by Assumption 1. A similar variant can be found in related work with a α-exp-concave assumption [37]. The FTAL algorithm for strongly convex functions is as follows.

Algorithm 7 FTAL for Strongly Convex Functions

Require: $\mathbf{x}_1 \in \mathcal{X}$, $\{H_t\}_{t=1}^T$, $\mathbf{Q}_0 = \mathbf{I}_n\epsilon$, $\epsilon > 0$ and $\mathbf{b}_0 = 0$

1: **for** $t = 1$ to T **do**
2: $\quad \mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$
3: $\quad \mathbf{v}_t = \frac{\sqrt{H_t}}{||\mathbf{y}_t||_2}\mathbf{y}_t$
4: $\quad \mathbf{Q}_t = \mathbf{v}_t\mathbf{v}_t^\top + \mathbf{Q}_{t-1}$
5: $\quad \mathbf{b}_t = \mathbf{v}_t\mathbf{v}_t^\top\mathbf{x}_t - \mathbf{y}_t + \mathbf{b}_{t-1}$
6: $\quad \mathbf{z}_{t+1} = \mathbf{Q}_t^{-1}\mathbf{b}_t$
7: $\quad \mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{X}}(\mathbf{z}_{t+1} - \mathbf{x})^\top\mathbf{Q}_t(\mathbf{z}_{t+1} - \mathbf{x})$
8: **end for**

3.2.1 Analysis

The regret bound proof for FTAL uses a proxy function $\tilde{f}_t$ that lower bounds each function f_t and is derived from Lemma 6. A bound on the difference between two sequential proxy functions $[\tilde{f}_t(\mathbf{x}_t) - \tilde{f}_t(\mathbf{x}_{t+1})]$ is obtained and then it is shown that this difference is greater than the difference between the optimal function value, $f_t(\mathbf{x}^*)$, and the function value generated by the algorithm, $f_t(\mathbf{x}_t)$. Finally, the sum of differences $[\tilde{f}_t(\mathbf{x}_t) - \tilde{f}_t(\mathbf{x}^*)]$ is bounded to obtain the regret bound of Theorem 6. We start with Lemma 7 that uses the proxy functions $\{\tilde{f}_t\}_{t=1}^T$ to bound the given functions $\{f_t\}_{t=1}^T$.

Lemma 7. *Let $f_t : \mathcal{X} \to \mathbb{R}$ be a cost function and let $\mathbf{x}_t \in \mathcal{X}$ for $t \in \{1, \ldots, T\}$. Let $\tilde{f}_t : \mathcal{X} \to \mathbb{R}$ be a proxy function for $t \in \{1, \ldots, T\}$ such that $f_t(\mathbf{x}_t) = \tilde{f}_t(\mathbf{x}_t)$ and for all $\mathbf{x} \in \mathcal{X}$ let $f_t(\mathbf{x}) \geq \tilde{f}_t(\mathbf{x})$. Then*

$$\sum_{t=1}^T f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^T f_t(\mathbf{x}) \leq \sum_{t=1}^T \tilde{f}_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{X}} h_T(\mathbf{x}) + \frac{1}{2}\mathbf{x}^{*\top}\mathbf{Q}_0\mathbf{x}^* \tag{3.43}$$

where $h_T(\mathbf{x}) = \sum_{t=1}^T \tilde{f}_t(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top\mathbf{Q}_0\mathbf{x}$ and $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^T f_t(\mathbf{x})$.

We now need to bound proxy function regret. Lemma 8 shows that proxy function regret can be upper bounded by the difference of proxy functions.

Lemma 8. *Let $\tilde{f}_t : \mathcal{X} \to \mathbb{R}$ for $t \in \{1, \ldots, T\}$ be a sequence of proxy functions and let $\mathbf{x}_{T+1} = \arg\min_{\mathbf{x}\in\mathcal{X}} h_T(\mathbf{x})$ where $h_T(\mathbf{x}) = \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_0 \mathbf{x}$. Then*

$$\sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{X}} h_T(\mathbf{x}) \leq \sum_{t=1}^{T} [\tilde{f}_t(\mathbf{x}_t) - \tilde{f}_t(\mathbf{x}_{t+1})] \tag{3.44}$$

The next theorem bounds the difference of differentiable univariate convex functions that leads to a logarithmic regret bound. The lower bounding proxy function resulting from Lemma 6 is formulated as a differentiable univariate convex function.

Theorem 5. *Assume for all $t \in \{1, \ldots, T\}$ the function $\tilde{f}_t : \mathcal{X} \to \mathbb{R}$ can be written as $\tilde{f}_t(\mathbf{x}) = g_t(\mathbf{u}_t^\top \mathbf{x})$ for a differentiable univariate convex function $g_t : \mathbb{R} \to \mathbb{R}$ and some vector $\mathbf{u}_t \in \mathbb{R}^n$. Assume that for some $a, b, \epsilon > 0$ some $c_t > 0$ where $t \in \{1, \ldots, T\}$ we have $||\mathbf{v}_t||_2 \leq c_t$ where $\mathbf{v}_t = \sqrt{g_t''(\mathbf{u}_t^\top \mathbf{x})}\mathbf{u}_t$. Assume for all $\mathbf{x}_t \in \mathcal{X}$ we have $|g_t'(\mathbf{u}_t^\top \mathbf{x}_t)| \leq b$ and for all $\mathbf{x} \in \mathcal{X}$ we have $|g_t'(\mathbf{u}_t^\top \mathbf{x})| \leq b$ and $g_t''(\mathbf{u}_t^\top \mathbf{x}) = \alpha_t^2 \geq a$. Then Algorithm 7 satisfies the follow bound:*

$$\sum_{t=1}^{T} [\tilde{f}_t(\mathbf{x}_t) - \tilde{f}_t(\mathbf{x}_{t+1})] \leq \frac{nb^2}{a} \log(\frac{1}{\epsilon} \sum_{t=1}^{T} c_t^2 + 1) \tag{3.45}$$

Finally, we bring these results together to obtain the final regret bound.

Theorem 6. *If $f_t : \mathcal{X} \to \mathbb{R}$ is a H_t-strongly convex function with respect to $||\cdot||_p$ for all $t \in \{1, \ldots, T\}$ where $p \in \{1, 2\}$, $H_t > 0$ and $H_{\min} = \min_t\{H_t\}$ subject to a convex set $\mathcal{X} \subseteq \mathbb{R}^n$ then Algorithm $\mathcal{A}_7$ has the following regret bound*

$$R_T(\mathcal{A}_7) \leq \frac{nL^2}{H_{\min}} \log(\frac{1}{\epsilon} \sum_{t=1}^{T} H_t + 1) + \frac{1}{2}\epsilon D^2 \tag{3.46}$$

The proofs of the previously states lemmas and theorems are as follows.

Proof (Lemma 7)**.** Let $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T} f_t(\mathbf{x})$ then

$$\begin{aligned} \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^*) &\leq \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_t) - \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}^*) - \frac{1}{2}\mathbf{x}^{*\top}\mathbf{Q}_0\mathbf{x}^* + \frac{1}{2}\mathbf{x}^{*\top}\mathbf{Q}_0\mathbf{x}^* & (3.47) \\ &\leq \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{X}} h_T(\mathbf{x}) + \frac{1}{2}\mathbf{x}^{*\top}\mathbf{Q}_0\mathbf{x}^* & (3.48) \end{aligned}$$

where $h_T(\mathbf{x}) = \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_0 \mathbf{x}$

□

Proof (Lemma 8)**.** We prove inductively that

$$\sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_{t+1}) \leq \min_{\mathbf{x}\in\mathcal{X}} h_T(\mathbf{x}) \tag{3.49}$$

By definition

$$\tilde{f}_1(\mathbf{x}_2) + \frac{1}{2}\mathbf{x}_2{}^\top \mathbf{Q}_0 \mathbf{x}_2 = \min_{\mathbf{x}\in\mathcal{X}} \tilde{f}_1(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_0 \mathbf{x} \tag{3.50}$$

given $\mathbf{Q}_0 \succ 0$ then

$$\tilde{f}_1(\mathbf{x}_2) \leq \min_{\mathbf{x}\in\mathcal{X}} \tilde{f}_1(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_0 \mathbf{x} \tag{3.51}$$

proving (3.49) for $T = 1$. Assuming correctness for $T - 1$ then

$$\begin{aligned}
\sum_{t=1}^{T-1} \tilde{f}_t(\mathbf{x}_{t+1}) &\leq \min_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T-1} \tilde{f}_t(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_0 \mathbf{x} && (3.52)\\
\sum_{t=1}^{T-1} \tilde{f}_t(\mathbf{x}_{t+1}) + \tilde{f}_T(\mathbf{x}_{T+1}) &\leq \min_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T-1} \tilde{f}_t(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_0 \mathbf{x} + \tilde{f}_T(\mathbf{x}_{T+1}) && (3.53)\\
\sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_{t+1}) &\leq \sum_{t=1}^{T-1} \tilde{f}_t(\mathbf{x}_{T+1}) + \frac{1}{2}\mathbf{x}_{T+1}{}^\top \mathbf{Q}_0 \mathbf{x}_{T+1} + \tilde{f}_T(\mathbf{x}_{T+1}) && (3.54)\\
&= \min_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_0 \mathbf{x} && (3.55)\\
&= \min_{\mathbf{x}\in\mathcal{X}} h_T(\mathbf{x}) && (3.56)
\end{aligned}$$

by definition. Completing the proof by induction. The result follows.

□

Proof (Theorem 5)**.** Let $h_t(\mathbf{x}) \equiv \sum_{i=1}^{t} \tilde{f}_i(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_0 \mathbf{x}$ where $\mathbf{Q}_0 = \mathbf{I}_n \epsilon$ and $\epsilon > 0$. Let Δ be the forward difference operator such that $\Delta \mathbf{x}_t \equiv (\mathbf{x}_{t+1} - \mathbf{x}_t)$ and $\Delta \nabla h_{t-1}(\mathbf{x}_t) \equiv \nabla h_t(\mathbf{x}_{t+1}) - \nabla h_{t-1}(\mathbf{x}_t)$. Note that $\nabla \tilde{f}_t(\mathbf{x}) = g_t'(\mathbf{u}_t^\top \mathbf{x})\mathbf{u}_t$ and let $\nabla_t \equiv \nabla \tilde{f}_t(\mathbf{x}_t)$. Consider now that

$$\begin{aligned}
\nabla h_t(\mathbf{x}_{t+1}) - \nabla h_t(\mathbf{x}_t) &= \sum_{i=1}^{t} \nabla \tilde{f}_i(\mathbf{x}_{t+1}) - \nabla \tilde{f}_i(\mathbf{x}_t) + \mathbf{Q}_0(\mathbf{x}_{t+1} - \mathbf{x}_t) && (3.57)\\
&= \sum_{i=1}^{t} [g_i'(\mathbf{u}_i^\top \mathbf{x}_{t+1}) - g_i'(\mathbf{u}_i^\top \mathbf{x}_t)]\mathbf{u}_i + \mathbf{Q}_0(\mathbf{x}_{t+1} - \mathbf{x}_t) && (3.58)
\end{aligned}$$

by applying a Taylor expansion of the function $g_i'(\mathbf{u}_i^\top \mathbf{x})$ at point $\mathbf{x}_t$ for some $\boldsymbol{\zeta}_t^i$ on the line segment between $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ such that

$$g_i'(\mathbf{u}_i^\top \mathbf{x}_{t+1}) = g_i'(\mathbf{u}_i^\top \mathbf{x}_t) + \nabla g_i'(\mathbf{u}_i^\top \boldsymbol{\zeta}_t^i)^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) \tag{3.59}$$

then

$$\begin{aligned}
\nabla h_t(\mathbf{x}_{t+1}) - \nabla h_t(\mathbf{x}_t) &= \sum_{i=1}^{t} \mathbf{u}_i [\nabla g_i'(\mathbf{u}_i^\top \boldsymbol{\zeta}_t^i)^\top (\mathbf{x}_{t+1} - \mathbf{x}_t)] + \mathbf{Q}_0(\mathbf{x}_{t+1} - \mathbf{x}_t) && (3.60)\\
&= \sum_{i=1}^{t} g_i''(\mathbf{u}_i^\top \boldsymbol{\zeta}_t^i) \mathbf{u}_i \mathbf{u}_i^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) + \mathbf{Q}_0(\mathbf{x}_{t+1} - \mathbf{x}_t) && (3.61)\\
&= (\sum_{i=1}^{t} g_i''(\mathbf{u}_i^\top \boldsymbol{\zeta}_t^i) \mathbf{u}_i \mathbf{u}_i^\top + \mathbf{Q}_0)(\mathbf{x}_{t+1} - \mathbf{x}_t) && (3.62)
\end{aligned}$$

Define $\mathbf{Q}_t \equiv \sum_{i=1}^{t} \alpha_i^2 \mathbf{u}_i \mathbf{u}_i^\top + \mathbf{Q}_0$ where $g_i''(\mathbf{u}_i^\top \boldsymbol{\zeta}_t^i) = \alpha_i^2$ and therefore

$$\nabla h_t(\mathbf{x}_{t+1}) - \nabla h_t(\mathbf{x}_t) = \mathbf{Q}_t \Delta \mathbf{x}_t \tag{3.63}$$

Given that $\alpha_i^2 \geq a$ and $\mathbf{Q}_0$ is positive definite then $\mathbf{Q}_t$ is positive definite. Now consider that

$$\begin{aligned}
\nabla h_t(\mathbf{x}_{t+1}) - \nabla h_t(\mathbf{x}_t) &= \nabla h_t(\mathbf{x}_{t+1}) - \sum_{i=1}^{t-1} \nabla \tilde{f}_i(\mathbf{x}_t) - \mathbf{Q}_0 \mathbf{x}_t - \nabla \tilde{f}_t(\mathbf{x}_t) && (3.64)\\
&= \nabla h_t(\mathbf{x}_{t+1}) - \nabla h_{t-1}(\mathbf{x}_t) - \nabla_t && (3.65)\\
&= \Delta \nabla h_{t-1}(\mathbf{x}_t) - \nabla_t && (3.66)
\end{aligned}$$

Now combining (3.66) and (3.63) we obtain

$$\mathbf{Q}_t \Delta \mathbf{x}_t = \Delta \nabla h_{t-1}(\mathbf{x}_t) - \nabla_t \tag{3.67}$$

multiplying on the left by $-\nabla_t^\top \mathbf{Q}_t^{-1}$ we get

$$-\nabla_t^\top \Delta \mathbf{x}_t = -\nabla_t^\top \mathbf{Q}_t^{-1} \Delta \nabla h_{t-1}(\mathbf{x}_t) + \nabla_t^\top \mathbf{Q}_t^{-1} \nabla_t \tag{3.68}$$

Now using the convexity of $\tilde{f}_t$ we obtain

$$\begin{aligned}
\tilde{f}_t(\mathbf{x}_t) - \tilde{f}_t(\mathbf{x}_{t+1}) &\leq -\nabla \tilde{f}_t(\mathbf{x}_t)^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) && (3.69)\\
&= -\nabla_t^\top \Delta \mathbf{x}_t && (3.70)
\end{aligned}$$

and therefore

$$\tilde{f}_t(\mathbf{x}_t) - \tilde{f}_t(\mathbf{x}_{t+1}) \leq -\nabla_t^\top \mathbf{Q}_t^{-1} \Delta \nabla h_{t-1}(\mathbf{x}_t) + \nabla_t^\top \mathbf{Q}_t^{-1} \nabla_t \tag{3.71}$$

We will now show that

$$-\nabla_t^\top \mathbf{Q}_t^{-1} \Delta \nabla h_{t-1}(\mathbf{x}_t) \leq 0 \tag{3.72}$$

Since $\mathbf{x}_i = \arg\min_{\mathbf{x}\in\mathcal{X}} h_{i-1}(\mathbf{x})$ we have

$$\nabla h_{i-1}(\mathbf{x}_i)^\top (\mathbf{x} - \mathbf{x}_i) \geq 0 \tag{3.73}$$

for any $\mathbf{x} \in \mathcal{X}$ using convexity and Lemma 16. Therefore

$$\begin{aligned}
0 &\leq \nabla h_t(\mathbf{x}_{t+1})^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) + \nabla h_{t-1}(\mathbf{x}_t)^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) && (3.74)\\
&= -\nabla h_t(\mathbf{x}_{t+1})^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) + \nabla h_{t-1}(\mathbf{x}_t)^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) && (3.75)\\
&= -[\nabla h_t(\mathbf{x}_{t+1}) - \nabla h_{t-1}(\mathbf{x}_t)]^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) && (3.76)\\
&= -[\Delta \nabla h_{t-1}(\mathbf{x}_t)]^\top \Delta \mathbf{x}_t && (3.77)
\end{aligned}$$

Using (3.67) to solve for $\Delta \mathbf{x}_t$ we obtain

$$\Delta \mathbf{x}_t = \mathbf{Q}_t^{-1}(\Delta \nabla h_{t-1}(\mathbf{x}_t) - \nabla_t) \tag{3.78}$$

and then

$$\begin{aligned}
0 &\leq -[\Delta \nabla h_{t-1}(\mathbf{x}_t)]^\top \mathbf{Q}_t^{-1}([\Delta \nabla h_{t-1}(\mathbf{x}_t)] - \nabla_t) && (3.79)\\
&= -[\Delta \nabla h_{t-1}(\mathbf{x}_t)]^\top \mathbf{Q}_t^{-1}[\Delta \nabla h_{t-1}(\mathbf{x}_t)] + [\Delta \nabla h_{t-1}(\mathbf{x}_t)]^\top \mathbf{Q}_t^{-1} \nabla_t && (3.80)
\end{aligned}$$

rearranging terms and given that $\mathbf{Q}_t^{-1}$ is positive definite we obtain

$$\begin{aligned}
-\nabla_t^\top \mathbf{Q}_t^{-1}[\Delta \nabla h_{t-1}(\mathbf{x}_t)] &\leq -[\Delta \nabla h_{t-1}(\mathbf{x}_t)]^\top \mathbf{Q}_t^{-1}[\Delta \nabla h_{t-1}(\mathbf{x}_t)] && (3.81)\\
&\leq 0 && (3.82)
\end{aligned}$$

This results in

$$\sum_{t=1}^{T} [\tilde{f}_t(\mathbf{x}_t) - \tilde{f}_t(\mathbf{x}_{t+1})] \leq \sum_{t=1}^{T} \nabla_t^\top \mathbf{Q}_t^{-1} \nabla_t \tag{3.83}$$

Given $\frac{\alpha_t^2}{a} \geq 1$ and $\nabla_t = g_t'(\mathbf{u}_t^\top \mathbf{x}_t)\mathbf{u}_t$ then

$$\begin{aligned}
\sum_{t=1}^{T} \nabla_t^\top \mathbf{Q}_t^{-1} \nabla_t &\leq \sum_{t=1}^{T} \frac{\alpha_t^2}{a} [g_t'(\mathbf{u}_t^\top \mathbf{x}_t)\mathbf{u}_t]^\top \mathbf{Q}_t^{-1} [g_t'(\mathbf{u}_t^\top \mathbf{x}_t)\mathbf{u}_t] && (3.84)\\
&= \sum_{t=1}^{T} \frac{g_t'(\mathbf{u}_t^\top \mathbf{x}_t)^2}{a} [\alpha_t \mathbf{u}_t]^\top \mathbf{Q}_t^{-1} [\alpha_t \mathbf{u}_t] && (3.85)
\end{aligned}$$

Let $\mathbf{v}_t \equiv \alpha_t \mathbf{u}_t$ and $|g_t'(\mathbf{u}_t^\top \mathbf{x}_t)| \leq b$ then

$$\sum_{t=1}^{T} \nabla_t^\top \mathbf{Q}_t^{-1} \nabla_t \;\; \leq \;\; \frac{b^2}{a} \sum_{t=1}^{T} \mathbf{v}_t^\top \mathbf{Q}_t^{-1} \mathbf{v}_t \tag{3.86}$$

where $\mathbf{Q}_t = \sum_{i=1}^{t} \mathbf{v}_i \mathbf{v}_i^\top + \mathbf{Q}_0$. Using Lemma 18 and $||\mathbf{v}_t||_2 \leq c_t$ then

$$\sum_{t=1}^{T} \mathbf{v}_t^\top \mathbf{Q}_t^{-1} \mathbf{v}_t \leq n \log(\frac{1}{\epsilon} \sum_{t=1}^{T} c_t^2 + 1) \tag{3.87}$$

and therefore

$$\sum_{t=1}^{T} [\tilde{f}_t(\mathbf{x}_t) - \tilde{f}_t(\mathbf{x}_{t+1})] \leq \frac{nb^2}{a} \log(\frac{1}{\epsilon} \sum_{t=1}^{T} c_t^2 + 1) \tag{3.88}$$

□

Proof (Theorem 6)**.** Using the H_t-strongly convex assumption and Lemma 6 we have $f_t(\mathbf{x}_t) = \tilde{f}_t(\mathbf{x}_t)$ and $f_t(\mathbf{x}) \geq \tilde{f}_t(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$ where

$$\tilde{f}(\mathbf{x}) \equiv f_t(\mathbf{x}_t) + \mathbf{y}_t^\top (\mathbf{x} - \mathbf{x}_t) + \frac{\alpha_t^2}{2} [\mathbf{y}_t^\top (\mathbf{x} - \mathbf{x}_t)]^2 \tag{3.89}$$

$\alpha_t = \frac{\sqrt{H_t}}{||\mathbf{y}_t||_2}$ and $\mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$. Given this, and using Lemma 7 we have the bound

$$\sum_{t=1}^{T} [f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] \leq \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_t) - \min_{\mathbf{x} \in \mathcal{X}} h_T(\mathbf{x}) + \frac{1}{2} \mathbf{x}^{*\top} \mathbf{Q}_0 \mathbf{x}^* \tag{3.90}$$

where $h_T(\mathbf{x}) = \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}) + \frac{1}{2} \mathbf{x}^\top \mathbf{Q}_0 \mathbf{x}$. Adding $\mathbf{x}^{*\top} \mathbf{Q}_0 \mathbf{x}^*$ to the result of Lemma 8 then

$$\sum_{t=1}^{T} [f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] \leq \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_t) - \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_{t+1}) + \frac{1}{2} \mathbf{x}^{*\top} \mathbf{Q}_0 \mathbf{x}^* \tag{3.91}$$

Without loss of generality we assume $0 \in \mathcal{X}$ and then using Assumption 1(b) we have $\frac{1}{2} \mathbf{x}^{*\top} \mathbf{Q}_0 \mathbf{x}^* \leq \frac{1}{2} \epsilon D^2$ giving

$$\sum_{t=1}^{T} [f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] \leq \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_t) - \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_{t+1}) + \frac{1}{2} \epsilon D^2 \tag{3.92}$$

Now to satisfy the conditions of Theorem 5 consider $\tilde{f}_t(\mathbf{x}) = g_t(\mathbf{y}_t^\top \mathbf{x})$ where

$$g_t(s) \equiv f_t(\mathbf{x}_t) + (s - \mathbf{y}_t^\top \mathbf{x}_t) + \frac{\alpha_t^2}{2} [s - \mathbf{y}_t^\top \mathbf{x}_t]^2 \tag{3.93}$$

and therefore $a = \beta$ since $g_t''(s) = \alpha_t^2 \geq \beta$ by Lemma 6. Then

$$\begin{aligned} ||\mathbf{v}_t||_2 &= ||\sqrt{g_t''(\mathbf{u}_t^\top \mathbf{x})}\mathbf{u}_t||_2 && (3.94) \\ &= ||\alpha_t \mathbf{y}_t||_2 && (3.95) \\ &= \sqrt{H_t} && (3.96) \end{aligned}$$

and therefore $c_t = \sqrt{H_t}$. Then finally $|g_t'(\mathbf{y}_t^\top \mathbf{x}_t)| = |1 + \alpha_t^2 \mathbf{y}_t^\top (\mathbf{x}_t - \mathbf{x}_t)| = 1$ and therefore $b = 1$. Using these parameters, Theorem 5 , equation (3.92) and $\beta = \frac{H_{\min}}{L^2}$ we have our result

$$R_T(\mathcal{A}_7) \leq \frac{nL^2}{H_{\min}} \log(\frac{1}{\epsilon} \sum_{t=1}^{T} H_t + 1) + \frac{1}{2}\epsilon D^2 \tag{3.97}$$

□

Chapter 4

New Block Diagonal Quasi-Newton Algorithms

The algorithms analyzed in Chapter 3 have an update step that can be implemented in $\mathcal{O}(n^2)$ space and time. This results from the need to store and calculate an approximate hessian matrix with n^2 elements. In this chapter we develop and analyze an algorithmic extension to a block diagonal, or fully diagonal, hessian approximation. This results in a new algorithm family where complexity can be managed while still obtaining the same theoretical bounds. The update step for a fully diagonal approximation requires $\mathcal{O}(n)$ space and time and therefore matches the update complexity of Online Gradient Descent.

A similar known algorithm uses a mirror descent framework and applies to strongly convex functions with respect to $||\cdot||_p$ where $p \geq 2$ [23]. Our new algorithms use a Follow the Leader framework, apply to $p \leq 2$ and does not rely on the knowledge of parameters.

In Section 4.3 we present how quasi-Newton algorithms can exploit problem structure to obtain a parallelized algorithm. In the constrained case, the projection step of our algorithms require a quadratic program to be solved. However, with a fully diagonal hessian approximation, all hessian eignvalues of the quadratic program are known by inspection. Algorithms that exploit this structure can be used to solve the quadratic program [2].

We adopt the following notation to help present block diagonal matrices. A vector $\mathbf{v}_t \in \mathcal{X}$ and be decomposed into $R \in \{1, \ldots, n\}$ sub-vectors $\mathbf{v}_t^r \in \mathbb{R}^m$ where $r \in \{1, \ldots, R\}$ and $n = R \times m$ such that $\mathbf{v}_t = [{\mathbf{v}_t^1}^\top, \ldots, {\mathbf{v}_t^R}^\top]^\top$.

4.1 Block Diagonal ONS for Strongly Convex Functions

We now present a block diagonal Online Newton Step (ONS) algorithm for strongly convex functions and the resulting regret bound. Similar to the algorithms developed in Chapter 3, this algorithm does not depend on the parameters D or L as defined by Assumption 1.

The algorithm achieves the same theoretical guarantee but allows for block diagonal hessian approximations. The choice of the initialization parameter ϵ, and the resulting bound, is addressed in Section 3.1.2.

Algorithm 8 Block Diagonal Matrix ONS for Strongly Convex Functions

Require: $\mathbf{x}_1 \in \mathcal{X} \subseteq \mathbb{R}^n$, $\{H_t\}_{t=1}^T$, $\mathbf{Q}_0 = \mathbf{I}_n\epsilon$, $\epsilon > 0$ and $R \in \{i | i \in \{1, \ldots, n\}, n = i \times m\}$

1: **for** $t = 1$ to T **do**

2: $\quad \mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$

3: $\quad \mathbf{v}_t = \frac{\sqrt{H_t}}{||\mathbf{y}_t||_2}\mathbf{y}_t$

4: $\quad \mathbf{Q}_t = \begin{bmatrix} \mathbf{v}_t^1\mathbf{v}_t^{1\top} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{v}_t^R\mathbf{v}_t^{R\top} \end{bmatrix} + \mathbf{Q}_{t-1}$

5: $\quad \mathbf{z}_{t+1} = \mathbf{x}_t - \mathbf{Q}_t^{-1}\mathbf{y}_t$

6: $\quad \mathbf{x}_{t+1} = \Pi_{\mathcal{X}}^{\mathbf{Q}_t}(\mathbf{z}_{t+1}) = \arg\min_{\mathbf{x}\in\mathcal{X}}(\mathbf{z}_{t+1} - \mathbf{x})^\top \mathbf{Q}_t(\mathbf{z}_{t+1} - \mathbf{x})$

7: **end for**

4.1.1 Analysis

Lemma 9 presents a block diagonal version of Lemma 6. This lemma exploits strong convexity to obtain a lower bound on the objective function.

Lemma 9. *Given a H_t-strongly convex function with respect to $||\cdot||_p$ where $p \in \{1, 2\}$ then for all $\mathbf{z}, \mathbf{x}_t \in \mathcal{X}$ and any $\mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$ and some R and m such that $n = R \times m$ where $||\mathbf{y}_t||_2 \leq L$ we have*

$$f_t(\mathbf{z}) \geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z} - \mathbf{x}_t) + \frac{1}{2}(\mathbf{z} - \mathbf{x}_t)^\top \mathbf{D}_{t,R}(\mathbf{z} - \mathbf{x}_t) \tag{4.1}$$

where

$$\mathbf{D}_{t,R} \equiv \begin{bmatrix} \mathbf{v}_t^1\mathbf{v}_t^{1\top} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{v}_t^R\mathbf{v}_t^{R\top} \end{bmatrix} \tag{4.2}$$

and equivalently

$$f_t(\mathbf{z}) \geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z} - \mathbf{x}_t) + \frac{1}{2}\sum_{r=1}^{R}[\mathbf{v}_t^{r\top}(\mathbf{z}^r - \mathbf{x}_t^r)]^2 \tag{4.3}$$

where $\mathbf{v}_t^r \in \mathbb{R}^m$ for all $r \in \{1, \ldots, R\}$ and $\mathbf{v}_t = [\mathbf{v}_t^{1\top}, \ldots, \mathbf{v}_t^{R\top}]^\top$, $\mathbf{v}_t^r = \alpha_t\mathbf{y}_t^r$ and $\alpha_t = \frac{\sqrt{H_t}}{||\mathbf{y}_t||_2}$. Also, given $\beta = \frac{H_{\min}}{L^2}$ where $H_{\min} = \min_t H_t$ then $\frac{\alpha_t^2}{\beta} \geq 1$.

Next we present the main theorem using Lemma 9 and Lemmas 18 and 19 of Appendix B. The regret bound is presented with an unknown $\epsilon > 0$ and the choice of ϵ is addressed in Section 3.1.2. As with the algorithms presented in Chapter 3 the strongly convex constant $H > 0$ is allowed to change at each iteration.

Theorem 7. *If $f_t : \mathcal{X} \to \mathbb{R}$ is a H_t-strongly convex function with respect to $||\cdot||_p$ for all $t \in \{1, \ldots, T\}$ where $p \in \{1, 2\}$, $H_t > 0$, $H_{\min} = \min_t H_t$ with some R and m such that $n = R \times m$ subject to the convex set $\mathcal{X} \subseteq \mathbb{R}^n$ then Algorithm $\mathcal{A}_8$ has the following regret bound*

$$R_T(\mathcal{A}_8) \leq \frac{nL^2}{2H_{\min}} \log(\frac{1}{\epsilon}\sum_{t=1}^{T} H_t + 1) + \frac{\epsilon D^2}{2} \tag{4.4}$$

where $\epsilon > 0$ is an initialization parameter.

The proofs for the above results are as follows.

Proof (Lemma 9)**.** Consider Theorem 7.7.3 of Horn and Johnson[42]. Given $\mathbf{A} \in \mathbb{S}^n_{++}$ and $\mathbf{B} \in \mathbb{S}^n_{+}$ then $\mathbf{A} \succeq \mathbf{B}$ if and only if $\lambda_{\max}(\mathbf{B}\mathbf{A}^{-1}) \leq 1$. Let $\mathbf{A} = \mathbf{I}_n$ and let $\mathbf{B} = \frac{1}{H_t}\mathbf{D}_{t,R}$ for any $R \in \{1, \ldots, n\}$ where

$$\mathbf{D}_{t,R} \equiv \begin{bmatrix} \mathbf{v}_t^1 \mathbf{v}_t^{1\top} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{v}_t^R \mathbf{v}_t^{R\top} \end{bmatrix} \tag{4.5}$$

Therefore if $\lambda_{\max}(\mathbf{B}) \leq 1$ then $\mathbf{I}_n \succeq \mathbf{B}$.

$$\begin{aligned} \text{trace}(\mathbf{B}) &= \frac{1}{||\mathbf{y}_t||_2^2}\sum_{i=1}^{n}(y_t(i))^2 && (4.6)\\ &= \frac{1}{||\mathbf{y}_t||_2^2}||\mathbf{y}_t||_2^2 && (4.7)\\ &= 1 && (4.8) \end{aligned}$$

We have $\text{trace}(\mathbf{B}) = \sum_{i=1}^n \lambda_i = 1$ and given $\mathbf{B} \succeq 0$ then $\lambda_i \geq 0$ for all $i \in \{1, \ldots, n\}$. Therefore $\lambda_i \leq 1$ for all $i \in \{1, \ldots, n\}$ and therefore $\lambda_{max}(\mathbf{B}) \leq 1$. This implies that $H_t\mathbf{I}_n \succeq \mathbf{D}_{t,R}$. Now, using strong convexity and Lemma 2 and given $||\cdot||_1^2 \geq ||\cdot||_2^2$, we have for some $p \in \{1, 2\}$ that

$$\begin{aligned} f_t(\mathbf{z}) &\geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z} - \mathbf{x}_t) + \frac{H_t}{2}||\mathbf{z} - \mathbf{x}_t||_p^2 && (4.9)\\ &\geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z} - \mathbf{x}_t) + \frac{H_t}{2}||\mathbf{z} - \mathbf{x}_t||_2^2 && (4.10) \end{aligned}$$

Now consider

$$f_t(\mathbf{z}) \geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z}-\mathbf{x}_t) + \frac{1}{2}(\mathbf{z}-\mathbf{x}_t)^\top H_t \mathbf{I}_n(\mathbf{z}-\mathbf{x}_t) \tag{4.11}$$

$$\geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z}-\mathbf{x}_t) + \frac{1}{2}(\mathbf{z}-\mathbf{x}_t)^\top \mathbf{D}_{t,R}(\mathbf{z}-\mathbf{x}_t) \tag{4.12}$$

then

$$f_t(\mathbf{z}) \geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z}-\mathbf{x}_t) + \frac{1}{2}\sum_{r=1}^{R}[\mathbf{v}_t^{r\top}(\mathbf{z}^r-\mathbf{x}_t^r)]^2 \tag{4.13}$$

and $\mathbf{v}_t^r = \alpha_t \mathbf{y}_t^r$ for all $r \in \{1,\ldots,R\}$ and $\alpha_t = \frac{\sqrt{H_t}}{||\mathbf{y}_t||_2}$. Now consider that $\alpha_t^2 = \frac{H_t}{||\mathbf{y}_t||_2^2}$ and $\beta = \frac{H_{\min}}{L^2}$ we then have

$$\alpha_t^2 = \frac{H_t}{||\mathbf{y}_t||_2^2} \tag{4.14}$$

$$\geq \frac{H_{\min}}{L^2} \tag{4.15}$$

$$= \beta \tag{4.16}$$

and therefore $\frac{\alpha_t^2}{\beta} \geq 1$.

□

Proof (Theorem 7)**.** Starting with the definition of $\mathbf{z}_{t+1}$ we have

$$\mathbf{z}_{t+1} - \mathbf{x}^* = \mathbf{x}_t - \mathbf{x}^* - \mathbf{Q}_t^{-1}\mathbf{y}_t \tag{4.17}$$

$$\mathbf{Q}_t(\mathbf{z}_{t+1} - \mathbf{x}^*) = \mathbf{Q}_t(\mathbf{x}_t - \mathbf{x}^*) - \mathbf{y}_t \tag{4.18}$$

multiplying on the left by (4.17) we obtain

$$(\mathbf{z}_{t+1}-\mathbf{x}^*)^\top\mathbf{Q}_t(\mathbf{z}_{t+1}-\mathbf{x}^*) = (\mathbf{x}_t-\mathbf{x}^*-\mathbf{Q}_t^{-1}\mathbf{y}_t)^\top\mathbf{Q}_t(\mathbf{x}_t-\mathbf{x}^*) - (\mathbf{x}_t-\mathbf{x}^*-\mathbf{Q}_t^{-1}\mathbf{y}_t)^\top\mathbf{y}_t \tag{4.19}$$

$$(\mathbf{z}_{t+1}-\mathbf{x}^*)^\top\mathbf{Q}_t(\mathbf{z}_{t+1}-\mathbf{x}^*) = (\mathbf{x}_t-\mathbf{x}^*)^\top\mathbf{Q}_t(\mathbf{x}_t-\mathbf{x}^*) - 2\mathbf{y}_t^\top(\mathbf{x}_t-\mathbf{x}^*) + \mathbf{y}_t^\top\mathbf{Q}_t^{-1}\mathbf{y}_t \tag{4.20}$$

rearranging terms we have

$$2\mathbf{y}_t^\top(\mathbf{x}_t-\mathbf{x}^*) = ||\mathbf{x}_t-\mathbf{x}^*||_{\mathbf{Q}_t}^2 - ||\mathbf{z}_{t+1}-\mathbf{x}^*||_{\mathbf{Q}_t}^2 + \mathbf{y}_t^\top\mathbf{Q}_t^{-1}\mathbf{y}_t \tag{4.21}$$

Given $\mathbf{Q}_t^{-1}$ is positive definite then using $\frac{\alpha_t^2}{\beta} \geq 1$ of Lemma 9 and let $\mathbf{v}_t \equiv \alpha_t\mathbf{y}_t$ we have

$$2\mathbf{y}_t^\top(\mathbf{x}_t-\mathbf{x}^*) \leq ||\mathbf{x}_t-\mathbf{x}^*||_{\mathbf{Q}_t}^2 - ||\mathbf{z}_{t+1}-\mathbf{x}^*||_{\mathbf{Q}_t}^2 + \frac{\alpha_t^2}{\beta}\mathbf{y}_t^\top\mathbf{Q}_t^{-1}\mathbf{y}_t \tag{4.22}$$

$$= ||\mathbf{x}_t-\mathbf{x}^*||_{\mathbf{Q}_t}^2 - ||\mathbf{z}_{t+1}-\mathbf{x}^*||_{\mathbf{Q}_t}^2 + \frac{1}{\beta}\mathbf{v}_t^\top\mathbf{Q}_t^{-1}\mathbf{v}_t \tag{4.23}$$

Using Lemma 19 we have

$$||\mathbf{z}_{t+1} - \mathbf{x}^*||^2_{\mathbf{Q}_t} \geq ||\mathbf{x}_{t+1} - \mathbf{x}^*||^2_{\mathbf{Q}_t} \tag{4.24}$$

and then

$$2\mathbf{y}_t^\top(\mathbf{x}_t - \mathbf{x}^*) \leq ||\mathbf{x}_t - \mathbf{x}^*||^2_{\mathbf{Q}_t} - ||\mathbf{x}_{t+1} - \mathbf{x}^*||^2_{\mathbf{Q}_t} + \frac{1}{\beta}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \tag{4.25}$$

summing together all T inequalities

$$\begin{aligned} 2\sum_{t=1}^{T}\mathbf{y}_t^\top(\mathbf{x}_t - \mathbf{x}^*) \;\leq\; & ||\mathbf{x}_1 - \mathbf{x}^*||^2_{\mathbf{Q}_1} + \sum_{t=2}^{T}(\mathbf{x}_t - \mathbf{x}^*)^\top(\mathbf{Q}_t - \mathbf{Q}_{t-1})(\mathbf{x}_t - \mathbf{x}^*) - \\ & ||\mathbf{x}_{T+1} - \mathbf{x}^*||^2_{\mathbf{Q}_T} + \frac{1}{\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \end{aligned} \tag{4.26}$$

As defined by Lemma 9 we have $\mathbf{Q}_t - \mathbf{Q}_{t-1} = \mathbf{D}_{t,R}$ and then

$$\begin{aligned} 2\sum_{t=1}^{T}\mathbf{y}_t^\top(\mathbf{x}_t - \mathbf{x}^*) \;\leq\; & (\mathbf{x}_1 - \mathbf{x}^*)^\top[\mathbf{Q}_1 - \mathbf{D}_{1,R}](\mathbf{x}_1 - \mathbf{x}^*) + \sum_{t=1}^{T}(\mathbf{x}_t - \mathbf{x}^*)^\top\mathbf{D}_{t,R}(\mathbf{x}_t - \mathbf{x}^*) - \\ & ||\mathbf{x}_{T+1} - \mathbf{x}^*||^2_{\mathbf{Q}_T} + \frac{1}{\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \end{aligned} \tag{4.27}$$

$$\begin{aligned} 2\sum_{t=1}^{T}\mathbf{y}_t^\top(\mathbf{x}_t - \mathbf{x}^*) \;\leq\; & (\mathbf{x}_1 - \mathbf{x}^*)^\top\mathbf{Q}_0(\mathbf{x}_1 - \mathbf{x}^*) + \sum_{t=1}^{T}(\mathbf{x}_t - \mathbf{x}^*)^\top\mathbf{D}_{t,R}(\mathbf{x}_t - \mathbf{x}^*) - \\ & ||\mathbf{x}_{T+1} - \mathbf{x}^*||^2_{\mathbf{Q}_T} + \frac{1}{\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \end{aligned} \tag{4.28}$$

given Assumption 1(b) , $\mathbf{Q}_0 = \mathbf{I}_n\epsilon$ and dropping negative terms we obtain

$$2\sum_{t=1}^{T}\mathbf{y}_t^\top(\mathbf{x}_t - \mathbf{x}^*) \leq \epsilon D^2 + \sum_{t=1}^{T}||\mathbf{x}_t - \mathbf{x}^*||_{\mathbf{D}_{t,R}} + \frac{1}{\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \tag{4.29}$$

using the lower bound of Lemma 9 then

$$\begin{aligned} \sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] \;\leq\; & \frac{\epsilon D^2}{2} + \frac{1}{2}\sum_{t=1}^{T}||\mathbf{x}_t - \mathbf{x}^*||_{\mathbf{D}_{t,R}} - \frac{1}{2}\sum_{t=1}^{T}||\mathbf{x}_t - \mathbf{x}^*||_{\mathbf{D}_{t,R}} + \\ & \frac{1}{2\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t \end{aligned} \tag{4.30}$$

$$\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] \;\leq\; \frac{1}{2\beta}\sum_{t=1}^{T}\mathbf{v}_t^\top \mathbf{Q}_t^{-1}\mathbf{v}_t + \frac{\epsilon D^2}{2} \tag{4.31}$$

let

$$\mathbf{Q}_t \equiv \begin{bmatrix} \mathbf{Q}_t^1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{Q}_t^R \end{bmatrix} \tag{4.32}$$

where for all $r \in \{1, \ldots, R\}$

$$\mathbf{Q}_t^r = \sum_{i=1}^{t} \mathbf{v}_i^r \mathbf{v}_i^{r\top} + \epsilon \mathbf{I}_m$$

then

$$\sum_{t=1}^{T} \mathbf{v}_t^\top \mathbf{Q}_t^{-1} \mathbf{v}_t = \sum_{t=1}^{T} \sum_{r=1}^{R} \mathbf{v}_t^{r\top} (\mathbf{Q}_t^r)^{-1} \mathbf{v}_t^r \tag{4.33}$$

using Lemma 18 and $||\mathbf{v}_t^r|| \leq ||\mathbf{v}_t||_2 \leq \sqrt{H_t}$ then for each $r \in \{1, \ldots, R\}$

$$\sum_{t=1}^{T} \mathbf{v}_t^{r\top} (\mathbf{Q}_t^r)^{-1} \mathbf{v}_t^r \leq m \log(\frac{1}{\epsilon} \sum_{t=1}^{T} H_t + 1) \tag{4.34}$$

finally given $\beta = \frac{H_{\min}}{L^2}$ we have the result

$$R_T(\mathcal{A}_8) \leq \frac{L^2}{2H_{\min}} \sum_{r=1}^{R} m \log(\frac{1}{\epsilon} \sum_{t=1}^{T} H_t + 1) + \frac{\epsilon D^2}{2} \tag{4.35}$$

$$R_T(\mathcal{A}_8) \leq \frac{L^2}{2H_{\min}} Rm \log(\frac{1}{\epsilon} \sum_{t=1}^{T} H_t + 1) + \frac{\epsilon D^2}{2} \tag{4.36}$$

$$R_T(\mathcal{A}_8) \leq \frac{nL^2}{2H_{\min}} \log(\frac{1}{\epsilon} \sum_{t=1}^{T} H_t + 1) + \frac{\epsilon D^2}{2} \tag{4.37}$$

□

4.2 Block Diagonal FTAL for Strongly Convex Functions

This section presents a block diagonal FTAL algorithm for strongly convex functions. This algorithm is similar to previous work on general convex functions [33]. However, their algorithm requires knowledge of bounds on the feasible set. Also, our algorithm exploits strong convexity to obtain better worst case bounds.

Algorithm 9 Block Diagonal Matrix FTAL for Strongly Convex Functions

Require: $\mathbf{x}_1 \in \mathcal{X}$, $\{H_t\}_{t=1}^T$, $\mathbf{Q}_0 = \mathbf{I}_n\epsilon$, $\epsilon > 0$, $\mathbf{b}_0 = 0$ and $R \in \{i | i \in \{1, \ldots, n\}, n = i \times m\}$

1: **for** $t = 1$ to T **do**

2: $\quad \mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$

3: $\quad \mathbf{v}_t = \frac{\sqrt{H_t}}{||\mathbf{y}_t||_2}\mathbf{y}_t$

4: $\quad \mathbf{D}_{t,R} = \begin{bmatrix} \mathbf{v}_t^1\mathbf{v}_t^{1\top} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{v}_t^R\mathbf{v}_t^{R\top} \end{bmatrix}$

5: $\quad \mathbf{Q}_t = \mathbf{D}_{t,R} + \mathbf{Q}_{t-1}$

6: $\quad \mathbf{b}_t = \mathbf{D}_{t,R}\mathbf{x}_t - \mathbf{y}_t + \mathbf{b}_{t-1}$

7: $\quad \mathbf{z}_{t+1} = \mathbf{Q}_t^{-1}\mathbf{b}_t$

8: $\quad \mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{X}}(\mathbf{z}_{t+1} - \mathbf{x})^\top\mathbf{Q}_t(\mathbf{z}_{t+1} - \mathbf{x})$

9: **end for**

4.2.1 Analysis

The analysis from Chapter 3 can be used to prove a theoretical bound for Algorithm 9. The proxy function $\tilde{f}_t$ is derived from Lemma 9 and lower bounds f_t with a block diagonal matrix approximation. Theorem 8 then uses Lemma 7 , Lemma 8 and Theorem 5 to prove the bound for Algorithm 9.

Theorem 8. *If $f_t : \mathcal{X} \to \mathbb{R}$ is a H_t-strongly convex function with respect to $||\cdot||_p$ for all $t \in \{1, \ldots, T\}$ where $p \in \{1, 2\}$, $H_t > 0$ and $H_{\min} = \min_t\{H_t\}$ and some $R, m \in \{1, \ldots, n\}$ such that $n = R \times m$ subject to a convex set $\mathcal{X} \subseteq \mathbb{R}^n$ then Algorithm $\mathcal{A}_9$ has the following regret bound*

$$R_T(\mathcal{A}_9) \leq \frac{nL^2}{H_{\min}}\log(\frac{1}{\epsilon}\sum_{t=1}^{T} H_t + 1) + \frac{1}{2}\epsilon D^2 \tag{4.38}$$

The proofs of the previously states lemmas and theorems are as follows.

Proof (Theorem 8)**.** Using the H_t-strongly convex assumption and Lemma 9 we have $f_t(\mathbf{x}_t) = \tilde{f}_t(\mathbf{x}_t)$ and $f_t(\mathbf{x}) \geq \tilde{f}_t(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$ where

$$\tilde{f}(\mathbf{x}) \equiv f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{x} - \mathbf{x}_t) + \frac{\alpha_t^2}{2}\sum_{r=1}^{R}[\mathbf{y}_t^{r\top}(\mathbf{x}^r - \mathbf{x}_t^r)]^2 \tag{4.39}$$

$\alpha_t = \frac{\sqrt{H_t}}{||\mathbf{y}_t||_2}$ and $\mathbf{y}_t \in \partial f_t(\mathbf{x}_t)$. Given this, and using Lemma 7 we have the bound

$$\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] \leq \sum_{t=1}^{T}\tilde{f}_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{X}} h_T(\mathbf{x}) + \frac{1}{2}\mathbf{x}^{*\top}\mathbf{Q}_0\mathbf{x}^* \tag{4.40}$$

where $h_T(\mathbf{x}) = \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}_0 \mathbf{x}$. Adding $\mathbf{x}^{*\top}\mathbf{Q}_0\mathbf{x}^*$ to the result of Lemma 8 then

$$\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] \leq \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_t) - \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_{t+1}) + \frac{1}{2}\mathbf{x}^{*\top}\mathbf{Q}_0\mathbf{x}^* \tag{4.41}$$

Without loss of generality we assume $0 \in \mathcal{X}$ and then using Assumption 1(b) we we have $\frac{1}{2}\mathbf{x}^{*\top}\mathbf{Q}_0\mathbf{x}^* \leq \frac{1}{2}\epsilon D^2$ giving

$$\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)] \leq \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_t) - \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_{t+1}) + \frac{1}{2}\epsilon D^2 \tag{4.42}$$

Now to satisfy the conditions of Theorem 5 consider $\tilde{f}_t(\mathbf{x}) = \sum_{r=1}^{R} g_t^r(\mathbf{y}_t^{r\top}\mathbf{x}^r)$ where

$$g_t^r(s^r) \equiv \frac{1}{R} f_t(\mathbf{x}_t) + (s^r - \mathbf{y}_t^{r\top}\mathbf{x}_t^r) + \frac{\alpha_t^2}{2}[s^r - \mathbf{y}_t^{r\top}\mathbf{x}_t^r]^2 \tag{4.43}$$

and therefore $a = \beta$ since $g_t^{r\prime\prime}(s^r) = \alpha_t^2 \geq \beta$ by Lemma 9. Then

$$\begin{aligned} ||\mathbf{v}_t^r||_2 &= ||\sqrt{g_t^{r\prime\prime}(\mathbf{u}_t^{r\top}\mathbf{x}^r)}\mathbf{u}_t^r||_2 && (4.44)\\ &= \alpha_t||\mathbf{y}_t^r||_2 && (4.45)\\ &\leq \alpha_t||\mathbf{y}_t||_2 && (4.46)\\ &= \sqrt{H_t} && (4.47) \end{aligned}$$

and therefore $c_t = \sqrt{H_t}$. Then finally $|g_t^{r\prime}(\mathbf{y}_t^{r\top}\mathbf{x}_t^r)| = |1 + \alpha_t^2 \mathbf{y}_t^{r\top}(\mathbf{x}_t^r - \mathbf{x}_t^r)| = 1$ and therefore $b = 1$. Using these parameters, Theorem 5 , equation (4.42) and $\beta = \frac{H_{\min}}{L^2}$ we have our result

$$\begin{aligned} R_T(\mathcal{A}_9) &\leq \sum_{t=1}^{T}\sum_{r=1}^{R} g_t^r(\mathbf{y}_t^{r\top}\mathbf{x}_t^r) - \sum_{t=1}^{T}\sum_{r=1}^{R} g_t^r(\mathbf{y}_t^{r\top}\mathbf{x}_{t+1}^r) + \frac{1}{2}\epsilon D^2 && (4.48)\\ R_T(\mathcal{A}_9) &\leq \sum_{r=1}^{R}\sum_{t=1}^{T}\Big[g_t^r(\mathbf{y}_t^{r\top}\mathbf{x}_t^r) - g_t^r(\mathbf{y}_t^{r\top}\mathbf{x}_{t+1}^r)\Big] + \frac{1}{2}\epsilon D^2 && (4.49)\\ R_T(\mathcal{A}_9) &\leq \sum_{r=1}^{R}\frac{mL^2}{H_{\min}}\log(\frac{1}{\epsilon}\sum_{t=1}^{T} H_t + 1) + \frac{1}{2}\epsilon D^2 && (4.50)\\ &= \frac{nL^2}{H_{\min}}\log(\frac{1}{\epsilon}\sum_{t=1}^{T} H_t + 1) + \frac{1}{2}\epsilon D^2 && (4.51) \end{aligned}$$

□

4.3 Parallelization of Quasi-Newton Algorithms

We now present the parallelized forms of the algorithms found in Chapter 3 given the objective function structure described in Chapter 1. Bertsekas [43] provides an in-depth review of specialized subgradient methods for optimization problems with a summed objective function. We now highlight how parallelism can be exploited in certain structured optimization problems. We do not present parallelized algorithms with a block diagonal hessian approximation. However, the implications with regard to algorithmic complexity will be discussed and given the following presentation the form of these algorithms should be clear to the reader.

Coupled Objective Function

Consider the online optimization problem of Definition 1, but assume the sequence of convex functions $\{f_1, f_2, \dots\}$ satisfy the following assumption.

Assumption 2. *The function $f_t : \mathcal{X} \to \mathbb{R}$ has the following form*

$$f_t(\mathbf{x}) = \sum_{r=1}^{R} f_t^r(\mathbf{x}) \tag{4.52}$$

where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$.

A problem with this objective function has coupling through the function arguments and may have coupling through the constraints, but parallelism can still be employed through the subgradient calculation. By utilizing a subgradient property we have

$$\partial f_t(\mathbf{x}) = \sum_{r=1}^{R} \partial f_t^r(\mathbf{x}) \tag{4.53}$$

as found in Appendix A. Therefore each subgradient $\mathbf{y}_t^r \in \partial f_t^r(\mathbf{x}_t^r)$ can be calculated in parallel on R slave processes. These subgradients are then sent to a master process that aggregates the subgradients, calculates the projection onto the convex set and then transmits the new feasible arguments back to the slave processes.

Decoupled Objective Function

Consider an online convex optimization problem with a function that only has coupling through the constraints.

Assumption 3. *The function $f_t : \mathcal{X} \to \mathbb{R}$ has the following form*

$$f_t(\mathbf{x}) = \sum_{r=1}^{R} f_t^r(\mathbf{x}^r) \tag{4.54}$$

where $[\mathbf{x}^1, \dots, \mathbf{x}^R] \in \mathcal{X} \subseteq \mathbb{R}^n$ and $\mathbf{x}^r \in \mathbb{R}^m$ for all $r \in \{1, \dots, R\}$

This problem structure allows greater algorithmic decoupling and therefore permits greater parallelism. All calculations can be done in parallel on R slave processes with the exception of the projection step calculated by a master process. Given the following result, it is clear that the algorithms can be parallelized, as described in Section 4.3.1, and that the previously presented theoretical results still hold.

Lemma 10. *Given a H_t-strongly convex function with respect to $||\cdot||_p$ where $p \in \{1,2\}$ with the function structure of Assumption 3 then for all $\mathbf{z}, \mathbf{x}_t \in \mathcal{X}$ and for all $\mathbf{y}_t \in \partial f(\mathbf{x}_t)$ then*

$$\sum_{r=1}^{R} f_t^r(\mathbf{z}_t^r) \geq \sum_{r=1}^{R} f_t^r(\mathbf{x}_t^r) + \sum_{r=1}^{R} \mathbf{y}_t^{r\top}(\mathbf{z}^r - \mathbf{x}_t^r) + \sum_{r=1}^{R} \frac{1}{2}[\mathbf{v}_t^{r\top}(\mathbf{z}^r - \mathbf{x}_t^r)]^2 \tag{4.55}$$

where $\mathbf{y}_t = [\mathbf{y}_t^1, \ldots, \mathbf{y}_t^R]^\top$, $\mathbf{x}_t = [\mathbf{x}_t^1, \ldots, \mathbf{x}_t^R]^\top$, $\mathbf{z} = [\mathbf{z}^1, \ldots, \mathbf{z}^R]^\top$, $\mathbf{v}_t^r = \alpha_t^r \mathbf{y}_t^r$, $\alpha_t^r = \frac{\sqrt{H_t}}{||\mathbf{y}_t^r||_2}$. Also, given $\beta = \frac{H_{\min}}{L^2}$ then $\frac{(\alpha_t^r)^2}{\beta} \geq 1$.

Proof (Lemma 10)**.** Using strong convexity, Lemma 2, and the function structure of Assumption 3 we have for some $p \in \{1,2\}$

$$f_t(\mathbf{z}) \geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z} - \mathbf{x}_t) + \frac{H_t}{2}||\mathbf{z} - \mathbf{x}_t||_p^2 \tag{4.56}$$

$$f_t(\mathbf{z}) \geq f_t(\mathbf{x}_t) + \mathbf{y}_t^\top(\mathbf{z} - \mathbf{x}_t) + \frac{H_t}{2}||\mathbf{z} - \mathbf{x}_t||_2^2 \tag{4.57}$$

$$\sum_{r=1}^{R} f_t^r(\mathbf{z}^r) \geq \sum_{r=1}^{R} f_t^r(\mathbf{x}_t^r) + \sum_{r=1}^{R} \mathbf{y}_t^{r\top}(\mathbf{z}^r - \mathbf{x}_t^r) + \frac{H_t}{2}\sum_{r=1}^{R} ||\mathbf{z}^r - \mathbf{x}_t^r||_2^2 \tag{4.58}$$

which are R summed H_t-strongly convex functions. The remaining proof follows from the proof of Lemma 6.

□

We now present parallel quasi-Newton algorithms for strongly convex functions.

4.3.1 Parallel ONS and FTAL for Strongly Convex Functions

A parallel version of Algorithm 6 is presented as Algorithm 10 assuming functions of the form given by Assumption 3. The algorithm portion that can be executed in parallel is the loop over the computing platforms R. To make the algorithm clearer and more compact consider the notation $\mathbf{z}_{t+1} \equiv [\mathbf{z}_{t+1}^{1\top}, \ldots, \mathbf{z}_{t+1}^{R\top}]^\top$, $\mathbf{x}_{t+1} \equiv [\mathbf{x}_{t+1}^{1\top}, \ldots, \mathbf{x}_{t+1}^{R\top}]^\top$, $\mathbf{y}_{t+1} \equiv [\mathbf{y}_{t+1}^{1\top}, \ldots, \mathbf{y}_{t+1}^{R\top}]^\top$ and

$$\mathbf{Q}_t \equiv \begin{bmatrix} \mathbf{Q}_t^1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{Q}_t^R \end{bmatrix} \qquad \mathbf{b}_t \equiv \begin{bmatrix} \mathbf{b}_t^1 \\ \vdots \\ \mathbf{b}_t^R \end{bmatrix} \tag{4.59}$$

Algorithm 10 Parallel ONS for Strongly Convex Functions

Require: $\mathbf{x}_1 \in \mathcal{X}$, $\{H_t\}_{t=1}^T$, $\mathbf{Q}_0 = \mathbf{I}_n\epsilon$ and $\epsilon > 0$
for $t = 1$ to T **do**
  for $r = 1$ to R **do**
    $\mathbf{y}_t^r \in \partial f_t^r(\mathbf{x}_t^r)$
    $\mathbf{v}_t^r = \sqrt{H_t}\frac{\mathbf{y}_t^r}{||\mathbf{y}_t^r||_2}$
    $\mathbf{Q}_t^r = \mathbf{v}_t^r \mathbf{v}_t^{r\top} + \mathbf{Q}_{t-1}^r$
    $\mathbf{z}_{t+1}^r = \mathbf{x}_t^r - (\mathbf{Q}_t^r)^{-1}\mathbf{y}_t^r$
  end for
  $[\mathbf{x}_{t+1}^{1\top}, \ldots, \mathbf{x}_{t+1}^{R\top}]^\top = \arg\min_{[\mathbf{x}^{1\top},\ldots,\mathbf{x}^{R\top}]^\top \in \mathcal{X}} \sum_{r=1}^R (\mathbf{z}_{t+1}^r - \mathbf{x}^r)^\top \mathbf{Q}_t^r (\mathbf{z}_{t+1}^r - \mathbf{x}^r)$
end for

The parallelized version of Algorithm 7 is presented as Algorithm 11.

Algorithm 11 Parallel FTAL for Strongly Convex Functions

Require: $\mathbf{x}_1 \in \mathcal{X}$, $\{H_t\}_{t=1}^T$, $\mathbf{Q}_0 = \mathbf{I}_n\epsilon$, $\epsilon > 0$ and $\mathbf{b}_0 = 0$
for $t = 1$ to T **do**
  for $r = 1$ to R **do**
    $\mathbf{y}_t^r \in \partial f_t^r(\mathbf{x}_t^r)$
    $\mathbf{v}_t^r = \sqrt{H_t}\frac{\mathbf{y}_t^r}{||\mathbf{y}_t^r||_2}$
    $\mathbf{Q}_t^r = \mathbf{v}_t^r \mathbf{v}_t^{r\top} + \mathbf{Q}_{t-1}$
    $\mathbf{b}_t^r = \mathbf{v}_t^r \mathbf{v}_t^{r\top} \mathbf{x}_t^r - \mathbf{y}_t^r + \mathbf{b}_{t-1}^r$
    $\mathbf{z}_{t+1}^r = (\mathbf{Q}_t^r)^{-1}\mathbf{b}_t^r$
  end for
  $[\mathbf{x}_{t+1}^{1\top}, \ldots, \mathbf{x}_{t+1}^{R\top}]^\top = \arg\min_{[\mathbf{x}^{1\top},\ldots,\mathbf{x}^{R\top}]^\top \in \mathcal{X}} \sum_{r=1}^R (\mathbf{z}_{t+1}^r - \mathbf{x}^r)^\top \mathbf{Q}_t^r (\mathbf{z}_{t+1}^r - \mathbf{x}^r)$
end for

As discussed in Section 2.6, the ONS and FTAL with a general strongly convex objective function can be implemented in $\mathcal{O}(n^2)$ time and space not including the projection step. Given the structured objective function of Assumption 3 and counting calculations done in parallel as a single calculation, the parallel ONS and FTAL can be implemented in time and space $\mathcal{O}(\frac{n^2}{R})$ without the projection step. This result indicates that for the problem class where $n << R$ parallelism may results in improved performance. Furthermore by applying the fully diagonal algorithms from Sections 4.1 and 4.2, the complexity scales with $\mathcal{O}(m)$ not including the projection step. This algorithm is particularly efficient when $m << R$ and the projection step is computationally efficient. If one were to use an interior point method a barrier function would be constructed by placing all the constraints in the objective function [44]. This would result in a coupled objective function where you are no longer able to exploit the problems structure for parallelism.

The structured objective function of Assumption 3 is discussed again in Chapter 7 when we address online vehicle routing. Although online vehicle routing is not a convex problem, the objective function satisfies Assumption 3 and permits parallel computation.

Chapter 5

Online Portfolio Optimization

One application of online convex optimization is online stock portfolio optimization. We seek to maximize our wealth using a constant-rebalanced portfolio (CRP) [45]. A CRP strategy rebalances a fixed amount of wealth among a set of stocks at regular trading periods. We will measure the performance of strategy by minimizing regret, where regret is the difference between an portfolio chosen at each iteration and the best portfolio chosen in hindsight. A investment strategy is said to be *universal* if it achieves sub-linear regret [46].

Regret bounds that are a function of the variation or volatility in stock have been derived. This is in contrast to a bound that is a function of an algorithm's iterations [37]. Meta algorithms have been developed that integrate many heuristic algorithms with a single universal algorithm to obtain a new algorithm that is also universal [47]. Our new universal algorithms presented in Chapter 3 and 4 could be used within this framework. Our work does not include transaction costs as in the work of Blum and Kalai [48].

We present two models for online portfolio management. The first is based on the α-exp-concave convex functions that motivated the logarithmic regret algorithms in Chapter 2. These algorithms were shown to out perform other known universal algorithms [49]. The second model we present is a ℓ_2-norm regularized model that is strongly convex. This model penalizes portfolios where all wealth is placed in just a few stocks and thus encourages balanced portfolios.

5.1 Models for Online Portfolio Optimization

Let there be n stocks in a portfolio. At every trading period $t \in \{1, \ldots, T\}$ an investor observes a relative price vector $\mathbf{r}_t \in \mathbb{R}^n$, where $r_t(i)$ is the ratio of the closing price of stock i on day t onto the closing price on day $t-1$. If an investor chooses a portfolio $\mathbf{x}_t$, then during trading period t their wealth changes by a factor of $\mathbf{r}_t^\top \mathbf{x}_t$. Therefore after T trading periods the wealth achieved per dollar invested is $\prod_{t=1}^T \mathbf{r}_t^\top \mathbf{x}_t$ and the logarithmic growth ratio is then $\sum_{t=1}^T \log(\mathbf{r}_t^\top \mathbf{x}_t)$. We will assume that the change in stock price over every trading period is

bounded above and below as stated by Asummption 4 [49].

Assumption 4. *A price relative vector* $\mathbf{r}_t \in \mathbb{R}^n$ *is such that* $||\mathbf{r}_t||_\infty \leq M$ *and* $\min_{i\in\{1,\ldots,n\}} r_t(i) = B$ *where* $B > 0$ *for all for* $t \in \{1, \ldots, T\}$.

The best constant-rebalanced portfolio (CRP) in hindsight at trading period T is defined as

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{S}^n} \sum_{t=1}^{T} \log(\mathbf{r}_t^\top \mathbf{x}) \tag{5.1}$$

where $\mathcal{S}^n$ is the n-dimensional simplex. This model is concave however from this point forward we will consider regret minimization of an online convex optimization problem. Therefore, to maximize our wealth, we will use an algorithm $\mathcal{A}$ to minimize regret as follows:

$$\sum_{t=1}^{T} g_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{S}^n} \sum_{t=1}^{T} g_t(\mathbf{x}) = R_T(\mathcal{A}, \{g_1, \ldots, g_T\}) \tag{5.2}$$

where the function $g_t : \mathcal{S}^n \to \mathbb{R}$ is defined as

$$g_t(\mathbf{x}) \equiv -\log(\mathbf{r}_t^\top \mathbf{x}) \tag{5.3}$$

This function is α-exp-concave function for all $\alpha \in (0, 1]$ as proven by Lemma 4 of Das and Banerjee [47]. Given that a n-dimentional simplex is bounded and Lemma 11, then the results presented in Chapter 2 can be applied to model 5.3.

Lemma 11. *Let the function* $g_t : \mathcal{S}^n \to \mathbb{R}$ *be defined by* $g_t(\mathbf{x}) = -\log(\mathbf{r}_t^\top \mathbf{x})$ *where* $\mathcal{S}^n$ *is the* n*-dimensional simplex. Given Assumption 4 then* $||\nabla g_t(\mathbf{x})||_2 \leq \frac{\sqrt{n}M}{B}$ *for all* $\mathbf{x} \in \mathcal{S}^n$.

Proof (Lemma 11)**.** Consider

$$\begin{aligned} ||\nabla g_t(\mathbf{x})||_2 &= || - \frac{1}{\mathbf{r}_t^\top \mathbf{x}} \mathbf{r}_t||_2 && (5.4)\\ &= \frac{1}{|\mathbf{r}_t^\top \mathbf{x}|} ||\mathbf{r}_t||_2 && (5.5) \end{aligned}$$

Now, using $\min_j \mathbf{r}_t(j) \geq B$ we have $|\mathbf{r}_t^\top \mathbf{x}| \geq \sum_{i=1}^n Bx(i)$ then using $\mathbf{x} \in \mathcal{S}^n$ we have $\sum_{i=1}^n x(i) = 1$ and therefore $|\mathbf{r}_t^\top \mathbf{x}| \geq B$. Finally given that $||\mathbf{r}_t||_2 \leq \sqrt{n}||\mathbf{r}_t||_\infty$ and $||\mathbf{r}_t||_\infty \leq M$ we have

$$\begin{aligned} ||\nabla g_t(\mathbf{x})||_2 &= \frac{1}{|\mathbf{r}_t^\top \mathbf{x}|} ||\mathbf{r}_t||_2 && (5.6)\\ &\leq \frac{1}{B} ||\mathbf{r}_t||_2 && (5.7)\\ &\leq \frac{\sqrt{n}}{B} ||\mathbf{r}_t||_\infty && (5.8)\\ &\leq \frac{\sqrt{n}M}{B} && (5.9) \end{aligned}$$

giving our result.

□

5.1.1 ℓ_2-norm Regularized Model

This section presents a ℓ_2-norm regularized model for online portfolio management. In a financial context minimizing $||\mathbf{x}||_2^2$ where $\mathbf{x} \in \mathcal{S}^n$ corresponds roughly to maximizing the effective number of assets N_e where $N_e \leq n$. That is,

$$||\mathbf{x}||_2^2 \approx \frac{1}{N_e} \tag{5.10}$$

where $\mathbf{x} \in \mathcal{S}^n$. By maximizing the effective number of assets, N_e , portfolio diversification is encouraged. Portfolio diversification counteracts instability by acting as a regularizer [50]. Therefore we will define the model $f_t : \mathcal{S}^n \to \mathbb{R}$ as

$$f_t(\mathbf{x}) \equiv -\log(\mathbf{r}_t^\top \mathbf{x}) + \frac{H}{2}||\mathbf{x}||_2^2 \tag{5.11}$$

where H is a tuned parameter. The regret we desire to minimize is then

$$\sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{S}^n} \sum_{t=1}^{T} f_t(\mathbf{x}) = R_T(\mathcal{A}, \{f_1, \ldots, f_T\}) \tag{5.12}$$

A larger H will result in a more diversified portfolio while a smaller H will result in a less diversified portfolio. A two dimensional simplex for different values of H can be seen in Figure 5.1. When $H = 0$ then we obtain the standard model for online portfolio management (5.3). As the H parameter increases the best portfolio is forced from the extremes to a more diversified portfolio.

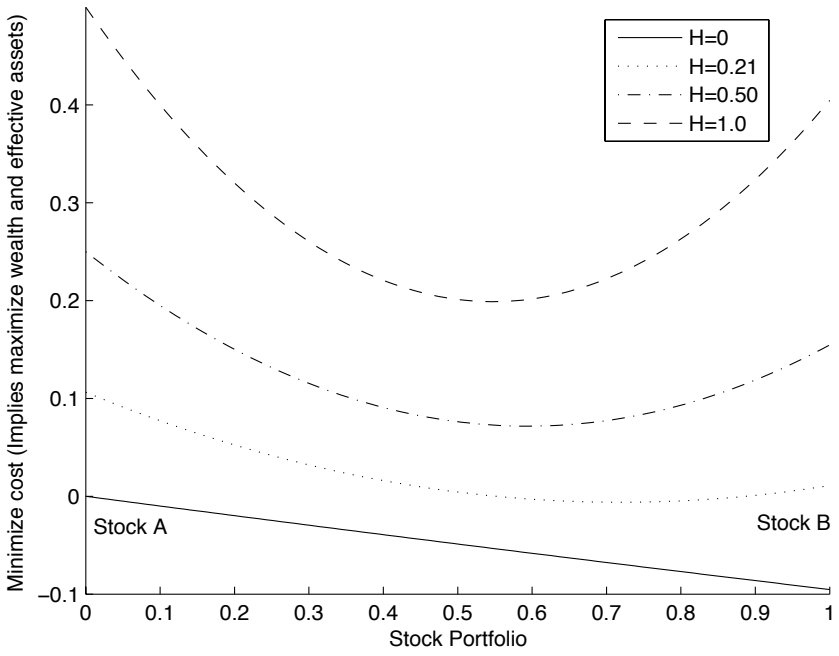


Figure 5.1: Two dimensional simplex of ℓ_2-regularized model for different H values, Stock B is 10% better than Stock A

For a fixed $H = 0.21$ two three dimensional simplexes is plotted in Figure 5.2. Each extreme point of the simplex represents the allocation of all wealth into a single stock. Figure 5.2 represents two different cases possible over one trading day. In the first case stock A is 10% better than stock B and C while the second case shows stock B and C is 10% better than stock A. The second case shows that if two stocks are equivalently better than another stock the portfolio allocation will be split between them. These figures show the model's primary advantage. With essentially equivalent but a slightly different stock values model (5.3) recommends moving all the stock value back and forth between the two stocks. On the other hand model (5.11) keeps the stock value split nearly even between the two different stocks. The two different models use two distinct strategies to optimize a constant-rebalanced portfolio. Model (5.3) finds and tracks a few of the best stocks, while model (5.11) deviates slightly from a uniform constant-rebalanced portfolio and shifts the allocation toward the better performing stocks.

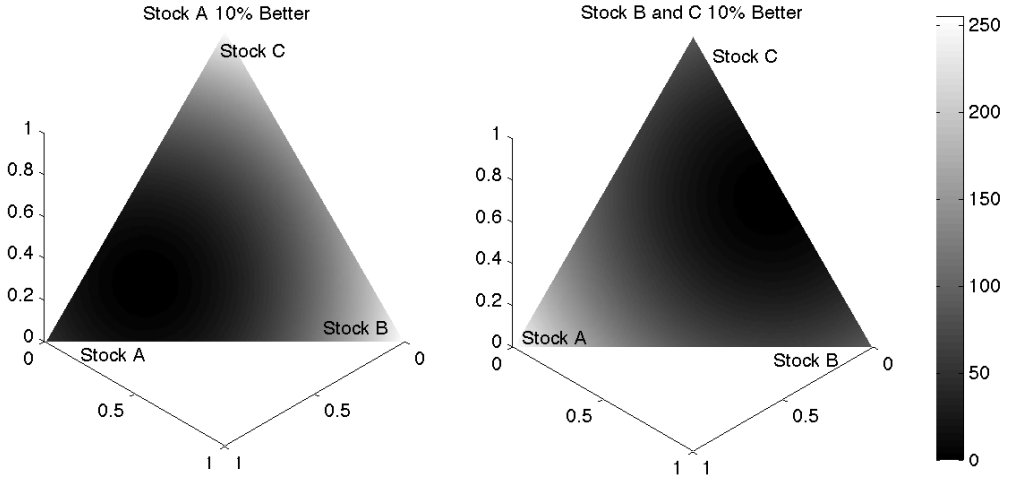


Figure 5.2: 3-dimensional simplex of ℓ_2-regularized model with $H = 0.21$

The model defined by function (5.11) is H-strongly convex using Lemma 17. Given Lemma 12 and the n-dimensional simplex is bounded, then the results presented in Chapter 3 can be applied to this model.

Lemma 12. *Let the function $f_t : \mathcal{S}^n \to \mathbb{R}$ be defined by $f_t(\mathbf{x}) = -\log(\mathbf{r}_t^\top \mathbf{x}) + \frac{H}{2}||\mathbf{x}||_2^2$ where $\mathcal{S}^n$ is an n-dimensional simplex. Given Assumption 4 then $||\nabla f_t(\mathbf{x})||_2 \leq \frac{\sqrt{n}M}{B} + H$ for all $\mathbf{x} \in \mathcal{S}^n$.*

Proof (Lemma 12)**.** Consider

$$||\nabla f_t(\mathbf{x})||_2 = || - \frac{1}{\mathbf{r}_t^\top \mathbf{x}}\mathbf{r}_t + H\mathbf{x}||_2 \tag{5.13}$$

$$\leq \frac{1}{|\mathbf{r}_t^\top \mathbf{x}|}||\mathbf{r}_t||_2 + H||\mathbf{x}||_2 \tag{5.14}$$

using the triangle inequality. Given that $||\mathbf{x}||_2 \leq ||\mathbf{x}||_1$ and $||\mathbf{x}||_1 = 1$ for all $x \in \mathcal{S}^n$ we have

$$||\nabla f_t(\mathbf{x})||_2 \leq \frac{1}{|\mathbf{r}_t^\top \mathbf{x}|}||\mathbf{r}_t||_2 + H \tag{5.15}$$

The result follows from Lemma 11.

□

5.2 Numerical Experiments

We will now numerically compare three algorithms. We compare a specialized FTAL algorithm for online portfolio management presented in Agarwal et al. [49], FTAL for strongly convex functions, Algorithm 7, and a uniform constant rebalanced portfolio. A uniform constant rebalanced portfolio is defined such that $\mathbf{x}_t = \frac{1}{n}\mathbf{1}$ for all $t \in \{1, \ldots, T\}$. A summary of the algorithms and the abbreviation used for each algorithm is presented in Table 5.1.

Table 5.1: Abbreviations for algorithms

Abbreviation	Description	Reference
FTAL-OPO	FTAL for online portfolio optimization	Agarwal et al.[49] with Model 5.3
FTAL-SC	FTAL for strongly convex functions	Algorithm 7 with Model 5.11
UCRP	Uniform constant-rebalanced portfolio	$\mathbf{x}_t = \frac{1}{n}\mathbf{1}$ for all $t \in \{1, \ldots, T\}$

The FTAL-OPO algorithm uses model (5.3) while FTAL-SC uses model (5.11). Our numerical experiments use historical New York Stock Exchange (NYSE) data for 19 stocks over a 44-year period (11178 trading days) ending in 2006 [51]. The data from July 3, 1962 to December 31, 1984 is identical to that used by Cover and Helmbold et al. [45, 52]. After that period relative price vectors are calculated from the nominal closing prices, including dividends and splits. The parameters used in our numerical experiments are found in Table 5.2.

Table 5.2: Parameters used in numerical experiments

Algorithm	Parameters
FTAL-OPO	$\beta = 1$, $\delta = \frac{1}{8}$, $\eta = 0$ and $\mathbf{x}_1 = \frac{1}{n}\mathbf{1}$
FTAL-SC	$H = 0.21$, $\epsilon = 0.05$ and $\mathbf{x}_1 = \frac{1}{n}\mathbf{1}$

Performance Metrics

We measure performance using three metrics: wealth, Annual Percentage Yield (APY) and volatility. Let V_i be the initial value of an investment and V_f be the final value after T trading days and let $T_{years} = T/365$. Wealth is defined as, $W \equiv \frac{V_f}{V_i}$ where $V_i = 1$, the return per dollar of initial investment. Annual Percentage Yield (APY) is calculated by the following formula

$$APY \equiv \left[(V_f/V_i)^{\frac{1}{T_{years}}} - 1\right] \times 100 \tag{5.16}$$

Volatility is calculated by taking the standard deviation of the sequence $\{\mathbf{r}_t^\top \mathbf{x}_t\}_{t=1}^T$ for a given algorithm.

5.2.1 Results

The performance metrics for the NYSE dataset are found in Table 5.3. As the table shows, FTAL-SC performs significantly better than FTAL-OPO. However, we shall see that the two algorithms and models use different strategies and FTAL-SC is not always better.

Table 5.3: Daily trading period NYSE dataset

Algorithm	Wealth	APY	Volatility
FTAL-OPO	713.9	23.93	0.0126
FTAL-SC	3815.7	30.90	0.0118
UCRP	503.3	22.52	0.0098

Figure 5.3 presents the growth of logarithmic wealth over time. Both the FTAL-OPO and FTAL-SC algorithms out perform the UCRP algorithm.

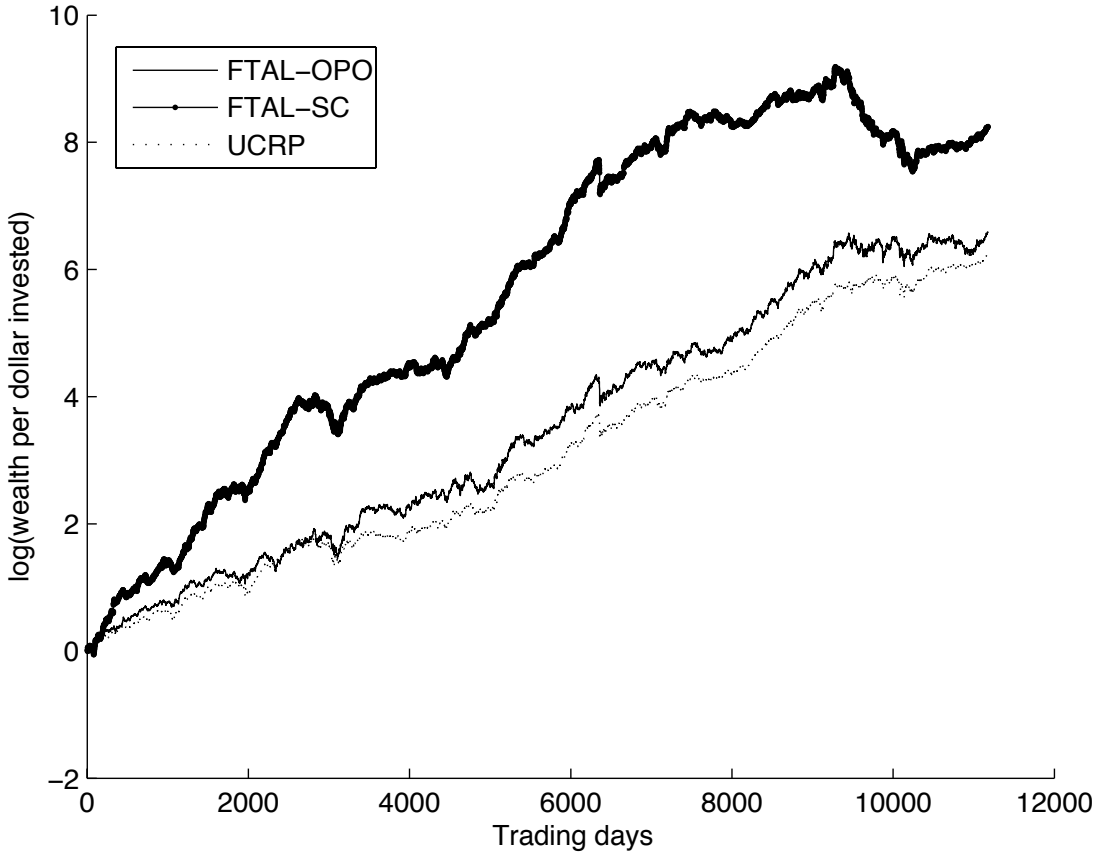


Figure 5.3: Logorithmic growth of wealth plotted vs. trading days

To gain insight into the algorithms' strategies we run numerical experiments with a portfolio of Coke, IBM and General Electric. The stock allocations chosen by the algorithms for each trading day are plotted on a three dimensional simplex in Figure 5.4. The UCRP algorithm is not shown since it would always choose the portfolio in the center of the simplex. This figure illuminates the distinct strategies of FTAL-OPO and FTAL-SC. The FTAL-OPO algorithm tracks a few of the best performing stocks and the FTAL-SC algorithm shifts the "center of gravity" toward the best performing stocks. In this case neither algorithm performs significantly better or worse then the other as shown in Table 5.4. Both FTAL-OPO and FTAL-SC slightly out perform the UCRP strategy.

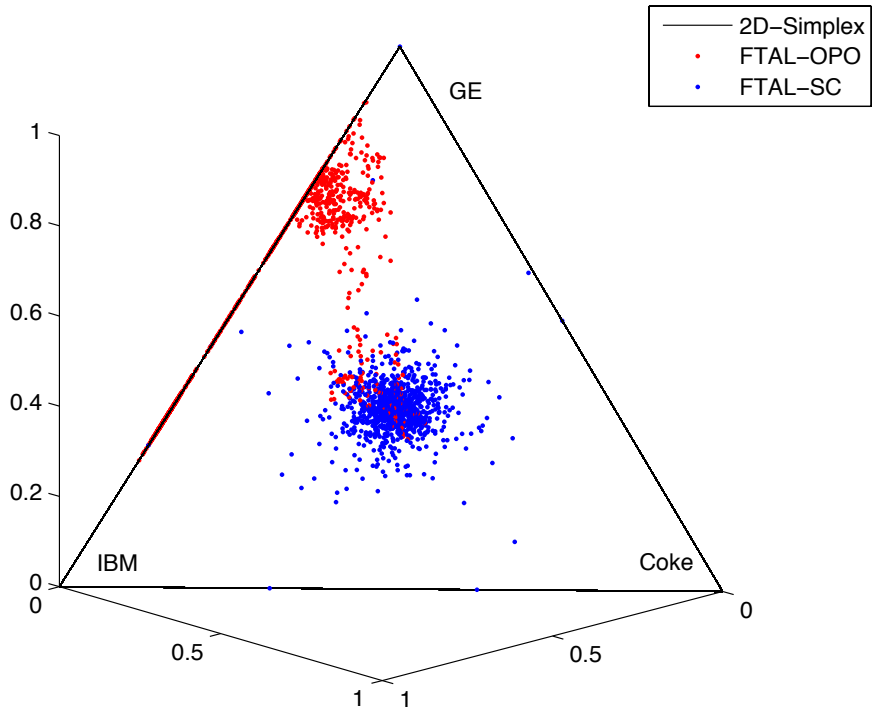


Figure 5.4: Portfolio over time for Coke, IBM and GE

Table 5.4: Daily Trading Period with Coke, IBM and GE

Algorithm	Wealth	APY	Volatility
FTAL-OPO	299.44	20.47	0.0139
FTAL-SC	292.40	20.37	0.0120
UCRP	283.60	20.25	0.0120

For this specific example Figure 5.5 shows that the FTAL-OPO algorithm performs better with a larger portfolio because there is a greater chance of finding and tracking high performing stocks. FTAL-OPO also performs better with a large or short trading period.

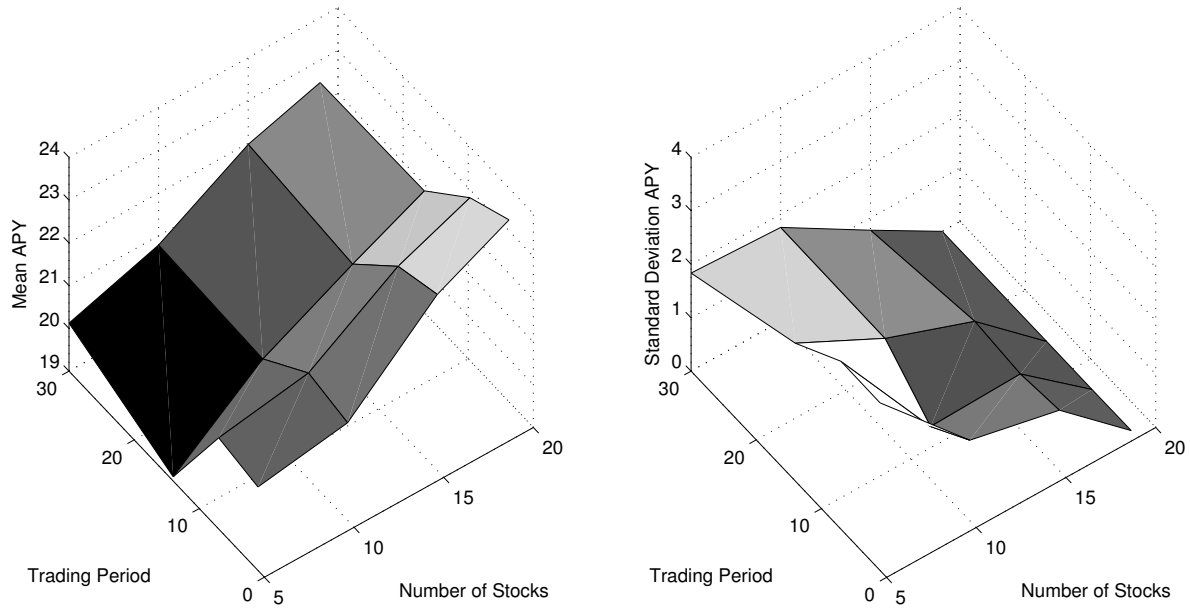


Figure 5.5: FTAL-OPO Annual Percentage Yield vs. trading period and portfolio size

Figure 5.6 shows that the FTAL-SC algorithm performs best with a short trading period and a large stock portfolio. The FTAL-OPO algorithm out performs the FTAL-SC algorithm for a large stock portfolio and a long trading period.

Figure 5.6: FTAL-SC Annual Percentage Yield vs. trading period and portfolio size

Figure 5.7 and 5.8 present the volatility associated with different portfolio sizes and trading periods. Figure 5.7 shows that the FTAL-OPO algorithm has low volatility for large stock portofilios because it finds and tracks the best stocks. The more stocks it has to choose from increases the likelihood that it will find good stocks to track. Figure 5.8 shows that the FTAL-SC algorithm constantly adjusts the "center of gravity" toward the best performing stocks resulting in increased volatility for larger portfolios and trading periods.

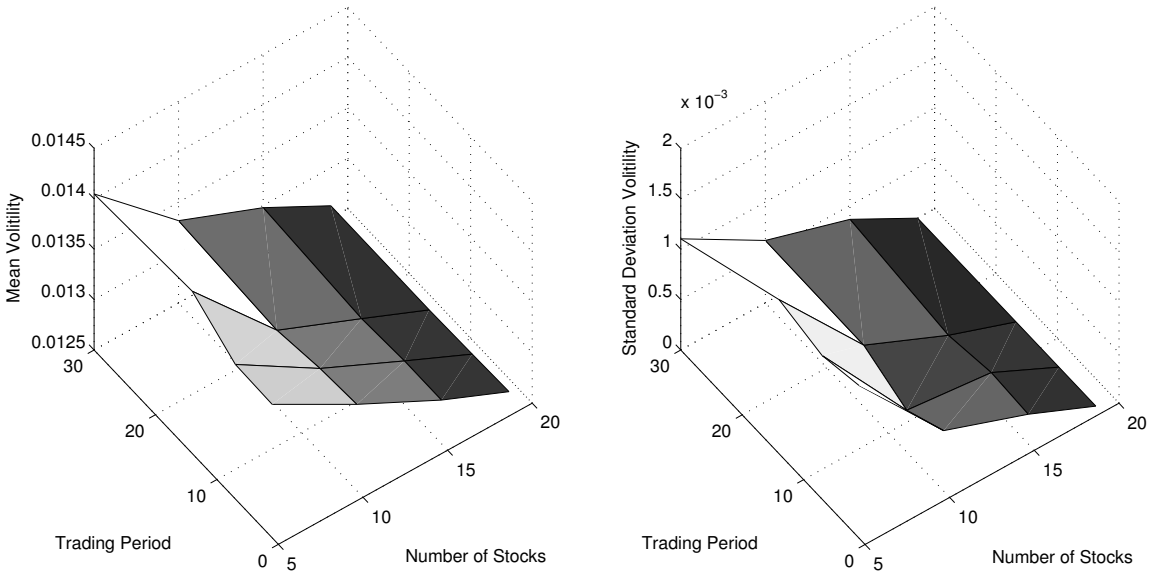


Figure 5.7: FTAL-OPO volatility vs. trading period and portfolio size

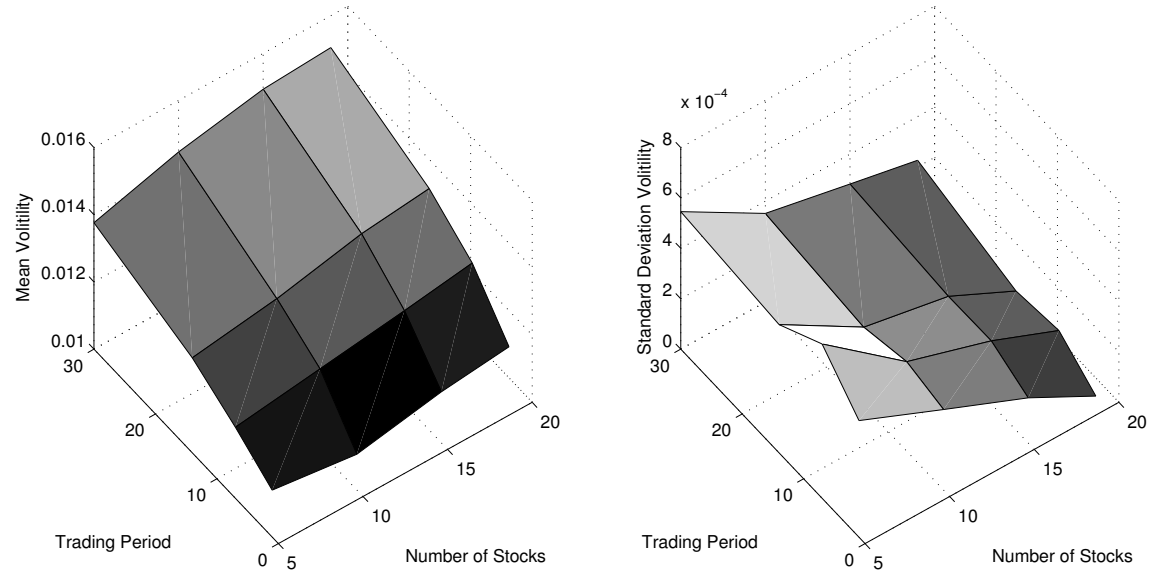


Figure 5.8: FTAL-SC volatility vs. trading period and portfolio size

Chapter 6

Numerical Experiments in Convex Optimization

As discussed in Section 2.1.1, the results of Chapters 2 through 4 apply to the optimization of a non-smooth strongly convex function. In this chapter we use numerical experiments to study the performance of the algorithms from Chapters 2 through 4 and related work.

6.1 Experimental Setup

We adopt a MATLAB testing environment designed for benchmarking non-smooth optimization algorithms [19]. The testing environment was then adapted and expanded to include new algorithms and test functions. We ran our experiments on a Quad Core Intel Xeon Processor 2.33GHz using 64bit Red Hat Linux with 8GB of RAM. The algorithms found in Table 6.1 and the functions found in Table 6.3 were implemented in MATLAB R2008a.

Although the theory developed earlier in this dissertation allows for the optimization of constrained non-smooth strongly convex functions we limit our tests to unconstrained non-smooth strongly convex functions. Specifically, we study the non-smooth strongly convex test functions in Table 6.3. The test functions can be defined for any dimension n. We test the algorithms using four values of n. Given our implementation and available computational resources, we will refer to cases where $(n = 10)$ as very small scale, $(n = 100)$ as small scale, $(n = 1000)$ as a medium scale and $(n = 10000)$ as a large scale. Given memory limitations, algorithms with $\mathcal{O}(n^2)$ complexity are tested on very small and small scale problems while algorithms with $\mathcal{O}(n)$ complexity are tested on medium and large scale problems. We will study and compare algorithms performance by looking at the error at termination defined by

$$f_{error} = \min_{i \in [1, T_{total}]} f_i(\mathbf{x}_i) - f(\mathbf{x}^*) \tag{6.1}$$

and the total number of iterations T_{total} at termination. For a given function, problem

dimension and algorithm set, each algorithm is given a fixed amount of time to solve a randomly generated problem instance. The time given to a problem instance depends on problem dimension. A problem of dimension ($n = 10$) is given 0.1 seconds, ($n = 100$) is given 1 second, ($n = 1000$) is given 10 seconds, and ($n = 10000$) is given 100 seconds. Time instead of function error is used as a termination condition because some algorithms require a prohibitive amount of time to reach the error achieved by other algorithms.

Given a set of 100 randomly generated problem instances the average error and iterations at termination is calculated with a 95% confidence interval for each algorithm and test function. A problem instance is composed of function parameters and an initial function argument. Test function parameters are randomly generated as described in Section 6.3. An initial condition is randomly generated from a unit normal distribution and then added to the optimal argument of the test problem. The optimal argument is known a priori by definition of the function.

6.2 Algorithms for Numerical Experiments

We compare the performance of the algorithms in Table 6.1. The table lists each algorithm's abbreviation and a reference. As specified in Table 6.2, with the exception of AdaGradMD and OGD-SC, all algorithms use the same initial hessian approximation. The initial hessian approximation for AdaGradMD was derived from its regret bound using optimization in the same way as our algorithm in Section 3.1.2 [23]. To accurately compare performance Algorithm 5 is initialized in the same way as Algorithm 7 and referred to as FTAL-EC-I. During testing, Algorithm 5 without the initial hessian was consistently outperformed by the same algorithm with an initial hessian and therefore the algorithm without an initial hessian has been omitted.

Table 6.1: Abbreviations used for different algorithms in the numerical experiments

Abbreviation	Description	Reference
FTAL-SC	FTAL for strongly convex functions	Algorithm 7
FTAL-SC-D	Diagonal Matrix FTAL for strongly convex functions	Algorithm 9
ONS-SC	ONS for strongly convex functions	Algorithm 6
ONS-SC-D	Diagonal Matrix ONS for strongly convex functions	Algorithm 8
ONS-EC	ONS for α-exp-concave functions	Algorithm 3
FTAL-EC-I	FTAL for α-exp-concave functions with initial hessian	Algorithm 5
AdaGradMD	AdaGrad mirrored descent	Duchi et al. [23]
OGD-SC	Gradient descent for strongly convex functions	Algorithm 2

The functions outlined in Table 6.3 are all constructed such that they are H-strongly convex with respect to $||\cdot||_2$ where $H = 1$. Although the new algorithms developed in

Chapter 3 do not require function parameter information, other than the strongly convex parameter H, the previously known algorithms do require function parameter information. As a compromise, we determine the function parameters from the initial argument and initial subgradient and then allow the new algorithms, developed Chapter 3 and 4, to use these parameters to calculate its initial hessian approximation. The function parameters used by each algorithm and the update complexity of each algorithm are found in Table 6.2. Let the randomly generated initial argument be $\mathbf{x}_1$ and let the initial subgradient be $\mathbf{y}_1$. The function parameters are then $L = ||\mathbf{y}_1||_2$, $L_\infty = ||\mathbf{y}_1||_\infty$ and $D = ||\mathbf{x}_1 - \mathbf{x}^*||_2$. The optimal argument $\mathbf{x}^*$ is known a priori via the test problem definition.

Table 6.2: Update complexity and parameter choices

Algorithm	Update Complexity	Parameters
FTAL-SC	$\mathcal{O}(n^2)$	$\epsilon = \frac{L^2}{HD^2}$
FTAL-SC-D	$\mathcal{O}(n)$	$\epsilon = \frac{L^2}{HD^2}$, $R = 1$
ONS-SC	$\mathcal{O}(n^2)$	$\epsilon = \frac{L^2}{HD^2}$
ONS-SC-D	$\mathcal{O}(n)$	$\epsilon = \frac{L^2}{HD^2}$, $R = 1$
ONS-EC	$\mathcal{O}(n^2)$	$\epsilon = \frac{L^2}{HD^2}$, $\alpha = \frac{L^2}{H}$
FTAL-EC-I	$\mathcal{O}(n^2)$	$\mathbf{Q}_0 = \frac{L^2}{HD^2}\mathbf{I}_n$, $\alpha = \frac{L^2}{H}$
AdaGradMD	$\mathcal{O}(n)$	$\eta = \frac{L_\infty^2}{H}$, $\delta = \frac{1}{D^2}$
OGD-SC	$\mathcal{O}(n)$	$\eta_t = \frac{1}{Ht}$ for all $t \in \{1, \ldots, T\}$

6.3 Non-smooth Strongly Convex Test Functions

The test functions are outlined in Table 6.3. These functions can be defined for any dimension n and are all H-strongly convex by construction where $H = 1$. The test functions were adopted from those found in earlier related work on non-smooth functions [53, 54, 18, 19]. Functions F1-F3 require only $\mathcal{O}(n)$ scalar variables to store the parameters and therefore are used for numerical experiments on very small, small, medium and large scale problems while functions F4-F5 require $\mathcal{O}(n^2)$ scalar variables and therefore are only used for very small, small and medium scale problems.

The remainder of this section describes each function and the method for generating function parameters. Figures 6.1 through 6.5 show a two dimensional contour plot of each function for a randomly generated problem instance and the iterates chosen by the FTAL-SC algorithm.

Table 6.3: Test functions

Abbreviation	Description	Function Definition
F1	ℓ_∞-norm plus quadratic	$f(\mathbf{x}) = \max_{1\leq i\leq n} \lvert a_i x_i\rvert + \frac{1}{2}\sum_{i=1}^n (b_i x_i)^2$
F2	Low memory piecewise quadratic	$f(\mathbf{x}) = \max_{1\leq i\leq n} \frac{1}{2}\mathbf{x}^\top \mathbf{D}_i \mathbf{x} + \mathbf{b}_i^\top \mathbf{x} + c_i$
F3	ℓ_1-norm plus quadratic	$f(\mathbf{x}) = \sum_{i=1}^n \lvert a_i x_i\rvert + \frac{1}{2}\sum_{i=1}^n (b_i x_i)^2$
F4	Piecewise quadratic	$f(\mathbf{x}) = \max_{1\leq i\leq n} \frac{1}{2}\mathbf{x}^\top \mathbf{A}_i \mathbf{x} + \mathbf{b}_i^\top \mathbf{x} + c_i$
F5	Convex partly smooth	$f(\mathbf{x}) = \sqrt{\mathbf{x}^\top \mathbf{A}\mathbf{x}} + \mathbf{x}^\top \mathbf{B}\mathbf{x}$

ℓ_∞-norm plus quadratic (F1)

By inspection the optimal argument is $\mathbf{x}^* = \mathbf{0}$ and the optimal solution is $f(\mathbf{x}^*) = 0$. The parameters a_i are drawn uniformly from the interval $[n, 0]$ and b_i are drawn uniformly from the interval $[n, 1]$ for all $i \in \{1, \ldots, n\}$.

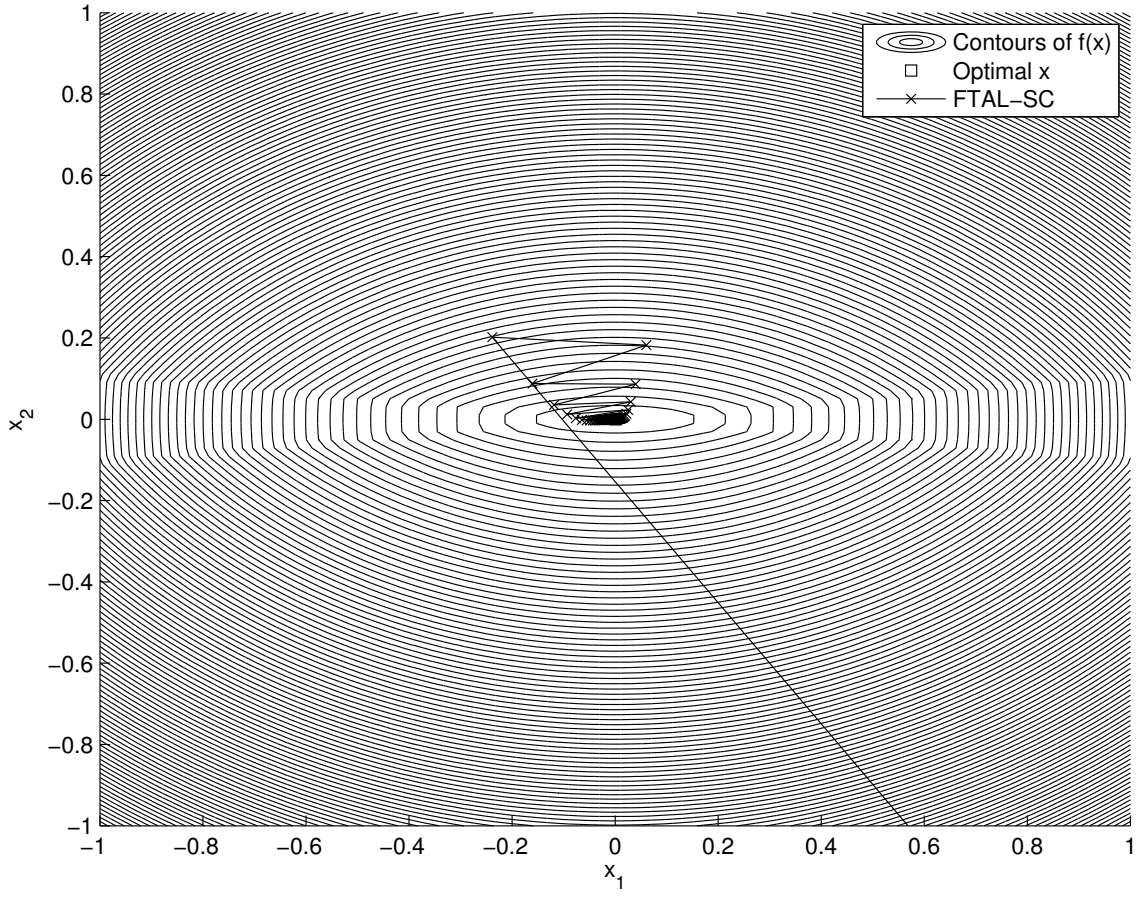


Figure 6.1: Test function F1

Low memory piecewise quadratic (F2)

The piecewise quadratic parameters for each $i \in \{1, \ldots, n\}$ are generated as follows. $\mathbf{D}_i$ is a diagonal positive definite matrix with elements on the diagonal drawn uniformly from the interval $[1, 4.8i]$ and with all other elements zero. The elements of the vector $\mathbf{b}_i$ are drawn uniformly from the interval $[i-1, i]$. Let $\bar{x} = -\mathbf{D}_n^{-1}\mathbf{b}_n$ and then finally $a_i = -\frac{1}{2}\bar{\mathbf{x}}^\top \mathbf{D}_i \bar{\mathbf{x}} - \mathbf{b}_i^\top \bar{\mathbf{x}}$. The problem then has the optimal solution $\mathbf{x}^* = \bar{\mathbf{x}}$ with the optimal value $f(\mathbf{x}^*) = 0$. This problem is potentially challenging because all the quadratic functions are active at the optimal.

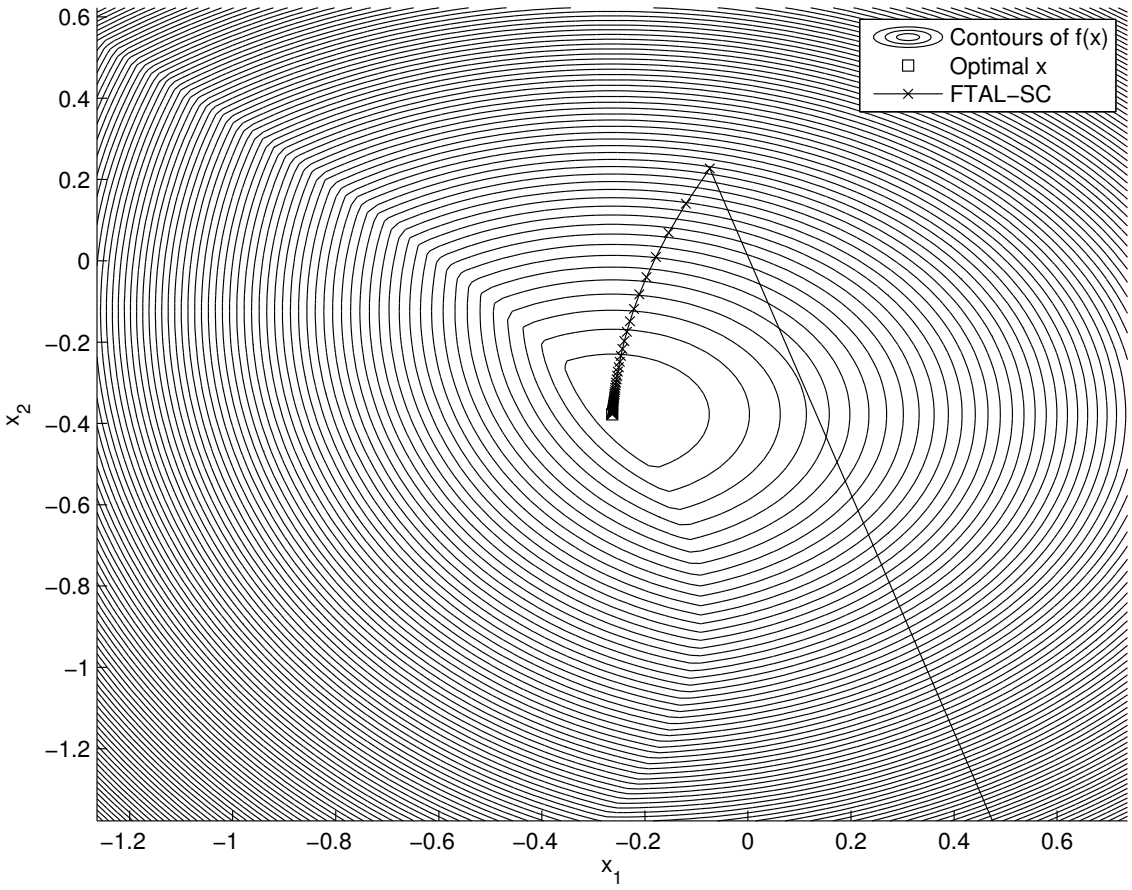


Figure 6.2: Test function F2

ℓ_1-norm plus quadratic (F3)

By inspection the optimal argument is $\mathbf{x}^* = \mathbf{0}$ and the optimal solution is $f(\mathbf{x}^*) = 0$. The parameters a_i are drawn uniformly from the interval $[n, 0]$ and b_i are drawn uniformly from the interval $[n, 1]$ for all $i \in \{1, \ldots, n\}$.

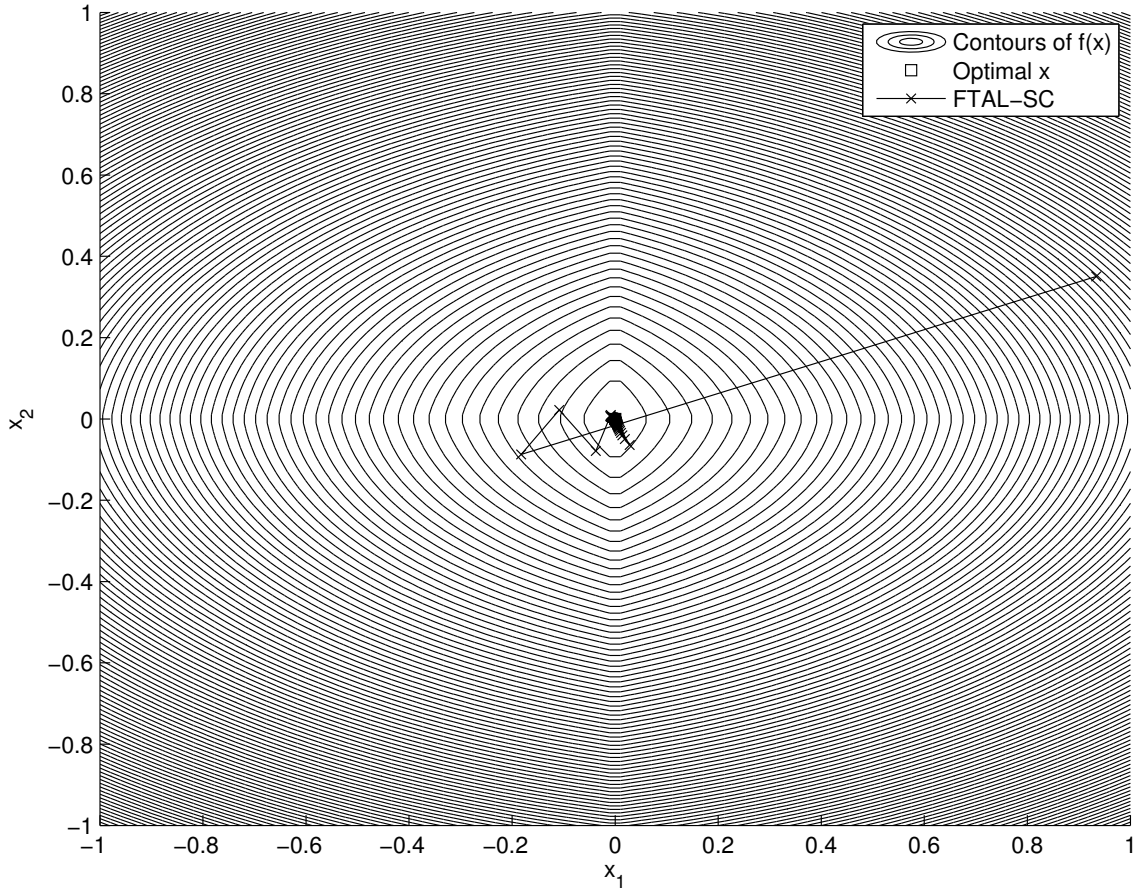


Figure 6.3: Test function F3

Piecewise quadratic (F4)

The piecewise quadratic parameters for each $i \in \{1, \ldots, n\}$ are generated as follows. $\mathbf{A}_i$ is a positive definite matrix with eignvalues drawn uniformly from the interval $[1, 4.8i]$. The elements of the vector $\mathbf{b}_i$ are drawn uniformly from the interval $[i-1, i]$. Let $\bar{x} = -\mathbf{A}_n^{-1}\mathbf{b}_n$ and then finally $a_i = -\frac{1}{2}\bar{\mathbf{x}}^\top \mathbf{A}_i \bar{\mathbf{x}} - \mathbf{b}_i^\top \bar{\mathbf{x}}$. The problem then has the optimal solution $\mathbf{x}^* = \bar{\mathbf{x}}$ with the optimal value $f(\mathbf{x}^*) = 0$. This problem is potentially challenging because all the quadratic functions are active at the optimal.

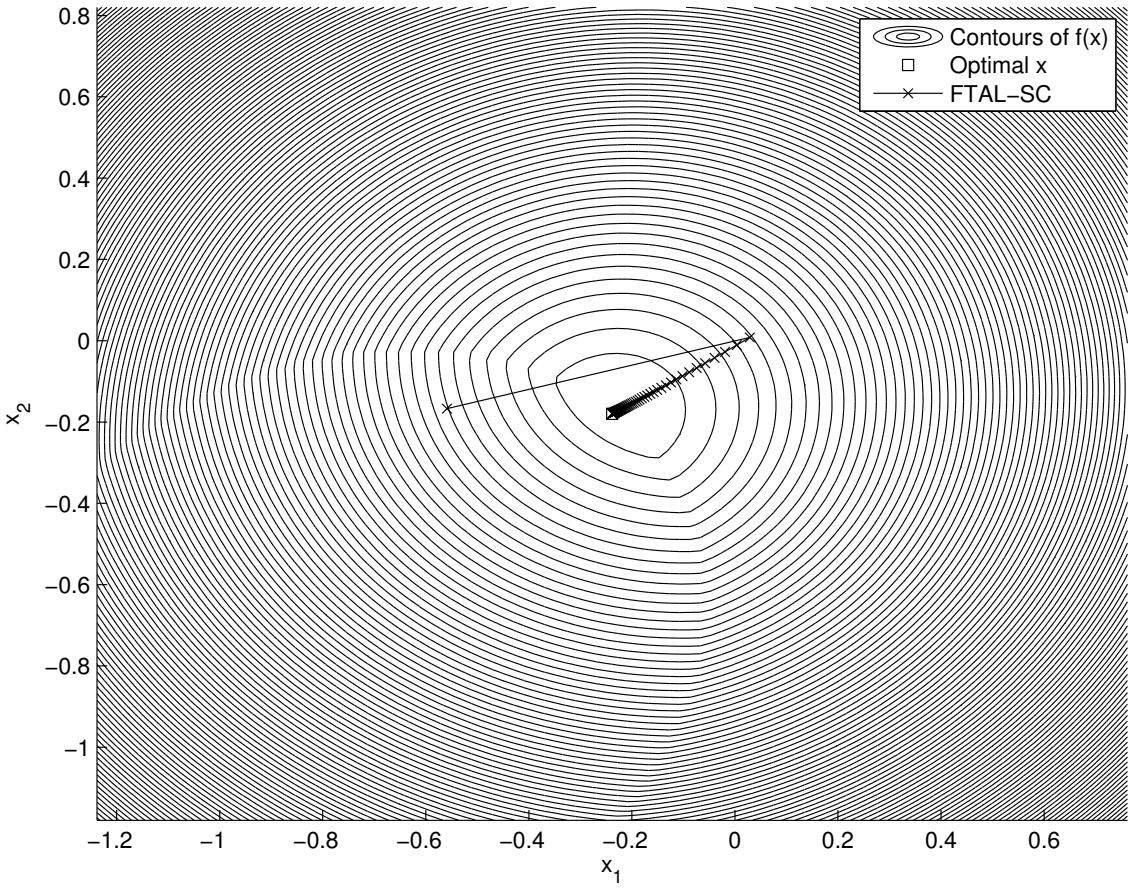


Figure 6.4: Test function F4

Convex partly smooth (F5)

By inspection the optimal argument is $\mathbf{x}^* = \mathbf{0}$ and the optimal solution is $f(\mathbf{x}^*) = 0$. The parameters are calculated as follows: $\mathbf{A} = \begin{bmatrix} \mathbf{M} & 0 \\ 0 & 0 \end{bmatrix}$ where $\mathbf{M} \in \mathbb{R}^{n/2 \times n/2}$ is a randomly generated symmetric positive definite matrix with condition number $[n/2]^2$ and $\mathbf{B} \in \mathbb{R}^{n \times n}$ is a randomly generated symmetric positive definite matrix with condition number n^2 and a minimum eignvalue of one.

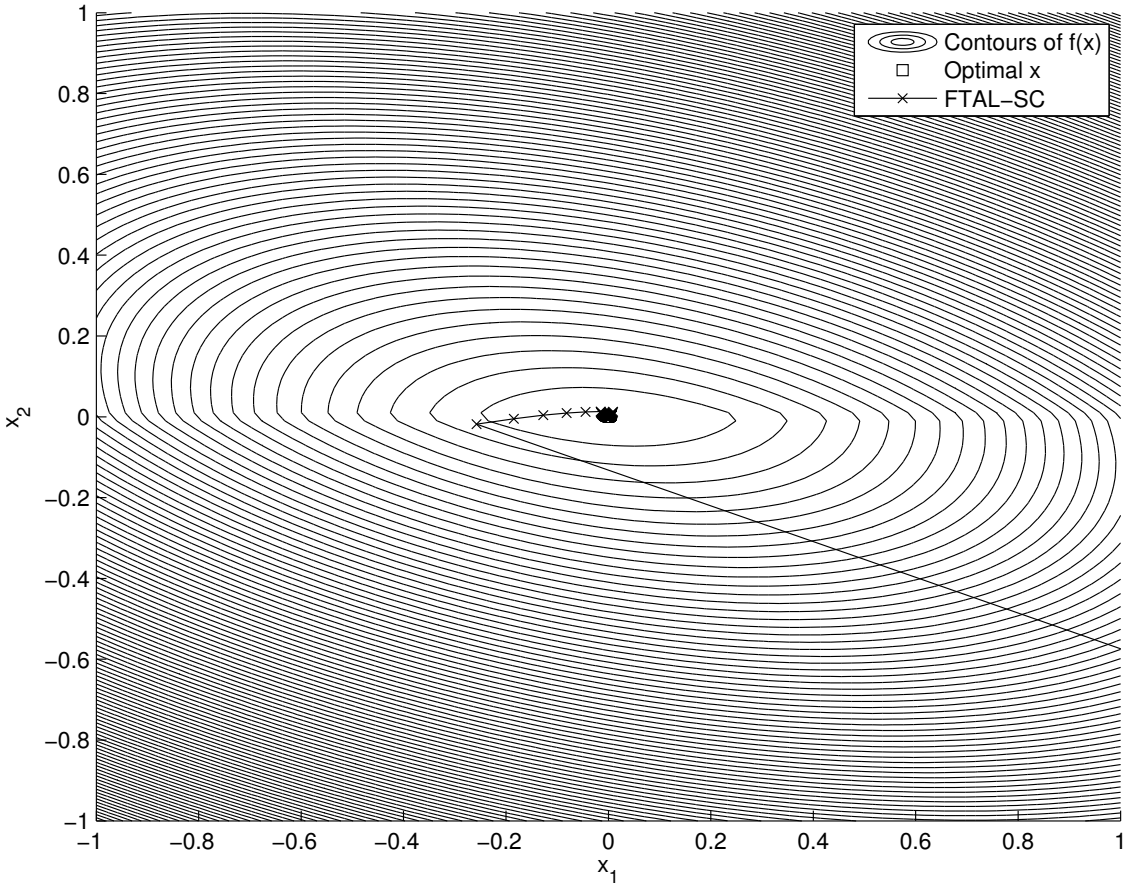


Figure 6.5: Test function F5

6.4 Results

We now discuss the results of our experiments. We first consider both $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ algorithms for very small scale and small scale problems in Sections 6.4.1 and 6.4.2. We then consider $\mathcal{O}(n)$ algorithms for medium and large scale problems in Sections 6.4.3 and 6.4.4. Given that each test function is artificially constructed we are concerned with the relative mean error, instead of the absolute error, of each function. Therefore, we normalize the mean error across algorithms for each function. The raw data used to construct each figure is found in tables at the end of each section. A summary of our results is found in Section 6.4.5.

6.4.1 Very Small Scale ($n = 10$)

For a very small scale problem Figure 6.6 gives the normalized mean error for all algorithms in Table 6.1 and all functions in Table 6.3. Figure 6.6 shows that the algorithms with lowest mean error for F5 are ONS-SC and FTAL-SC. It is unclear however which algorithms have the lowest error for functions F1-F4. Therefore consider Figure 6.7 that plots the mean error with a 95% confidence interval for the algorithms FTAL-SC, ONS-SC and OGD-SC.

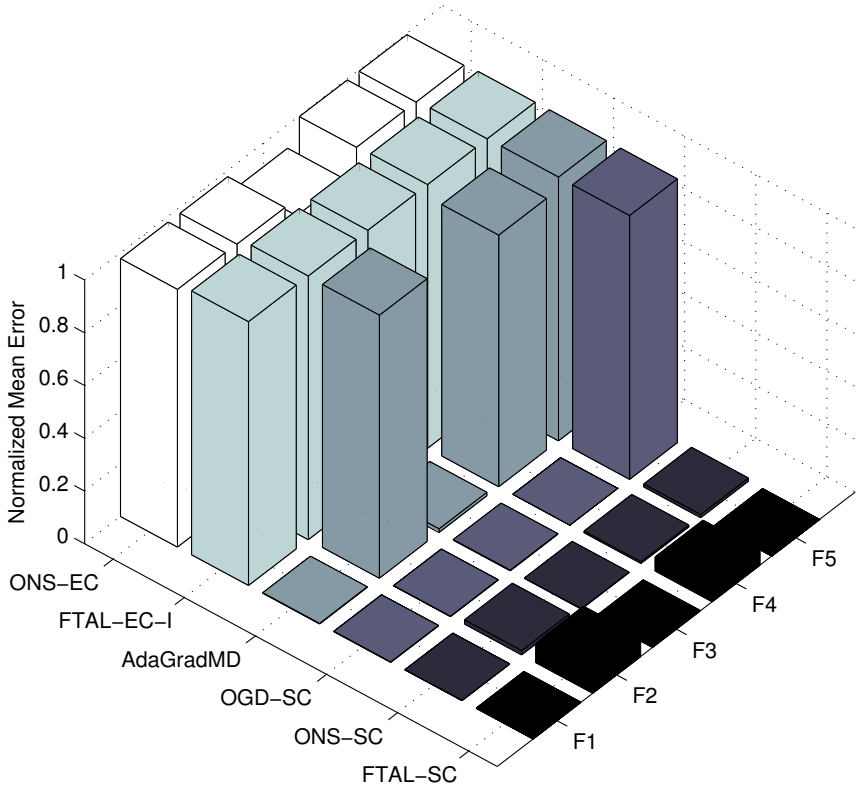
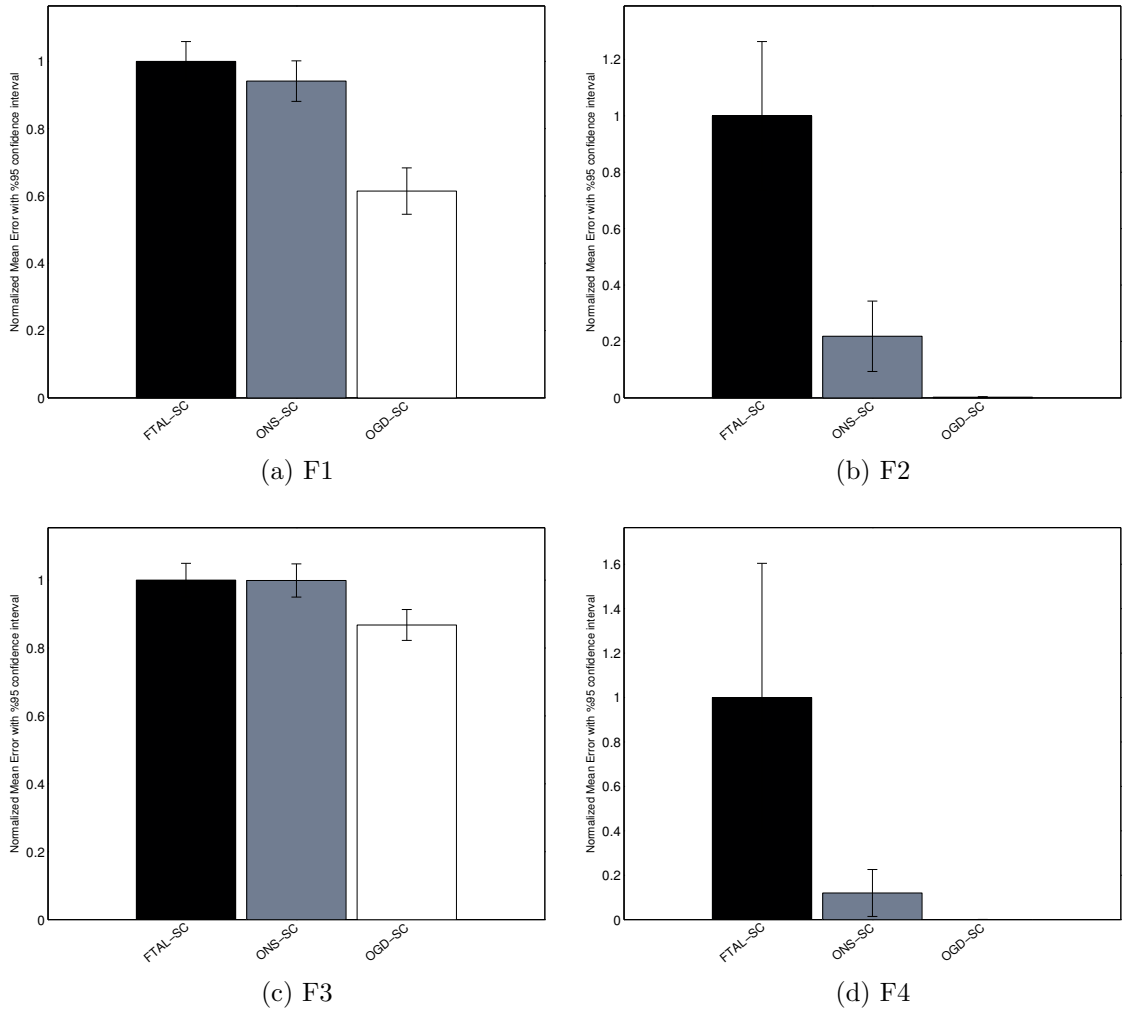


Figure 6.6: Mean error for all algorithms ($n = 10$)

For test functions F1-F4 Figure 6.7 shows OGD-SC has a lower mean error than FTAL-SC and ONS-SC. One possible explanation for this is that OGD-SC is a $\mathcal{O}(n)$ algorithm while FTAL-SC and ONS-SC are both $\mathcal{O}(n^2)$ algorithms. The algorithms were run for the same amount of time and therefore OGD-SC is allowed to execute more iterations. Therefore in Figure 6.8 we compare the diagonal matrix algorithms FTAL-SC-D and ONS-SC-D to OGD-SC and AdaGradMD.



(a) F1

(b) F2

(c) F3

(d) F4

Figure 6.7: Mean error with 95% confidence interval for test functions F1-F4 ($n = 10$)

Figure 6.8 shows that ONS-SC-D and FTAL-SC-D outperform the other algorithms for test function F5. In Figure 6.9 we compare the $\mathcal{O}(n)$ algorithm for functions F1-F4.

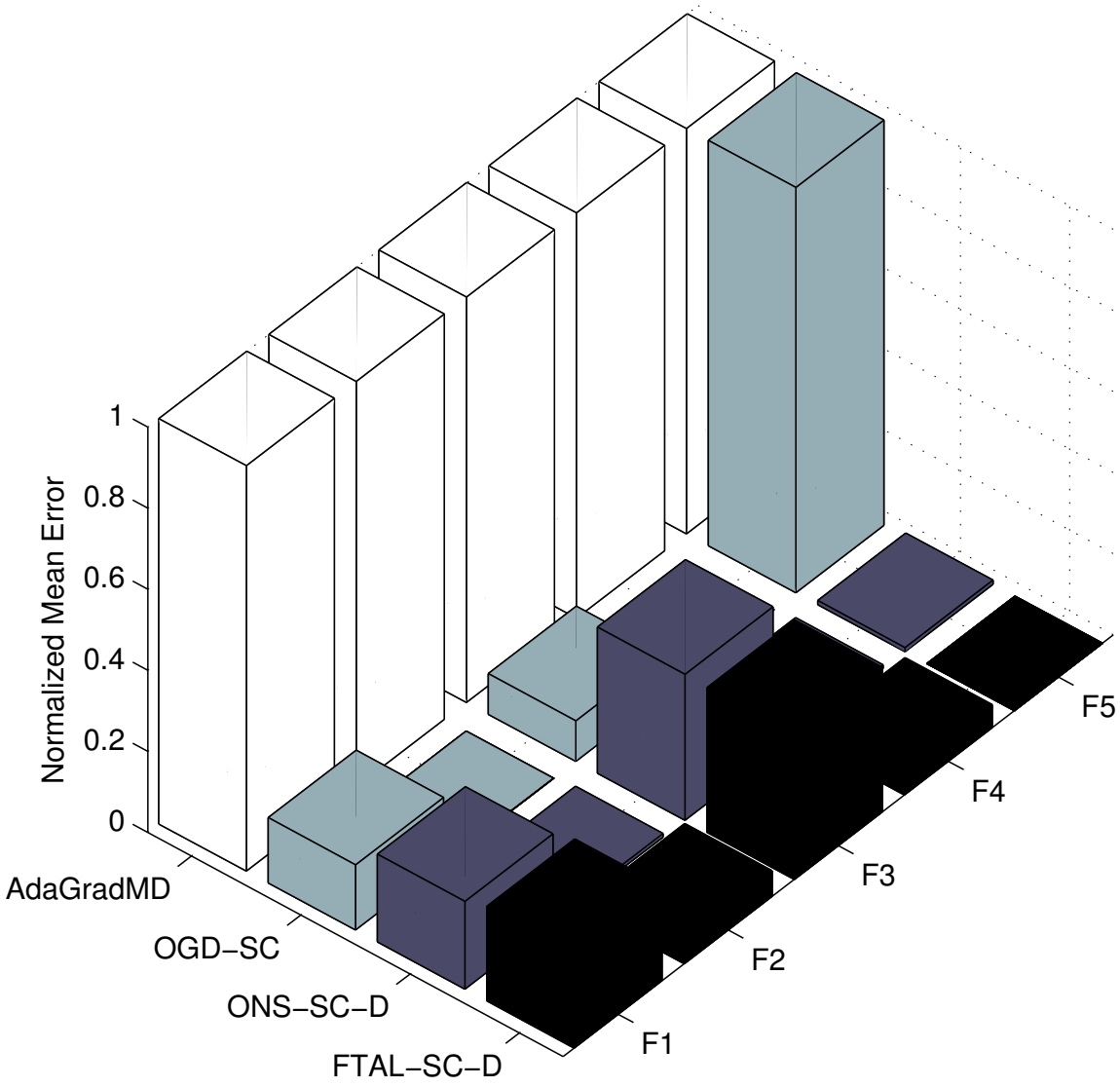
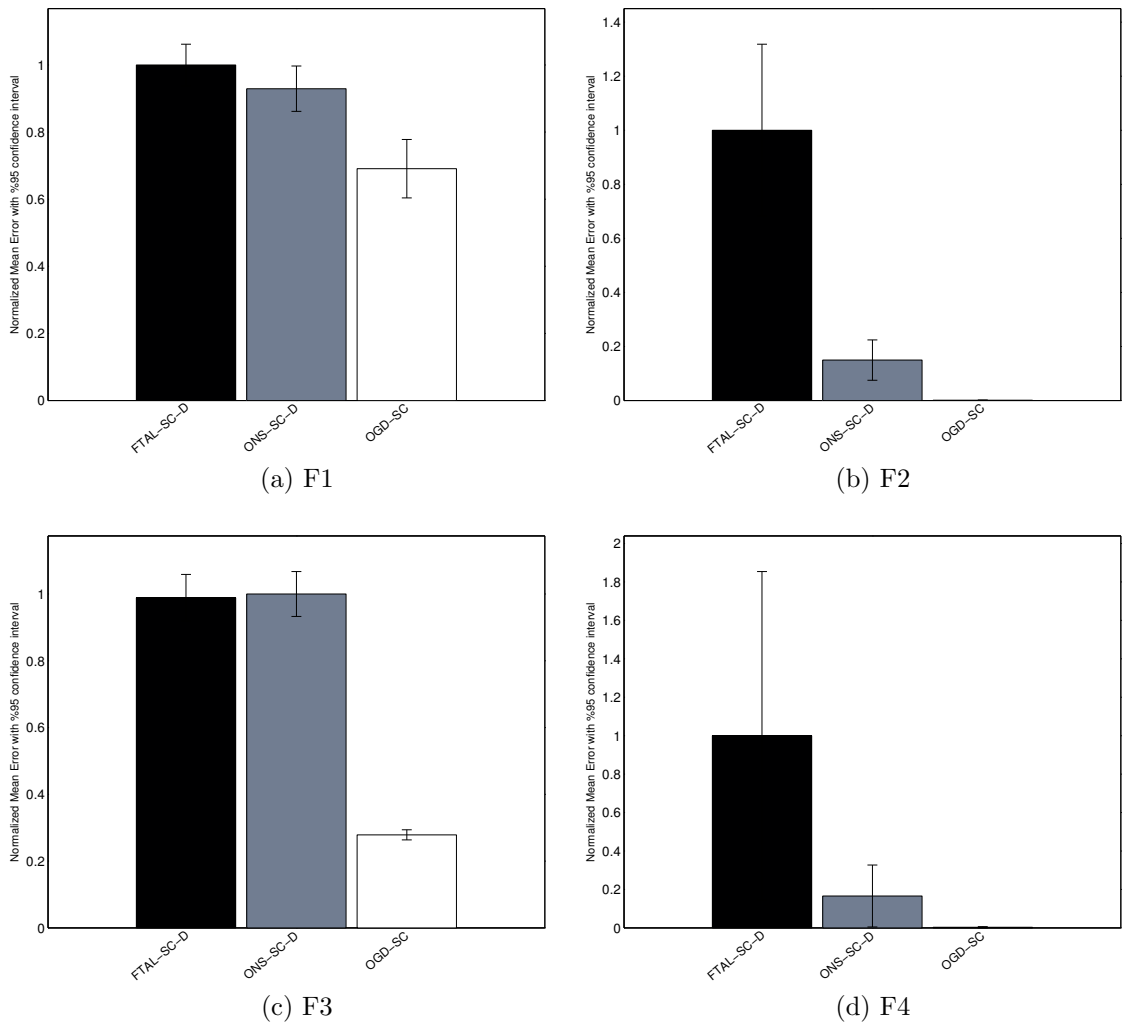


Figure 6.8: Mean error for $\mathcal{O}(n)$ algorithms ($n = 10$)

The relative mean error shown in Figure 6.9 is quite similar to Figure 6.7. In Figure 6.9 FTAL-SC-D and ONS-SC-D are shown to have marginally lower or effectively equivalent mean error as FTAL-SC and ONS-SC in Figure 6.7.



(a) F1 (b) F2

(c) F3 (d) F4

Figure 6.9: Mean error with 95% confidence interval for test functions F1-F4 and $\mathcal{O}(n)$ algorithms ($n = 10$)

From these results we conclude that for problems of dimension ($n = 10$) on test function F5 algorithms that FTAL-SC and ONS-SC have lower mean error than any other algorithm considered. On the other test functions OGD-SC has the lowest mean error followed by FTAL-SC and ONS-SC, AdaGradMD and then finally ONS-EC and FTAL-EC-I. Table 6.4 presents the data used to compose Figure 6.6. Table 6.5 presents the data used to create Figure 6.8. Table 6.4 and 6.5 both have columns for algorithms AdaGradMD and OGD-SC. The two tables were created from two distinct experiments however the results for AdaGradMD and OGD-SC should be statistically equivalent. The numbers are different because two different sets of test problems were randomly generated.

Table 6.4: Mean error and total iterations with 95% confidence interval for all algorithms ($n = 10$)

		FTAL-SC	ONS-SC	OGD-SC	AdaGradMD	FTAL-EC-I	ONS-EC
F1	$\hat{f}_{error}$	6.62e-02 ±3.91e-03	6.23e-02 ±3.98e-03	4.07e-02 ±4.56e-03	2.51e-01 ±2.91e-02	1.77e+02 ±2.11e+01	1.73e+02 ±2.12e+01
	$\hat{T}_{inter}$	473.44 ±1.34	493.13 ±0.51	634.31 ±0.37	558.79 ±0.51	516.69 ±0.53	536.77 ±0.57
F2	$\hat{f}_{error}$	1.11e+01 ±2.91e+00	2.42e+00 ±1.38e+00	2.90e-02 ±2.07e-02	1.47e+02 ±1.79e+01	1.48e+02 ±1.95e+01	1.44e+02 ±1.96e+01
	$\hat{T}_{inter}$	271.22 ±0.08	276.87 ±0.12	317.23 ±0.18	298.66 ±0.17	285.21 ±0.17	290.77 ±0.17
F3	$\hat{f}_{error}$	2.45e-01 ±1.21e-02	2.44e-01 ±1.20e-02	2.12e-01 ±1.11e-02	1.87e+00 ±1.65e-01	1.40e+02 ±2.85e+01	1.27e+02 ±2.57e+01
	$\hat{T}_{inter}$	552.81 ±0.20	574.29 ±0.27	781.96 ±0.48	670.82 ±0.16	609.55 ±0.47	634.43 ±0.48
F4	$\hat{f}_{error}$	7.70e+00 ±4.65e+00	9.24e-01 ±8.14e-01	4.57e-03 ±3.67e-03	1.37e+02 ±2.07e+01	1.43e+02 ±1.77e+01	1.43e+02 ±1.78e+01
	$\hat{T}_{inter}$	269.97 ±0.09	275.19 ±0.11	316.16 ±0.17	297.18 ±0.15	283.51 ±0.12	289.24 ±0.14
F5	$\hat{f}_{error}$	4.58e-04 ±9.11e-05	5.72e+00 ±8.17e-01	5.01e+02 ±5.34e+01	5.01e+02 ±5.34e+01	5.01e+02 ±5.34e+01	4.98e+02 ±5.41e+01
	$\hat{T}_{inter}$	499.39 ±0.20	514.22 ±0.19	677.57 ±0.17	594.45 ±0.16	543.03 ±0.13	563.14 ±0.13

Table 6.5: Mean error and total iterations with 95% confidence interval for $\mathcal{O}(n)$ algorithms ($n = 10$)

		FTAL-SC-D	ONS-SC-D	OGD-SC	AdaGradMD
F1	$\hat{f}_{error}$	5.77e-02 ±3.57e-03	5.36e-02 ±3.90e-03	3.98e-02 ±5.02e-03	2.47e-01 ±2.96e-02
	$\hat{T}_{inter}$	547.06 ±1.44	562.57 ±0.61	634.52 ±0.49	556.80 ±0.48
F2	$\hat{f}_{error}$	9.01e+00 ±2.87e+00	1.35e+00 ±6.71e-01	1.14e-02 ±5.63e-03	1.43e+02 ±1.76e+01
	$\hat{T}_{inter}$	292.93 ±0.12	295.79 ±0.12	316.19 ±0.17	297.57 ±0.15
F3	$\hat{f}_{error}$	7.62e-01 ±5.40e-02	7.71e-01 ±5.19e-02	2.15e-01 ±1.17e-02	2.14e+00 ±2.13e-01
	$\hat{T}_{inter}$	653.26 ±0.51	671.98 ±0.36	783.80 ±0.57	669.09 ±0.17
F4	$\hat{f}_{error}$	8.79e+00 ±7.51e+00	1.46e+00 ±1.42e+00	2.70e-02 ±3.03e-02	1.56e+02 ±2.51e+01
	$\hat{T}_{inter}$	291.40 ±0.13	294.57 ±0.12	315.28 ±0.17	296.51 ±0.15
F5	$\hat{f}_{error}$	4.00e-04 ±8.50e-05	4.80e+00 ±7.51e-01	4.70e+02 ±4.70e+01	4.70e+02 ±4.70e+01
	$\hat{T}_{inter}$	574.57 ±0.29	587.37 ±0.20	677.95 ±0.19	595.01 ±0.15

6.4.2 Small Scale ($n = 100$)

We now investigate small scale problems. In Figure 6.10 we can see that FTAL-SC has a lower mean error than all other algorithms for test functions F4 and F5. Figure 6.10 also shows FTAL-SC, ONS-SC and OGD-SC have lower mean error than FTAL-EC, ONS-EC and AdaGradMD for all test functions. Now consider Figure 6.11 that compares FTAL-SC, ONS-SC and OGD-SC with test functions F1-F3.

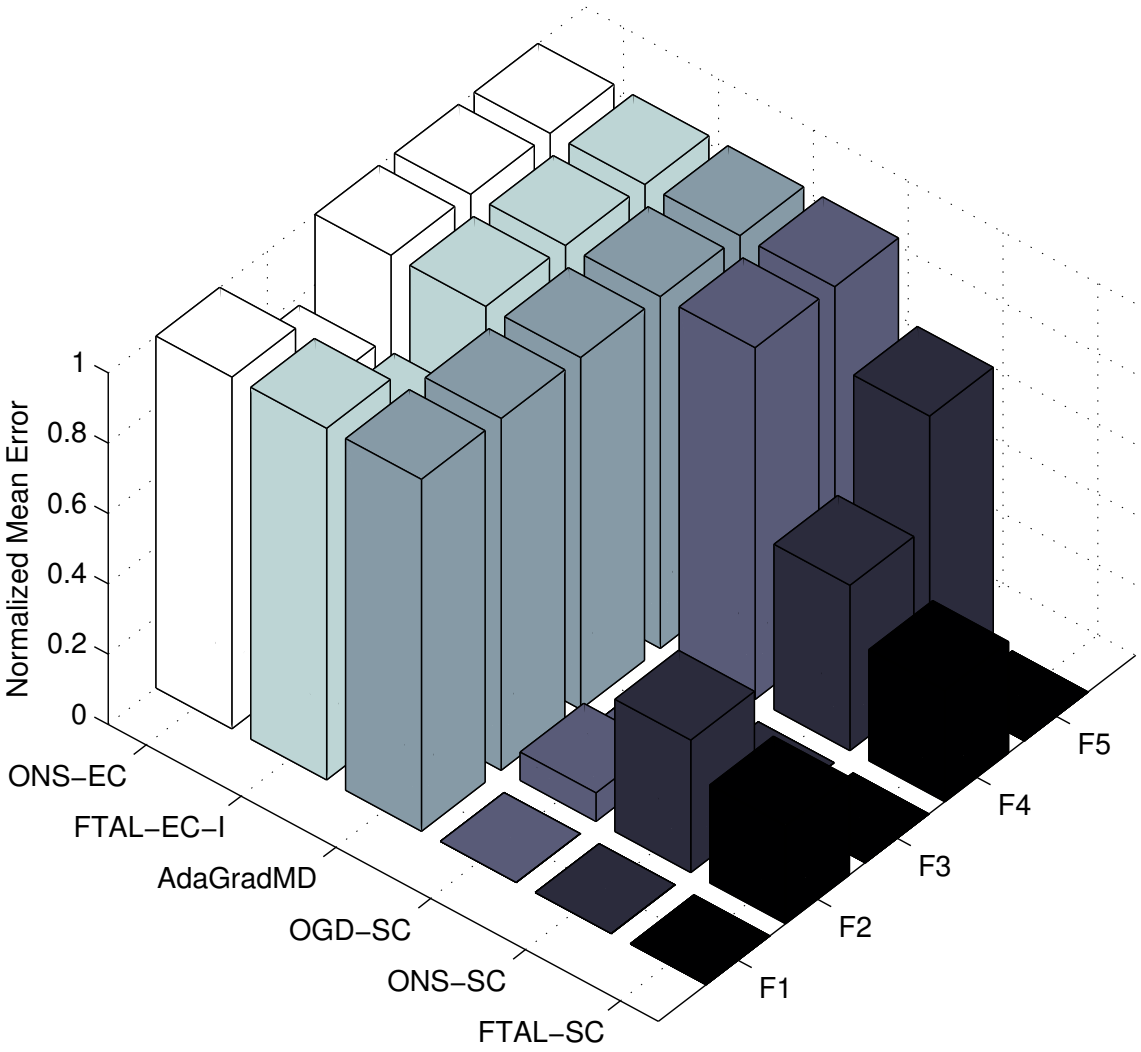
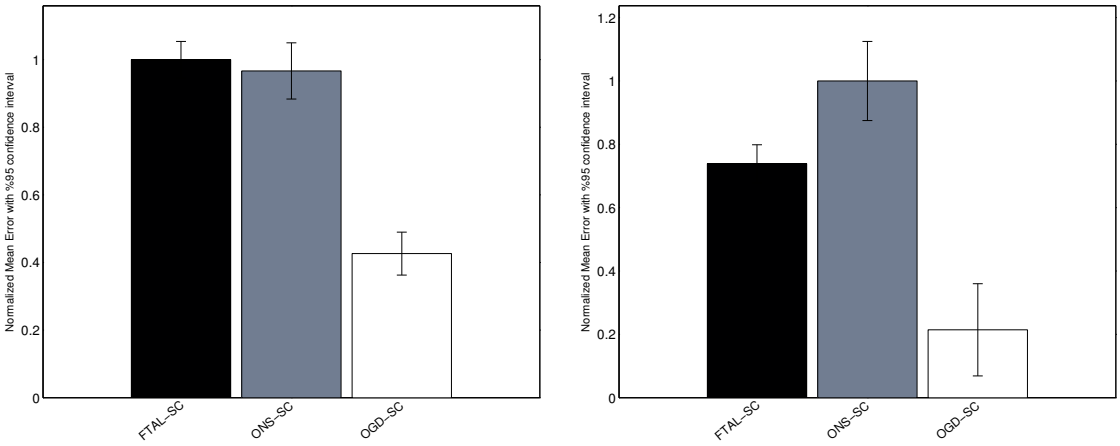


Figure 6.10: Mean error for all algorithms ($n = 100$)

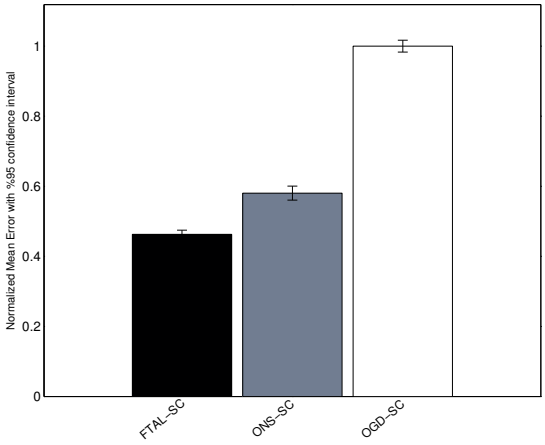
It is seen in Figure 6.11 FTAL-SC and ONS-SC have lower mean error than OGD-SC for test functions F3 while OGD-SC has lower mean error on functions F1 and F2. For problems of larger dimension, algorithmic complexity should have more of an effect on performance so

we should compare algorithms with the same algorithmic complexity. Consider Figure 6.12 for a comparison of algorithms with $\mathcal{O}(n)$ complexity for problem dimension ($n = 100$).



(a) F1

(b) F2



(c) F3

Figure 6.11: Mean error with 95% confidence interval for test functions F1-F3 ($n = 100$)

For the algorithms with $\mathcal{O}(n)$ complexity the relative mean error for all functions is seen in Figure 6.10. Let's look closer at the relative mean error of ONS-SC-D, FTAL-SC-D and OGD-SC in Figure 6.13.

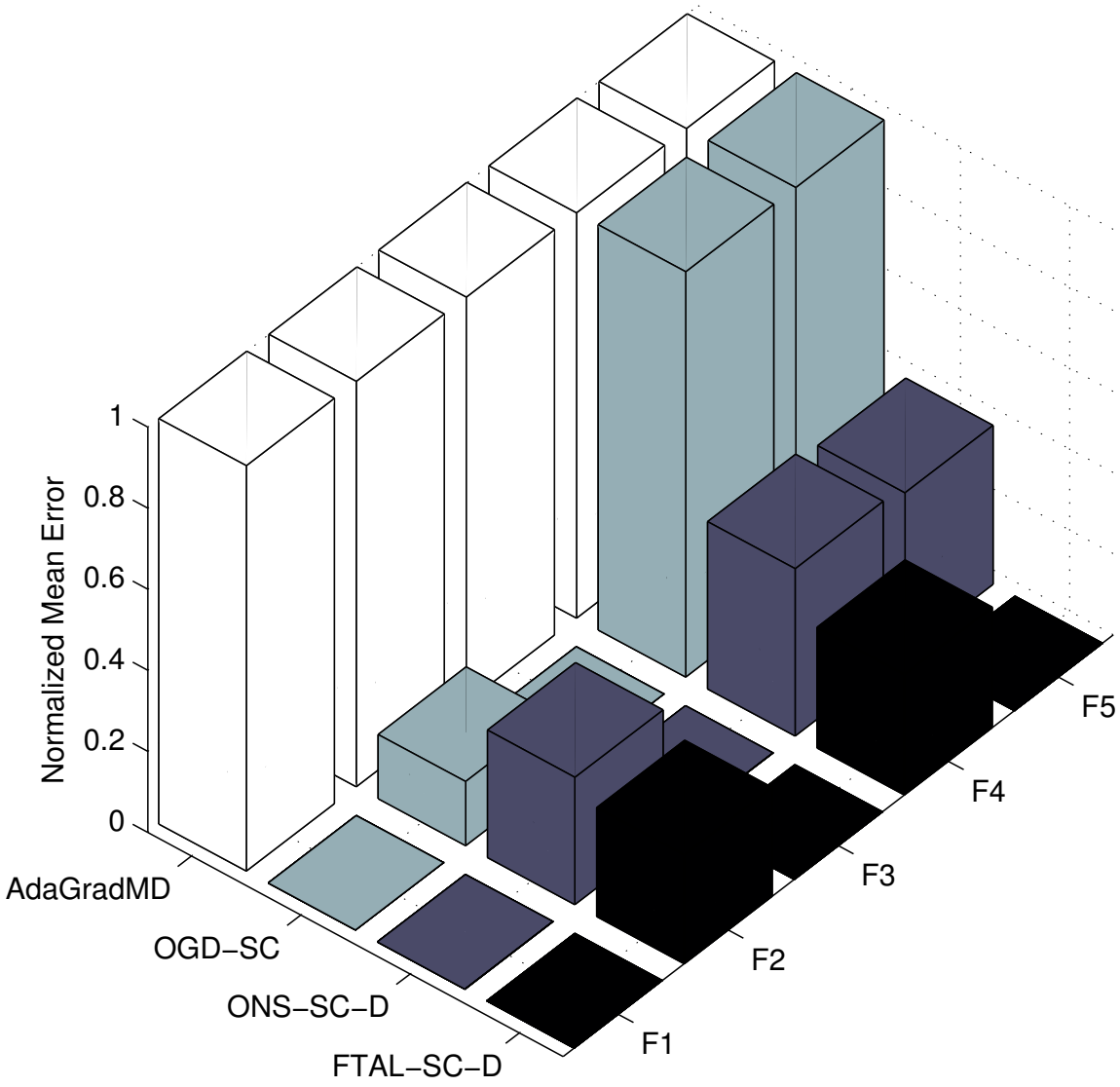
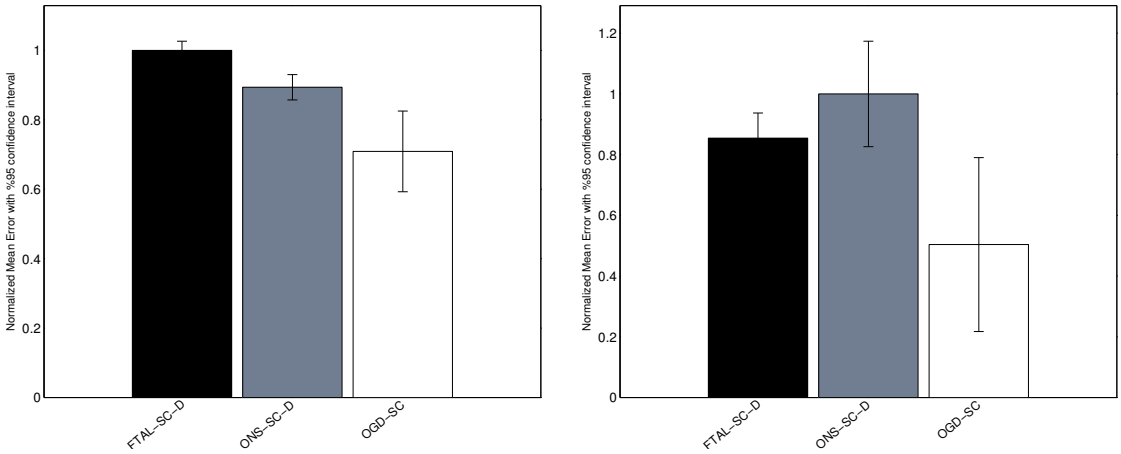
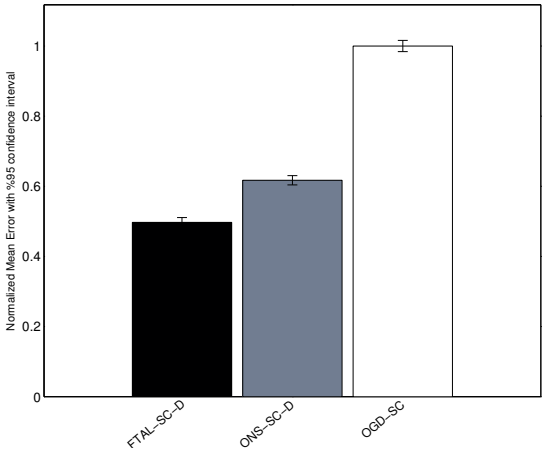


Figure 6.12: Mean error for $\mathcal{O}(n)$ algorithms ($n = 100$)

Figure 6.13 presents similar results to Figure 6.11 however as a result of the increased iterations executed by ONS-SC-D and FTAL-SC-D the performance difference between the algorithms has decreased.



(a) F1 (b) F2



(c) F3

Figure 6.13: Mean error with 95% confidence interval for test functions F1-F3 and $\mathcal{O}(n)$ algorithms ($n = 100$)

In summary, for small scale problems FTAL-SC-D has the lowest mean error for test functions F3, F4 and F5 while OGD-SC has the lowest mean error for test functions F1 and F2. Tables 6.6 and 6.7 both have columns for algorithms AdaGradMD and OGD-SC. The two tables were created from two distinct experiments however the results for AdaGradMD and OGD-SC should be statistically equivalent. The numbers are different because two different sets of test problems were randomly generated.

Table 6.6: Mean error and total iterations with 95% confidence interval for all algorithms ($n = 100$)

		FTAL-SC	ONS-SC	OGD-SC	AdaGradMD	FTAL-EC-I	ONS-EC
F1	$\hat{f}_{error}$	1.12e+00 ±6.01e-02	1.08e+00 ±9.30e-02	4.77e-01 ±7.10e-02	1.73e+05 ±6.42e+03	1.73e+05 ±6.42e+03	1.73e+05 ±6.42e+03
	$\hat{T}_{inter}$	1210.13 ±22.74	1273.36 ±13.23	5815.03 ±2.14	4952.34 ±11.30	1245.24 ±12.62	1284.36 ±8.52
F2	$\hat{f}_{error}$	5.82e+03 ±4.71e+02	7.88e+03 ±9.84e+02	1.69e+03 ±1.15e+03	2.09e+04 ±2.26e+03	1.64e+04 ±1.33e+03	1.62e+04 ±1.36e+03
	$\hat{T}_{inter}$	387.46 ±1.03	393.18 ±0.42	513.34 ±0.48	507.82 ±0.64	390.44 ±0.34	394.21 ±0.37
F3	$\hat{f}_{error}$	1.26e+01 ±3.24e-01	1.57e+01 ±5.41e-01	2.71e+01 ±4.61e-01	1.70e+05 ±7.31e+03	1.70e+05 ±7.31e+03	1.70e+05 ±7.31e+03
	$\hat{T}_{inter}$	1265.99 ±1.22	1326.62 ±0.15	6956.14 ±3.58	6101.77 ±1.62	1297.53 ±0.17	1337.21 ±0.14
F4	$\hat{f}_{error}$	5.17e+03 ±2.14e+03	7.67e+03 ±1.68e+03	1.63e+04 ±2.00e+03	1.63e+04 ±2.00e+03	1.63e+04 ±2.00e+03	1.63e+04 ±2.00e+03
	$\hat{T}_{inter}$	174.15 ±0.15	175.80 ±0.10	197.19 ±0.08	195.83 ±0.10	175.34 ±0.09	176.31 ±0.09
F5	$\hat{f}_{error}$	9.12e-03 ±3.64e-04	4.01e+05 ±1.40e+04	5.16e+05 ±1.62e+04	5.16e+05 ±1.62e+04	5.16e+05 ±1.62e+04	5.16e+05 ±1.62e+04
	$\hat{T}_{inter}$	1238.61 ±3.62	1299.92 ±0.22	5863.08 ±0.99	5236.04 ±1.18	1273.35 ±0.22	1310.18 ±0.12

Table 6.7: Mean error and total iterations with 95% confidence interval for $\mathcal{O}(n)$ algorithms ($n = 100$)

		FTAL-SC-D	ONS-SC-D	OGD-SC	AdaGradMD
F1	$\hat{f}_{error}$	5.58e-01 ±1.48e-02	4.99e-01 ±2.04e-02	3.96e-01 ±6.49e-02	1.72e+05 ±7.22e+03
	$\hat{T}_{inter}$	5024.71 ±1.96	5179.28 ±2.73	5833.88 ±2.36	4961.58 ±11.74
F2	$\hat{f}_{error}$	5.19e+03 ±5.03e+02	6.07e+03 ±1.05e+03	3.06e+03 ±1.74e+03	1.93e+04 ±2.30e+03
	$\hat{T}_{inter}$	505.76 ±0.55	506.29 ±0.47	512.26 ±0.43	506.27 ±0.51
F3	$\hat{f}_{error}$	1.34e+01 ±3.70e-01	1.66e+01 ±3.58e-01	2.69e+01 ±4.33e-01	1.71e+05 ±6.64e+03
	$\hat{T}_{inter}$	5869.23 ±1.10	6019.10 ±3.02	6954.81 ±3.83	6094.32 ±1.15
F4	$\hat{f}_{error}$	4.89e+03 ±1.87e+03	6.79e+03 ±1.19e+03	1.65e+04 ±1.55e+03	1.65e+04 ±1.55e+03
	$\hat{T}_{inter}$	197.02 ±0.11	197.13 ±0.11	198.98 ±0.10	197.05 ±0.08
F5	$\hat{f}_{error}$	8.75e-04 ±3.93e-05	2.00e+05 ±8.68e+03	5.09e+05 ±1.72e+04	5.09e+05 ±1.72e+04
	$\hat{T}_{inter}$	4977.00 ±1.30	5104.76 ±0.93	5841.57 ±1.10	5197.40 ±0.91

6.4.3 Medium Scale ($n = 1000$)

Consider now medium scale problems. Given limitations on computational resources and the results obtained in the previous sections, we only compare the algorithms with $\mathcal{O}(n)$ complexity to the test functions with $\mathcal{O}(n)$ memory requirements. The $\mathcal{O}(n)$ algorithms are compared with test functions F1-F3 in Figure 6.14. From this figure it is clear that FTAL-SC-D has the lowest mean error for test function F2. Also it is clear that OGD-SC, ONS-SC-D and FTAL-SC-D out perform AdaGradMD for all test functions. Figure 6.15 compares algorithms OGD-SC, ONS-SC-D and FTAL-SC-D for test functions F1 and F3.

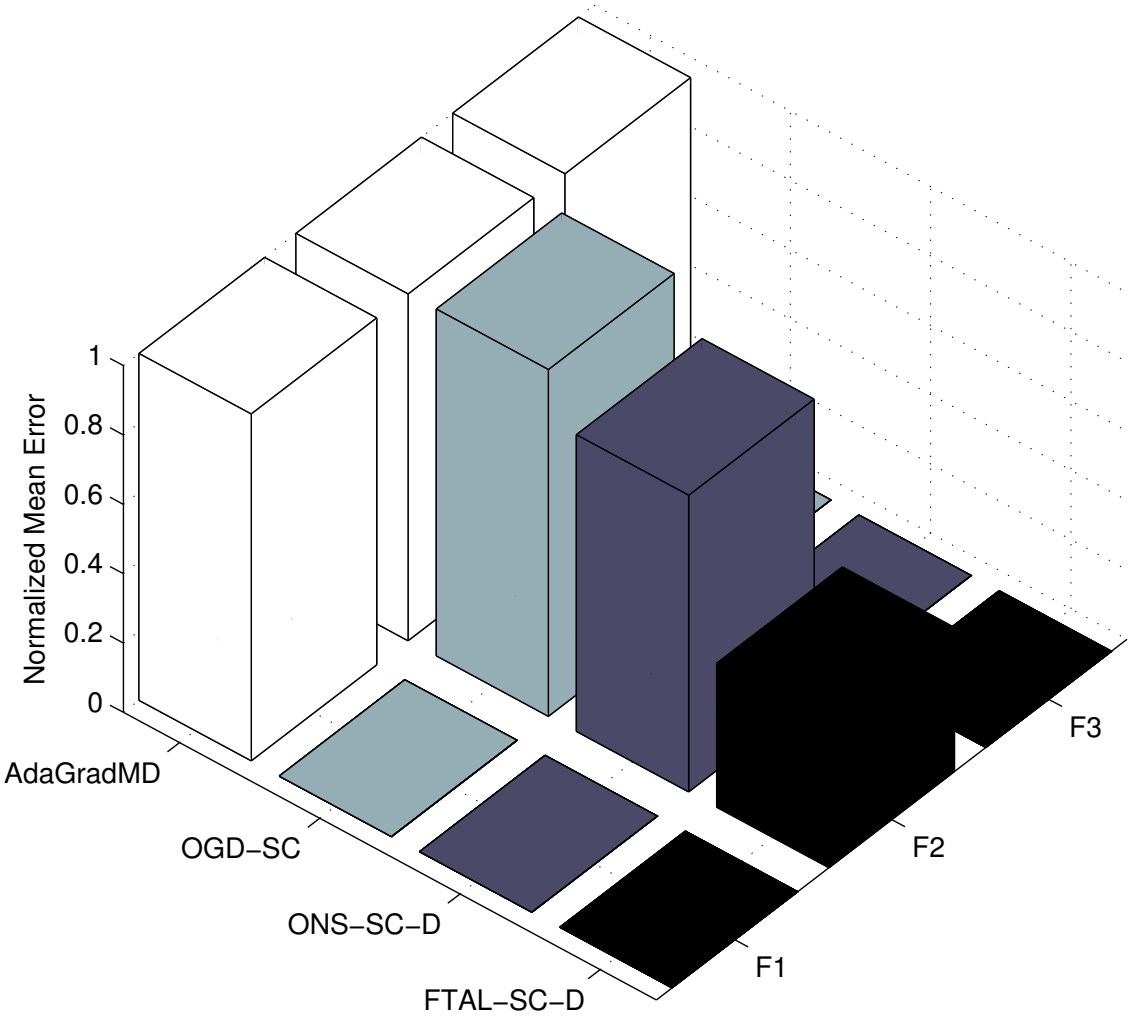
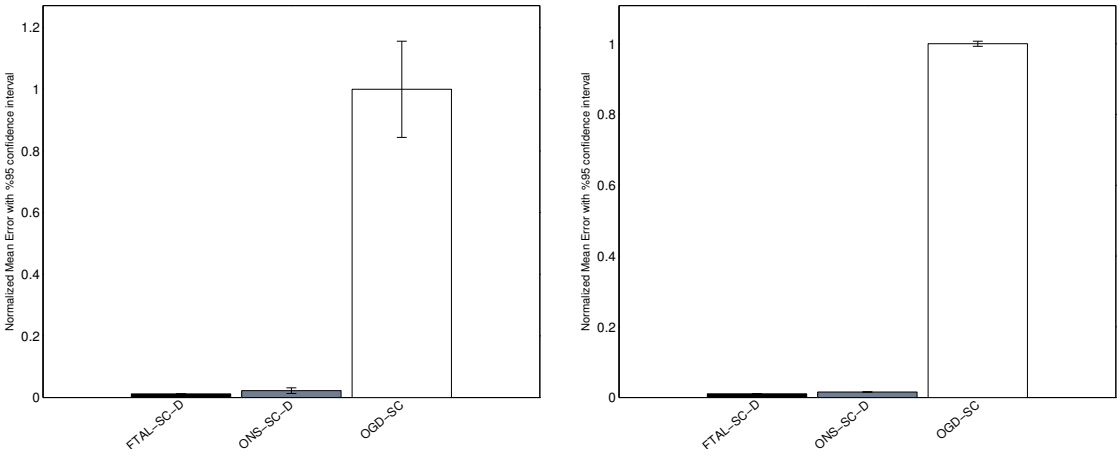


Figure 6.14: Mean error for $\mathcal{O}(n)$ algorithms ($n = 1000$)

Figure 6.15 shows that FTAL-SC-D has the lowest mean error for F1 and F3 followed by ONS-SC-D and then OGD-SC.



(a) F1 (b) F3

Figure 6.15: Mean error with 95% confidence interval for test functions F1, F3 and $\mathcal{O}(n)$ algorithms ($n = 1000$)

In summary, the FTAL-SC-D algorithm has the lowest mean error with test functions F1-F3 for problems of dimension ($n = 1000$). Table 6.8 presents the data, and a 95% confidence interval, used to plot Figures 6.14 and 6.15.

Table 6.8: Mean error and total iterations with 95% confidence interval for $\mathcal{O}(n)$ algorithms ($n = 1000$)

		FTAL-SC-D	ONS-SC-D	OGD-SC	AdaGradMD
F1	$\hat{f}_{error}$	1.64e+00 ±5.15e-02	3.10e+00 ±1.27e+00	1.39e+02 ±2.16e+01	1.67e+08 ±2.03e+06
	$\hat{T}_{inter}$	33830.02 ±19.87	34660.49 ±17.88	19879.64 ±63.09	24385.07 ±131.47
F2	$\hat{f}_{error}$	8.64e+05 ±4.31e+04	1.78e+06 ±2.18e+05	2.08e+06 ±2.24e+05	2.08e+06 ±2.24e+05
	$\hat{T}_{inter}$	359.21 ±0.44	359.00 ±0.34	360.45 ±0.28	358.53 ±0.33
F3	$\hat{f}_{error}$	1.32e+02 ±1.22e+00	1.94e+02 ±1.00e+01	1.24e+04 ±9.02e+01	1.65e+08 ±2.26e+06
	$\hat{T}_{inter}$	34612.84 ±14.27	35270.45 ±13.90	41234.12 ±15.78	36847.17 ±13.33

6.4.4 Large Scale ($n = 10000$)

We now consider solving large scale test problems. The results for all $\mathcal{O}(n)$ algorithms and test functions F1-F3 are shown in Figure 6.16. This figure shows that FTAL-SC-D has the lowest mean error for test function F2. It is also clear that ONS-SC-D and FTAL-SC-D have lower mean error than algorithms AdaGradMD and OGD-SC for test functions F1 and F3. In Figure 6.17 we compare the performance of algorithms ONS-SC-D and FTAL-SC-D on test functions F1 and F3.

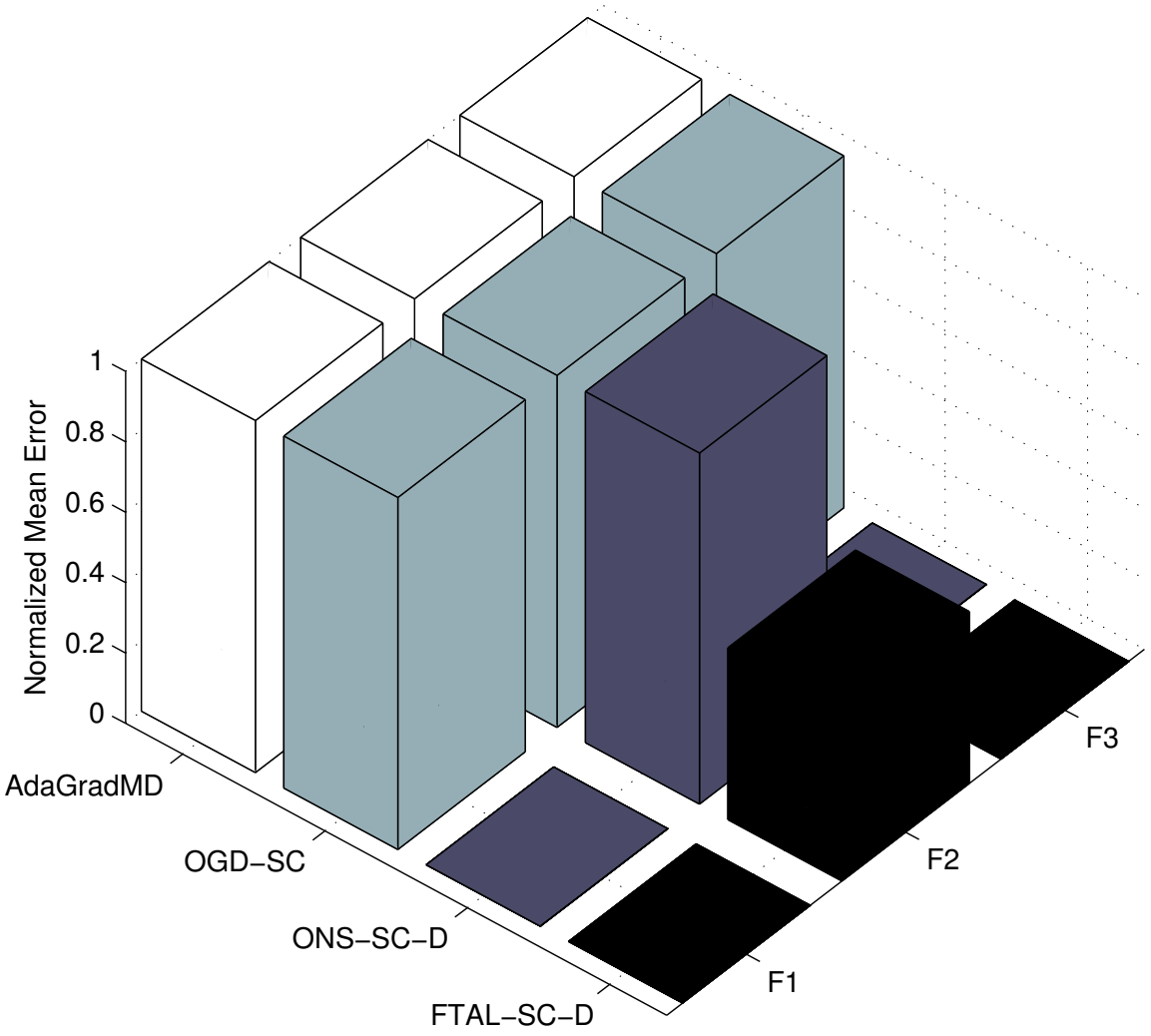
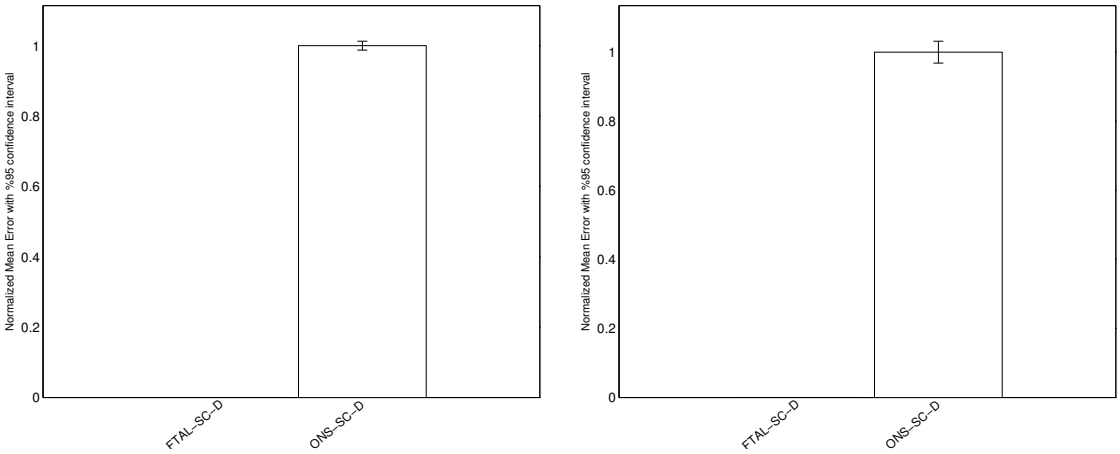


Figure 6.16: Mean error for $\mathcal{O}(n)$ algorithms ($n = 10000$)

Figure 6.17 shows that for large scale problems FTAL-SC-D has the lowest mean error for test functions F1-F2 followed by ONS-SC-D.



(a) F1 (b) F3

Figure 6.17: Mean error with 95% confidence interval for test functions F1, F3 and $\mathcal{O}(n)$ algorithms ($n = 10000$)

In summary, the FTAL-SC-D algorithm has the lowest mean error on test functions F1-F3 for problems of dimension ($n = 10000$). Table 6.9 presents the data, and a 95% confidence interval, used to compose Figures 6.16 and 6.17.

Table 6.9: Mean error and total iterations with 95% confidence interval for $\mathcal{O}(n)$ algorithms ($n = 10000$)

		FTAL-SC-D	ONS-SC-D	OGD-SC	AdaGradMD
F1	$\hat{f}_{error}$	1.97e+00 ±6.27e-02	2.85e+06 ±3.63e+04	1.66e+11 ±7.41e+08	1.66e+11 ±7.41e+08
	$\hat{T}_{inter}$	110116.13 ±109.78	114223.01 ±117.72	113921.05 ±298.85	55092.68 ±392.19
F2	$\hat{f}_{error}$	1.10e+08 ±4.07e+06	2.26e+08 ±2.35e+07	2.26e+08 ±2.35e+07	2.26e+08 ±2.35e+07
	$\hat{T}_{inter}$	56.61 ±0.32	55.46 ±0.33	57.14 ±0.22	56.18 ±0.25
F3	$\hat{f}_{error}$	1.32e+03 ±3.63e+00	5.29e+06 ±1.68e+05	1.68e+11 ±7.33e+08	1.68e+11 ±7.33e+08
	$\hat{T}_{inter}$	91593.12 ±51.24	94211.67 ±48.31	130984.68 ±187.74	105307.78 ±55.07

6.4.5 Summary

Table 6.10 gives a summary of our numerical results. For very small scale problems ($n = 10$) OGD-SC achieves the lowest mean error for the test functions we considered with the exception of test function F5 where FTAL-SC-D has the lowest mean error. For small scale problems ($n = 100$) FTAL-SC has the lowest mean error for test function F3, FTAL-SC-D for F4-F5 and OGD-SC for F1-F2. For medium scale problems ($n = 1000$) the $\mathcal{O}(n)$ algorithms were used to optimize test functions F1-F3. In all cases the FTAL-SC-D algorithm had the lowest mean error. Finally for large scale problems ($n = 10000$), where again the $\mathcal{O}(n)$ algorithms were used to optimize test functions F1-F3, the FTAL-SC-D algorithm has the lowest mean error.

Table 6.10: Algorithms with lowest mean error for a given function and problem size. NA stands for not applicable.

	$(n = 10)$	$(n = 100)$	$(n = 1000)$	$(n = 10000)$
F1	OGD-SC	OGD-SC	FTAL-SC-D	FTAL-SC-D
F2	OGD-SC	OGD-SC	FTAL-SC-D	FTAL-SC-D
F3	OGD-SC	FTAL-SC	FTAL-SC-D	FTAL-SC-D
F4	OGD-SC	FTAL-SC-D	NA	NA
F5	FTAL-SC-D	FTAL-SC-D	NA	NA

Chapter 7

A Heuristic Algorithm for Online Vehicle Routing

Online vehicle routing seeks to find the optimal time ordering of tasks for each vehicle given an optimal assignment of tasks to vehicles and changing problem parameters. Our abstract model can be interpret as optimizing a fleet of vehicles to pickup packages from customers while using the minimum amount of fuel. Our research draws from multiagent auctioning research [55], vehicle routing research [56] and machine learning [14] to solve the online vehicle routing problem [57]. Our research uses a variant of the classic vehicle routing problem (VRP) where a vehicle is not required to return its start location. This variant is referred to as a wandering vehicle routing problem (WVRP) or exploration problem [58, 55]. This application does not fit within the theoretical framework of Chapters 2 though 6 since the WVRP problem is non-convex. However, our theoretical work was inspired by, and has drawn from, experimental work on online vehicle routing for collaborative unmanned aerial vehicle (UAV) systems. We highlight connections between our heuristic WVRP algorithm and adaptive subgradient based algorithms.

Our first algorithm for solving the WVRP problem was inspired by auction based multiagent routing [55] and led to an experimental implementation demonstrated in summer 2006 [1, 59]. A next generation collaborative control algorithm was developed and then implemented in summer 2009 [60, 57, 61]. This new algorithm combined multiagent auctioning and optimization based vehicle routing [56]. A simplified form of the core algorithm demonstrated in summer 2009 is presented here as a heuristic algorithm for online vehicle routing.

Within the last decade there has been significant effort devoted to understanding and solving different variations of online vehicle routing problems for unmanned aerial vehicles. Researchers have proposed robust task allocation algorithms [62] with complex constraints such as time windows [63]. In the following sections, we highlight connections between our heuristic and subgradient based algorithms for online convex optimization. Online convex optimization has been applied to the online shortest path problem. As in our formulation,

edge weights are allowed to change over time [39]. However, unlike the shortest path problem with positive weights, the VRP and the WVRP are NP-hard [58, 55].

The TSP is a well studied problem and our research does not seek to create better solutions for single vehicle routing. Instead we solve a sequence of single vehicle routing problems, using known methods, to solve the online WVRP. A significant body of research exists to solve the classic VRP such as that of Fisher [64] that draws on the work of Held and Karp [65] on the TSP.

Given that WVRP is NP-hard, approximations are used to obtain a practical solution. We first present a formal definition of WVRP in Section 7.1. In Section 7.2 decomposition is used to pose the WVRP as a relatively low dimensional function subject to relatively few constraints. In Section 7.3 we approximate this function with a piecewise affine function. Finally, we formally define the online vehicle routing problem in Section 7.4 and then present a heuristic algorithm for online vehicle routing in Section 7.5.

7.1 Vehicle Routing Problem

We first present a time invariant Binary Integer Programming (BIP) formulation of the WVRP. As stated previously, this problem is NP-hard. Let there be R vehicles, m task locations in Euclidean space and let $n \equiv R \times m$. Let the binary vector $\mathbf{z} \in \{0,1\}^n$ represent an allocation of tasks to vehicles and the binary vector $\mathbf{e} \in \{0,1\}^{R\times(m^2+m)}$ represent the ordering of tasks for each vehicle. If $z^r(i) = 1$ then task i will be executed by vehicle r, otherwise $z^r(i) = 0$. The decision variable $e^r(i,j) = 1$ if vehicle r will visit location j after location i, otherwise $e^r(i,j) = 0$. The cost $c^r(i,j)$ is the Euclidean distance for vehicle r to travel to location j from location i. This problem has no resource constraints and therefore is always feasible [56, 66]. Consider the following Binary Integer Programming formulation

of the WVRP:

$$\underset{\mathbf{e},\mathbf{z}}{\text{minimize}} \quad \sum_{r=1}^{R} \sum_{i} \sum_{j} c^r(i,j) e^r(i,j) \tag{7.1}$$

$$\text{subject to:} \quad \sum_{r=1}^{R} z^r(i) = 1 \qquad \forall i \in \{1,\ldots,m\} \tag{7.2}$$

For each vehicle $\forall r \in \{1,\ldots,R\}$ then

$$\sum_{i=1, i\neq j}^{m} e^r(i,j) + e^r(r,j) = z^r(j) \qquad \forall j \in \{1,\ldots,m\} \tag{7.3}$$

$$\sum_{i=1, i\neq j}^{m} e^r(j,i) \leq z^r(j) \qquad \forall j \in \{1,\ldots,m\} \tag{7.4}$$

$$\sum_{i=1}^{m} e^r(r,i) \leq 1 \tag{7.5}$$

$$\sum_{i,j\in\mathbf{S}} e^r(i,j) \leq |\mathbf{S}| - 1 \qquad \forall \mathbf{S} \subseteq \{1,\ldots,m\} : |\mathbf{S}| \geq 2 \tag{7.6}$$

$$\mathbf{z} \in \{0,1\}^n \qquad \mathbf{e} \in \{0,1\}^{R\times(m^2+m)} \tag{7.7}$$

Constraints (7.2) require that each task is allocated to one and only one vehicle; constraints (7.3) require that if a task is allocated to a vehicle then the vehicle must enter that task; constraints (7.4) require that if a task is allocated to a vehicle then the vehicle must not exit that task more than once; constraint (7.5) requires that a vehicle must not leave its start location more than once and (7.6) denotes the classic subtour constraints [66]. For an intuitive understanding of an optimal solution for two vehicles and three tasks in $\mathbb{R}^2$ see Figure 7.1.

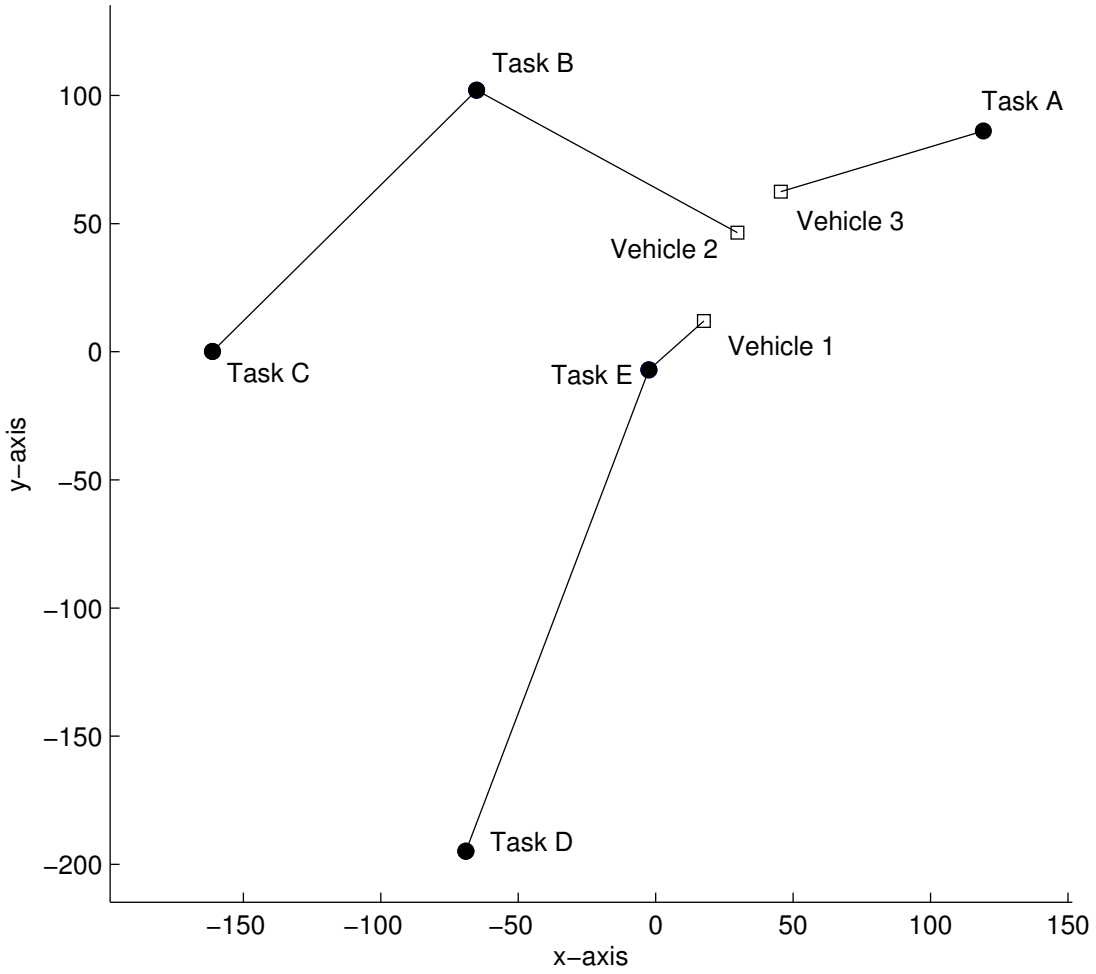


Figure 7.1: A graphical representation of a typical wandering vehicle routing problem with 3 vehicles and 5 tasks in two dimensional Euclidean space

7.2 Decomposition of Vehicle Routing Problem

Problem (7.1)-(7.7) can be reformulated as a function of lower dimension with fewer constraints. However, this function is difficult to evaluate because a Binary Integer Program

(BIP) must be solved. The WVRP can be written equivalently as follows:

$$\underset{\mathbf{z}_r, r\in\{1,\dots,R\}}{\text{minimize}} \quad \sum_{r=1}^{R} g^r(\mathbf{z}^r) \tag{7.8}$$

$$\text{subject to:} \quad \sum_{r=1}^{R} \mathbf{z}^r = \mathbf{1} \tag{7.9}$$

$$\mathbf{z}^r \in \{0,1\}^m \qquad \forall r \in \{1,\dots,R\} \tag{7.10}$$

where $\mathbf{z}^r = [z^r(1),\dots,z^r(m)]^\top$ for each vehicle $r \in \{1,\dots,R\}$. The function g^r is defined as the optimization problem:

$$g^r(\mathbf{z}^r) \equiv \underset{\mathbf{e}^r}{\text{minimize}} \quad \mathbf{c}^{r\top}\mathbf{e}^r \tag{7.11}$$

$$\text{subject to:} \quad \mathbf{D}^r\mathbf{e}^r = \mathbf{z}^r \tag{7.12}$$

$$\mathbf{B}^r\mathbf{e}^r \leq \mathbf{z}^r \tag{7.13}$$

$$\mathbf{H}^r\mathbf{e}^r \leq \mathbf{h}^r \tag{7.14}$$

$$\mathbf{e}^r \in \{0,1\}^{m^2} \tag{7.15}$$

where $\mathbf{z}^r$ is given. The constraints (7.12)-(7.14) describe a single vehicle routing problem and correspond to the constraints (7.3)-(7.6) for a given vehicle $r \in \{1,\dots,R\}$. Constraints (7.12) corresponds to constraints (7.3) and require that a vehicle enter an assigned task. Constraints (7.13) correspond to constraints (7.4) and require that a vehicle leave an assigned task no more than once. Constraints (7.14) correspond to constraints (7.5)-(7.6) and require that a vehicle leave its start location no more than once and eliminates all possible subtours. The tasks assigned to vehicle r is given by the binary vector $\mathbf{z}^r$. For any feasible task assignment to vehicle r, the problem (7.11)-(7.15) is always feasible because tasks can always be ordered [56]. To simplify notation we define the set:

$$\mathbf{Z} \equiv \left\{ [\mathbf{z}^{1\top},\dots,\mathbf{z}^{R\top}]^\top \in \{0,1\}^n \,\middle|\, \sum_{r=1}^{R} \mathbf{z}^r = \mathbf{1} \right\} \tag{7.16}$$

Given set (7.16) the original problem (7.1)-(7.7) can be written compactly as

$$\min_{\mathbf{z}\in\mathbf{Z}} g(\mathbf{z}) \tag{7.17}$$

where

$$g(\mathbf{z}) \equiv \sum_{r=1}^{R} g^r(\mathbf{z}^r) \tag{7.18}$$

Note that this function is decoupled and satisfies Assumption 3.

7.3 Heuristic Convex Approximation

We will develop a heuristic convex approximation of function (7.18). This convex approximation is not guaranteed to be a lower bound of the function. Attempts to develop practical Lagrangian cuts for this function have not been successful [56, 67]. Our heuristic approximation has drawn from multiagent auctioning research [55] and heuristics to solve the vehicle routing problem [56]. Although it is unclear if there is a direct connection between our approximation and Lagragian duality theory, it is clear that the goal of both methods is to obtain a "price" for each vehicle's task assignment.

We now describe the procedure for constructing the piecewise affine approximation of function (7.18). First the problem (7.11)-(7.15) is solved for a given allocation $\mathbf{z}_i \in \mathbf{Z}$ with an efficient TSP heuristic. This heuristic is chosen to meet the speed and accuracy requirements of a specific application.

Let the feasible solution corresponding to the allocation $\mathbf{z}_i^r$ obtained by the efficient heuristic be denoted by $\hat{\mathbf{e}}_i^r$. The heuristic "prices" for the allocation $\mathbf{z}_i^r$ are calculated as follows

$$\hat{\mathbf{y}}_i^r = \mathbf{D}^r(\mathbf{c}^r \circ \hat{\mathbf{e}}_i^r) \tag{7.19}$$

where $\circ$ is the Hadamard (elementwise) product. The nonzero elements of the vector $\mathbf{c}^r \circ \hat{\mathbf{e}}_i^r$ are the cost edges traversed by each vehicle. The matrix $\mathbf{D}^r$ maps the edges traversed while entering a task to the elements of $\hat{\mathbf{y}}_i^r$ corresponding to vehicle's assigned tasks. If a vehicle is not assigned to a task that element is zero. Using these "prices" we obtain the convex approximation $\hat{f} : \mathcal{X} \to \mathbb{R}$ of function (7.18) such that

$$\hat{f}(\mathbf{x}) = \sum_{r=1}^{R} \hat{f}^r(\mathbf{x}^r) \tag{7.20}$$

$$= \sum_{r=1}^{R} \max_{i=1,\ldots,K} \mathbf{c}_i^{r\top} \hat{\mathbf{e}}_i^r + \hat{\mathbf{y}}_i^{r\top}(\mathbf{x}^r - \mathbf{z}_i^r) \tag{7.21}$$

where $\mathcal{X}$ is the convex hull of the point in $\mathbf{Z}$, that is $\mathbf{Z} \subseteq \mathcal{X}$.

7.4 Online Vehicle Routing Problem

The WVRP in Section 7.1 is not time varying. We now extend this formulation to a time varying problem and relate it to online convex optimization. Online convex optimization, by Definition 1, can accommodate a time varying objective function but not time varying constraints. First consider the non-convex vehicle routing problem denoted by problem (7.17). We assume the VRP constraints remain unchanged over time but that a new cost

vector $\mathbf{c}_t$ is given at each iteration. Consider, for a given $\mathbf{z} \in \mathbf{Z}$ at time t, the R parallel wandering TSPs denoted by the function:

$$\begin{aligned} g_t(\mathbf{z}) \equiv \underset{\mathbf{e}}{\text{minimize}} \quad & \mathbf{c}_t^\top \mathbf{e} && (7.22)\\ \text{subject to:} \quad & \mathbf{De} = \mathbf{z} && (7.23)\\ & \mathbf{Be} \leq \mathbf{z} && (7.24)\\ & \mathbf{He} \leq \mathbf{h} && (7.25)\\ & \mathbf{e} \in \{0,1\}^{R\times(m^2+m)} && (7.26) \end{aligned}$$

where the elements $\mathbf{c}_t$ are the cost to traverse an edge at time t. By allowing the vector $\mathbf{c}_t$ to change over time, we consider the case where task locations, vehicle start locations and vehicle end locations may change over time. If fuel is the cost metric then our online vehicle routing formulation minimizes the total fuel consumed by all vehicles over all time for a fixed set constraints. We desire to optimize the infinite sequence of non-convex function $\{g_1, g_2, \dots\}$ with an algorithm $\mathcal{A}$. The regret $R_T(\mathcal{A})$ of the online vehicle routing problem is

$$\sum_{t=1}^{T} g_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathbf{Z}} \sum_{t=1}^{T} g_t(\mathbf{x}) = R_T(\mathcal{A}) \tag{7.27}$$

Given the functions $\{g_1, g_2, \dots\}$ are non-convex and the set $\mathbf{Z}$ defined by (7.16) is not convex this is not an online convex optimization problem by Definition 1. To help solve this problem we approximate the online WVRP problem with the heuristic convex approximation (7.21) subject to non-convex constraints.

$$\sum_{t=1}^{T} \hat{f}_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathbf{Z}} \sum_{t=1}^{T} \hat{f}_t(\mathbf{x}) = \hat{R}_T(\mathcal{A}) \tag{7.28}$$

Given that the set $\mathbf{Z}$ defined by (7.16) is not a convex set this is still not a online convex optimization as defined by Definition 1. Given the non-convex constraints, we present a heuristic algorithm below, in Section 7.5, to solve the online WVRP.

7.5 A Heuristic Algorithm for Online Vehicle Routing

Our heuristic algorithm has a similar form to those presented in Chapters 4 and exploits the structure of Assumption 3 to create a parallelized algorithm. The algorithm includes a ℓ_2-norm projection onto a non-convex set. Projections onto non-convex sets are in general difficult but given the problem's special structure we show that it can be solved efficiently.

This algorithm was validated with numerical and experimental results. The algorithm was compared to a multiagent auctioning algorithm and the optimal solution [61]. The

algorithm was also implemented and demonstrated on a fleet of fully autonomous unmanned aerial vehicles in 2009 [60]. The author's involvement with this research led to our work on quasi-Newton algorithms for online convex optimization.

Algorithm 12 Heuristic for Online WVRP with Quadratic BIP

Require: $\mathbf{x}_1 \in \mathbf{Z}$, $\mathbf{D}_0 = \mathbf{I}_n$ and $\mathbf{b}_0 = 0$

1: **for** $t = 1$ to T **do**
2: **for** $r = 1$ to R **do**
3: $\mathbf{y}_t^r \in \partial \hat{f}_t^r(\mathbf{x}_t^r)$
4: $\mathbf{D}_t^r = \begin{bmatrix} x_t^r(1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & x_t^r(m) \end{bmatrix} + \mathbf{D}_{t-1}^r$
5: $\mathbf{b}_t^r = -\mathbf{y}_t^r + \mathbf{b}_{t-1}^r$
6: $\mathbf{z}_{t+1}^r = (\mathbf{D}_t^r)^{-1}\mathbf{b}_t^r$
7: **end for**
8: $\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathbf{Z}} \sum_{r=1}^{R} (\mathbf{z}_{t+1}^r - \mathbf{x}^r)^\top (\mathbf{z}_{t+1}^r - \mathbf{x}^r)$
9: **end for**

The ℓ_2-norm projection step onto a non-convex set is a quadratic binary integer program. We show below that this quadratic binary integer program can also be solved in parallel with m sorting algorithms. We first prove Lemma 13 that highlights a special property of the set (7.16).

Lemma 13. *Let the set* $\mathbf{Z}$ *be defined by (7.16) then* $||\mathbf{z}||_2^2 = m$ *for all* $\mathbf{z} \in \mathbf{Z}$.

Proof. Using the definition of $\mathbf{Z}$ then $z^r(i) \in \{1, 0\}$ and

$$\sum_{r=1}^{R} z^r(i) = 1 \tag{7.29}$$

Therefore $[z^r(i)]^2 = z^r(i)$ and

$$\begin{aligned} ||\mathbf{z}||_2^2 &= \sum_{r=1}^{R}\sum_{i=1}^{m} [z^r(i)]^2 && (7.30)\\ &= \sum_{i=1}^{m}\sum_{r=1}^{R} z^r(i) && (7.31)\\ &= \sum_{i=1}^{m} 1 && (7.32)\\ &= m && (7.33) \end{aligned}$$

□

The projection step of Algorithm 12 can be written as

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathbf{Z}} \sum_{r=1}^{R} (\mathbf{z}_{t+1}^r - \mathbf{x}^r)^\top (\mathbf{z}_{t+1}^r - \mathbf{x}^r) \tag{7.34}$$

$$= \arg\min_{\mathbf{x}\in\mathbf{Z}} \sum_{r=1}^{R} \mathbf{x}^{r\top}\mathbf{x}^r - 2\mathbf{b}_t^{r\top}(\mathbf{D}_t^r)^{-1}\mathbf{x}^r + \mathbf{b}_t^{r\top}(\mathbf{D}_t^r)^{-2}\mathbf{b}_t^r \tag{7.35}$$

For all $\mathbf{x} \in \mathbf{Z}$ by Lemma 13 we have $\mathbf{x}^\top\mathbf{x} = m$ and therefore dropping all constant terms and coefficients we have

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathbf{Z}} \sum_{r=1}^{R} -\mathbf{b}_t^{r\top}(\mathbf{D}_t^r)^{-1}\mathbf{x}^r \tag{7.36}$$

which is a BIP with totally unimodular constraints and therefore solvable as the following linear program

$$\underset{\mathbf{x}}{\text{minimize}} \qquad \sum_{i=1}^{m}\sum_{r=1}^{R} -\frac{b^r(i)}{d^r(i)} x^r(i) \tag{7.37}$$

$$\text{subject to:} \qquad \sum_{r=1}^{R} x^r(i) = 1 \qquad \forall i \in \{1,\dots,m\} \tag{7.38}$$

$$x^r(i) \geq 0 \qquad \forall i \in \{1,\dots,m\} \tag{7.39}$$

Given decoupled constraints, by inspection, the linear program is solved with m parallel problems. For each $i \in \{1,\dots,m\}$ we have

$$r^*(i) = \arg\min_{r\in\{1,\dots,R\}} [-\frac{b^r(i)}{d^r(i)}] \tag{7.40}$$

and then for each task $i \in \{1,\dots,m\}$ and each vehicle $r \in \{1,\dots,R\}$

$$x^r(i) = \begin{cases} 1, & \text{if } r = r^*(i) \\ 0, & \text{if } r \neq r^*(i) \end{cases} \tag{7.41}$$

This shows that the ℓ_2-norm projection onto the non-convex set (7.16) can be solved with a set of m parallel sorting algorithms. The complete heuristic algorithm for online vehicle routing with sorting is as follows.

Algorithm 13 Heuristic for Online WVRP with Sorting

Require: $\mathbf{x}_1 \in \mathbf{Z}$, $d_0^r(i) = 1 \quad \forall i \in \{1, \dots m\} \quad$ and $\quad \forall r \in \{1, \dots R\}$ and $\mathbf{b}_0 = 0$

for $t = 1$ to T **do**
  for $r = 1$ to R **do**
    $\mathbf{y}_t^r \in \partial \hat{f}_t^r(\mathbf{x}_t^r)$
    $\mathbf{d}_t^r = \mathbf{x}_t^r + \mathbf{d}_{t-1}^r$
    $\mathbf{b}_t^r = -\mathbf{y}_t^r + \mathbf{b}_{t-1}^r$
  end for
  $\mathbf{x}_{t+1} = \mathbf{0}$
  for $i = 1$ to m **do**
    $r^* = \arg\min_{r \in \{1,\dots,R\}} [-\frac{b^r(i)}{d^r(i)}]$
    $x_{t+1}^{r^*}(i) = 1$
  end for
end for

As previously discussed, allowing for parallelism, Algorithm 13 calculates updates in $\mathcal{O}(m)$ space and time not including the projection step. Also, we have shown that the projection step can be solved with a sorting algorithm. Using a standard sorting implementation problem (7.40) can be solved in $\mathcal{O}(R \log(R))$. The algorithmic updates are coupled only by the tasks and the projection step is coupled only by the vehicles. This shows that Algorithm 13 is highly parallelized while still enabling collaboration between vehicles.

Chapter 8

Conclusions

We have presented subgradient based methods and online convex optimization as a promising framework for solving online, structured and large scale optimization problems. We presented previous work on logarithmic regret algorithms and then derived specialized ONS and FTAL algorithms for strongly convex functions. We then extended the ONS and FTAL algorithms for strongly convex functions to a block diagonal hessian approximation. This allows the algorithms to be use for solving large scale problems. In addition, we demonstrated how these algorithms can be parallelized given a summed objective function.

We applied these algorithms to online portfolio optimization with a ℓ_2-norm regularized constant-rebalanced portfolio model that is strongly convex. Also, within a classic optimization framework we performed numerical experiments that compared the performance of our new algorithms to known algorithms on a set of non-smooth strongly convex test problems. These numerical experiments show that in the majority of cases we consider our algorithms out perform other known algorithms. We also presented a heuristic algorithm for online vehicle routing. This online vehicle routing research led to and motivated our work on online convex optimization.

8.1 Further Work

We see at least two future research directions. First, extending the algorithms to a decentralized architecture that can be implemented, with theoretical guarantees, across a communication network with delays. Additionally, we see the potential use of online convex optimization to solve classical control problems.

8.1.1 Algorithm Decentralization

Although we presented parallelized algorithms, more work is necessary to fully distribute the algorithms over a communication network with delays [2]. Given the algorithm's quadratic

form we suggest consideration of a Kalman Filter, or its dual the Information Filter, for algorithm decentralization over a communication network using work such as Nettleton et al. [68]. This approach would require a connected tree network but could result in a robust algorithm for decentralized optimization. It is also interesting to consider adapting distributed dual averaging for convex functions to strongly convex functions [69]. The use of strongly convex functions should result in faster convergence over a connected communication network. Finally, it may be possible to use randomized incremental subgradient methods to extend these algorithm across a communication network [70].

8.1.2 Online Convex Optimization and Control

We recommend the consideration of the online convex optimization formalism for classic control problems. Classical control problems include applications to the low level control of dynamic systems, such as cars and aircraft. As discussed in Section 2.1.1, if a function does not change through time, then an algorithm can be developed that will converge to the optimal solution of a single convex function, that is, the bound will converge to zero. However, if the function is allowed to change arbitrarily over time, then the regret bound will grow at a bounded rate. What if we were to construct a model of how the function changes over time? Can we obtain a bound that does not converge to zero, or grow to infinity, but instead the bound changes with the function's variation and evolves through time? In Chapter 3 we took a first step in this direction by analyzing the case where the strongly convex parameters change over time. Function propagation models or other highly structured optimization formulations may result in faster methods applicable to classical real-time control problems with time constants of ~ 20 milliseconds. Also, inspired by the online vehicle routing problem, it is interesting to consider generalizing online convex optimization to a sequence of time varying convex functions with *time varying* convex constraints.

Bibliography

[1] A. Ryan, J. Tisdale, M. F. Godwin, D. Coatta, D. Nguyen, S. Spry, R. Sengupta, and J. Hedrick, "Decentralized control of unmanned aerial vehicle collaborative sensing missions," in *American Control Conference*, pp. 4672–4677, 2007.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: Numerical methods*. Prentice Hall, 1989. Republished in 1997 by Athena Scientific.

[3] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," *Machine Learning*, vol. 20, no. 2, p. 928, 2003.

[4] E. Hazan, A. Kalai, S. Kale, and A. Agarwal, "Logarithmic regret algorithms for online convex optimization," *Learning Theory*, pp. 499–513, 2006.

[5] S. Kakade and S. Shalev-Shwartz, "Mind the duality gap: Logarithmic regret algorithms for online optimization," *Advances in Neural Information Processing Systems*, vol. 22, 2008.

[6] S. Kakade, S. Shalev-Shwartz, and A. Tewari, "On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization," *Unpublished Manuscript, http://ttic.uchicago.edu/shai/papers/KakadeShalevTewari09.pdf*, 2009.

[7] S. Kakade and A. Tewari, "On the generalization ability of online strongly convex programming algorithms," *Advances in Neural Information Processing Systems*, vol. 22, 2009.

[8] Y. Nesterov, *Introductory lectures on convex optimization: a basic course*. Kluwer Academic Publishers, 2004.

[9] H. Robbins, "Asymptotically subminimax solutions of compound statistical decision problems," in *Second Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 131–149, 1951.

[10] J. Hannan, "Approximation to bayes risk in repeated play," *Contributions to the Theory of Games*, vol. 3, pp. 97–139, 1957.

[11] N. Merhav and M. Feder, "Universal sequential learning and decision from individual data sequences," in *5th Annual Workshop on Computational Learning Theory*, pp. 413–427, 1992.

[12] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[13] A. Gaivoronski and F. Stella, "Stochastic nonstationary optimization for finding universal portfolios," *Annals of Operations Research*, vol. 100, pp. 165–188, 2000.

[14] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Machine Learning*, vol. 69, pp. 169–192, 2007.

[15] R. Fletcher, "A new approach to variable metric algorithms," *The Computer Journal*, vol. 13, no. 3, p. 317, 1970.

[16] W. Schirotzek, "Shor, nz, minimization methods for non-differentiable functions," *Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik, Translation from the Russian*, vol. 66, no. 11, pp. 575–575, 1986.

[17] D. Yudin and A. Nemirovski, "Evaluation of the information complexity of mathematical programming problems," *Russian, Ekonomika i Matematicheskie Metody*, vol. 12, pp. 128–142, 1976.

[18] A. Lewis and M. Overton, "Nonsmooth optimization via bfgs," *SIAM Journal on Optimization, submitted for publication*, 2009.

[19] A. Skajaa, "Limited memory bfgs for nonsmooth optimization," Master's thesis, New York University, 2010.

[20] J. Yu, S. Vishwanathan, S. Günter, and N. Schraudolph, "A quasi-newton approach to nonsmooth convex optimization problems in machine learning," *The Journal of Machine Learning Research*, vol. 11, pp. 1145–1200, 2010.

[21] A. Bordes, L. Bottou, and P. Gallinari, "Sgd-qn: Careful quasi-newton stochastic gradient descent," *The Journal of Machine Learning Research*, vol. 10, pp. 1737–1754, 2009.

[22] J. More, "The levenberg-marquardt algorithm: implementation and theory," *Numerical analysis*, pp. 105–116, 1978.

[23] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," in *23rd Annual Conference on Learning Theory*, 2010.

[24] Y. Nesterov, "Primal-dual subgradient methods for convex problems," *Mathematical Programming*, vol. 120, no. 1, pp. 221–259, 2009.

[25] L. Xiao, "Dual averaging methods for regularized stochastic learning and online optimization," *Journal of Machine Learning Research*, vol. 9999, pp. 2543–2596, December 2010.

[26] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical Programming*, vol. 103, pp. 127–152, 2005.

[27] A. Nemirovski and D. Yudin, *Problem Complexity and Efficiency in Optimization*. John Wiley, 1983.

[28] A. Iouditski and Y. Nesterov, "Primal-dual subgradient methods for minimizing uniformly convex functions." http://hal.archives-ouvertes.fr/docs/00/50/89/33/PDF/Strong-hal.pdf, 2010.

[29] E. Hazan and S. Kale, "Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization," in *24th Annual Conference on Learning Theory*, 2011.

[30] J. Duchi and Y. Singer, "Efficient online and batch learning using forward backward splitting," *The Journal of Machine Learning Research*, vol. 10, pp. 2899–2934, 2009.

[31] J. Duchi, S. Shalev-Shwartz, Y. Singer, A. Tewari, and T. Chicago, "Composite objective mirror descent," in *23rd Annual Conference on Computational Learning Theory*, 2010.

[32] A. Beck and M. Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex optimization," *Operations Research Letters*, vol. 31, pp. 167–175, May 2003.

[33] H. McMahan and M. Streeter, "Adaptive bound optimization for online convex optimization," *Arxiv preprint arXiv:1002.4908*, 2010.

[34] P. Bartlett, E. Hazan, and A. Rakhlin, "Adaptive online gradient descent," *Advances in Neural Information Processing Systems*, vol. 21, 2007.

[35] S. Shalev-Shwartz and Y. Singer, "Logarithmic regret algorithms for strongly convex repeated games," tech. rep., Hebrew University, 2007.

[36] S. Shalev-Shwartz, *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, Hebrew University, 2007.

[37] E. Hazan and S. Kale, "On stochastic and worst-case models for investing," *Advances in Neural Information Processing Systems (NIPS)*, vol. 22, pp. 709–717, 2010.

[38] E. Hazan and S. Kale, "Extracting certainty from uncertainty: regret bounded by variation in costs," *Machine Learning*, vol. 80, pp. 165–188, 2010.

[39] A. Kalai and S. Vempala, "Efficient algorithms for online decision problems," *Journal of Computer and System Sciences*, vol. 71, no. 3, pp. 291–307, 2005.

[40] D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, 1973.

[41] M. Brookes, "The matrix reference manual," *Imperial College London*, 2005.

[42] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge University Press, 1990.

[43] D. Bertsekas, "Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey," *Lab. for Information and Decision Systems Report LIDSP-2848, MIT*, 2010.

[44] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, March 2004.

[45] T. Cover, "Universal portfolios," *Mathematical Finance*, vol. 1, no. 1, pp. 1–29, 1991.

[46] A. Kalai and S. Vempala, "Efficient algorithms for universal portfolios," *The Journal of Machine Learning Research*, vol. 3, pp. 423–440, 2003.

[47] P. Das and A. Banerjee, "Meta optimization and its application to portfolio selection," in *International Conference on Knowledge Discovery and Data Mining*, 2011.

[48] A. Blum and A. Kalai, "Universal portfolios with and without transaction costs," *Machine Learning*, vol. 35, no. 3, pp. 193–205, 1999.

[49] A. Agarwal, E. Hazan, S. Kale, and R. Schapire, "Algorithms for portfolio management based on the newton method," in *23rd International Conference on Machine Learning*, pp. 9–16, ACM, 2006.

[50] J. Bouchaud and M. Potters, *Theory of Financial Risks: From Statistical Physics to Risk Management*, vol. 217. Cambridge University Press, 2000.

[51] Y. Singer and G. Gelencsér, "New york stock exchange datasets." http://www.cs.bme.hu/∼oti/portfolio/data.html, July 2011.

[52] D. Helmbold, R. Schapire, Y. Singer, and M. Warmuth, "On-line portfolio selection using multiplicative updates," *Mathematical Finance*, 1996.

[53] X. Chen and M. Fukushima, "Proximal quasi-newton methods for nondifferentiable convex optimization," *Mathematical Programming*, vol. 85, no. 2, pp. 313–334, 1999.

[54] M. Haarala, K. Miettinen, and M. Mäkelä, "New limited memory bundle method for large-scale nonsmooth optimization," *Optimization Methods and Software*, vol. 19, no. 6, pp. 673–692, 2004.

[55] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing," in *Robotics: Science and Systems*, June 2005.

[56] M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2, pp. 109–124, 1981.

[57] M. F. Godwin, "Robust collaboration of systems," Master's thesis, University of California, Berkeley, CA, USA, Spring 2007.

[58] P. Toth and D. Vigo, *The vehicle routing problem*, vol. 9. Society for Industrial Mathematics, 2002.

[59] M. F. Godwin, S. Spry, and J. K. Hedrick, "Distributed collaboration with limited communication using mission state estimates," in *American Control Conference*, pp. 14–16, 2006.

[60] J. M. Garvey, B. Kehoe, B. Basso, M. F. Godwin, J. Wood, J. Love, S. Liu, Z. Kim, S. Jackson, Y. Fallah, T. Fu, R. Sengupta, and J. K. Hedrick, "An autonomous unmanned aerial vehicle system for sensing and tracking," in *Infotech@Aerospace Conference*, American Institute of Aeronautics and Astronautics, 2011.

[61] M. F. Godwin and J. K. Hedrick, "Stochastic approximation of an online multiagent routing problem for autonomous aircraft," in *Infotech@Aerospace Conference*, American Institute of Aeronautics and Astronautics, March 2011.

[62] M. Alighanbari and J. P. How, "A robust approach to the uav task assignment problem," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 2, pp. 118–134, 2008.

[63] S. Ponda, J. Redding, H. Choi, J. How, M. Vavrina, and J. Vian, "Decentralized planning for complex missions with dynamic communication constraints," in *American Control Conference*, 2010.

[64] M. L. Fisher, "Optimal solution of vehicle routing problems using minimum k-trees," *Operations Research*, vol. 42, no. 4, pp. 626–642, 1994.

[65] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees," *Operations Research*, vol. 18, no. 6, pp. 1138–1162, 1970.

[66] E. L. Lawler, *The Traveling salesman problem : a guided tour of combinatorial optimization*. Wiley, 1985.

[67] M. L. Fisher and R. Jaikumar, "A decomposition algorithm for large-scale vehicle routing," Tech. Rep. 78-11-05, Decision Sciences Working Paper, July 1978.

[68] E. Nettleton, H. Durrant-Whyte, and S. Sukkarieh, "A robust architecture for decentralised data fusion," in *International Conference on Advanced Robotics*, June 2003.

[69] J. Duchi, A. Agarwal, and M. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, p. 1, 2011.

[70] B. Johansson, M. Rabi, and M. Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM Journal on Optimization*, vol. 20, pp. 1157–1170, August 2009.

[71] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 2nd ed., 1999.

[72] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, 1970.

Appendix A

Mathematical Background

In this chapter we present some well known mathematical tools and definitions that are used throughout this work. For a more thorough introduction see [8, 44, 71, 72]. The natural logarithm is denoted by log.

A.1 Norms

Definition 3 (Norm)**.** *A norm* $||\cdot||$ *on* $\mathbb{R}^n$ *is a mapping that assigns a scalar* $||\mathbf{x}||$ *to every* $\mathbf{x} \in \mathbb{R}^n$ *and has the following properties*

- $||\mathbf{x}|| \geq 0$ *for all* $\mathbf{x} \in \mathbb{R}^n$
- $||c\mathbf{x}|| = |c| \cdot ||\mathbf{x}||$ *for every* $c \in \mathbb{R}$ *and every* $\mathbf{x} \in \mathbb{R}^n$
- $||\mathbf{x}|| = 0$ *if and only if* $\mathbf{x} = 0$
- $||\mathbf{x} + \mathbf{z}|| \leq ||\mathbf{x}|| + ||\mathbf{z}||$ *for all* $\mathbf{x}, \mathbf{z} \in \mathbb{R}^n$

Typical norms include the Euclidean norm $||\cdot||_2$, the one norm $||\cdot||_1$, infinity norm $||\cdot||_\infty$ and the general p-norm $||\cdot||_p$. Two norms $||\mathbf{x}||_p$ and $||\mathbf{x}||_q$ are said to be *equivalent* if there exists positive real scalars $\alpha, \beta \in \mathbb{R}$ such that

$$\alpha||\mathbf{x}||_p \leq ||\mathbf{x}||_q \leq \beta||\mathbf{x}||_p$$

For example, let $\mathbf{x} \in \mathbb{R}^n$ then

$$||\mathbf{x}||_2 \leq ||\mathbf{x}||_1 \leq \sqrt{n}||\mathbf{x}||_2$$
$$||\mathbf{x}||_\infty \leq ||\mathbf{x}||_2 \leq \sqrt{n}||\mathbf{x}||_\infty$$

Lemma 14 (Hölder's Inequality)**.** *Let* $||\mathbf{x}||_p$ *and* $||\mathbf{z}||_q$ *be two norms where* $\mathbf{x}, \mathbf{z} \in \mathbb{R}^n$ *and* $\frac{1}{q} + \frac{1}{p} = 1$ *then*

$$|\mathbf{x}^\top \mathbf{z}| \leq ||\mathbf{x}||_q ||\mathbf{z}||_p \tag{A.1}$$

A.2 Generalized Inequalities

Assume $\mathcal{K}$ is a proper cone. When $\mathcal{K} = \mathbb{R}_+$ the partial ordering $\succeq_{\mathcal{K}}$ is the usual ordering $\geq$ on $\mathbb{R}$ and the strict partial ordering $\succ_{\mathcal{K}}$ is the same as the strict partial ordering $>$ on $\mathbb{R}$. Now let the nonnegative orthant cone be denoted by $\mathcal{K} = \mathbb{R}^n_+$. Given $\mathbf{z}, \mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \succeq_{\mathcal{K}} \mathbf{x}$ then $z(i) \geq x(i)$ for all $i \in \{1, \ldots, n\}$. Given two vectors $\mathbf{z}, \mathbf{x} \in \mathbb{R}^n$ the nonnegative orthant cone is denoted as $\mathbf{z} \geq \mathbf{x}$ and the associated strict inequality is denoted as $\mathbf{z} > \mathbf{x}$. The positive semidefinite cone $\mathbb{S}^n_+$ and the positive definite cone $\mathbb{S}^n_{++}$ are proper cones in the set of symmetric matrices $\mathbb{S}^n$. If $\mathbf{A} \succeq \mathbf{B}$ then $\mathbf{A} - \mathbf{B}$ is positive semidefinite and similarly if $\mathbf{A} \succ \mathbf{B}$ then $\mathbf{A} - \mathbf{B}$ is positive definite. An identity matrix of dimension n is denoted by $\mathbf{I}_n \in \mathbb{S}^n_{++}$.

A.3 Convex Analysis

We review a few definitions and concepts from convex analysis.

Definition 4 (Convex set)**.** *A set $\mathcal{X} \in \mathbb{R}^n$ is convex if*

$$\theta\mathbf{x} + (1-\theta)\mathbf{z} \in \mathcal{X} \qquad \forall \mathbf{x}, \mathbf{z} \in \mathcal{X}, \forall \theta \in [0,1] \tag{A.2}$$

Definition 5 (Convex function)**.** *Let $\mathcal{X} \in \mathbb{R}^n$ be a convex set. A function $f : \mathcal{X} \to \mathbb{R}$ is called convex if*

$$f(\theta\mathbf{x} + (1-\theta)\mathbf{z}) \leq \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{z}), \qquad \forall \mathbf{x}, \mathbf{z} \in \mathcal{X}, \forall \theta \in [0,1] \tag{A.3}$$

The function f is strictly convex if the above inequality is strict for all $\mathbf{x}, \mathbf{z} \in \mathcal{X}$ such that $\mathbf{x} \neq \mathbf{z}$ and all $\alpha \in (0,1)$.

The function f is called concave if $-f$ is convex. Jenson's inequality generalizes the basic inequality for convex functions.

Lemma 15 (Jenson's Inequality)**.** *If $f : \mathcal{X} \to \mathbb{R}$ is convex and $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathcal{X}$ and $\theta_1, \ldots, \theta_m \geq 0$ with $\sum_{i=1}^m \theta_i = 1$ then*

$$f(\sum_{i=1}^m \theta_i \mathbf{x}_i) \leq \sum_{i=1}^m \theta_i f(\mathbf{x}_i) \tag{A.4}$$

A.3.1 Gradients, Subgradients, Differential Sets and Optimality Condition

A subgradient is a generalization of the gradient for convex functions and is defined as follows.

Definition 6 (Subgradient)**.** *A vector* $\mathbf{y} \in \partial f(\mathbf{x})$ *is a subgraident of a function* f *at* $\mathbf{x}$ *if*

$$f(\mathbf{z}) \geq f(\mathbf{x}) + \mathbf{y}^\top(\mathbf{z} - \mathbf{x}) \quad \forall \mathbf{z} \in \mathcal{X} \tag{A.5}$$

where $\partial f(\mathbf{x})$ *is the differential set of subgradients, or subdifferential, of* f *at* $\mathbf{x}$*.*

A function is convex if and only if $\partial f(\mathbf{x})$ is a non-empty set. If $f : \mathcal{X} \to \mathbb{R}$ is convex and differentiable at $\mathbf{x}$ then $\partial f(\mathbf{x})$ contains a single vector that is the gradient of f at $\mathbf{x}$ and is denoted by $\nabla f(\mathbf{x})$. If a function is twice differentiable at $\mathbf{x}$ then the hessian is denoted by $\nabla^2 f(\mathbf{x})$. If $f : \mathbb{R} \to \mathbb{R}$ and f is differentiable then the derivative f at $x \in \mathbb{R}$ is denoted by $f'(x)$ and if f is twice differentiable then the second derivative is denoted by $f''(x)$. We will now present some basic properties of the subdifferential that are helpful for calculating subgradients.

- **Scaling:** For $\theta > 0$ we have $\partial(\theta f) = \theta \partial f$
- **Addition:** $\partial(f_1 + f_2) = \partial f_1 + \partial f_2$
- **Affine transformation:** Let $\mathbf{A}$ be a matrix, $\mathbf{b}$ a vector, if $g(\mathbf{x}) = f(\mathbf{A}\mathbf{x} + \mathbf{b})$ then $\partial g(\mathbf{x}) = \mathbf{A}^\top \partial f(\mathbf{A}\mathbf{x} + \mathbf{b})$
- **Pointwise maximum:** If $f(\mathbf{x}) = \max_{i \in [1,\dots,m]} f_i(\mathbf{x})$ then $\partial f(\mathbf{x})$ is the convex hull of the set $\left\{ \bigcup_{i=1}^m \{\partial f_i(\mathbf{x}) | f_i(\mathbf{x}) = f(\mathbf{x})\} \right\}$. That is, the set of subgradients is the convex hull of the the union of subdifferentials of active functions at $\mathbf{x}$.

Lemma 16 (Optimality Condition)**.** *Let* $f : \mathcal{X} \to \mathbb{R}$ *be a convex and differentiable function. Then* $\mathbf{x}^*$ *is a global optimal if and only if* $\mathbf{x}^* \in \mathcal{X}$ *and*

$$\nabla f(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x} \in \mathcal{X} \tag{A.6}$$

A.3.2 Strong Convexity

Strong convexity is formally defined as follows.

Definition 7 (Strong Convexity)**.** *A continuous function* $f : \mathcal{X} \to \mathbb{R}$ *is* H*-strongly convex with respect to a norm* $||\cdot||$ *if for all* $\mathbf{z}, \mathbf{x} \in \mathcal{X}$ *and* $\theta \in [0, 1]$ *we have*

$$f(\theta\mathbf{z} + (1-\theta)\mathbf{x}) \leq \theta f(\mathbf{z}) + (1-\theta) f(\mathbf{x}) - \frac{H}{2}\theta(1-\theta)||\mathbf{z} - \mathbf{x}||^2 \tag{A.7}$$

If a strongly convex function is twice differentiable then there exists a $H > 0$ such that $\nabla^2 f(\mathbf{x}) \succeq \mathbf{I}_n H$. The addition of a differentiable strongly convex function and a convex function results in a strongly convex function as stated in Lemma 17. Lemma 17 follows from [36].

Lemma 17. *Assume that $f : \mathcal{X} \to \mathbb{R}$ is differentiable and H-strongly convex with respect to $||\cdot||$ such that $H > 0$ and $h : \mathcal{X} \to \mathbb{R}$ is convex. If $g = h + f$ then g is H-strongly convex with respect to $||\cdot||$.*

Proof (Lemma 17)**.** Let $\mathbf{x}, \mathbf{z} \in \mathcal{X}$ and $\mathbf{y}_2 \in \partial g(\mathbf{x})$. Since $\partial g(\mathbf{x}) = \partial h(\mathbf{x}) + \partial f(\mathbf{x})$ there exists a $\mathbf{y}_1 \in \partial h(\mathbf{x})$ such that $\mathbf{y}_2 = \mathbf{y}_1 + \nabla f(\mathbf{x})$. Using the convexity of h and g we have

$$g(\mathbf{z}) - g(\mathbf{x}) - \mathbf{y}_2^\top(\mathbf{z} - \mathbf{x}) \geq f(\mathbf{z}) - f(\mathbf{x}) - \nabla f(\mathbf{x})^\top(\mathbf{z} - \mathbf{x}) \tag{A.8}$$

using that f is H-strongly convex we have

$$\begin{aligned} g(\mathbf{z}) - g(\mathbf{x}) - \mathbf{y}_2^\top(\mathbf{z} - \mathbf{x}) &\geq f(\mathbf{z}) - f(\mathbf{x}) - \nabla f(\mathbf{x})^\top(\mathbf{z} - \mathbf{x}) && \text{(A.9)} \\ &\geq \frac{H}{2}||\mathbf{z} - \mathbf{x}||^2 && \text{(A.10)} \end{aligned}$$

and therefore g is H-strongly convex with respect to $||\cdot||$ where $H > 0$.

□

Appendix B

Technical Lemmas

The following results of Hazan et al. [14] are used throughout this work and are presented here for completeness.

Lemma 18. *Let* $\mathbf{v}_t \in \mathbb{R}^n$ *and* $||\mathbf{v}_t||_2 \leq d_t$ *for all* $t \in \{1, \ldots, T\}$ *and* $\mathbf{Q}_T = \sum_{t=1}^{T} \mathbf{v}_t \mathbf{v}_t^\top + \epsilon \mathbf{I}_n$ *then*

$$\sum_{t=1}^{T} \mathbf{v}_t^\top \mathbf{Q}_t^{-1} \mathbf{v}_t \leq n \log \left(\frac{1}{\epsilon} \sum_{t=1}^{T} d_t^2 + 1 \right) \tag{B.1}$$

Proof (Lemma 18)**.** Let $\mathbf{A}, \mathbf{B} \in \mathbb{S}_{++}^n$ be positive definite matrices such that $\mathbf{A} \succeq \mathbf{B} \succ \mathbf{0}$ then given Lemma 12 of [14] we have

$$\text{trace}(\mathbf{A}^{-1}[\mathbf{A} - \mathbf{B}]) \leq \log \left(\frac{\det(\mathbf{A})}{\det(\mathbf{B})} \right) \tag{B.2}$$

now consider

$$\begin{aligned}
\sum_{t=1}^{T} \mathbf{v}_t^\top \mathbf{Q}_t^{-1} \mathbf{v}_t &= \sum_{t=1}^{T} \text{trace}(\mathbf{v}_t^\top \mathbf{Q}_t^{-1} \mathbf{v}_t) && \text{(B.3)} \\
&= \sum_{t=1}^{T} \text{trace}(\mathbf{Q}_t^{-1} \mathbf{v}_t \mathbf{v}_t^\top) && \text{(B.4)} \\
&= \sum_{t=1}^{T} \text{trace}(\mathbf{Q}_t^{-1} [\mathbf{Q}_t - \mathbf{Q}_{t-1}]) && \text{(B.5)} \\
&\leq \sum_{t=1}^{T} \log \left(\frac{\det(\mathbf{Q}_t)}{\det(\mathbf{Q}_{t-1})} \right) && \text{(B.6)} \\
&= \log \left(\frac{\det(\mathbf{Q}_T)}{\det(\mathbf{Q}_0)} \right) && \text{(B.7)}
\end{aligned}$$

Given that $\mathbf{Q}_T = \sum_{t=1}^{T} \mathbf{v}_t \mathbf{v}_t^\top + \epsilon \mathbf{I}_n$ and $||\mathbf{v}_t||_2 \leq d_t$ then $\lambda_{\max}(\mathbf{Q}_T) \leq \sum_{t=1}^{T} d_t^2 + \epsilon$ and therefore $\det(\mathbf{Q}_T) \leq (\sum_{t=1}^{T} d_t^2 + \epsilon)^n$ resulting in

$$\begin{aligned} \sum_{t=1}^{T} \mathbf{v}_t^\top \mathbf{Q}_t^{-1} \mathbf{v}_t &\leq \log\left(\frac{(\sum_{t=1}^{T} d_t^2 + \epsilon)^n}{\epsilon^n}\right) && \text{(B.8)} \\ &\leq n \log\left(\frac{1}{\epsilon}\sum_{t=1}^{T} d_t^2 + 1\right) && \text{(B.9)} \end{aligned}$$

giving our result.

□

Lemma 19. *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex set, $\mathbf{z} \in \mathbb{R}^n$ and $\mathbf{x} = \Pi_{\mathcal{X}}^{Q}(\mathbf{z})$ be the projection of $\mathbf{z}$ onto $\mathcal{X}$ according to the positive semidefinite matrix $\mathbf{Q} \succeq 0$. Then for any point $\mathbf{u} \in \mathcal{X}$*

$$(\mathbf{z}-\mathbf{u})^\top \mathbf{Q}(\mathbf{z}-\mathbf{u}) \geq (\mathbf{x}-\mathbf{u})^\top \mathbf{Q}(\mathbf{x}-\mathbf{u}) \tag{B.10}$$

Proof (Lemma 19)**.** Let $g : \mathcal{X} \to \mathbb{R}$ be a convex function such that $g(\mathbf{u}) = (\mathbf{z}-\mathbf{u})^\top \mathbf{Q}(\mathbf{z}-\mathbf{u})$. By definition of the projection $\mathbf{x} = \Pi_{\mathcal{X}}^{Q}(\mathbf{z})$ the point $\mathbf{x}$ minimizes the convex function g over a convex set and therefore using Lemma 16 we have

$$\nabla g(\mathbf{x})^\top (\mathbf{u}-\mathbf{x}) \geq 0 \qquad \forall \mathbf{u} \in \mathcal{X} \tag{B.11}$$

which implies

$$2(\mathbf{x}-\mathbf{z})^\top \mathbf{Q}(\mathbf{u}-\mathbf{x}) \geq 0 \tag{B.12}$$

and then

$$2(\mathbf{x}-\mathbf{z})^\top \mathbf{Q}\mathbf{u} \geq 2(\mathbf{x}-\mathbf{z})^\top \mathbf{Q}\mathbf{x} \tag{B.13}$$

now consider the relation

$$\begin{aligned} (\mathbf{z}-\mathbf{u})^\top \mathbf{Q}(\mathbf{z}-\mathbf{u}) - (\mathbf{x}-\mathbf{u})^\top \mathbf{Q}(\mathbf{x}-\mathbf{u}) &= \mathbf{z}^\top \mathbf{Q}\mathbf{z} - \mathbf{x}^\top \mathbf{Q}\mathbf{x} + 2\mathbf{u}^\top \mathbf{Q}(\mathbf{x}-\mathbf{z}) && \text{(B.14)} \\ &\geq \mathbf{z}^\top \mathbf{Q}\mathbf{z} - \mathbf{x}^\top \mathbf{Q}\mathbf{x} + 2\mathbf{x}^\top \mathbf{Q}(\mathbf{x}-\mathbf{z}) && \text{(B.15)} \\ &= \mathbf{z}^\top \mathbf{Q}\mathbf{z} + \mathbf{x}^\top \mathbf{Q}\mathbf{x} - 2\mathbf{x}^\top \mathbf{Q}\mathbf{z} && \text{(B.16)} \\ &= (\mathbf{x}-\mathbf{z})^\top \mathbf{Q}(\mathbf{x}-\mathbf{z}) && \text{(B.17)} \end{aligned}$$

Finally given that $\mathbf{Q} \succeq 0$ we have our result.

□