Optimization Methods for Structured Machine Learning Problems

Nikolaos Tsipinakis

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Statistical Science

University College London

February 7, 2019

I, Nikolaos Tsipinakis, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Solving large-scale optimization problems lies at the core of modern machine learning applications. Unfortunately, obtaining a sufficiently accurate solution quickly is a difficult task. However, the problems we consider in many machine learning applications exhibit a particular structure. In this thesis we study optimization methods and improve their convergence behavior by taking advantage of such structures. In particular, this thesis constitutes of two parts:

In the first part of the thesis, we consider the Temporal Difference learning (TD) problem in off-line Reinforcement Learning (RL). In off-line RL, it is typically the case that the number of samples is small compared to the number of features. Therefore, recent advances have focused on efficient algorithms to incorporate feature selection via ℓ_1-regularization which effectively avoids over-fitting. Unfortunately, the TD optimization problem reduces to a fixed-point problem where convexity of the objective function cannot be assumed. Further, it remains unclear whether existing algorithms have the ability to offer good approximations for the task of policy evaluation and improvement (either they are non-convergent or do not solve the fixed-point problem). In this part of the thesis, we attempt to solve the ℓ_1-regularized fixed-point problem with the help of Alternating Direction Method of Multipliers (ADMM) and we argue that the proposed method is well suited to the structure of the aforementioned fixed-point problem.

In the second part of the thesis, we study multilevel methods for large-scale optimization and extend their theoretical analysis to self-concordant functions. In par-

ticular, we address the following issues that arise in the analysis of second-order optimization methods based either on sampling, randomization or sketching: (a) *the analysis of the iterates is not scale-invariant* and (b) *lack of global fast convergence rates without restrictive assumptions*. We argue that, with the analysis undertaken in this part of the thesis, the analysis of randomized second-order methods can be considered on-par with the analysis of the classical Newton method. Further, we demonstrate how our proposed method can exploit typical spectral structures of the Hessian that arise in machine learning applications to further improve the convergence rates.

Impact Statement

In the third chapter we aim to solve the Temporal Difference (TD) learning problem in off-line Reinforcement Learning (RL) which typically is a difficult task and for this reason it cannot find many practical applications specifically when sparsity is required. Our proposed method is tested in a complex environment and our preliminary numerical results show an encouraging performance of our method (ADMM-TD). However, a proof of convergence of ADMM-TD is still open. We believe that our encouraging numerical results will drive other researchers within academia to establish a complete theory of ADMM-TD. On the other hand, outside academia, on-line TD learning is already widely used in practice and in many machine learning applications is producing sufficiently accurate approximations. With the work undertaken in this chapter we believe that off-line TD learning can be efficiently compared to other techniques applied in machine learning problems. In the fourth chapter of this thesis we study the multilevel methods and we propose YAWN, a variant of the Newton method. In large-scale optimization, randomized variants of the Newton method have concentrated the main interest of the research community due to their fast convergence rates. However, their analysis suffers from one the following shortfalls: (a) is not scale invariant, (b) is not global, (c) absence of super-linear convergence rates —we note that all three characteristics are involved in the analysis of the classical Newton method. In this part of the thesis, we claim that our proposed method is able to address all three issues. Hence, with the analysis undertaken in this chapter, the analysis of the randomized variants of the Newton method can be considered on par with the classical analysis of the Newton method.

The super-linear convergence of YAWN can be further improved by estimating specific parameters of the algorithm, something that can attract the interest of other researchers within academia. On the other hand, outside academia, we argue that YAWN can be directly applied in practice and be able to produce accurate results quickly. This is also demonstrated in our initial numerical experiments which suggest that YAWN outperforms state-of-the-art methods.

“All paths lead to the same goal: to convey to others what we are. And we must pass through solitude and difficulty, isolation and silence in order to reach forth to the enchanted place where we can dance our clumsy dance and sing our sorrowful song”

Pablo Neruda

Acknowledgements

My Ph.D. journey started almost four years ago. For me being today at the position of submitting my thesis is very much ought to my first supervisor Dr. James D.B. Nelson. When I was applying for the Ph.D position in University College London I was not believing that I would be accepted in such a prestigious institution. It was James then that trusted me and offered me this position. I would like to express my deepest gratitude to my supervisor James Nelson. Unfortunately, sad things happen in life. James is not with us anymore. I would like to express my deepest condolences to his family and once more to mention that I am extremely grateful to him.

After James loss the situation was difficult for me and the rest of his Ph.D students. We had to seek new supervisors. By this time, I had already been in discussions with Dr. Panos Parpas, Imperial College London, in order for us to begin a collaboration since we are both interested in computational optimization. I asked Panos to supervise me for the rest of my Ph.D. and I after explained him the situation he immediately accepted to co-supervise me. He, of course, had no obligation to accept and unofficially supervise a student from another university. I would like to express that I am extremely thankful to Panos for this decision. Since I was a rookie in optimization, Panos was the best teacher I could have. His amazing research experience in optimization methods brought me today in the position of submitting my thesis. I shall mention that the great job that appears in Chapter 4 of this thesis is ought to him. I could never be able to produce such good results if I did not have

his guidance. Panos is a great person and supervisor and I would clearly like to continue collaborating with him in the future either officially or unofficially.

During my Ph.D. studies, I have to admit to myself that I feel extremely lucky to meet great people and the situation cannot be different with Dr. Paul Northrop. Working with Panos was really important for me to keep my research on track. However, I had to also seek an official supervisor from UCL. Most of the professors in the Statistical Department, UCL, reasonably declined to supervise a student who had been already working with another supervisor and, importantly, in a research field that is not of their direct interest. At this point, Paul, who was responsible of helping me find an official supervisor, proposed himself to take this role. I am heartily thankful to Paul for accepting this unfamiliar and uncomfortable role. I am completely aware about the risk he took that moment and for this reason I extremely admire and respect him. He decided to supervise me unconditionally, something that not many professors in academia would do.

I would also like to express my gratitude to my uncle, Dr. Michael Tsingelis. For Michael to me has always been an example of a great mathematician. He has been standing by my side since my early years in the university. Further, with his deep knowledge in pure maths (Algebra and Group Theory), Michael helped me a lot through my Ph.D. years by reviewing my work and many times proposing solutions in difficult theoretical tasks. I am also grateful to my family for their unconditional support all these years.

Last I am thankful to Defence Science and Technology Laboratory (DTSL) for supporting and funding my Ph.D program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"There are reasons to be sad, disconsolate, bitter, but there is not a single reason to be hopeless."

Nazim Hikmet Ran

Modern machine learning applications often require optimizing large-scale models. In this domain, the ability to obtain sufficiently accurate solutions quickly is crucial. Examples of such problems can be found in [5, 6, 7, 8]. In the context of computational optimization, there have been many developments for reducing the computational burden of solving large-scale optimization problems. In general, for an arbitrary optimization problem, aiming for extremely fast convergence rates is typically a very difficult task [9]. Therefore, much emphasis has been given in designing methods that take advantage of the problem structure. In addition to the fast solvers, the goal of accuracy in prediction is equally important. For this reason, exploitation of the prior knowledge about regularity in datasets, such as sparsity and smoothness, has become necessary. In particular, regularization in statistical regression setting has attracted an increasing interest during the last decade. Statistical regularization, or alternatively, penalization of the standard least-squares problem, has been efficiently applied in many diverse fields, such as classification, prediction on multivariate datasets (eg., graphical models), image and signal processing and

compressive sensing. To this end, the goal of this thesis is to develop and study optimization methods that take advantage of the structures of optimization models that arise in machine learning applications.

1.1 ADMM for Reinforcement Learning

Problems with large datasets are encountered in almost all applied fields such as AI, statistics, machine learning, etc. Here, we discuss Alternating Direction Method of Multipliers (ADMM), a general optimization framework which has recently received a lot of attention for various large regularized regression problems [10, 11].

ADMM is most useful when applied to optimization problems with a separable objective function. Regularized regression problems, such as *Lasso*, ridge regression and basis pursuit, fall into this category. In particular, since the function subject to minimization is separable, it can be split into two parts, and thus the algorithm can handle each part completely separately (i.e., each iteration can be viewed as an independent subproblem). For instance, each ADMM iteration implies a small convex optimization problem for which is often the case that it can be calculated analytically, thus yielding an efficient algorithm in terms of time complexity.

The algorithm was developed in 1970s and is closely related to algorithms such as *Douglas-Rachford splitting*, Dykstra's alternating projections and the method of multipliers [12]. ADMM has a fairly well established theory in the context of convex optimization and hence, within this domain, it has found numerous applications in different fields such as in [13] for compressive sensing, [14] for graphical models and [15] for signal processing and control problems. However, convexity of the objective functions does not always hold in many applied fields. For instance, the Reinforcement Learning problem we discuss below considers solving a fixed-point problem which in turn does not correspond to any other convex optimization problem.

Reinforcement Learning

Reinforcement Learning (RL) is a sub-field of machine learning [16]. As the name suggests, it considers learning problems where, when interacting with the environment, we learn what to do or what actions to take in order to optimize the desirable outcome. In particular, RL considers an agent which interacts with the environment. The agent finds itself in a current state and faces the dilemma of what action to select, since it is not told which action performs optimally. Moreover, after executing an action, the environment returns a new state together with a reward (indicating how good the selected action was) and the procedure continues in the same manner. In practice, we face problems where a sequence of actions (policy) needs to be taken in order to achieve our goal. The desirable outcome (or goal) in RL is to maximize the sum of the expected reward, or equivalently, to find the assignment of actions to each state that, when executed, maximizes the sum of the expected reward (optimal policy).

Furthermore, RL can be considered as a part of *Artificial Intelligence* (AI) and as such it aims for autonomous systems (or intelligent agents) that can make decisions and eventually achieve the determined goals. During the last years RL has successfully found many applications in AI such as robotics [17], autonomous helicopter [18] and *TD-Gammon* [19]. It has also been applied in fields such as Control theory and Operational Research (inverted pendulum [20] and shop scheduling [21], respectively).

Here, we consider off-line (batch) RL [22] where the agent is not allowed to interact with the environment in order to obtain the optimal policy but instead is given a fixed set of sampled states and actions, typically finite. Using the given information, the agent forms a random policy which is then used to interact with the environment; the policy is fixed and does not improve during the procedure. Hence, the goal in off-line RL is, given the existing data, to find the policy that maximizes the sum of rewards. The goal of maximizing the sum of rewards can be attained by computing the so called value functions.

A core problem in off-line RL emerges in situations where the state space is large, where explicit computation of the value functions becomes infeasible. Instead, approximation techniques provide the only way forward. In this work we are interested in linear representation of the value functions since the updates reduce to a simple form when employing first- or second-order methods [23, 16]. Least-squares and regularized least-squares methods have been proposed to solve the RL problem [24, 25, 26, 3, 27, 28, 29, 30, 31]. However, none of the proposed methods have been able to solve the RL problem, especially when the goal is to find the optimal policy. In particular, least-squares methods are known to be vulnerable to over-fitting and thus lead to poor predictions. On the other hand, ℓ_1-regularized least-squares methods, although have been found to overcome the over-fitting issue, only converge under some strong assumptions that rarely hold in practice.

In this thesis, we propose to use ADMM for solving the ℓ_1-regularized least-squares optimization problem. This problem, however, reduces to a fixed-point problem and as such it does not correspond to any convex optimization problem. Therefore, we modify the standard ADMM in order to now solve the aforementioned fixed-point problem. Our theoretical analysis shows that our proposed method is able to return efficient solutions. Incorporating ℓ_1-regularization into the fixed-point solution means that the underlying optimization problem is separable and thus can be handled efficiently by ADMM producing fast iterations. Another advantage is that ADMM yields closed-form solutions for the subproblems of the ℓ_1-regularized least-squares problem that are easy to compute. However, since the optimization problem we consider here is not convex, a proof of convergence is a difficult task and remains open. Finally, we perform preliminary numerical experiments which indicate the effectiveness of our proposed method.

The main contributions of this work have been published in:

Nikos Tsipinakis and James D.B. Nelson. Sparse temporal difference learning via alternating direction method of multipliers. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pages 220–225. IEEE,

2015.

1.2 Multilevel Methods

Multilevel methods in optimization arise from the general field of multigrid methods that were first introduced for solving (non-)linear Partial Differential Equations (PDEs) [33, 34, 35, 36]. In this domain, multigrid methods, in order to overcome the computational burden, attempt to offer approximate solutions from coarser discretizations of a mesh. They construct a hierarchy of different-sized discretization problems where the idea is to use the information of the smaller problems (in lower dimensionality) to solve the exact problem. Problems of lower dimensions are often called *coarse* problems. The advantage of this procedure is clear: coarse problems are typically much easier to be optimized because of their significantly reduced dimensionality; we note that, directly solving for the exact solution in the context of PDEs is expensive.

When the discussion comes to the context of large-scale optimization in machine learning applications the situation is similar: optimizing the exact model is often an intractable task (see [8] for examples in applications such as background extraction in video processing, and face recognition). To this end, the multigrid idea (solving coarse models in order to obtain a solution of the exact model) was introduced into optimization where many authors adopted the name *multilevel* [37, 38, 39, 40]. Importantly, the performance of multilevel methods has been found very efficient and in many cases it has been shown to outperform classical optimization methods.

Classical optimization methods such as first order methods, stochastic, proximal, accelerated or otherwise, are the most popular class of algorithms for the large-scale optimization models that arise in modern machine learning applications. The ease of implementation in distributed architectures and the ability to obtain a reasonably accurate solution quickly are the main reasons for the dominance of first-order methods in machine learning applications. In the last few years, second-order meth-

ods based on variants of the Newton method have also been proposed. Second-order methods, such as the Newton method, offer the potential of quadratic convergence rates (the holy grail in optimization algorithms). Unfortunately, the conventional Newton method has huge storage and computational demands and does not scale to applications that have both large and dense Hessian matrices. To improve the convergence rates, and robustness of the optimization algorithms used in machine learning applications many authors have recently proposed modifications of the classical Newton method [41, 42, 43, 44]. However, the current state-of-the-art methods discussed previously suffer from either of the following shortfalls: **Shortfall I:** *Lack of scale-invariant convergence analysis without restrictive assumptions*, and, **Shortfall II:** *Lack of global super-linear rates without ad-hoc assumptions regarding the spectral properties of the input data*.

In this thesis, we propose a general unconstrained optimization method based on the multilevel framework and we attempt to address both shortcomings listed above. Our theoretical analysis is based on the theory of self-concordant functions and we are able to prove a super-linear convergence rate without relying on unknown parameters (scale invariant analysis). Thus, we argue that with the results presented in this thesis, the theory of the variants of the Newton methods can be considered to be on-par with the theory of the classical Newton method. These fundamental results are achieved by drawing parallels between the second-order methods used in machine learning, and the so-called Galerkin model from the multilevel optimization literature. To the best of our knowledge, this is the first multilevel optimization method that captures the advantages of the multigrid theory (i.e., fast global convergence rates) and in parallel does not suffer from either of the shortfalls listed above.

The main contents of this work are currently in preparation with title:

Nikos Tsipinakis and Panos Parpas. *Exploiting coarse-grained models for faster, scale-invariant convex optimization*.

Chapter 2

Background Theory

In this chapter we present the most relevant theory of optimization methods required for this thesis. In particular, in the first section we present the theory of convex and self-concordant functions. In the second section, we review the most relevant first- and second-order methods for unconstrained convex optimization. In the third, we present the Alternating Direction Method of Multipliers and we describe the setting of the equality constrained convex optimization. In section four, we present optimization methods for approximating matrices with low-rank structure. We would like to emphasize that the goal of this chapter is not to provide a complete review of the methods that will be discussed. Our purpose, nevertheless, is to provide the reader with the necessary theoretical knowledge required before moving forward to the core of this work.

2.1 Preliminaries

In this section we collect some general, fundamental, results that will be useful throughout this thesis.

For any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ the standard inner product is defined by

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^{n} x_i y_i.$$

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be proper when $\operatorname{dom} f \neq \emptyset$, where $\operatorname{dom} f = \{\mathbf{x} \in \mathbb{R}^n \ : \ f(\mathbf{x}) < +\infty\}$. Additionally, a function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be closed when its epigraph, $\operatorname{epi} f$, is a closed set, where

$$\operatorname{epi} f = \{(\mathbf{x}, t) \ : \ \mathbf{x} \in \operatorname{dom} f, t \in \mathbb{R}, f(\mathbf{x}) \leq t\}.$$

A function $f : \mathbb{R}^n \to \mathbb{R}^+$ is called a norm if for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we have that: (i) $f(\mathbf{x}) = 0$, then $x = 0$; (ii) $f(\lambda\mathbf{x}) = |\lambda|\mathbf{x}, \lambda \in \mathbb{R}$; (iii) the triangle inequality holds, i.e., $f(\mathbf{x}+\mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$. A vector space $\mathcal{H}$ with a norm that satisfies the above conditions is called a normed vector space. Let $p \geq 1$. The ℓ_p-norm is defined as follows

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}.$$

Let $\mathcal{H} = (\mathcal{H}, \|\cdot\|)$ be a normed vector space. A mapping $f : \mathcal{H} \to \mathcal{H}$ is called contraction if for any $\mathbf{x}, \mathbf{y} \in \mathcal{H}$ and $\gamma \in (0, 1)$ we have that

$$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq \gamma \|\mathbf{x} - \mathbf{y}\|.$$

The Banach space is defined as a complete normed vector space, where every Cauchy sequence is convergent (i.e., $\lim_{n\to\infty} \sup_{m \geq n} \|a_n - a_m\| = 0$, where $\{a_n\}_{n\geq 0}$ is a sequence on $\mathcal{H}$).

Theorem 2.1.1 (Banach's fixed-point theorem [1])**.** *Let V be a Banach space and $T : V \to V$ be a contraction mapping. Then T has a unique fixed-point.*

For more details on the preliminary theory discussed above see [45, 46, 1].

2.1.1 Convex Functions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex, if, for all $\mathbf{x}, \mathbf{y} \in \operatorname{dom} f$ and some $\theta \in [0, 1]$, we have that

$$f(\theta\mathbf{x} + (1 - \theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1 - \theta) f(\mathbf{y}),$$

where $\mathrm{dom}\, f$ is a convex set. Based on the first-order information, for a differentiable function f, the above definition becomes

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}). \tag{2.1}$$

Inequality (2.1) constitutes a necessary and sufficient condition for a function f to be convex. Further, a function is called strictly convex if (2.1) holds with strict inequality. Importantly, note that if $\nabla f(\mathbf{x})^T = 0$ then we have $f(\mathbf{x}) \leq f(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathrm{dom}\, f$ which means that $\mathbf{x}$ is global minimizer of f. In addition to the first-order convexity condition, the second-order condition, assuming a twice differentiable f is given by

$$\nabla^2 f(\mathbf{x}) \succeq 0,$$

for all $\mathbf{x} \in \mathrm{dom}\, f$. In other words, a twice differentiable function is convex when the Hessian matrix is positive semi-definite. If, in addition, the Hessian matrix is positive definite then f is strictly convex.

A twice differentiable is strongly convex if there exists a constant $\mu > 0$ such that

$$\nabla^2 f(\mathbf{x}) \succeq \mu \mathbf{I}_{n\times n}, \tag{2.2}$$

where $\mathbf{I}_{n\times n}$ is the identity matrix. A direct consequence of strong convexity is that the Hessian matrix is also bounded above, i.e., there exists $M > 0$ such that $\nabla^2 f(\mathbf{x}) \preceq M\mathbf{I}_{n\times n}$. Combining both bounds of the Hessian matrix we have that

$$f(\mathbf{x})+\nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x})+\frac{\mu}{2}\|\mathbf{y}-\mathbf{x}\|_2 \leq f(\mathbf{y}) \leq f(\mathbf{x})+\nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x})+\frac{M}{2}\|\mathbf{y}-\mathbf{x}\|_2. \tag{2.3}$$

For a more refined analysis on convex functions we refer the reader to [45, 9].

2.1.2 Self-Concordant Functions

In this section we recall some of the main properties and inequalities of the class of self-concordant functions. We follow similar notation as in the books [9, 45] (for a

complete theory on self-concordant functions see [9]).

A univariate convex function $\phi : \mathbb{R} \to \mathbb{R}$ is called self-concordant if

$$|\phi'''(x)| \leq 2\phi''(x)^{3/2}. \tag{2.4}$$

Examples of such functions include but not are limited to linear, quadratic and logarithmic. Further, consider a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$ and also fix $\mathbf{x} \in \operatorname{dom} f$ and a direction $\mathbf{u} \in \mathbb{R}^n$. Then, $\phi(t) = f(\mathbf{x} + t\mathbf{u})$ is called self-concordant for all $\mathbf{x}$ and $\mathbf{u}$ if it is self-concordant along every line in its domain. Importantly, self-concordance is preserved under composition with any affine function.

Next, given $\mathbf{x} \in \operatorname{dom} f$ and assuming that $\nabla^2 f(\mathbf{x})$ is positive-definite we can define the following norms

$$\|\mathbf{u}\|_{\mathbf{x}} = \langle \nabla^2 f(\mathbf{x})\mathbf{u}, \mathbf{u}\rangle^{1/2} \text{ and } \|\mathbf{v}\|_{\mathbf{x}}^* = \langle [\nabla^2 f(\mathbf{x})]^{-1}\mathbf{v}, \mathbf{v}\rangle^{1/2}, \tag{2.5}$$

where it holds that $|\langle \mathbf{u}, \mathbf{v}\rangle| \leq \|\mathbf{u}\|_{\mathbf{x}}^* \|\mathbf{v}\|_{\mathbf{x}}$. The Newton decrement is defined as

$$\lambda_f(\mathbf{x}) = \|\nabla f(\mathbf{x})\|_{\mathbf{x}}^* = \|[\nabla^2 f(\mathbf{x})]^{-1/2}\nabla f(\mathbf{x})\|_2. \tag{2.6}$$

In addition, we take into consideration two auxiliary functions, both introduced in [9]. Define the univariate functions ω and ω_* such that

$$\omega(x) = x - \log(1 + x) \quad \text{and} \quad \omega_*(x) = -x - \log(1 - x), \tag{2.7}$$

with $\operatorname{dom}\omega = \{x \in \mathbb{R} : x \geq 0\}$ and $\operatorname{dom}\omega_* = \{x \in \mathbb{R} : 0 \leq x < 1\}$, respectively. Note that both functions are convex and their range is the set of positive real numbers.

Now, from the definition (2.4), we have that

$$\left| \frac{d}{dt} \left(\phi''(t)^{-1/2} \right) \right| \leq 1,$$

from which, after integration, we obtain the following bounds

$$\frac{\phi''(0)}{(1 + t\phi''(0)^{1/2})^2} \leq \phi''(t) \leq \frac{\phi''(0)}{(1 - t\phi''(0)^{1/2})^2} \tag{2.8}$$

where the lower bound holds for $t \geq 0$ and the upper bound for $t \in [0, \phi''(0)^{-1/2})$, with $t \in \operatorname{dom} \phi$. Consider now functions on $\mathbb{R}^n$. For $\mathbf{x} \in \operatorname{dom} f$, and for any $\mathbf{y} \in S(\mathbf{x})$, where $S(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}} < 1\}$, we have that

$$(1 - \|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}})^2 \nabla^2 f(\mathbf{x}) \preceq \nabla^2 f(\mathbf{y}) \preceq \frac{1}{(1 - \|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}})^2} \nabla^2 f(\mathbf{x}). \tag{2.9}$$

Finally, let us state one last pair of inequalities that will be useful in our analysis. For $\mathbf{x}$ and $\mathbf{y}$ from $\operatorname{dom} f$ it holds that

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \omega(\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\|_{\mathbf{y}}^*)$$

and if also $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x})\|_{\mathbf{y}}^* < 1$, then

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \omega_*(\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\|_{\mathbf{y}}^*). \tag{2.10}$$

For more details on self-concordant functions see [45, 9].

2.2 Unconstrained Convex Optimization Methods

In this section we are interested in solving the following unconstrained optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}), \tag{2.11}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function. Further, we assume that f is twice differentiable and a minimizer $\mathbf{x}^*$ exists. In the unconstrained case, $\mathbf{x}^*$ is an optimal

Algorithm 2.1 Gradient Descent

1: **Initialize:** $\mathbf{x}_0 \in \mathbb{R}^n$
2: **for** $k = 0, 1, \ldots$ **do**
3: Compute the direction as $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$
4: Choose t_k through inexact line search Algorithm 2.2
5: Update

$$\mathbf{x}_{k+1} := \mathbf{x}_k + t_k \mathbf{d}_k$$

6: **end for**
7: **return** $\mathbf{x}_{h,k}$

point if and only if

$$\nabla f(\mathbf{x}^*) = 0. \tag{2.12}$$

Therefore, the goal is to seek points that satisfy (2.12). In practice, this can be achieved via an iterative scheme by producing a sequence of k points; the iterative procedure terminates, at some iteration k, if $\|\nabla f(\mathbf{x})\|_2 < \epsilon$, for some tolerance $\epsilon > 0$.

2.2.1 Gradient Descent Method

In this section we discuss first-order methods for solving the convex program (2.11). Specifically, we will concentrate on the gradient descent method [46, 45, 9], a method which is frequently well suited to large-scale optimization problems since, by definition, it uses "cheap" iterations based on the first-order information (i.e., gradients) of the objective function.

Consider the optimization problem in (2.11). The gradient descent method builds iterates using the first-order information. In particular, the negative gradient is chosen as search direction, that is, $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$, and thus we take the following iterative scheme

$$\mathbf{x}_{k+1} = \mathbf{x}_k + t_k \mathbf{d}_k.$$

Algorithm 2.2 Armijo Rule

1: **Input:** $\alpha \in (0, 0.5),\ \beta \in (0, 1)$ and a descent direction $\mathbf{d}$
2: $t := 1$
3: **while** $f(\mathbf{x} + t\mathbf{d}) > f(\mathbf{x}) + \alpha t \nabla f(\mathbf{x})^T \mathbf{d}$ **do**
4: $\quad t := \beta t$
5: **end while**

This choice of search direction produces a descent algorithm since

$$\nabla f(\mathbf{x}_k)\mathbf{d}_k = -\|\nabla f(\mathbf{x}_k)\|_2 < 0$$

which means that, at each iteration, we expect $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$, unless $\mathbf{x}_k$ is optimal. The gradient descent method has been analyzed assuming f is a m-strongly convex function and enjoys a linear convergence rate. In Algorithm 2.1, the step size is computed through the inexact line search method. If we instead assume that f has a L-Lipschitz continuous gradient, with L known, then we can use constant step size as $t = 1/L$ (similarly, when assuming m-strongly convex function, there exists parameter M such that the constant step yields $t = 1/M$). Since in most practical problems such constants are typically unknown, for computing t_k, we consider the Armijo rule or inexact line search method, see Algorithm 2.2.

2.2.2 Newton Method

In this section we consider the Newton method [46, 45, 9], a second-order method for solving (2.11). In general, second-order methods make use of the information which emerges from the second derivative of the objective function. We discuss the Newton method with analysis based on both classical theory (Lipschitz continuity and strongly convex functions) and theory of self-concordant functions.

We are interested in solving the convex optimization problem (2.11). The Newton method builds the iterates based on the second-order Taylor approximation of f, i.e.,

$$f(\mathbf{x}_k + \mathbf{d}) \approx f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{d} \rangle + \frac{1}{2}\mathbf{d}^T \nabla^2 f(\mathbf{x}_k)\mathbf{d}.$$

Algorithm 2.3 Newton Method

1: **Initialize:** $\mathbf{x}_0 \in \mathbb{R}^n$
2: **for** $k = 0, 1, \ldots$ **do**
3: Compute direction and decrement by
$\mathbf{d}_k = -[\nabla f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k), \quad \lambda(\mathbf{x}_k)^2 = \nabla f(\mathbf{x}_k)^T[\nabla^2 f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k)$
4: Choose t_k through inexact line search Algorithm 2.2
5: Update

$$\mathbf{x}_{k+1} := \mathbf{x}_k + t_k\mathbf{d}_k$$

6: **end for**
7: **return** $\mathbf{x}_{h,k}$

Since $\nabla^2 f(\mathbf{x}_k)$ is positive definite, we can minimize the right-hand side (which is a convex quadratic function) to obtain the Newton direction

$$\mathbf{d}_k = -[\nabla^2 f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k).$$

Positive definiteness of $\nabla^2 f(\mathbf{x}_k)$ also implies that the Newton step is a descent direction

$$\nabla f(\mathbf{x}_k)^T\mathbf{d} = -\nabla f(\mathbf{x}_k)^T[\nabla^2 f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k) < 0 \tag{2.13}$$

unless $\mathbf{x}_k$ is a minimizer. It is easy to see that $\mathbf{d}_k$ is what we need to add to the current point $\mathbf{x}_k$ so that to minimize the right-hand side of the Taylor approximation. The intuition indicates that if f in (2.11) is a quadratic function, then the point $\mathbf{x}_k + \mathbf{d}_k$ is exactly the minimizer of f and thus the Newton method converges in one iteration. Relation (2.13) leads us to the definition of the Newton decrement

$$\lambda(\mathbf{x}_k) = \left[\nabla f(\mathbf{x}_k)^T[\nabla^2 f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k)\right]^{1/2}.$$

The Newton decrement plays important role in the analysis of the Newton method and can be used as an exit condition (i.e., $\lambda(\mathbf{x}_k)^2/2 \leq \epsilon$ for some small $\epsilon > 0$).

One important aspect of the Newton method is that the Newton step builds iterates that are invariant of affine transformation of variables. This means that the convergence rate is not affected by the input data.

Analysis with classical theory

In addition to the basic assumptions (convexity and twice differentiable function), the analysis of the Newton method has been conducted by further assuming the following

Assumption 2.2.1. *Function f is strongly convex. Then, there exist positive constants m and M such that*

$$m\mathbf{I}_{n\times n} \leq \nabla^2 f(\mathbf{x}) \leq M\mathbf{I}_{n\times n}.$$

where $\mathbf{I}_{n\times n}$ is the identity matrix. Further, f possesses L-Lipschitz continuous Hessian, i.e.,

$$\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\|_2 \leq L\|\mathbf{x}-\mathbf{y}\|, \quad \mathbf{x},\mathbf{y} \in \operatorname{dom} f.$$

The above assumption is typical when analyzing descent methods based on second-order information. It has been proved that the Newton method can achieve quadratic convergence rate. In particular, convergence is split into two phases according to the magnitude of $\|\nabla f(\mathbf{x}_k)\|_2$: (a) the damped Newton phase where Algorithm 2.3 can choose step size $t_k < 1$ and (b) the quadratic phase where the convergence is extremely fast and the step is always chosen as $t_k = 1$.

Theorem 2.2.2 ([45])**.** *Suppose that Algorithm 2.3 is performed and let Assumption 2.2.1 hold. There exists $\eta \in (0, m^2/L]$ such that*

1. *if $\|\nabla f(\mathbf{x}_k)\|_2 > \eta$, then there exists $\gamma = \alpha\beta\frac{m}{M^2}$ such that*

$$f(\mathbf{x_{k+1}}) - f(\mathbf{x_k}) \leq -\gamma$$

2. *if $\|\nabla f(\mathbf{x}_k)\|_2 \leq \eta$, then $t_k = 1$ and*

$$\|\nabla f(\mathbf{x}_{k+1})\|_2 \leq \frac{L}{2m^2}\left(\|\nabla f(\mathbf{x}_k)\|_2\right)^2$$

Theorem 2.2.2 describes the convergence behavior of the Newton method. If the current point $\mathbf{x}_k$ is far from the optimizer, then the damped Newton phase is performed which guarantees reduction of the value function according to some constant γ. If $\mathbf{x}_k$ is sufficiently close to the minimizer, then we obtain quadratic reduction in the value of $\|\nabla f(\mathbf{x}_{k+1})\|_2$.

Although the Newton method, under Assumption 2.2.1, achieves quadratic convergence rate, we cannot say much about both regions of convergence since constants L and m are typically unknown in practice. Intuition suggests that sufficiently small values in L yield extremely fast reduction in $\|\nabla f(\mathbf{x}_{k+1})\|_2$. In the next section we see how to obtain explicit expressions about the quadratically convergence phase of Newton method.

Analysis with self-concordant functions

The analysis of the Newton method conducted in the previous section has two important shortcomings: (a) complexity bounds involve constants m, M and L which are typically not known in practice and thus we cannot obtain an explicit bound on the number of iterations; (b) Although the Newton step produces an affine invariant method with respect to the change of coordinates, its analysis is not affine invariant, i.e., if we change coordinates all constants m, M and L change. Therefore we should seek a theory that is independent of the affine transformation of variables. The way forward for achieving this goal is to replace Assumption 2.2.1 with the elegant theory of self-concordant functions.

Therefore, we are interested in solving the optimization problem (2.11) where, now, we assume that the objective function f is a strictly self-concordant function. The idea of the analysis of Newton method remains the same but now the results do not depend on any unknown constants. In contrast to the classical analysis, the region of quadratic convergence depends on the magnitude of the Newton decrement (in place of the norm of the gradient).

Theorem 2.2.3 ([45]). *Suppose that Algorithm 2.3 is performed and let f be a strictly self-concordant function. Then, for $\eta \in (0, 1/4]$,*

1. *if $\lambda(\mathbf{x}_k) > \eta$, then there exists $\gamma = \alpha\beta\eta^2 \frac{\eta^2}{1+\eta}$ such that*

$$f(\mathbf{x_{k+1}}) - f(\mathbf{x_k}) \leq -\gamma$$

2. *if $\lambda(\mathbf{x}_k) \leq \eta$, then $t_k = 1$ and*

$$\lambda(\mathbf{x}_k) \leq 2\lambda(\mathbf{x}_k)^2$$

Theorem 2.2.3 shows that the Newton method enjoys a quadratic convergence rate. In particular, we come up with an explicit expression of the region of the quadratically convergent phase (independent of any unknown constants). This mean that Algorithm 2.3 is affine invariant with respect to the change of coordinates.

To this end, the Newton method, either analyzed using the classical theory or the theory of self-concordant functions, has been found to outperform many algorithms due to its quadratically convergent phase. However, the main drawback arises in large-scale optimization problems since handling $\nabla^2 f(\mathbf{x})$ is typically infeasible. On the other hand, for moderate-sized optimization problems, Newton method can be considered as one of the best method to be applied to, due to its extremely fast convergent behavior and its advanced theoretical analysis.

2.3 Equality Constrained Convex Optimization

Consider the optimization problem of the form

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \end{aligned} \tag{2.14}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function with primal optimal value p^*. The Lagrangian [46, 10, 11, 45] associated with the problem (2.14) is a function $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ defined as

$$L(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} - \mathbf{b}),$$

where $\mathbf{y} \in \mathbb{R}^m$ is called the dual variable or the Lagrange multiplier of the equality constraint. The Lagrange dual function $g : \mathbb{R}^m \to \mathbb{R}$ is defined as the infimum of the Lagrangian, that is

$$g(\mathbf{y}) = \inf_{\mathbf{x} \in \mathbb{R}^n} \{f(\mathbf{x}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} - \mathbf{b})\}.$$

There are two important properties regarding the Lagrangian. First, the dual function, $g(\mathbf{y})$, is always concave —this is true even in the case where the optimization problem (2.14) is not convex. Moreover, it implies lower bounds on the primal optimal value p^*, for any $\mathbf{y} \in \mathbb{R}^m$, i.e.,

$$g(\mathbf{y}) \leq p^*.$$

Therefore, this leads one to search the best available lower bound. The best bound can be obtained by the following unconstrained optimization problem, called the Lagrange dual problem

$$\max_{\mathbf{y} \in \mathbb{R}^m} g(\mathbf{y}). \tag{2.15}$$

We denote the dual optimal value of the above optimization problem as d^*. Note that (2.15) is always convex since it is a maximization problem over a concave function. When the best lower bound obtained by the dual problem is equal to the primal optimal value of the initial problem, i.e., $d^* = p^*$, we say either that the duality gap is zero or that strong duality holds.

KKT Optimality Conditions

We shall now examine the sufficient and necessary conditions of the problem (2.14). Consider $\mathbf{x}^*$ and $\mathbf{y}^*$ as the primal and dual optimal points, respectively, and also that strong duality holds. We have the following optimality conditions associated with the optimization problem (2.14) and its dual problem (2.15)

$$\begin{aligned} \partial f(\mathbf{x}^*) + \mathbf{A}^T\mathbf{y}^* &\ni 0 \\ \mathbf{A}\mathbf{x}^* &= \mathbf{b}. \end{aligned}$$

The above optimality conditions are called *Karush-Kuhn-Tucker* (KKT) conditions. Any $\mathbf{x}^*$ and $\mathbf{y}^*$ must satisfy the above conditions when duality gap is zero [45].

Note that the first equation is obtained by taking the gradient of the Lagrangian over $\mathbf{x}$ and the second equation because of the fact that the equality constraint must always hold. The operator ∂ denotes the subdifferential of a function since $f(\mathbf{x})$ might not be differentiable. Additionally, ∂f is set-valued and hence we use $\in$ instead of $=$. When $f(\mathbf{x})$ is differentiable, the subdifferential symbol, ∂, can be replaced by the gradient, and the inclusion symbol by equality (for more background on subdifferential calculus see [47]).

Augmented Lagrangian

Consider now the optimization problem of the form

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}. \end{aligned} \tag{2.16}$$

For any feasible point $\mathbf{x}$, the above optimization problem is equivalent to (2.14) — the term $\frac{\rho}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$ will be equal to zero. Therefore, the Lagrangian of (2.16) is

$$L_\rho(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2, \tag{2.17}$$

where $\rho > 0$ denotes the penalty parameter. Equivalently, equation (2.17) can be viewed as the *augmented Lagrangian* of the problem (2.14) [10, 11]. Note that for $\rho = 0$ the augmented Lagrangian yields the standard Lagrangian.

2.3.1 Alternating Direction Method of Multipliers

In this section we discuss problems of the form as in (2.14) where the objective function is separable, i.e., $f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$. The Alternating Direction Method of Multipliers (ADMM) is a simple but powerful algorithm as it has been demonstrated to be very efficient for problems with separable objective functions in the context of large scale optimization [10, 11]. Consider the following composite problem

$$\begin{aligned} &\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad h(\mathbf{x}) + g(\mathbf{z}) \\ &\text{subject to} \quad \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}, \end{aligned} \tag{2.18}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{B} \in \mathbb{R}^{m\times p}$, $\mathbf{z} \in \mathbb{R}^p$ and $\mathbf{c} \in \mathbb{R}^m$. Moreover, functions h and g are convex, proper and closed functions. The variable $\mathbf{z}$ emerges by performing variable splitting over $\mathbf{x}$, i.e., $\mathbf{x}$ has been split into two parts, namely $\mathbf{x}$ and $\mathbf{z}$, and then, for feasibility purposes, we must incorporate $\mathbf{z}$ into the equality constraint.

The augmented Lagrangian associated with the problem (2.18) is of the form

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = h(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|^2,$$

where $\mathbf{y} \in \mathbb{R}^m$ denotes the dual variable and $\rho > 0$ is the penalty parameter. The ADMM iterations are functions of the augmented Lagrangian and ρ can be considered as the step-size parameter of the algorithm. In particular, we have the following

ADMM iterations

$$\begin{aligned}\mathbf{x}_{k+1} &:= \underset{\mathbf{x}\in\mathbb{R}^n}{\operatorname{argmin}}\{L_\rho(\mathbf{x}, \mathbf{z}_k, \mathbf{y}_k)\}\\ \mathbf{z}_{k+1} &:= \underset{\mathbf{z}\in\mathbb{R}^p}{\operatorname{argmin}}\{L_\rho(\mathbf{x}_{k+1}, \mathbf{z}, \mathbf{y}_k)\}\\ \mathbf{y}_{k+1} &:= \mathbf{y}_k + \rho(\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1} - \mathbf{c}).\end{aligned} \tag{2.19}$$

The above iterations are often difficult to calculate and thus is more convenient to express the Lagrangian in the following form

$$\begin{aligned}&L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) =\\ &h(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|^2 =\\ &h(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|^2 + \frac{1}{2\rho}\|\mathbf{y}\|^2 - \frac{1}{2\rho}\|\mathbf{y}\|^2 =\\ &h(\mathbf{x}) + g(\mathbf{z}) + \frac{\rho}{2}(\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|^2 + \frac{2}{\rho}\mathbf{y}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{1}{\rho^2}\|\mathbf{y}\|^2) - \frac{1}{2\rho}\|\mathbf{y}\|^2 =\\ &h(\mathbf{x}) + g(\mathbf{z}) + \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c} + \frac{1}{\rho}\mathbf{y}\|^2 - \frac{1}{2\rho}\|\mathbf{y}\|^2 =\\ &h(\mathbf{x}) + g(\mathbf{z}) + \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}\|^2 - \frac{\rho}{2}\|\mathbf{u}\|^2,\end{aligned}$$

where $\mathbf{u} \in \mathbb{R}^m$ denotes the scaled dual variable, $\mathbf{u} = \frac{1}{\rho}\mathbf{y}$. Hence, by replacing the above result in (2.19), we have that

$$\begin{aligned}\mathbf{x}_{k+1} &:= \underset{\mathbf{x}\in\mathbb{R}^n}{\operatorname{argmin}}\{h(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_k - \mathbf{c} + \mathbf{u}_k\|^2\}\\ \mathbf{z}_{k+1} &:= \underset{\mathbf{z}\in\mathbb{R}^p}{\operatorname{argmin}}\{g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}_k\|^2\}\\ \mathbf{u}_{k+1} &:= \mathbf{u}_k + \mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1} - \mathbf{c}.\end{aligned} \tag{2.20}$$

As a result, both ADMM forms, (2.19) and (2.20), are equivalent and also it is evident that the right hand side of the latter ADMM form can be often easily evaluated since the minimization part is simpler (easier differentiation of such a function) —this form is also called as scaled form due to the scaled dual variable $\mathbf{u}$.

Optimality Conditions and Convergence

At the optimal points $\mathbf{x}^*$, $\mathbf{z}^*$ and $\mathbf{y}^*$, ADMM must satisfy the following optimality conditions

$$\mathbf{A}\mathbf{x}^* + \mathbf{B}\mathbf{z}^* - \mathbf{c} = 0 \tag{2.21}$$

$$\partial h(\mathbf{x}^*) + \mathbf{A}^T\mathbf{y}^* \ni 0 \tag{2.22}$$

$$\partial g(\mathbf{z}^*) + \mathbf{B}^T\mathbf{y}^* \ni 0. \tag{2.23}$$

It has been shown that the optimality conditions associated with the ADMM iterations (2.19) are

$$0 \in \partial h(\mathbf{x}_{k+1}) + \mathbf{A}^T\mathbf{y}_{k+1} + \rho\mathbf{A}^T\mathbf{B}(\mathbf{z}_k - \mathbf{z}_{k+1}) \tag{2.24}$$

$$0 \in \partial g(\mathbf{z}_{k+1}) + \mathbf{B}^T\mathbf{y}_{k+1}. \tag{2.25}$$

For more details on the derivation of the aforementioned optimality conditions see [10]. It is obvious that equation (2.25) always satisfies optimality condition (2.23). On the other hand, equation (2.22) will only be satisfied when

$$\rho\mathbf{A}^T\mathbf{B}(\mathbf{z}_{k+1} - \mathbf{z}_k) \in \partial h(\mathbf{x}_{k+1}) + \mathbf{A}^T\mathbf{y}_{k+1}.$$

The quantity on the left-hand side of the above relation is called the dual residual and we define

$$\mathbf{q}_{k+1} = \rho\mathbf{A}^T\mathbf{B}(\mathbf{z}_{k+1} - \mathbf{z}_k).$$

Furthermore, optimality conditions (2.21) indicate that

$$\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1} - \mathbf{c} = 0,$$

which is called primal residual, and thus we define

$$\mathbf{r}_{k+1} = \mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1} - \mathbf{c}.$$

ADMM has been shown to converge under the most general assumptions. In particular, the extended functions $h : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ need to be convex, proper and closed functions and moreover the standard Lagrangian (L_0) to have a saddle point, i.e.,

$$L(\mathbf{x}^*, \mathbf{z}^*, \mathbf{y}) \leq L(\mathbf{x}^*, \mathbf{z}^*, \mathbf{y}^*) \leq L(\mathbf{x}, \mathbf{z}, \mathbf{y}^*).$$

Note that functions h and g can take the value $+\infty$ and also that there no assumptions on matrices $\mathbf{A}$ and $\mathbf{B}$. Under these assumptions it has been proved that the algorithm converges to the optimal solution p^* and, further, the dual variable y to its optimal point y^* as $k \to +\infty$. Additionally, it has been shown that both primal and dual residuals converge to zero in the limit.

2.3.2 Proximal ADMM

In this section we discuss the proximal form of ADMM. We start by providing the relevant theory on proximal operators.

The proximal operator of a convex, proper and closed function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is defined as

$$\mathbf{prox}_{\mu f}(\mathbf{v}) = \underset{\mathbf{x} \in \mathbb{R}^n}{\operatorname{argmin}} \{f(\mathbf{x}) + \frac{1}{2\mu}\|\mathbf{x} - \mathbf{v}\|^2\},$$

where $\mathbf{prox}_{\mu f} : \mathbb{R}^n \to \mathbb{R}^n$ [11]. Moreover, $\mu > 0$ denotes the step-size parameter indicating how fast we want to move towards the optimal point.

The resolvent of an operator $\mathbf{H}$ with scalar μ is defined as $\mathbf{J}_{\mu\mathbf{H}} = (\mathbf{I} + \mu\mathbf{H})^{-1}$ —note that the resolvent is a relation (for more details see [48]). In the case of proximal operators, we have that the resolvent of the subdifferential operator and the proximal operator coincide, so we have that

$$\mathbf{prox}_{\mu f} = \mathbf{J}_{\mu \partial f}.$$

For f convex, proper and closed and since ∂f is a maximal monotone operator,

the resolvent of the subdifferential is single-valued even though ∂f is set-valued. Furthermore, the application of the proximal operator can be viewed as a fixed-point iteration. In particular, it has been shown that for any maximal monotone operator $\mathbf{H}$, $\mathbf{x}$ minimizes $\mathbf{H}$, i.e., $0 \in \mathbf{H}(\mathbf{x})$, if and only if $\mathbf{x} = \mathbf{J}_{\mu\mathbf{H}}(\mathbf{x})$, which in turn yields

$$0 \in \partial f(\mathbf{x}) \;\Leftrightarrow\; \mathbf{x} = \mathbf{prox}_{\mu f}(\mathbf{x}). \tag{2.26}$$

For a proof of the theorem and details on monotone operators see [12]. As a result, equation (2.26) implies the *proximal point algorithm*, that is

$$\mathbf{x}_{k+1} := \mathbf{prox}_{\mu f}(\mathbf{x}_k).$$

The proximal point algorithm has not been found effective in most applications since it requires minimization over $f(\mathbf{x}) + \frac{1}{2\mu}\|\mathbf{x} - \mathbf{v}\|^2$ at each iteration. For this reason, it is more useful to apply either *proximal gradient method* or ADMM in its proximal form, also known as *Douglas-Rachford splitting method*. The former method considers the unconstrained optimization problem of the form

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}}\; h(x) + g(x), \tag{2.27}$$

where, again, the objective function f has been split into h and g and moreover we require h to be differentiable. The proximal gradient method consists of the following iterations

$$\mathbf{x}_{k+1} := \mathbf{prox}_{\mu_k g}(\mathbf{x}_k - \mu_k \nabla h(\mathbf{x}_k)),$$

where $\mu_k > 0$ denotes the step-size parameter. It has been shown that the algorithm also converges for a fixed step-size, μ, and moreover, in the case where ∇h is L-Lipschitz continuous, μ can take values in $(0, \frac{2}{L}]$ (for more discussion about the proximal gradient method and its accelerated form see [11]).

On the other hand, the proximal form of ADMM performs variable splitting to the

unconstrained problem (2.27)

$$\begin{aligned} &\text{minimize} \;\; h(\mathbf{x}) + g(\mathbf{z}) \\ &\text{subject to} \;\; \mathbf{x} = \mathbf{z}, \end{aligned}$$

where we now introduce the equality constraint requiring variables $\mathbf{x}$ and $\mathbf{z}$ to be equal. Thus, we have the following iterations

$$\begin{aligned} \mathbf{x}_{k+1} &:= \mathbf{prox}_{\mu h}(\mathbf{z}_k - \mathbf{u}_k) \\ \mathbf{z}_{k+1} &:= \mathbf{prox}_{\mu g}(\mathbf{x}_{k+1} + \mathbf{u}_k) \\ \mathbf{u}_{k+1} &:= \mathbf{u}_k + \mathbf{x}_{k+1} - \mathbf{z}_{k+1}, \end{aligned}$$

where $\mathbf{x}, \mathbf{z}, \mathbf{u} \in \mathbb{R}^n$, with $\mathbf{u}$ denoting the dual variable.

The advantage of ADMM against the proximal gradient method is that the former evaluates $h(\mathbf{x})$ and $g(\mathbf{z})$ completely separately, in many cases, yielding the algorithm to perform more efficiently in terms of time complexity. Additionally, none of the functions $h(\mathbf{x})$ and $g(\mathbf{z})$ are required to be differentiable. Finally, it is easy to see that, by replacing $\mathbf{u} = \frac{1}{\rho}\mathbf{y}$, $\mu = \frac{1}{\rho}$ and matrices $\mathbf{A}, \mathbf{B}$ with the identity matrix, the ADMM version in (2.19) is equivalent with the proximal version presented above, and thus proximal ADMM can be viewed as a special case of the standard ADMM in Section 2.3.1.

2.4 Low-Rank Approximation Methods

A key feature in modern (large-scale) machine learning problems is the limitation of storing excessively big matrices. Fortunately, in many applications (see [49] and references therein), these matrices exhibit a low-rank structure. First, consider a general $\mathbf{A} \in \mathbb{R}^{n \times m}$ matrix. The Singular Value Decomposition of $\mathbf{A}$ is summarized in the following theorem.

Theorem 2.4.1 ([50]). *Let $\mathbf{A} \in \mathbb{R}^{n \times m}$ and assume that $n < m$. Then, there ex-*

ist unitary matrices $\mathbf{U} \in \mathbb{R}^{n\times n}$ *and* $\mathbf{V} \in \mathbb{R}^{m\times m}$ *and a diagonal matrix* $\mathbf{\Sigma}_n = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ *with* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$ *such that*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}, \quad \mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_n & \mathbf{0} \end{pmatrix}$$

where $\mathbf{\Sigma}$ *is a diagonal* $n \times m$ *matrix.*

The positive constants $\sigma_1, \ldots, \sigma_n$ are called the singular values of $\mathbf{A}$ and when $\mathrm{rank}(\mathbf{A}) = n$ they coincide with the square roots of the eigenvalues of the matrix $\mathbf{A}\mathbf{A}^T$. If, further, matrix $\mathbf{A}$ is known to be of rank-p $< n$, the above theorem applies with singular values of $\mathbf{\Sigma}_n$ be as $\sigma_1 \geq \cdots \geq \sigma_p > 0 = \sigma_{p+1} = \cdots = \sigma_n$.

For the purposes of this work, we consider low-rank approximations of a positive semi-definite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$. We can obtain an rank-$p$ approximation matrix, $\mathbf{A}_p$, of $\mathbf{A}$ by solving the following optimization problem

$$\min_{\mathbf{A}_p \in \mathbb{R}^{n\times n}} \|\mathbf{A} - \mathbf{A}_p\|_2 \quad \text{s.t.} \quad \mathrm{rank}(\mathbf{A}_p) = p, \quad p < n.$$

It is known that the above optimization problem can be analytically solved through the eigenvalue decomposition [51, 49]. That is,

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T = \begin{pmatrix} \mathbf{U}_p & \mathbf{U}_{n-p} \end{pmatrix} \mathrm{diag}(\mathbf{\Sigma}_p, \mathbf{\Sigma}_{n-p}) \begin{pmatrix} \mathbf{U}_p & \mathbf{U}_{n-p} \end{pmatrix}^T$$

where $\mathbf{\Sigma}_p \in \mathbb{R}^{p\times p}$, $\mathbf{\Sigma}_{n-p} \in \mathbb{R}^{(n-p)\times(n-p)}$ are diagonal matrices containing the eigenvalues of $\mathbf{A}$ and $\mathbf{U}_p \in \mathbb{R}^{n\times p}$, $\mathbf{U}_{n-p} \in \mathbb{R}^{n\times(n-p)}$ are unitary matrices containing the corresponding eigenvectors. Then, we construct $\mathbf{A}_p$ as

$$\mathbf{A}_p = \mathbf{U}_p\mathbf{\Sigma}_p\mathbf{U}_p^T.$$

Note that this method is just a special case of Theorem 2.4.1 for positive semi-definite matrices. It can be also found in literature under the name Truncated-SVD (T-SVD) where, by positive semi-definiteness of $\mathbf{A}$, SVD coincides with the eigen-

value decomposition. The name "truncated" refers to the truncation step, i.e., first compute the eigenvalue decomposition and then perform the truncation step so that to retain only the first p eigenvalues by zeroing all the last $(n-p)$ eigenvalues.

2.4.1 Randomized SVD

Deterministic algorithms [52] for computing the T-SVD of a matrix are typically expensive (of order $\mathcal{O}(pn^2)$) but they offer effective approximations. Randomized methods have lately drawn much attention due to their decreased computational cost and, in parallel, they have been developed enough so that to offer competitive error bounds. In a survey paper [49], the authors discuss, among others, the pros and cons of the Randomized SVD method. As discussed, the main limitation (computational burden) arises in cases where the singular values decay slowly. In such case, Randomized SVD addresses this difficulty by incorporating q number of power iterations and a random Gaussian test matrix (q equal 1 or 2 typically suffices in practice). This choice of test matrix has been shown to almost always produce efficient approximations. This is illustrated in the following theorem —for more details on randomized methods see [49].

Theorem 2.4.2 ([49]). *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a positive semi-definite matrix. Select a target rank $2 \leq p \leq n/2$ and a number q of power iterations. Execute the Randomized SVD to obtain a rank-$2p$ factorization $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ of $\mathbf{A}$. If, further, we incorporate the truncation step to retain only the first p eigenvalues and vectors we obtain the following bound*

$$\mathbb{E}\left\|\mathbf{A} - \mathbf{U}_p\mathbf{\Sigma}_p\mathbf{U}_p^T\right\|_2 \leq \lambda_{p+1} + \left[1 + 4\sqrt{\frac{2n}{p-1}}\right]^{1/(2q+1)} \lambda_{p+1},$$

where the expectation is taken over the randomness of the test matrix and λ_{p+1} denotes the $(p+1)^{th}$ eigenvalue of $\mathbf{A}$.

Note that the above bound depends on the $(p+1)^{\text{th}}$ eigenvalue of $\mathbf{A}$ which is typically a very small positive real number (in practice, we encounter matrix structures

such that there is a large gap between the p^{th} and the $(p+1)^{\text{th}}$ eigenvalues). Thus, the Randomized SVD method produces an efficient low rank approximation. The Randomized SVD requires $\mathcal{O}(n^2 \log(p))$ computations which constitutes a clear advantage over the deterministic methods. Finally, we emphasize that Theorem 2.4.2 can be applied to matrices which are not positive semi-definite, i.e., it applies to any $\mathbf{A} \in \mathbb{R}^{n \times m}$ matrix.

2.4.2 Nyström method

In the context of computational complexity, the Nyström method obtains a good low rank approximation of a positive semi-definite matrix $\mathbf{A}$ with cheap per-iteration cost [53, 54, 55]. Let a set $S_N = \{1, 2, \ldots, n\}$ with $S_p \subseteq S_n$ and denote s_i be the i^{th} element of S_p, where $i = 1, 2, \ldots, p$ and $p \leq n$. The method comprises the following steps

1. Construct matrix $\mathbf{B} \in \mathbb{R}^{p \times n}$ such that the i^{th} row of $\mathbf{B}$ is the s_i row of $\mathbf{A}$

2. Construct matrix $\mathbf{C} \in \mathbb{R}^{p \times p}$ such that the $(i,j)^{\text{th}}$ element of $\mathbf{C}$ is the $(s_i, s_j)^{\text{th}}$ element of $\mathbf{A}$ and then compute the pseudo-inverse $\mathbf{C}^{+}$

3. Construct matrix $\mathbf{D} \in \mathbb{R}^{n \times p}$ such that the i^{th} column of $\mathbf{D}$ is the s_i column of $\mathbf{A}$

Then, the Nyström method builds a low-rank approximation of $\mathbf{A}$ as

$$\mathbf{A}_p = \mathbf{D}\mathbf{C}^{+}\mathbf{B}. \tag{2.28}$$

In general, the set S_p can be constructed using different sampling methods [53, 54, 55]. In this work we consider the naive Nyström method which is based on uniform sampling without replacement. In particular, we can construct a matrix $\mathbf{P} \in \mathbb{R}^{n \times p}$ such that i^{th} column of $\mathbf{P}$ is the s_i column of the identity matrix $\mathbf{I}_{n \times n}$.

Then, equation (2.28) can alternatively be seen as

$$\mathbf{A}_p = (\mathbf{AP}) \left(\mathbf{P}^T\mathbf{AP}\right)^+ (\mathbf{AP})^T . \tag{2.29}$$

It has been shown that, with high probability, the error bound of the naive Nyström method is governed by the λ_{p+1} eigenvalue of $\mathbf{A}$ [54]. This means that we can obtain accurate approximations when there is a large gap between the p^{th} and the $(p+1)^{\text{th}}$ eigenvalues.

Chapter 3

Sparse Temporal Difference Learning via Alternating Direction Method of Multipliers

Convex optimization methods have found many applications in myriad applied fields. Unfortunately, there exist a number of optimization problems where the convexity of the objective function cannot be assumed. In this setting, local optima are not guaranteed to offer "good" solutions. As a fixed-point problem, least-squares temporal difference for Reinforcement Learning falls under this category. More interestingly, recent work in off-line Reinforcement Learning has focused on efficient algorithms to incorporate feature selection, via ℓ_1-regularization, into the Bellman operator fixed-point estimators. These developments now mean that over-fitting can be avoided when the number of samples is small compared to the number of features. However, it remains unclear whether existing algorithms have the ability to offer good approximations for the task of policy evaluation and improvement. In this chapter, we propose a new algorithm for approximating the ℓ_1-regularized fixed-point based on the Alternating Direction Method of Multipliers (ADMM). We argue that the new ADMM is well suited to the aforementioned fixed-point problem, even though it reduces to a non-convex optimization problem, by demonstrating, with

experimental results, that the proposed algorithm is more stable for policy iteration compared to prior work. Furthermore, we derive a theoretical result that states the proposed algorithm obtains a solution which satisfies the optimality conditions for the fixed-point problem.

3.1 Introduction

Reinforcement Learning (RL) is a sub-field of machine learning [16]. As the name suggests, it considers learning problems where, when interacting with the environment, we learn what to do or what actions to take in order to optimize the desirable outcome. In particular, RL considers an agent which interacts with the environment. The agent finds itself in a current state and faces the dilemma of what action to select, since it is not told which action performs optimally. Moreover, after executing an action, the environment returns a new state together with a reward (indicating how good the selected action was) and the procedure continues in the same manner. In practice, we face problems where a sequence of actions (policy) needs to be taken in order to achieve our goal. The desirable outcome (or goal) in RL is to maximize the sum of the expected reward, or equivalently, to find the assignment of actions to each state that, when executed, maximizes the sum of the expected reward (optimal policy).

Learning what actions to execute is of central importance in on-line RL. Without knowing which actions yield the total expected reward, the agent must first gain experience by executing actions that have not been selected in the past. To this end, the agent exploits the acquired knowledge and thus selects actions that perform optimal in the long-term run. This is the so called exploration/exploitation trade-off which, in essence, balances the task of exploration and exploitation.

In off-line (or batch) RL [22], the situation is slightly different in the sense that the information is collected a priori. In this case, the agent is not allowed to interact with the environment in order to obtain the optimal policy but instead is given a fixed

set of sampled states and actions, typically finite. Using the given information, the agent forms a random policy which is then used to interact with the environment —the policy is fixed and does not improve during the procedure. Hence, the goal in off-line RL is, given the existing data, to find the policy that maximizes the sum of rewards.

A core problem in off-line RL emerges in situations where the state space is large. In such cases, explicit computation of the value functions becomes infeasible — rewards are represented via the value functions which are then subject to maximization. Instead, approximation techniques provide the only way forward. In particular, common choices to represent the value functions are those of linear architecture [23] where the hypothesis space $\mathcal{F}$ is defined by a set of feature vectors. In this domain, Least-Squares Temporal Difference (LSTD) algorithms [24, 25, 26] attempt to find the fixed-point of the projected Bellman operator, ΠT, by using a rich number of samples. Unfortunately, for off-line learning, it is typically the case in practice that the amount of available data is not sufficient, leading LSTD to poor predictions. Indeed, in the regression setting, when only a small number of samples is available relative to the number of features, the least-squares method is known to be very vulnerable to over-fitting. A typical way to overcome this issue is by incorporating ℓ_1- and/or ℓ_2-regularization known as *Lasso* [56] and *ridge*-regression [57], respectively. The former turns out to be of particular interest in the context of high-dimensional problems since it produces sparse solutions and therefore performs feature selection.

Many authors have explored regularized approximations for the value functions [3, 27, 28, 29, 30, 31] as a means to address the over-fitting problem in RL. However, the majority of these methods are not able to both produce sparse solutions and treat the function approximation as a fixed-point problem. In particular, in [3] the authors perform ℓ_1-regularization to the Bellman Residual Minimization (BRM). This means that the method performs feature selection, however, the optimization problem is not treated as a fixed-point problem. In [27] the ℓ_2-regularized fixed-

point problem is studied which overcomes the over-fitting issue but is not able to return sparse solutions. In [28, 29] the authors take a different route by solving a convex optimization problem. As such, the ℓ_1-regularization can be easily incorporated but, again, the optimization problem loses its interpretation as a fixed-point problem. On the other hand, several recent methods have been proposed which add an ℓ_1 penalty to the fixed-point of the composed Bellman operator. In [31] the authors were the first to introduce the ℓ_1-regularization of the least-squares fixed-point. As the name suggests, their LARS-TD algorithm is inspired by the Least Angle Regression (LARS) algorithm. However, as is shown, the algorithm only converges to the fixed-point under some strong assumptions which rarely hold in the context of policy iteration (see Section 3.2.5) where the goal is to find the optimal policy (also known as control problem). Further, LARS as a homotopy method needs to compute the complete path of regularization parameters. Therefore, the optimal regularization parameter is guaranteed to be selected, however, in cases where a dense solution is required, this fact may result in an inefficient method in terms of time complexity. Next, in [30] the authors compute the same fixed-point using the linear complementarity formulation but again the algorithm shares the same conditions with LARS-TD.

To this end, for such methodology to be practically feasible when aiming for the optimal policy, there still remains an apparent need to introduce new algorithms to TD learning in order to efficiently evaluate the ℓ_1-regularized fixed-point problem within policy iteration. In this chapter we propose solving the ℓ_1-regularized fixed-point problem with the help of the Alternating Direction Method of Multipliers (ADMM) which has been shown to be able to efficiently handle large problems (for details on ADMM see Section 2.3 and [10, 11]). Thus, our goal is to develop a method that not only overcomes time complexity issues but also offers optimal solutions to the context of policy iteration —an area of off-line TD learning which still suffers. To be precise, our contributions are as follows:

- We employ ADMM for solving the ℓ_1-regularized fixed-point problem. To

the best of our knowledge this is the first ADMM method to be applied on TD learning. For this reason we name our algorithm ADMM-TD.

- In terms of computational complexity, since the ℓ_1-regularized fixed-point problem is a separable, ADMM is able to exploit its structure by introducing new variables. This means that ADMM handles each subproblem independently thus yielding an efficient algorithm in the context of large-scale optimization. We note that if, further, the regularization parameter is known a-priori, we come up with an even faster optimization method.

- We aim to establish theoretical guarantees of ADMM in the TD context. In particular, we show that the ADMM-TD solution satisfies the ℓ_1-regularized fixed-point problem optimality conditions. This means that we should expect efficient approximate solutions comparable with the state-of-the-art, see LARS-TD in [31]. Note that the TD learning problem, as a fixed-point problem, it does not correspond to any convex optimization problem and hence, a proof of convergence is a challenging task and remains open.

- Our preliminary numerical experiments illustrate the efficacy of ADMM-TD. In particular, for the prediction problem, ADMM-TD compares similarly to LARS-TD both yielding the same approximation error (this fact was also explained by our theoretical analysis). However, one should expect for LARS-TD to return better approximate solutions since it searches the full regularization path (and is able to find the optimal parameter) while ADMM-TD searches only a small path. This indicates that searching for the full path is possibly not always necessary. In turn, this means that LARS-TD performs redundant iterations which result in an increased computational cost. This is a fact that ADMM-TD exploits by searching only a small subset of the full path. Further, as noticed above, if the optimal regularization parameter is known at hand, then LARS-TD loses its efficiency in comparison to non-homotopy methods.

- In the context of policy iteration, our experimental results show a more stable

performance for ADMM-TD compared to LARS-TD.

The rest of this chapter is organized as follows. In Section 3.2, we provide the required background. Specifically, we first review the basic RL problem together with the necessary theory. We then discuss how the value functions can be approximated. Further we discuss state-of-the-art methods and we present the so called ℓ_1-regularized fixed-point problem.

In Section 3.3, we present the new ADMM algorithm for TD learning. We start by deriving the algorithm steps and then prove that the solution obtained from the algorithm satisfies the optimality conditions. We also derive the stopping criteria of ADMM-TD. Furthermore, we validate the efficiency of the algorithm through several experiments.

In Section 3.4, we discuss our contributions in the context of the most relevant, recent work, we provide a general discussion about the proposed method.

3.2 Reinforcement Learning: Background Knowledge

The RL problem is modeled by a Markov Decision Process (MDP). It considers an *agent* which interacts in a given *environment*. In particular, the agent finds itself in a *state* at a certain point in time, and the interaction occurs when executing an *action* from the set of the available actions. Subsequently, the agent observes a new state and receives a *reward*. Based on this new state, the agent faces a similar task, where now the set of actions might not be the same (see Figure 3.1). The set of executed actions at each state defines a *policy*. In this context, the agent's goal is to find an optimal policy by maximizing the sum of the expected reward.

More formally, a MDP [2] is defined as the tuple $\langle S, A, P, R, \gamma \rangle$, where S denotes the set of states and $A(s)$ the set of actions available at each state; a policy, $\pi : S \rightarrow$

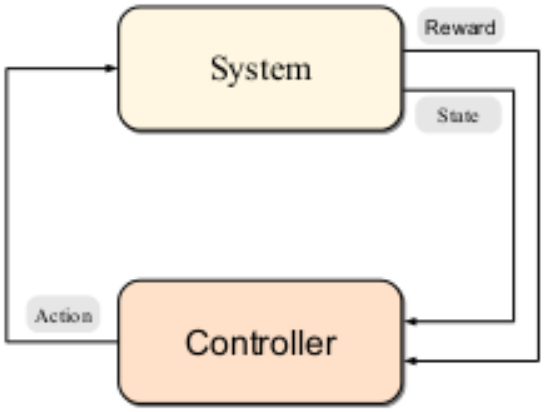


Figure 3.1: A basic reinforcement learning problem. The agent (controller) interacts with the environment (system), by executing an action, which then returns a new state along with the associated reward (plot taken from [1]).

A, is a mapping from states to actions; $P : S \times A \times S \rightarrow p(s'|s, \pi(s)) \in [0, 1]$ are the transition probabilities of moving to a new state, s', after executing an action $\pi(s) \in A(s)$. When reaching a new state the system returns a reward (or a cost), $R(s, \pi(s), s') : S \rightarrow \mathbb{R}$; $\gamma \in [0, 1)$ is the discount factor. For ease, we consider discrete state space and finite MDPs. We further assume that the rewards are bounded by a scalar $M < \infty$, for all $s \in S$ and $\pi(s) \in A$, i.e., $\sup \{R(s, \pi(s), s')\} \leq M$.

Let $t = 0, 1, 2, ..$ be the discrete time step, e.g., $s_t = s$ and $s_{t+1} = s'$. The sequence of rewards received by the agent at each state is denoted as $r_{t+1}, r_{t+2}, ...$, and thus the goal is to maximize the discounted sum of rewards, that is

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

It is easy to see that maximizing the above equation in absence of the discount factor, the sum may be infinite. Moreover, γ can be viewed as a weight on future reward, e.g., as γ approaches zero, the agent accounts more for the early returns. Moreover, the expected reward, at a given time step t, state s and when executing

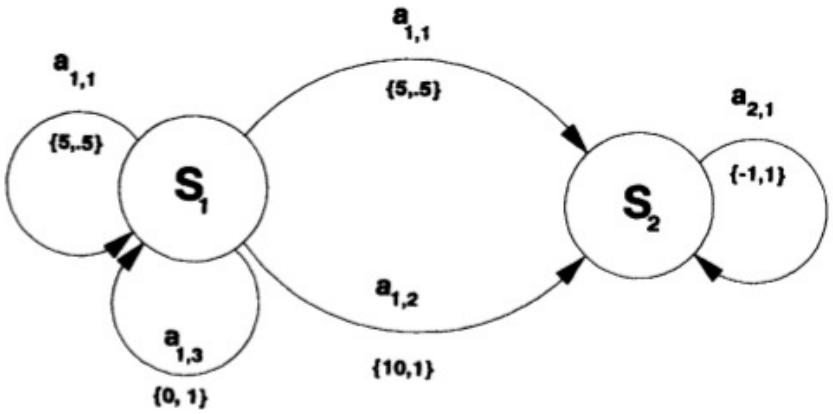


Figure 3.2: A simple environment of two states (plot taken from [2]).

an action a, can be calculated by the following equation

$$R_t(s,a) = \sum_{s' \in S} r(s,a,s')p(s'|s,a).$$

In what follows, we revise two simple examples.

Example 1 (Recycling Robot, [16]): We describe a simple real-world example, that is a robot which has as a job to collect empty cans in an office environment. The robot is supplied with a "hand" for collecting the cans and a bin to recycle them while it is moving. Moreover, it has sensors for detecting empty cans, a navigation system and a rechargeable battery. The robot is controlled by a reinforcement learning agent and in order to complete its task the agent acts according to the level of the battery and not according to any factors of the external environment. As a result, we define the set of states as the current charge level of the battery, i.e., $S = (s_1, s_2)$, where s_1 and s_2 respectively denote the *high* and the *low* level of the battery. The action set available to the agent consists of three actions, $A = (a_1, a_2, a_3)$, where a_1 denotes that the agent takes the decision to actively search and collect empty cans, a_2 to remain in the same position and wait for someone to collect a can for itself and a_3 to recharge the battery. Furthermore, when a_1 is executed (searching for cans) it is likely that the robot runs out of battery, while when staying stationary, a_2, the battery level remains the same. If the battery becomes depleted, the robot must be

rescued and returned to the charging point (yielding a negative reward). We can also define the action set available to the agent at each state, that is, $A(s_1) = (a_1, a_2)$ and $A(s_2) = (a_1, a_2, a_3)$ —note that a_3 is not included in the $A(s_1)$ set since recharging the robot while the battery level remains high is meaningless.

Additionally, the transition probabilities and the rewards of moving to a new state s' are the following: (i) $P(s_1, a_1, s_1) = \alpha$, that is, the probability that the robot is currently at high level of battery (s_1), completes the searching task (a_1) and the battery still remains fully charged (s_2) is α, while $P(s_1, a_1, s_2) = 1 - \alpha$. In both cases, the agent is rewarded the same i.e., $R(s_1, a_1, s_1) = R(s_1, a_1, s_2) = r_1$; (ii) $P(s_2, a_1, s_2) = \beta$ and $P(s_2, a_1, s_1) = 1-\beta$ while the rewards are $R(s_2, a_1, s_2) = r_1$ and $R(s_2, a_1, s_1) = -1$ (in the latter case the reward is negative since the robot ran out of battery and had to be rescued); (iii) $P(s_1, a_2, s_1) = 1$ and $R(s_1, a_2, s_1) = r_2$, i.e, the level of the battery always remains the same when robot is waiting (a_2) which yields a reward r_2. Note that $r_1 > r_2,\ r_1, r_2 > 0$ since the robot is able to collect more empty cans when searching. Finally, when robot recharges the battery we assume that no cans are collected and thus it yields a zero-reward. See Table 3.1 for the full list of transition probabilities and rewards.

Table 3.1: Recycling Robot: Transition probabilities and expected rewards.

s	s'	a	$P(s, a, s')$	$R(s, a, s')$
s_1	s_1	a_1	α	r_1
s_1	s_2	a_1	$1 - \alpha$	r_1
s_1	s_1	a_2	1	r_2
s_1	s_2	a_2	0	r_2
s_2	s_1	a_1	$1 - \beta$	-1
s_2	s_2	a_1	β	r_1
s_2	s_1	a_2	0	r_2
s_2	s_2	a_2	1	r_2
s_2	s_1	a_3	1	0
s_2	s_2	a_3	0	0

Example 2 [2]: Figure 3.2 illustrates a simple two-state stationary model, where rewards and transition probabilities remain the same at each time step. In state s_1 the agent is allowed to select from three available actions ($a_{1,1}, a_{1,2}, a_{1,3}$), while state s_2

supplies the agent with only one action $a_{2,1}$. The arrows indicate the possible transitions. The braces under each arrow indicate the rewards and transition probabilities of executing an action, respectively. For instance, when executing action $a_{1,1}$ the system evolves in states s_1 and s_2, both, with probabilities 0.5, respectively, and the agent receives a reward of 5 units in both states. We can now formally define the stationary model (assuming only transition probabilities) of Figure 3.2 as follows

Set of states :

$$S = \{s_1, s_2\}.$$

Set of actions :

$$A(s_1) = \{a_{1,1}, a_{1,2}, a_{1,3}\},\ A(s_2) = \{a_{2,1}\}.$$

Rewards :

$$\begin{array}{ll} r(s_1, a_{1,1}, s_1) = 5 & r(s_1, a_{1,2}, s_2) = 10 \\ r(s_1, a_{1,1}, s_2) = 5 & r(s_1, a_{1,3}, s_1) = 0 \end{array} \qquad r(s_2, a_{2,1}, s_2) = -1.$$

Transition probabilities :

$$\begin{array}{ll} p(s_1, a_{1,1}, s_1) = 0.5 & p(s_1, a_{1,2}, s_2) = 1 \\ p(s_1, a_{1,1}, s_2) = 0.5 & p(s_1, a_{1,3}, s_1) = 1 \end{array} \qquad p(s_2, a_{2,1}, s_2) = 1.$$

Note that transition probabilities $p(s_2, a_{2,1}, s_1),\ p(s_1, a_{1,2}, s_1),\ p(s_1, a_{1,3}, s_2)$ are equal to zero. We can now calculate the expected rewards. **Expected rewards** :

$$\begin{aligned} R(s_1, a_{1,1}) &= p(s_1, a_{1,1}, s_1) r(s_1, a_{1,1}, s_1) + p(s_1, a_{1,1}, s_2) r(s_1, a_{1,1}, s_2) \\ &= 0.5 \times 5 + 0.5 \times 5 = 5, \end{aligned}$$

similarly we find

$$R(s_1, a_{1,2}) = 10 \qquad R(s_2, a_{2,1}) = -1.$$
$$R(s_1, a_{1,3}) = 0 \qquad \square$$

Therefore, the goal is to find the policy that returns the optimal behavior. This can be achieved by computing the so called value functions, which we review in the following section.

3.2.1 Value Functions and Optimal Value functions

A policy can be evaluated through the value functions [16]. The *state-value* function, $V : S \rightarrow \mathbb{R}$, as the name indicates, denotes the value of a state for a given policy and is defined as

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[R_t | s_t\right] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t\right].$$

Note that the index π of $\mathbb{E}_{\pi}$ refers to the given policy. The state-value function can be written in a recursive form showing the relationship between the current and successor states, i.e.,

$$V^{\pi}(s) = \sum_{a \in A(s)} \pi(s,a) \sum_{s' \in S} P(s,a,s') \left[R(s,a,s') + \gamma V^{\pi}(s')\right], \tag{3.1}$$

where $\pi(s,a)$ is the probability of selecting an action a. However, in practical problems we only consider deterministic policies where $\pi(s,a) = 1$, i.e., the agent always executes the same action in a given policy and thus we often denote $a = \pi(s)$.

Alternatively, one can evaluate a policy using the *action-value* function, $Q : S \times A \rightarrow \mathbb{R}$. The action-value function indicates "how good" an action is, and is defined

as

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}_{\pi}\left[R_t|s_t,a_t\right] \\ &= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t,a_t\right] \\ &= \sum_{s'}P(s,a,s')\left[R(s,a,s)+\gamma V^{\pi}(s')\right], \end{aligned} \tag{3.2}$$

which effectively connects the state- and the action-value functions. The value functions in their recursive form are also known as the *Bellman's Equations* for V^{π} and Q^{π}, respectively.

The optimal state- and action-value functions are defined as

$$\begin{aligned} V^*(s) &= \max_{\pi} V^{\pi}(s), \quad \text{for all } s \in S, \\ Q^*(s,a) &= \max_{\pi} Q^{\pi}(s,a), \quad \text{for all } s \in S \text{ and } a \in A(s), \end{aligned}$$

respectively. Again, Q^* can be written in terms of V^* as

$$Q^*(s,a) = \mathbb{E}\left[r_{t+1}+\gamma V^*(s')|s_t,a_t\right].$$

Subsequently, we can define Bellman's optimality equations for V and Q which satisfy the following recursive forms

$$\begin{aligned} V^*(s) &= \max_{a\in A(s)} Q^{\pi^*}(s,a) \\ &= \max_{a\in A(s)} \sum_{s'} P(s,a,s')\left[R(s,a,s')+\gamma V^*(s')\right], \end{aligned}$$

and

$$\begin{aligned} Q^*(s,a) &= \mathbb{E}\left[r_{t+1}+\gamma \max_{a'} Q^*(s_{t+1},a')|s_t,a_t\right] \\ &= \sum_{s'} P(s,a,s')\left[R(s,a,s')+\gamma \max_{a'} Q^*(s',a')\right], \end{aligned}$$

respectively. Equivalently, Bellman's optimality equation for Q can be written as

$$Q^*(s,a) = R(s,a,s') + \gamma \sum_{s'} P(s,a,s')V^*(s').$$

Using the optimal value functions one can obtain the optimal policy. We say a policy π^* is optimal if and only if $V^{\pi^*} \geq V^{\pi}$, for any policy π, i.e., it obtains the largest expected reward, and we denote $\pi^* \geq \pi$. An optimal policy π^* always exists, but is not necessarily unique even though the optimal value functions are unique (for more details see [16]). The formula for π^* is given by the following equation

$$\begin{aligned}\pi^*(s) &= \operatorname*{argmax}_a \; Q(s,a) \\ &= \operatorname*{argmax}_a \sum_{s'} P(s,a,s') \left[R(s,a,s') + \gamma V^{\pi}(s')\right].\end{aligned}$$

In addition, Bellman's equation for V can be viewed as a linear system of equations, and since we assume large but finite S and A sets it can be expressed in matrix form. In particular, assuming deterministic policies ($\pi(s,a) = 1$), equation (3.1) can be written as

$$V^{\pi} = R + \gamma P^{\pi} V^{\pi}, \tag{3.3}$$

which is a set of $|S|$ linear equations, $V \in \mathbb{R}^{|S|}$, where $|S|$ is the total number of states. Furthermore, R is the vector of rewards, $R \in \mathbb{R}^{|S|}$,

$$R = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_{|S|} \end{bmatrix} = \begin{bmatrix} \sum_{s' \in S} P(s_1,a,s')R(s_1,a,s') \\ \sum_{s' \in S} P(s_2,a,s')R(s_2,a,s') \\ \vdots \\ \sum_{s' \in S} P(s_{|S|},a,s')R(s_{|S|},a,s') \end{bmatrix}, \quad s \in S,$$

and P^π is the matrix of transition probabilities, $P \in \mathbb{R}^{|S|\times|S|}$,

$$P^\pi = \begin{bmatrix} P(s_1) \\ P(s_2) \\ \vdots \\ P(s_{|S|}) \end{bmatrix}, \quad P(s_i)^T \in \mathbb{R}^{|S|} \text{ for } i = 1, \ldots, |S|.$$

Finally, note that the Bellman's optimality equations for V and Q are not linear.

3.2.2 Bellman Operator

We consider the Bellman operator in order to calculate the value and the optimal value functions. Let $B(s) \in \mathbb{R}^{|S|}$ be the set of all state-value functions, i.e.,

$$B(s) = \{V | V : S \to \mathbb{R}\}, \ ||V||_\infty < \infty\}, \quad ||V||_\infty = \max_{s\in S} \{|V(s)|\}.$$

The Bellman operator [1], $T_\pi : B(s) \to B(s)$, is defined as

$$(T^\pi V)(s) = R(s, \pi(s), s') + \gamma \sum_{s'\in S} P(s, \pi(s), s')V^\pi(s'), \quad s \in S.$$

From the above equation and equation (3.3) we have that

$$T^\pi V^\pi = V^\pi. \tag{3.4}$$

Moreover, it has been shown that T^π is a maximum norm contraction (L-Lipschitz continuous with $L \in (0, 1)$), i.e.,

$$\|T^\pi W - T^\pi V\|_\infty \leq \gamma \|W - V\|_\infty,$$

for $0 < \gamma < 1$. Therefore, Banach's fixed-point theorem applies, which in turn implies the existence and uniqueness of the fixed-point of equation $T^\pi V^\pi = V^\pi$ —for more details see [1].

Alternatively, one can compute the state-value function by solving a linear system of equations. Equation (3.4) implies that $V^\pi = R + \gamma P^\pi V^\pi$, and when R and P are known, which is typically not true in practice, it can be solved analytically yielding

$$V^\pi = (I - \gamma P^\pi)^{-1} R.$$

Similarly, we can define the Bellman operator for the action-value function. Let now $B(s,a) \in \mathbb{R}^{|S|\times|A|}$ be the set of all action-value functions such that

$$B(s,a) = \{Q | Q : S \times A \to \mathbb{R}\}, \ ||Q||_\infty < \infty\}, \quad ||Q||_\infty = \max_{s\in S} \{|Q(s,a)|\}.$$

Hence, the Bellman operator, $T_\pi : B(s,a) \to B(s,a)$, is defined as

$$T^\pi Q^\pi(s,a) = R(s,a,s') + \gamma \sum_{s'\in S} P(s,a,s') Q(s', \pi(s')), \quad (s,a) \in S \times A,$$

which, as before, yields

$$T^\pi Q^\pi = R + \gamma P^\pi Q^\pi.$$

Again, due to Banach's theorem, we have that the fixed-point of $T^\pi Q^\pi = Q^\pi$ exists and is unique, for $\gamma \in (0,1)$.

Finally, Banach's fixed-point theorem also holds for the Bellman's optimality operator T^*, for both V^* and Q^*. Define the Bellman optimality operator for V^* as $T^*_V : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ and for Q^* as $T^*_Q : \mathbb{R}^{|S|\times|A|} \to \mathbb{R}^{|S|\times|A|}$. It has been shown that the fixed-point equations

$$T^*_V V^* = V^* \quad \text{and} \quad T^*_Q Q^* = Q^*,$$

have unique solutions since operators T^*_V and T^*_Q are maximum norm contractions for $\gamma \in (0,1)$.

3.2.3 Function Approximation

In many practical problems, we often deal with large state spaces and thus storing and filling arrays of the above form becomes infeasible due to time and memory constraints. In such situations, approximation of the value functions is necessary. The most common choice to represent the value functions are those of linear architecture using a set of features, i.e,

$$\hat{V}^{\pi} = \Phi w,$$

where $\Phi \in \mathbb{R}^{|S| \times n}$ is the feature matrix, that is

$$\Phi = \begin{bmatrix} \phi(s_1)^T \\ \phi(s_2)^T \\ \vdots \\ \phi(s_{|S|})^T \end{bmatrix},$$

where $\phi(s_i) \in \mathbb{R}^n,\ i = 1, \ldots, |S|$ and $w \in \mathbb{R}^n$ is the vector of weights. Similarly, action-value function can be approximated as follows

$$\hat{Q}^{\pi} = \Phi w = \sum_{i=1}^{n} w_i \phi_i(s),$$

where $\hat{Q}^{\pi} \in \mathbb{R}^{|S||A|},\ \Phi \in \mathbb{R}^{|S||A| \times n}$ and $w \in \mathbb{R}^n$.

Gaussian Radial Basis Functions

Throughout this chapter, we will be representing the value functions using Gaussian Radial Basis functions (RBFs) as features,

$$f_{t_0, r_0}(x) = e^{-t_0 (x - r_0)^2},$$

where t_0 denotes the scale factor of the RBF and r_0 its location. Given the scale factor t_0, the location r_0, N-number of states and n-number of features we have

that

$$\phi(s) = \begin{bmatrix} 1 & f_{t,r_1}(s) & \cdots & f_{t,r_{n-1}}(s) \end{bmatrix}^T,$$

where $s \in S$, $t = \frac{r_0(n-2)}{N-1}$ and $r_j = 1 + (j-1)\frac{N-1}{n-2}$ with $j = 1, \ldots, n-1$. Furthermore, it is common to use two dimensional RBFs when the state space lies in two dimensions, i.e.,

$$f_{t_0,r_0}(x, y) = e^{-(t_x(x-r_x)^2 + t_y(y-r_y)^2)},$$

where, now, $t_0 = (t_x, t_y)$ is the scale factor of the RBF and $r_0 = (r_x, r_y)$ denotes its location in $\mathbb{R}^2$, respectively. Additionally, we define the mapping between states $s \in S$ and the $\mathbb{R}^2$ space as

$$(x, y) \in [0, x_{max}] \times [0, y_{max}].$$

Again, given N-number of states, a $g \times g$ grid, the scale $t_0 = (t_x, t_y)$ and the location $r_0 = (r_x, r_y)$, the feature vector is given by

$$\phi(s = x, y) = \begin{bmatrix} 1 & f_{t,r_1}(s) & \cdots & f_{t,r_{n-1}}(s) \end{bmatrix}^T,$$

where $s \in S$, $t = (\frac{t_x(g+1)}{x_{\max}}, \frac{t_y(d+1)}{y_{\max}})$ and, $r_j \in \{(\frac{x_{\max}}{g+1}k, \frac{y_{\max}}{g+1}d) \mid k, d = 1, \ldots, g\}$, $j = 1, \ldots, g^2$, thus total number of features is given by $n = g^2 + 1$. It is also possible to concatenate RBFs at different scales (or otherwise at different grids). We call these multilevel RBFs, where now the total number of features is given by $n = 1 + \sum_{i=1}^{q} g_i^2$, where q denotes the total number of different levels. For example, let $[2, 4, 8, 16, 32]$ be the two dimensional grid, that is, we concatenate the RBFs at different scales plus a constant offset as discussed above. As a result, the total number of features is $n = 1 + \sum_{i=1}^{5} g_i^2 = 1365$ (for more details see [4, Section 5.2.5].

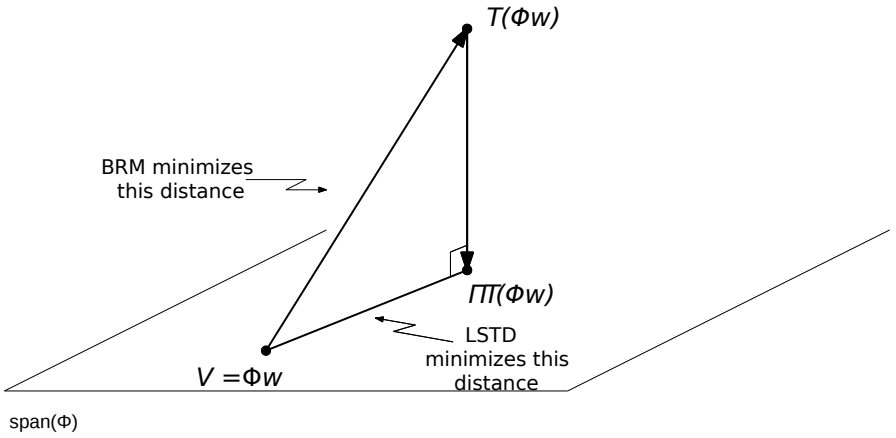


Figure 3.3: LSTD fixed-point. Φw lies on the hypothesis space $\mathcal{F}$ spanned by the columns of Φ (i.e., $\mathrm{span}(\Phi)$). However, when applying the Bellman operator T, $T(\Phi w)$ does not necessarily lie onto $\mathrm{span}(\Phi)$. Hence, LSTD first minimizes the distance $\|T(\Phi w) - y\|,$ for any $y \in \mathcal{F}$, yielding the projection $\Pi T(\Phi w)$ onto $\mathrm{span}(\Phi)$, and then minimizes $\|\Phi w - \Pi T(\Phi w)\|$. On the other hand, Bellman Residual Minimization (BRM) minimizes $\|\Phi w - T(\Phi w)\|$ directly, however, this optimization problem does not reduces to the fixed-point problem of TD learning (for more details on BRM see [3]) —plot taken from [4].

3.2.4 Least-Squares Temporal Difference

Approximating value function as discussed above, defines a hypothesis space spanned by the columns of $\mathbf{\Phi}$, i.e., $\mathcal{F} = \{\Phi w, w \in \mathbb{R}^n\}$. However, when applying the Bellman operator, the point $T^\pi \hat{V}^\pi$ does not necessarily lie onto $\mathcal{F}$. LSTD [26] solves the problem of approximating the vector w by projecting $T^\pi \hat{V}^\pi$ back onto the hypothesis space. The objective function subject to approximation is therefore defined as $\hat{V}^\pi = \Pi T^\pi(\hat{V}^\pi)$, where Π is the projection operator. As a consequence, LSTD searches for the fixed-point, $\hat{V}^\pi$, of the composed operator ΠT^π. The latter fixed-point problem can be then written as an optimization problem, i.e.,

$$w = f(w) = \theta = \underset{\theta \in \mathbb{R}^n}{\mathrm{argmin}} \|\Phi\theta - (R + \gamma P \Phi w)\|_\xi^2, \tag{3.5}$$

where $\xi \in \mathbb{R}^{|S|\times|S|}$ is a diagonal matrix with entries representing the stationary distribution of the states (where $\|x\|_\xi^2 = x^T\xi x$). To make things more clear, the above fixed-point problem is effectively a nested optimization problem, that is

$$\begin{aligned} \theta_w &= \underset{\theta\in\mathbb{R}^n}{\operatorname{argmin}} \|\Phi\theta - (R + \gamma P\Phi w)\|^2 \\ w^* &= \underset{w\in\mathbb{R}^n}{\operatorname{argmin}} \|\Phi w - \Phi\theta_w\|^2, \end{aligned} \tag{3.6}$$

where the first equation accounts for the projection while the second for the minimization (for intuition, see Figure 3.3).

The LSTD fixed-point problem (3.5) requires knowledge of P and the construction of a large matrix, Φ. To this end, LSTD collects m samples of the form, $\{s_i, a_i, r_i s_i'\}_{i=1,\dots,m}$, possibly sampled over several trajectories. This results in the sampled matrices

$$\tilde{\Phi} = \begin{pmatrix} \phi(s_1)^T \\ \phi(s_2)^T \\ \vdots \\ \phi(s_m)^T \end{pmatrix}, \quad \tilde{\Phi}' = \begin{pmatrix} \phi(s_1')^T \\ \phi(s_2')^T \\ \vdots \\ \phi(s_m')^T \end{pmatrix}, \quad \tilde{R} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix},$$

where $\Phi' = P\Phi$. Replacing the sampled features and reward matrices in (3.5) we have that

$$w = \underset{\theta\in\mathbb{R}^n}{\operatorname{argmin}} \|\tilde{\Phi}\theta - (\tilde{R} + \gamma\tilde{\Phi}'w)\|^2.$$

The above problem can be solved explicitly using the nested formulation of the fixed-point problem. Hence, equation (3.6), using the sampled features and rewards matrices replaced, implies

$$\begin{aligned} \tilde{\Phi}^T(\tilde{\Phi}\theta_w - (\tilde{R} + \gamma\tilde{\Phi}'w)) = 0 \\ \tilde{\Phi}^T(\Phi w - \Phi\theta_w) = 0 \end{aligned} \quad \Rightarrow \quad \begin{aligned} \tilde{\Phi}^T\tilde{\Phi}\theta_w - \tilde{\Phi}^T\tilde{R} - \gamma\tilde{\Phi}^T\tilde{\Phi}'w = 0 \\ w = \theta_w \end{aligned},$$

yielding the following set of linear equations

$$\tilde{A}w = \tilde{b} \;\Rightarrow\; w = \tilde{A}^{-1}\tilde{b} \tag{3.7}$$

where $\tilde{A} \in \mathbb{R}^{n\times n}$, $\tilde{b} \in \mathbb{R}^n$ are defined as

$$\begin{aligned}\tilde{A} &= \tilde{\Phi}^T\left(\tilde{\Phi} - \gamma\tilde{\Phi}'\right)\\ \tilde{b} &= \tilde{\Phi}^T\tilde{R},\end{aligned}$$

respectively. Alternatively, one can obtain the solution of the fixed-point problem by solving analytically the equation

$$\begin{aligned}\hat{V}^\pi &= \Pi T^\pi(\hat{V}^\pi)\\ \tilde{\Phi}w &= \tilde{\Phi}(\tilde{\Phi}^T\tilde{\Phi})^{-1}\tilde{\Phi}^T(\tilde{R} + \gamma\tilde{\Phi}'w)\end{aligned}$$

where the projection is given explicitly by $\Pi = \tilde{\Phi}(\tilde{\Phi}^T\tilde{\Phi})^{-1}\tilde{\Phi}^T$.

Similarly to V function, the objective function subject to approximation for Q is $\hat{Q}^\pi = \Pi T^\pi(\hat{Q}^\pi)$ which, again, can be solved analytically yielding the same set of linear equation, (3.7). The only difference is that the sampled feature matrices are now defined as follows

$$\tilde{\Phi} = \begin{pmatrix}\phi(s_1,a_1)^T\\ \phi(s_2,a_2)^T\\ \vdots\\ \phi(s_m,a_m)^T\end{pmatrix}, \quad \tilde{\Phi}' = \begin{pmatrix}\phi(s_1',\pi(s_1))^T\\ \phi(s_2',\pi(s_2))^T\\ \vdots\\ \phi(s_m',\pi(s_m))^T,\end{pmatrix}$$

where again $\Phi' = P\Phi$. Hence, the algorithm for the Q function is called LSTDQ.

For finding the optimal policy, the authors in [25] introduced Least-Squares Policy Iteration (LSPI) algorithm which is based on the general policy iteration framework for dynamic programming. Using the Q value, the optimal policy can be obtained

Algorithm 3.1 LSPI

Input:
$\{s_i, a_i, r_i, s'_i\}_{i=1,\ldots,n}$ and form $\tilde{\Phi}$
Initialize:
$\gamma \in [0, 1]$
$w_0 \leftarrow 0$ and obtain π_0
$\pi_{\text{new}} \leftarrow \pi_0$
repeat
  $\pi \leftarrow \pi_{\text{new}}$
  $\pi_{\text{new}} \leftarrow \underset{a \in A}{\text{argmax}}\{\tilde{\Phi} w\}$ (w evaluated by LSTDQ using policy π)
until convergence, i.e., $\pi_{\text{new}} \approx \pi$
return π

from

$$\pi^*(s, a) = \underset{a \in A}{\text{argmax}} \ \hat{Q}(s, a) = \underset{a \in A}{\text{argmax}} \ \sum_{i=1}^{n} \phi_i(s, a)^T w_i. \tag{3.8}$$

The LSPI algorithm is an "off-policy" mechanism in the sense that the policy used to estimate Q function is different from the one used to sample state-action pairs. LSPI works as follows: a random policy π (subject to evaluation), characterized by the sampled features and the weight parameter w, enters the LSTDQ together with the samples of the form $\{s_i, a_i, r_i, s'_i\}_{i=1,\ldots,m}$. Then, the maximization equation (3.8) is performed to identify the new policy π_{new} yielding also, through LSTDQ, an improved w_{new} vector. Afterwards, the algorithm continues in the same manner until either two successor policies to be equivalent or $\|w_{\text{new}} - w\| < \varepsilon$, for some small $\varepsilon > 0$. LSPI is shown in Algorithm 3.1.

In summary, LSTD has been found to perform very well when the number of samples is large and the number of features is small. However, there is a number of drawbacks that accompany LSTD. The main one emerges when $m < n$ (fewer samples compared to features). In this case, least-squares is prone to over-fitting and results in poor approximations. Moreover, the matrix $\tilde{A}$ need not be full column rank and thus its left-inverse is not always guaranteed to exist. Finally, when n is too large the method is not feasible due to memory and time constraints since

LSTD requires $O(n^2)$ computations.

3.2.5 LARS-TD

To overcome the limitations arising in LSTD, the authors in [31] proposed LARS-TD. They apply the ℓ_1-regularization penalty to the LSTD objective function which has the characteristic of avoiding over-fitting and performing sparse selection. More precisely, the penalty is added to the projection of $T^\pi V$ onto the hypothesis space $\mathcal{F}$, which yields the following optimization problem

$$w = \underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \|\tilde{\Phi}\theta - (\tilde{R} + \gamma \tilde{\Phi}' w)\|^2 + \lambda \|\theta\|_1, \tag{3.9}$$

or equivalently, it can be viewed in a nested form as

$$\begin{aligned} \theta_w &= \underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \|\tilde{\Phi}\theta - (\tilde{R} + \gamma \tilde{\Phi}' w)\|^2 + \lambda \|\theta\|_1 \\ w^* &= \underset{w \in \mathbb{R}^n}{\operatorname{argmin}} \|\tilde{\Phi} w - \tilde{\Phi}\theta_w\|^2. \end{aligned}$$

The above optimization problem can be alternatively written as a fixed-point

$$\tilde{\Phi} w = \tilde{\Pi}_{\ell_1} \tilde{T}(\tilde{\Phi} w),$$

for which it has been proved that the operator $\tilde{\Pi}_{\ell_1} \tilde{T}$ is a γ-contraction, which in turn ensures the existence and uniqueness of the fixed-point $\tilde{\Phi} w$, see [58] —although we note that w, itself, need not be unique.

To efficiently solve for the ℓ_1-regularized fixed-point in (3.9), the authors employ the Least Angle Regression (LARS) method. LARS is based on a homotopy method which allows the computation of the complete regularization path, for details see [59]. It has been shown that as long as $\tilde{A}$ is a P-matrix[1], each LARS-TD step satisfies the optimality conditions, and thus the algorithm always finds a solution to (3.9).

[1] A square matrix A, not necessarily symmetric, is a P-matrix when all its principle minors are positive.

It is instructive to review analytically the major steps of the optimality conditions since they play an important role for LARS-TD and our proposed algorithm (cf. Section 3.3).

Define $G(\theta) = \frac{1}{2}\|\tilde{\Phi}\theta - (\tilde{R} + \gamma\tilde{\Phi}'w)\|^2 + \lambda\|\theta\|_1$, and thus the optimality conditions for the convex problem are

$$0 \in \partial G(\theta), \tag{3.10}$$

where ∂ denotes the sub-differential (since $\|\theta\|_1$ is not differentiable). Moreover, we have that

$$\partial G(\theta) = \tilde{\Phi}^T(\tilde{\Phi}\theta - (\tilde{R} + \gamma\tilde{\Phi}'w)) + \lambda\partial\|\theta\|_1,$$

where

$$\partial\|\theta\|_1 \in \begin{cases} \{+1\}, & \theta_i > 0 \\ [-1, 1], & \theta_i = 0 \\ \{-1\}, & \theta_i < 0. \end{cases}$$

Therefore, equation (3.10) implies that

$$[\tilde{\Phi}^T(\tilde{\Phi}\theta - (\tilde{R} + \gamma\tilde{\Phi}'w))]_i \in \begin{cases} \{-\lambda\}, & \theta_i > 0 \\ [-\lambda, \lambda], & \theta_i = 0 \\ \{\lambda\}, & \theta_i < 0. \end{cases}$$

Now, setting $w = \theta$, as required at the fixed-point, the optimality conditions for the problem (3.9) become

$$[\tilde{\Phi}^T\tilde{R} - \tilde{\Phi}^T\left(\tilde{\Phi} - \gamma\tilde{\Phi}'\right)w]_i = [\tilde{b} - \tilde{A}w]_i \in \begin{cases} \{\lambda\}, & w_i > 0 \\ [-\lambda, \lambda], & w_i = 0 \\ \{-\lambda\}, & w_i < 0. \end{cases} \tag{3.11}$$

A solution w satisfying the optimality conditions yields the fixed-point $\tilde{\Phi}w$ of the composed operator $\tilde{\Pi}_{\ell_1}\tilde{T}^{\pi}$.

LARS-TD enjoys many of the benefits of LARS, in that it follows a homotopy path and hence it offers all the solutions $w^*(\lambda)$. Its computational complexity is $\mathcal{O}(mnk^2)$, where k denotes the cardinality of the active set. Therefore, if the solution is sparse enough, the algorithm can compute the fixed-point very efficiently, i.e., after a few numbers of iterations. On the other hand, LARS-TD also inherits the LARS drawbacks, too. If the whole path needs to be computed, the complexity reduces to a full least-squares, requiring to invert a nearly dense $\tilde{A}$, many times. Additionally, LARS-TD converges to the fixed-point under the assumption that $\tilde{A}$ is a P-matrix. However, $\tilde{A}$ is not necessarily a P-matrix when samples are collected off-policy where, inevitably, the distribution of states is different from the distribution of the underlying policy. To overcome this issue, the authors propose adding an ℓ_2-penalty to the fixed-point problem, known as elastic net [60], to ensure that $\tilde{A}$ is positive definite. Elastic net formulation of the problem (3.9) nevertheless comes at cost of reduced sparsity. More importantly, in the context of policy iteration, computing an almost complete regularization path could be inefficient, as also discussed in [30].

3.3 Sparse Temporal Difference Learning via ADMM

Our proposed approach is to apply ADMM to TD learning for solving the ℓ_1-regularized fixed-point problem. ADMM exploits some nice characteristics of the structure of (3.9) which match those of ℓ_1-regularization in linear regression. For this reason, we name the algorithm ADMM-TD.

3.3.1 ADMM-TD

We proceed by deriving the ADMM-TD steps. Consider the optimization problem (3.9) and note that it is convex over θ, i.e.,

$$\underset{\theta\in\mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\tilde{\Phi}\theta - (\tilde{R} + \gamma\tilde{\Phi}'w)\|^2 + \lambda\|\theta\|_1.$$

The above problem has the property of being separable and, hence, it can been split into two parts, namely $f(\cdot)$ and $g(\cdot)$. Furthermore, the requirement that the separate variables are equal yields the following equivalent problem

$$\text{minimize } \; f(\theta, w) + g(z) \quad \text{ subject to } \; \theta = z, \tag{3.12}$$

where $f(\theta, w) = \frac{1}{2}\|\tilde{\Phi}\theta - (\tilde{R} + \gamma\tilde{\Phi}'w)\|^2$ and $g(z) = \lambda\|z\|_1$. The ADMM-TD steps for the fixed-point problem can be derived through the augmented Lagrangian which, in terms of the proximal form, reduces to

$$w_{k+1} := f(w) = \theta_{k+1} \; = \; \mathbf{prox}_{\mu f}(z_k - u_k) \tag{3.13}$$

$$z_{k+1} \; := \; \mathbf{prox}_{\mu g}(w_{k+1} + u_k) \tag{3.14}$$

$$u_{k+1} \; := \; u_k + w_{k+1} - z_{k+1}, \tag{3.15}$$

where $u = \frac{1}{\rho}y, \; u \in \mathbb{R}^n$, denotes the scaled dual variable of the dual variable $y \in \mathbb{R}^n$ and $\mu = \frac{1}{\rho} > 0$ the step-size (or penalty) parameter (for more details on ADMM and its proximal version see [10, 11] and Section 2.3). The proximal operator of the first subproblem (3.13) is defined as

$$\theta_{k+1} := \underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \frac{1}{2}\|\tilde{\Phi}\theta - (\tilde{R} + \gamma\tilde{\Phi}'w)\|^2 + \frac{\rho}{2}\|\theta - z_k + u_k\|^2 \right\},$$

which can be solved by setting the gradient, w.r.t. θ, equal to zero, i.e.,

$$\nabla_\theta \left(\frac{1}{2}\|\tilde{\Phi}\theta - (\tilde{R} + \gamma\tilde{\Phi}'w)\|^2 + \frac{\rho}{2}\|\theta - z_k + u_k\|^2 \right) = 0,$$

which is equal to

$$\tilde{\Phi}^T(\tilde{\Phi}\theta - (\tilde{R} + \gamma\tilde{\Phi}'w)) + \rho(\theta - z_k + u_k) = 0.$$

At the fixed-point we require $w = \theta$ (as discussed in Section 3.2.4) and using simple algebra we have that

$$\begin{aligned} & \tilde{\Phi}^T\tilde{\Phi}w - \tilde{\Phi}^T\tilde{R} - \gamma\tilde{\Phi}^T\tilde{\Phi}'w + \rho w - \rho(z_k + u_k) = 0 \\ \Rightarrow\ & (\tilde{\Phi}^T\tilde{\Phi} - \gamma\tilde{\Phi}^T\tilde{\Phi}' + \rho I)w = \tilde{\Phi}^T\tilde{R} + \rho(z_k + u_k) \\ \Rightarrow\ & (\tilde{\Phi}^T(\tilde{\Phi} - \gamma\tilde{\Phi}') + \rho I)w = \tilde{\Phi}^T\tilde{R} + \rho(z_k + u_k), \end{aligned}$$

and thus it follows the fixed-point solution

$$w_{k+1} := \left(\tilde{A} + \rho I\right)^{-1}\left(\tilde{b} + \rho(z_k - u_k)\right), \tag{3.16}$$

where $\tilde{A} = \tilde{\Phi}^T\left(\tilde{\Phi} - \gamma\tilde{\Phi}'\right)$ and $\tilde{b} = \tilde{\Phi}^T\tilde{R}$. Note that ρ equal to zero yields exactly the LSTD fixed-point solution. For solving the second subproblem (3.14), we evaluate the proximal operator with respect to the previous iteration, i.e.,

$$z_{k+1} \ := \ \underset{z\in\mathbb{R}^n}{\operatorname{argmin}}\left\{\|z\|_1 + \frac{\rho}{2}\|w_{k+1} - z + u_k\|^2\right\}.$$

Using subdifferential theory we can obtain a closed form solution, that is,

$$0 \in \partial(\|z\|_1 + \frac{\rho}{2}\|w_{k+1} - z + u_k\|^2)$$

which reduces to the *soft-thresholding shrinkage operator* [61]

$$z_{k+1} \ := \ S_{\lambda/\rho}(w_{k+1} + u_k), \tag{3.17}$$

with

$$S_\beta(x) = \operatorname{sgn}(x) \odot \max\left\{|x| - \beta, 0\right\}, x \in \mathbb{R}^p, \tag{3.18}$$

and where $\mathrm{sgn}(x)$ is the signum function defined as

$$[\mathrm{sgn}(x)]_i = \begin{cases} +1, & x_i > 0 \\ 0, & x_i = 0 \\ -1, & x_i < 0. \end{cases} \tag{3.19}$$

The soft-thresholding is a component-wise operation, and thus $\odot$ denotes the component-wise multiplication. Equivalently, the soft thresholding operator can be seen in the following form

$$S_\beta(x) = (x - \beta)_+ - (-x - \beta)_+.$$

The ADMM-TD pseudocode is presented in Algorithm 3.2. Note that the vector w will be equal to z, and hence sparse, only in the limit as indicated by the subproplem (3.15).

Algorithm 3.2 ADMM-TD

Input:
$\{s_i, r_i, s_i'\}_{i=1,\ldots,n}$ and form $\tilde{\Phi}$ and $\tilde{\Phi}'$
Initialize:
$\gamma \in [0,1]$, $\lambda \geq 0$, $\rho > 0$
$\tilde{A} \leftarrow \tilde{\Phi}^T \left(\tilde{\Phi} - \gamma \tilde{\Phi}'\right)$, $\tilde{b} \leftarrow \tilde{\Phi}^T \mathbf{R}$
for $k = 0, 1, \ldots$ **do**
  $w^{k+1} := (\tilde{A} + \rho I)^{-1}(\tilde{b} + \rho(z^k - u^k))$
  $z^{k+1} := S_{\lambda/\rho}(w^{k+1} + u^k)$
  $u^{k+1} := u^k + w^{k+1} - z^{k+1}$
end for
return w

3.3.2 Stopping Criteria

The stopping criteria used for ADMM-TD are similar to those discussed by the authors in [10], see also Section 2.3. More precisely, for deriving the stopping criteria we first need to examine the necessary and sufficient optimality conditions

for the ADMM-TD problem. Those consist of primal feasibility

$$w^* - z^* = 0, \tag{3.20}$$

and dual feasibility

$$0 = \nabla_\theta f(w^*) + y^* \tag{3.21}$$

$$0 \in \partial g(z^*) - y^* \tag{3.22}$$

The above optimality conditions can be easily obtained from the augmented Lagrangian of the problem (3.12). In equation (3.21) we use the gradient, $\nabla_\theta f(w^*)$, and not the subdifferential, ∂, since $f(\cdot)$ is differentiable (subproblem 3.13).

First, primal feasibility, equation (3.20), implies that the primal residual

$$r_{k+1} = w_{k+1} - z_{k+1} \tag{3.23}$$

must be equal to zero as $k \to \infty$.

Moreover, we now have that θ_{k+1} minimizes $L_\rho(\theta, z_k, y_k)$. Taking the gradient w.r.t. θ we have that

$$\begin{aligned} 0 &= \nabla_\theta f(\theta_{k+1}, w) + y_k + \rho(\theta_{k+1} - z_k) \\ &= \nabla_\theta f(w_{k+1}) + y_k + \rho(w_{k+1} - z_k), \end{aligned}$$

where the second equation obtained by setting $w = \theta$ since we are searching the fixed-point. Incorporating now the dual variable update equation, $y_{k+1} = y_k + \rho(w_{k+1} - z_{k+1})$, into the last equation, we have that

$$\begin{aligned} 0 &= \nabla_\theta f(w_{k+1}) + y_{k+1} - \rho(w_{k+1} - z_{k+1}) + \rho(w_{k+1} - z_k) \\ &= \nabla_\theta f(w_{k+1}) + y_{k+1} + \rho(z_{k+1} - z_k), \end{aligned}$$

or

$$\rho(z_k - z_{k+1}) = \nabla_\theta f(w_{k+1}) + y_{k+1}.$$

The last two equations indicate that the quantity

$$d_{k+1} = \rho(z_k - z_{k+1}) \tag{3.24}$$

must vanish in the limit in order for the optimality condition (3.21) to hold. Moreover, d^{k+1} can be considered as the dual residual.

Furthermore, z_{k+1} minimizes $L_\rho(w_{k+1}, z, y_k)$ which yields

$$\begin{aligned} 0 &\in \partial g(z_{k+1}) - y_k - \rho(w_{k+1} - z_{k+1}) \\ &\in \partial g(z_{k+1}) - y_{k+1} + \rho(w_{k+1} - z_{k+1}) - \rho(w_{k+1} - z_{k+1}) \\ &\in \partial g(z_{k+1}) - y_{k+1}, \end{aligned}$$

implying that the optimality condition (3.22) is always satisfied.

As a result, primal and dual residuals (equations (3.23) and (3.24), respectively) can be chosen as the stopping criteria as both must converge to zero as $k \to \infty$. In particular, as similarly discussed in [10], we require primal and dual residual to be small quantities, that is

$$\|r^{k+1}\| \leq \varepsilon_{\text{primal}} \quad \text{and} \quad \|d^{k+1}\| \leq \varepsilon_{\text{dual}},$$

where the positive quantities $\varepsilon_{\text{primal}}$ and $\varepsilon_{\text{dual}}$ denote the primal dual feasibility tolerances, respectively. Note that these tolerances show how much suboptimal we would like to be compared to the optimal solution. Finally, primal and dual feasibility tolerances can be evaluated according to the relative and absolute tolerances,

ε_{abs} and ε_{rel} respectively, i.e.,

$$\begin{aligned}\varepsilon_{\text{primal}} &= \sqrt{n}\,\varepsilon_{\text{abs}} + \varepsilon_{\text{rel}} \max\{\|w_k\|, \|z_k\|\} \\ \varepsilon_{\text{dual}} &= \sqrt{n}\,\varepsilon_{\text{abs}} + \varepsilon_{\text{rel}}\|y_k\| \\ &= \sqrt{n}\,\varepsilon_{\text{abs}} + \varepsilon_{\text{rel}}\|\rho u_k\|,\end{aligned}$$

where ε_{abs} and ε_{rel} are positive quantities and $\sqrt{n}$ because of the fact that $w, z, u \in \mathbb{R}^n$.

3.3.3 Properties of ADMM-TD

In what follows, we show that the ADMM-TD fixed-point solution, w^*, is also a solution to the ℓ_1-regularized fixed-point problem, (3.9), with optimality conditions (3.11).

Lemma 3.3.1. *The fixed-point solution, w^*, as obtained from the ADMM-TD iterations in Algorithm 3.2, satisfies the optimality conditions (3.11), and is thus a solution to problem (3.9), for any $\lambda \geq 0$ and $\rho > 0$.*

Proof. At the fixed-point, ADMM-TD iterations satisfy the following equations

$$w^* = (\tilde{A} + \rho I)^{-1}(\tilde{b} + \rho(z^* - u^*)) \tag{3.25}$$

$$z^* = S_{\lambda/\rho}(w^* + u^*) \tag{3.26}$$

$$u^* = u^* + w^* - z^*. \tag{3.27}$$

Equation (3.27) implies that $w^* = z^*$. From (3.25) it follows that

$$\begin{aligned}&(\tilde{A} + \rho I)w^* = \tilde{b} + \rho w^* - \rho u^* \\ \Leftrightarrow\ &\rho u^* = \tilde{b} - \tilde{A}w^* \\ \Leftrightarrow\ &u^* = \frac{1}{\rho}(\tilde{b} - \tilde{A}w^*).\end{aligned} \tag{3.28}$$

Similarly, equation (3.26) can be rewritten as:

$$w^* \;=\; S_{\lambda/\rho}(w^* + u^*). \tag{3.29}$$

Now, combining (3.28) with (3.29) we have that

$$w^* \;=\; S_{\lambda/\rho}\bigg(w^* + \frac{1}{\rho}(\tilde{b} - \tilde{A}w^*)\bigg). \tag{3.30}$$

From this point, the proof parallels those in [61] and [62]. Using now the definition of the shrinkage operator (3.18), the right-hand side of the equation (3.30) can be written as

$$\operatorname{sgn}\left(w_i^* + \frac{1}{\rho}[\tilde{b} - \tilde{A}w^*]_i\right) \max\left\{\left|w_i^* + \frac{1}{\rho}[\tilde{b} - \tilde{A}w^*]_i\right| - \frac{\lambda}{\rho}, 0\right\}.$$

Since the $\max$ operator is nonnegative, the sign of operator sgn must agree with the sign of w_i^*. Therefore, if $w_i^* > 0$, it follows by definition of sgn operator, (3.19), that

$$\operatorname{sgn}\left(w_i^* + \frac{1}{\rho}[\tilde{b} - \tilde{A}w^*]_i\right) = 1,$$

and also that

$$\max\left\{\left|w_i^* + \frac{1}{\rho}[\tilde{b} - \tilde{A}w^*]_i\right| - \frac{\lambda}{\rho}, 0\right\} = w_i^* + \frac{1}{\rho}[\tilde{b} - \tilde{A}w^*]_i - \frac{\lambda}{\rho}.$$

Replacing the above results to the equation (3.30) we have that

$$\begin{aligned} w_i^* &= w_i^* + \frac{1}{\rho}[\tilde{b} - \tilde{A}w^*]_i - \frac{\lambda}{\rho} \\ \Leftrightarrow\; [\tilde{b} - \tilde{A}w^*]_i &= \lambda, \;\; w_i^* > 0, \end{aligned}$$

as the optimality conditions, (3.11), indicate. With similar operations one can show that $[\tilde{b} - \tilde{A}w^*]_i = -\lambda$, for any $w_i^* < 0$.

Finally, $w_i^* = 0$ implies either that

$$\operatorname{sgn}\left(\frac{1}{\rho}[\tilde{b} - \tilde{A}w^*]_i\right) = 0, \tag{3.31}$$

or

$$\max\left\{\left|\frac{1}{\rho}[\tilde{b} - \tilde{A}w^*]_i\right| - \frac{\lambda}{\rho}, 0\right\} = 0. \tag{3.32}$$

In the first case, (i), we must have that

$$[\tilde{b} - \tilde{A}w^*]_i = 0,$$

which satisfies the optimality conditions. From the second case, ((ii)), it follows that

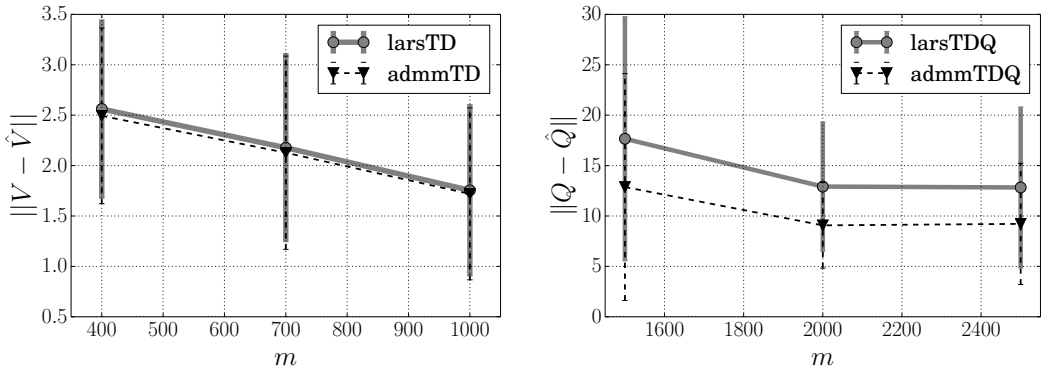
$$\begin{aligned} &\left|\frac{1}{\rho}[\tilde{b} - \tilde{A}w^*]_i\right| - \frac{\lambda}{\rho} \leq 0 \\ \Leftrightarrow\ &-\lambda \leq [\tilde{b} - \tilde{A}w^*]_i \leq \lambda \end{aligned}$$

which concludes the proof. □

The above lemma indicates that the ADMM-TD solves the fixed-point problem (3.9), and that the above proof also holds for $w = z$. A convergence proof of the ADMM-TD to the fixed-point remains outstanding. However, our experimental results in Section 3.3.4 below indicate comparable, if not better, behavior relative to the LARS-TD algorithm.

3.3.4 Experiments

The four-rooms grid problem, as discussed in [4, Section 5], was used to compare the proposed ADMM-TD with LARS-TD. The problem involves a two dimensional grid with total number of states $S = M \times N$, where M and N denote the rows and columns respectively and are chosen as the largest factors of S. The grid is split into four interconnected rooms where only the neighbor rooms are connected to



(a) State-Value Function **(b)** State-Action Value Function

Figure 3.4: (a) Averaged approximation error over 50 trials for V function using 1365 features versus samples m. As shown both methods perform similarly when samples are collected on-policy. (b) Averaged approximation error over 50 trials for Q function using 2728 features versus samples m. ADMM-TD is able to offer better approximations for Q function when samples are collected off-policy since LARS-TD, in order not to violate the optimality conditions, incorporates also an ℓ_2 regularization (elastic net).

each other. The grid maps a state $s \in S$ to the grid location, (i, j) as follows

$$s \mapsto \left(\lceil \frac{s}{N} \rceil, (s-1) \mod N \right), \; s = 1, \ldots, MN \qquad \text{and} \qquad (i,j) \mapsto (i-1)\, N + j.$$

Goal states are the states $S-1, S-2$. The agent receives a reward of 1 when it visits the goal states and receives -1 elsewhere. The action set available to the agent comprises eight actions —agents with these characteristics are called "king-move" agents due to the king player in the chess game which is able to move towards all possible directions, i.e., $A = (N, S, W, E, NW, NE, SW, SE)$. Each action has a probability of success of 0.85.

We used the four-rooms environment with a total of 25 states in all our experiments, and thus goal states are the states 24 and 23. The value functions are represented with Gaussian Radial Basis Functions (RBFs) concatenated over different two-dimensional grids (as exactly was applied in [31]. Training samples are collected in different episodes of 5 steps each. The step-size parameter is kept fixed for all the experiment, $\rho = 0.1$. To select the most effective value of the regularization

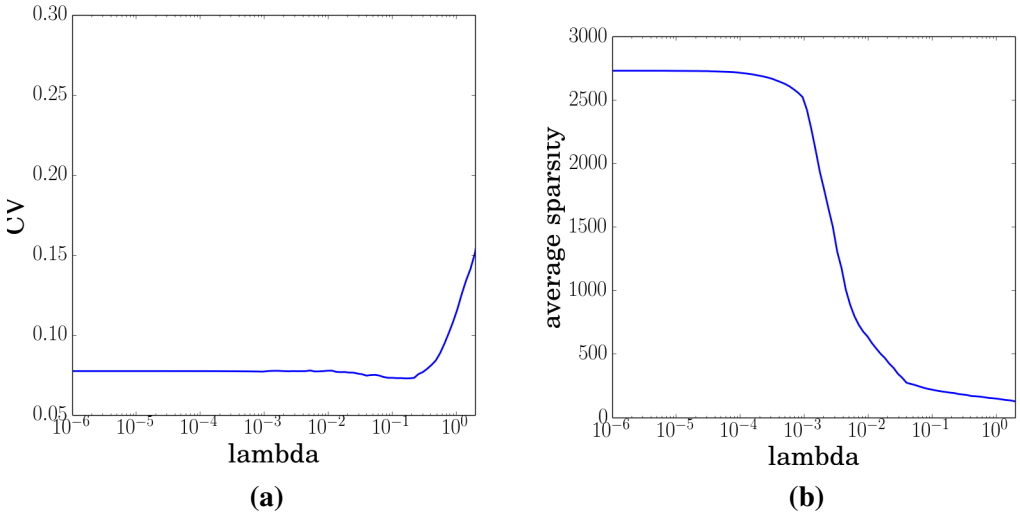


Figure 3.5: (a) 10-fold cross-validation (CV) versus regularization parameter λ; minimum CV achieved for $\lambda = 0.214$. (b) averaged sparsity versus regularization parameter λ; $\lambda = 0.214$ yields approximately 227 nonzero features.

parameter, we use K-fold cross-validation (CV), for a total of 100 values of λ, with K either 5 or 10 according to the magnitude of samples. As a stopping criterion for the ADMM-TD algorithm, we selected the values $\varepsilon_{\text{abs}} = 10^{-2}$ and $\varepsilon_{\text{rel}} = 10^{-4}$ for absolute and relative tolerance, respectively (see Section 3.3.2).

In the first experiment, Figure 3.4a, we approximate the state-value function using 1365 features concatenated over $[2, 4, 8, 16, 32]$ grids, which indicate the scales of RBFs. We compare the performance of the algorithms over different number of samples $(400, 700, 1000)$ collected on-policy, in 50 trials. Further, for obtaining the best value of the regularization parameter λ, we performed K-fold CV with $K = 5$. As expected for the prediction problem, where the samples are collected on-policy, LARS-TD and the *Lasso* formulation of ADMM-TD yielded similar averaged approximation errors and both were always able to find the fixed-point solution.

Subsequently, we test both algorithms in the context of action-value function approximation. In this experiment we supply both methods with 2728 features concatenated over $[2, 4, 8, 16]$ grids. Again, we average our results over 50 trials using different number of samples (see Figure 3.4b). This time, we collected our samples

Table 3.2: Mean simulated reward (20 trials) $\pm$ standard error between ADMM-TD and LARS-TD for $m = (1500, 2000)$ samples and 2728 features —the larger the averaged simulated reward the better the policy is. ADMM-TD yields better policies compared to the elastic net formulation of LARS-TD.

No. of samples	averaged simulated reward	
	ADMM-TD	LARS-TD
1500	73.06 ± 5.001	66.76 ± 6.91
2000	73.77 ± 3.47	67.78 ± 7.14

off-policy (executing a random policy at each episode). Under these circumstances, LARS-TD was not always able to find a solution. In particular, we found LARS-TD to violate the optimality conditions $27/150$ times, while ADMM-TD never failed. For this reason, the LARS-TD results, illustrated in Figure 3.4b, incorporate ℓ_2-regularization (elastic net) as also proposed in [31]. However, this modification in LARS-TD comes with the drawback of increased error and computational cost due to reduced sparsity. For instance, 10-fold cross-validation in the case of $m = 1500$ indicates $\lambda = 0.214$ producing about 200 nonzero features (Figures 3.5a, 3.5b), while for the same example, LARS-TD produces approximately 2000 nonzero features. As a result, ADMM-TD yields decreased averaged approximation error compared to the elastic net formulation of LARS-TD as illustrated in Figure 3.4b. We also performed the same experiment using even smaller number of samples, i.e., $m = 1000$. In this case, ADMM-TD was found to violate the optimality condition only once over 50 trials whilst LARS-TD 13 times over 50 trials which again indicates the improved performance of ADMM-TD compared to LARS-TD in the off-policy scenario.

In the final experiment, the ability of both algorithms to find good policies (policy iteration) is evaluated. We use $(1500, 2000)$ samples and, as before, the samples are collected in the same manner. The results are averaged over 20 trials where each policy iteration trial is run until either convergence to the optimal solution or a maximum of 15 steps is reached. In this setting, we found that LARS-TD violated the optimality conditions repeatedly, and hence was never able to find a good policy. This is understandable because the policy changes drastically at each step due to

the rich available action set. On the other hand, given enough samples, ADMM-TD never failed to reach the optimal policy —again, we only found ADMM-TD not satisfying the optimality conditions for $m \leq 1000$. Therefore, in order to compare both methods in terms of the simulated reward, we, again, apply the an ℓ_2 regularization in LARS-TD algorithm. Nevertheless, LARS-TD now requires storing and inverting a square matrix with almost 2000 entries (as described in the previous paragraph) many times at each policy iteration step. The fact that LARS-TD computes a complete homotopy path within policy iteration makes the algorithm inefficient with respect to time complexity (the same issue is also discussed in [30]. As a result, for practical purposes, we tuned both algorithms to produce no more than 200 nonzero features as indicated by the 10-fold cross-validation (Figure 3.5). In this context, ADMM-TD yielded better policies compared to LARS-TD, as shown in Table 3.2 (note that the largest value of averaged simulated reward denotes the best policy). Furthermore, we noted that approximately 5 policy iteration steps were needed for ADMM-TD to reach an optimal policy, while LARS-TD needed more than 10 steps in average.

3.4 Conclusion and Perspectives

In this chapter we proposed an alternative off-line algorithm for solving the ℓ_1 regularized fixed-point. We validated the efficacy of our algorithm against LARS-TD in a complex experimental environment with many available actions. Our results indicate that, given enough samples, ADMM-TD is able to find the fixed-point solution even within the policy iteration procedure. Furthermore, our initial experiments indicate that, for this particular example we consider in the previous section, our proposed algorithm is more efficient compared to LARS-TD modified to incorporate ℓ_2 regularization.

In particular, we showed that, for the state-value function, both algorithms performed equivalently offering similar approximations. Additionally, both algorithms always converged to the fixed-point solution. This was expected since the samples

are collected on policy. In this case, we form a positive definite matrix $\tilde{A}$ and thus both algorithms do not encounter any issues (specifically, given enough samples, $\tilde{A}$ is positive definite, and thus, it has been proved that LARS-TD converges to the fixed-point solution, while for ADMM-TD we can only guarantee convergence empirically). Furthermore, LARS-TD for state-value function, is shown to be more efficient in terms of time complexity (especially when high level of sparsity is needed) since it can offer good approximations without requiring cross-validation. On the other hand, ADMM-TD in order to provide a good approximation requires tunning the parameter λ, by performing CV, which in turn increases the computational cost (while in case where λ is known ADMM-TD can be extremely fast). Hence, ADMM-TD as a *Lasso* method is not able to return the complete set of solutions, as LARS-TD does being a homotopy method, but only returns a subset over a fixed grid.

The main drawbacks of LARS-TD emerge when collecting the samples off-policy. In fact, for approximating action-value function by following a random policy, we demonstrated through experiments that ADMM-TD is able to offer an improved performance over LARS-TD. In particular, LARS-TD diverged from the fixed-point solution $27/150$ times (for $1500, 2000$ and 2500 samples) and $13/50$ times (for 1000 samples). In contrast to LARS-TD, ADMM-TD was not able to find the ℓ_1 regularized fixed-point only once in 50 trials (for $m = 1000$), while for $m > 1000$ never failed (only for $m < 1000$ ADMM-TD encountered serious problems when trying to approximate Q).

To overcome this issue in LARS-TD, we applied the elastic net formulation (by adding an ℓ_2 penalty norm to the objective function). However, as illustrated in Section 3.3.4, this modification now implies reduced sparsity for the vector w and thus leads to a poorer approximation of the action-value function. Additionally, reduced sparsity increases the computational cost of LARS-TD. To see this, consider

the basic LARS-TD iteration

$$\Delta w_I = \tilde{A}_{I,I}^{-1} \operatorname{sign}(c_I).$$

As discussed before, subscript I denotes the active set which, in essence, indicates the level of sparsity at each iteration. Solving for the elastic net problem yields an almost full dense matrix $\tilde{A}$ which needs to be inverted at each iteration (in our experiments $\tilde{A}$ was almost a 2000×2000 matrix). It is natural then that LARS-TD loses its efficiency since it computes the inverse of a matrix with very large dimensions many times due to the decreased sparsity. More importantly, in the context of policy iteration, things become even more worse for LARS-TD since the same computations need to be performed at each policy iteration step. Therefore, it is not clear whether obtaining the optimal policy via LARS-TD (modified to incorporate the ℓ_2 penalty norm) is feasible or not.

On the other hand, given enough samples (> 1000), our experimental results show that ADMM-TD was able to calculate the ℓ_1-regularized fixed-point. Hence, we conjecture that our algorithm could be shown to converge to the fixed-point under much weaker assumption compared to the existing work. Our anticipation is driven, first, from the fact that the penalty parameter ρ is incorporated in the diagonal of $\tilde{A}$ and, second, due to the behavior of our algorithm in the context of policy iteration. More precisely, recall the ADMM-TD iteration

$$w_{k+1} \leftarrow (\tilde{A} + \rho I)^{-1}(\tilde{b} + \rho(z_k - u_k)).$$

In fact, in contrast to the basic iteration of LARS-TD, we assume that the presence of penalty ρ could improve the stability of the ADMM-TD iterations in the case where $\tilde{A}$ is not positive definite (since the eigenvalues of $\tilde{A}$ definitely play a significant role for the algorithm's convergence).

Furthermore, the advantage of ADMM-TD as a direct method committed to only a single value of λ is that, even in the case where the optimality conditions are vio-

lated, one may discard the coefficient for the specific λ without affecting the other solutions. However, when LARS-TD violates the optimality conditions, the complete homotopy path is affected. As also shown in [31], the homotopy path reaches discontinuities which make the algorithm return multiple fixed-points. We additionally note that, similar to the standard ADMM, the proposed ADMM-TD can be easily extended to allow other forms of regularization (eg., Tikhonov regularization or elastic net). Finally, our main goal in the future is to establish convergence of ADMM-TD and examine the efficacy of our method using real datasets.

Chapter 4

Multilevel Methods for Self-Concordant Functions

The analysis of second-order optimization methods based either on sampling, randomization or sketching has two serious shortcomings compared to conventional second-order methods. The first shortcoming is that the analysis of the iterates is not scale-invariant, and even if it is, restrictive assumptions are required on the problem structure. The second shortfall is that the fast convergence rates of second-order methods have only been established by making assumptions regarding the input data. These theoretical shortcomings have severe practical implications too. In this chapter, building upon the general framework of multigrid methods, we attempt to address these issues. In particular, we propose Yet Another Well-behaved Newton (YAWN) method and establish its super-linear convergence rate using the well-established theory of self-concordant functions. Taking advantage of the theory of multigrid optimization methods and the role of coarse-grained models together with self-concordance as our basic assumption, we come up with a method that is global, scale invariant and independent of unknown constants such as Lipschitz constants and strong convexity parameters. To the best of our knowledge this is the first multigrid optimization method to be analyzed using the theory of self-concordant functions and, in parallel, offers theoretical guarantees that capture

the efficient performance of the general multigrid methods developed for solving Partial Differential Equations (PDEs).

4.1 Introduction

Multigrid methods [33, 34, 35, 36] have been successfully studied for solving Partial Differential Equations (PDEs). In this domain, directly solving for the exact solution is typically expensive. Fortunately, multigrid methods attempt to offer approximate solutions through discretization of a mesh. This means that a hierarchy of discretized problems can be constructed and the multigrid idea is to use the information of the discretized problems to solve the exact problem. We adopt the traditional terminology of the multigrid community and we call the discretized problems as *coarse* problems while the exact problem is called *fine*. For instance, for the mesh refinement example (for details see [33]), the solution of the fine problem is produced as follows: at each level, ranging from the coarsest to fine, the corresponding solution yields the new starting point of the next level (next less coarse level in the hierarchy), and this process moves towards the fine level. There are two advantages of this idea: (a) the problems in coarser levels retain similar structure with the fine problem and thus accurate solutions can be produced, and (b) the computational cost of solving the coarse problems is significantly reduced (and varies according to the dimensionality at each level), which in turn accelerates the convergence of the fine problem. However, note that more accurate solution should be expected in higher dimensions.

Multigrid methods were further extended to solve large-scale optimization problems using the second-order information. In this context, dimensionality reduction is of great value since computing and inverting the exact Hessian of a large-scale problem is typically infeasible. Nash [37] brought the multigrid philosophy into the unconstrained convex optimization for infinite dimensional problems where a global convergence is provided. However, a "smoothing" step is required when switching to different levels, same as in solving PDEs. In [38], Gratton et al. ex-

pand the work of Nash for solving a recursive trust region problem without requiring the smoothing step and, further, they prove convergence for nonconvex problems. Additionally, Wen and Goldfarb [39] proposed a line search method with global sublinear convergence and also provide an extension of their results to nonconvex problems. They also remove the burden of the smoothing step by introducing new conditions which produce effective search directions. In a recent work, [40], which constitutes an extension of the work in [39] the authors come up with an, improved, composite convergence rate for strongly convex functions.

To this end, the performance of the multigrid methods in the domain of optimization has been found very efficient for infinite dimensional problems and in many cases outperform classical methods, see for instance [63, 40, 39]. Throughout this chapter we adopt the name *multilevel* instead of the traditional multigrid (as proposed in [39]), referring to the fine optimization problem (objective function) which can be discretized in different levels.

Other than multilevel methods, first order methods, stochastic, proximal, accelerated or otherwise, are the most popular class of algorithm for the large-scale optimization models that arise in modern machine learning applications. The ease of implementation in distributed architectures and the ability to obtain a reasonably accurate solution quickly are the main reasons for the dominance of first-order methods in machine learning applications. In the last few years, second-order methods based on variants of the Newton method have also been proposed. Second-order methods, such as the Newton method, offer the potential of quadratic convergence rates (the holy grail in optimization algorithms), and scale invariance. Both of these features are highly desirable in optimization algorithms and are not present in first-order methods.

Fast convergence rates do not need additional motivation, and they are particularly important for machine learning applications such as background extraction in video processing, and face recognition (see e.g. [8] for examples). Additionally, scale invariance is crucial because it means that the algorithm is not sensitive to the input

data (see [42] for a thorough discussion of the consequences of scale invariance in machine learning applications). Unfortunately, the conventional Newton method has huge storage and computational demands and does not scale to applications that have both large and dense Hessian matrices.

To improve the convergence rates, and robustness of the optimization algorithms used in machine learning applications many authors have recently proposed modifications of the classical Newton method. We refer the interested reader to the recent survey in [41] for a thorough review. Below we discuss the methods most related to our approach and discuss the theoretical and practical limitations of the current state-of-the-art. Despite the recent interest, and developments in the application of machine learning applications existing approaches suffer from one or both of the following shortfalls that we address in this chapter. These shortcomings have significant implications regarding the practical performance of second order methods in machine learning applications (see [64, 43] for additional discussion).

Shortfall I: Lack of scale-invariant convergence analysis without restrictive assumptions. The Newton algorithm can be analyzed using the elegant theory of self-concordant functions. Convergence proofs using the theory of self-concordant functions enable the derivation of convergence rates that are independent of problem constants such as the Lipschitz constants and strong convexity parameters [65]. In machine learning applications these constants are related to the input data of the problem (e.g., the dictionary in supervised learning applications), so having a theory that is not affected by the scaling of the data is quite important both for practical reasons (e.g., choice of step-sizes), and theory (rates derived using this approach do not depend on unknown constants). The analysis of a Newton algorithm based on sketching was undertaken in [42], but the authors assumed that the square-root of the Hessian matrix is known. In addition, [43] showed that the sub-sampled Newton method produces results that are better in practice. However, [43] rely on the properties of the conjugate-gradient method and therefore their results still depend on unknown problem parameters. Further, the theoretical guarantees of all the mul-

tilevel methods discussed above also require dependence on these parameters.

Shortfall II: Lack of global super-linear or quadratic convergence rates without ad-hoc assumptions regarding the spectral properties of the input data. In addition to scale invariance and a theory that does not rely on unknown constants, the second major feature of second-order methods is their extremely fast convergence rates. The authors in [42] showed a super-linear convergence rate but made assumptions regarding the square root of the Hessian matrix. While [44] do perform a global convergence rate analysis, they do not establish super-linear convergence of their method, and in addition, their analysis depends on unknown constants. As for the multilevel literature in optimization, in [37], although the rate is global, the smoothing step is required which reduces the usefulness of the method in the context of the large-scale optimization. In [38], Gratton et al. also show global convergence rate, nevertheless the total complexity of this method is the same as in gradient descent. Moreover, in [39], the authors only show a global sublinear convergence rate. On the other hand, in [40], the authors offer an improved, composite, convergence rate but their theory is local and depends on unknown parameters.

The main contribution of this chapter is to propose an optimization algorithm based on second-order information that can scale to realistic convex optimization models that arise in machine learning applications. The method is general in the sense that it does not assume that the objective function is a sum of functions, and we make no assumptions regarding the data regime. The proposed approach can easily be applied to the case where the constraints can be incorporated to the objective using a self-concordant barrier function. However, in this chapter we focus on the unconstrained case. Our theoretical analysis is based on the theory of self-concordant functions and we are able to prove the super-linear, and under some additional assumptions, the quadratic convergence of the algorithm without relying on unknown parameters. We emphasize that the main results are independent of the problem data. Thus with the results presented in this chapter, the theory of sub-sampled or sketched Newton methods can be considered to be on-par with the theory of

the classical Newton method. These fundamental results are achieved by drawing parallels between the second-order methods used in machine learning, and the so-called Galerkin model from the multigrid optimization literature. In addition, to the best of our knowledge, this is the first multilevel optimization method that captures the advantages of the multigrid theory (i.e., fast global convergence rates) and in parallel does not suffer from either of the shortfalls listed above.

To be precise our contributions are as follows.

- We propose Yet Another Well-behaved Newton (YAWN) method based on the multilevel framework for unconstrained convex optimization. In particular, we extend the results in [40, 39] to self-concordant functions. Our analysis is global and independent on any unknown problem parameters. To the best of our knowledge, this is the first multilevel method to be analyzed using the theory of self-concordant functions and the first second-order method with convergence analysis that is on par with the standard Newton method.

- As in [40, 38], we specify the coarse model to be the Galerkin model. As long as the first- and second-order coherency is retained, we discuss connections of our method with the classical Newton method and the other variable metric methods.

- Using self-concordance as the basic assumption for the coarse model too, we show that the coarse/Galerkin model achieves a global quadratic convergence rate. The convergence behavior of the coarse model is explicit in the sense that it does not depend on any unknown constants.

- We start our analysis by showing that YAWN can achieve a worse-case sub-linear rate which depends only on the choice of starting point. Later, we propose a super-linear convergence rate for our method with minimal assumptions. This result is global, invariant of the input data and we come up with an explicit region of super-linear convergence.

- We draw parallels between the Galerkin model and the low-rank approximation of the Hessian. In particular, with the help of the naive Nyström method, we show how our algorithm can be seen as a Newton algorithm which lies in a much lower subspace. Further, we show connections with the classical low-rank Singular Value Decomposition (SVD) algorithms and how the search direction can be obtained using these methods.

- Using SVD on the Hessian matrix of the fine model and some extra assumptions, we are able to prove quadratic convergence of our method with reduced computational cost compared to the Newton method. This result is still global and scale invariance is preserved.

- When SVD is applied to the Hessian matrix of the coarse model we show that the general super-linear convergence rate applies. By assuming a specific structure on the Hessian matrix, which is not restrictive for practical problems, we discuss how the super-linear rate can approach the fast rate of Newton method.

- In addition to the fast convergence rates above, we also show that YAWN is efficient enough to be applied to large-scale optimization models that arise in machine learning applications. Numerical experiments based on standard benchmark problems and other state-of-the-art second-order methods suggest that the method compares favorably with the state-of-the-art (it is typically several times faster) and more robust (it is also faster in different data regimes).

The rest of this chapter is organized as follows: In Section 4.2, we discuss the multilevel framework and the background knowledge required for building the multilevel scheme. We show connections with the variable metric methods and describe the so-called Galerkin model. In Section 4.3, we present YAWN method and we show that it enjoys a general super-linear rate with minimal assumptions. In Section 4.4 we aim to improve the convergence results by taking assumptions on the structure

of the problem. Specifically, we show connections of YAWN with low-rank decomposition methods. In Section 4.5, we provide complexity bounds for our methods. If the structure of the problem is as in Section 4.4, we explain why one should expect a convergence rate that approaches the fast rate of Newton method. In Section 4.6, we validate our theoretical guarantees through several numerical results.

4.2 Multilevel Models for Unconstrained Optimization

In this section we collect information and discuss the background of the general multilevel framework. Since our work is based on the theory of self-concordant functions, we start by discussing the important aspects of these functions. We then discuss how the multilevel method can be constructed and show connections with the variable metric methods. Finally, we present the Galerkin model together with some general technical results that emerge due to the presence of self-concordant functions.

4.2.1 Self-Concordant Functions

In this section we recall some main properties and inequalities about the class of self-concordant functions that was also discussed in the second chapter. We follow similar notation as in the books [9, 45] (for a more refined analysis one shall refer to [9]).

A univariate convex function $\phi : \mathbb{R} \to \mathbb{R}$ is called self-concordant if

$$|\phi'''(x)| \leq 2\phi''(x)^{3/2}. \tag{4.1}$$

Examples of such functions include but are not limited to linear, quadratic and logarithmic. Further, consider a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$ and also fix $\mathbf{x} \in \operatorname{dom} f$ and a direction $\mathbf{u} \in \mathbb{R}^n$. Then, $\phi(t) = f(\mathbf{x} + t\mathbf{u})$ is called self-

concordant for all $\mathbf{x}$ and $\mathbf{u}$ if it is self-concordant along every line in its domain. Self-concordance is preserved under composition with any affine function.

Next, given $\mathbf{x} \in \operatorname{dom} f$ and assuming that $\nabla^2 f(\mathbf{x})$ is positive-definite we can define the following norms

$$\|\mathbf{u}\|_{\mathbf{x}} = \langle \nabla^2 f(\mathbf{x})\mathbf{u}, \mathbf{u}\rangle^{1/2} \text{ and } \|\mathbf{v}\|_{\mathbf{x}}^* = \langle [\nabla^2 f(\mathbf{x})]^{-1}\mathbf{v}, \mathbf{v}\rangle^{1/2}, \tag{4.2}$$

where it holds that $|\langle \mathbf{u}, \mathbf{v}\rangle| \leq \|\mathbf{u}\|_{\mathbf{x}}^* \|\mathbf{v}\|_{\mathbf{x}}$. Therefore the Newton decrement can be written as

$$\lambda_f(\mathbf{x}) = \|\nabla f(\mathbf{x})\|_{\mathbf{x}}^* = \|[\nabla^2 f(\mathbf{x})]^{-1/2}\nabla f(\mathbf{x})\|_2. \tag{4.3}$$

In addition, we take into consideration two auxiliary functions, both introduced in [9]. Define the univariate functions ω and ω_* such that

$$\omega(x) = x - \log(1+x) \quad \text{and} \quad \omega_*(x) = -x - \log(1-x), \tag{4.4}$$

with $\operatorname{dom}\omega = \{x \in \mathbb{R} : x \geq 0\}$ and $\operatorname{dom}\omega_* = \{x \in \mathbb{R} : 0 \leq x < 1\}$, respectively. Note that both functions are convex and their range is the set of positive real numbers.

Now, from the definition (4.1), we have that

$$\left| \frac{d}{dt}\left(\phi''(t)^{-1/2}\right) \right| \leq 1,$$

from which, after integration, we obtain the following bounds

$$\frac{\phi''(0)}{(1+t\phi''(0)^{1/2})^2} \leq \phi''(t) \leq \frac{\phi''(0)}{(1-t\phi''(0)^{1/2})^2} \tag{4.5}$$

where the lower bound holds for $t \geq 0$ and the upper bound for $t \in [0, \phi''(0)^{-1/2})$, with $t \in \operatorname{dom}\phi$. Consider now functions on $\mathbb{R}^n$. For $\mathbf{x} \in \operatorname{dom} f$, and for any

$\mathbf{y} \in S(\mathbf{x})$, where $S(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}} < 1\}$, we have that

$$(1 - \|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}})^2 \nabla^2 f(\mathbf{x}) \preceq \nabla^2 f(\mathbf{y}) \preceq \frac{1}{(1 - \|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}})^2} \nabla^2 f(\mathbf{x}). \tag{4.6}$$

Finally, let us state one last pair of inequalities that will be useful in our analysis. For $\mathbf{x}$ and $\mathbf{y}$ from $\operatorname{dom} f$ it holds that

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \omega(\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\|_{\mathbf{y}}^*)$$

and if also $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x})\|_{\mathbf{y}}^* < 1$, then

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \omega_*(\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\|_{\mathbf{y}}^*). \tag{4.7}$$

With the basic definitions and assumptions in place we shall proceed to the core idea of multilevel methods.

4.2.2 Problem Framework and Settings

In this section, based on the idea of multigrid methods, we attempt to solve the following unconstrained optimization problem

$$\min_{\mathbf{x}_h \in \mathbb{R}^N} f_h(\mathbf{x}_h)$$

where $f_h : \mathbb{R}^N \to \mathbb{R}$ is a continuous, differentiable and strictly convex self-concordant function. Further, we suppose that f_h has a closed sublevel set and is bounded below so that a minimizer $\mathbf{x}_h^*$ exists.

Since this work constitutes an extension of the results in [40] to self-concordant functions, we choose to adopt similar notations. We clarify that the subscript h denotes the discretization level and specifically it refers to the *fine* level of the multigrid and hence f_h is considered to be the fine or exact model. In this chapter, unlike the idea of multigrid methods, where a hierarchy of several discretized problems is constructed according to the dimension of each level, we consider only two levels.

The model in the lower level (lower dimension) is called *coarse* model. Thus, the idea is to use information of the coarse model to solve the fine model. As with subscript h, we use H to refer to the coarse level and thus f_H refers to the coarse model. Moreover, the dimensions related to the fine and coarse models are denoted with N and n, respectively, that is, $\operatorname{dom} f_H = \{\mathbf{x}_H \in \mathbb{R}^n\}$ and $\operatorname{dom} f_h = \{\mathbf{x}_h \in \mathbb{R}^N\}$, where $n \leq N$.

To map information from coarse to fine model and vice versa we define $\mathbf{P}$ and $\mathbf{R}$ to be the prolongation and restriction operators, respectively, where matrix $\mathbf{P} \in \mathbb{R}^{N\times n}$ defines a mapping from coarse to fine level and matrix $\mathbf{R} \in \mathbb{R}^{n\times N}$ from fine to coarse. The following assumption on the aforementioned operators is typical for multilevel methods, see for instance [40, 39],

Assumption 4.2.1. *The restriction and prolongation operators* $\mathbf{R}$ *and* $\mathbf{P}$ *are connected via the following relation*

$$\mathbf{P} = \sigma \mathbf{R}^T,$$

where $\sigma > 0$*, and with* $\mathbf{P}$ *to be of full column rank, i.e.,*

$$\operatorname{rank}(\mathbf{P}) = n.$$

For simplification purposes and without loss of generality we assume that $\sigma = 1$.

To construct the coarse model we use the following notation: denote as $\mathbf{x}_{h,k}$ a vector that belongs in the fine level at some iteration k. Assuming that $\mathbf{x}_{h,k}$ is the current solution of the fine model with associated gradient $\nabla f_h(\mathbf{x}_{h,k})$, we move to the coarse level with initial point $\mathbf{x}_{H,0} := \mathbf{R}\mathbf{x}_{h,k}$. Thus, the optimization problem at the coarse level is constructed as

$$\min_{\mathbf{x}_H \in \mathbb{R}^n} \psi_H(\mathbf{x}_H) := \min_{\mathbf{x}_H \in \mathbb{R}^n} \left\{ f_H(\mathbf{x}_H) + \langle \mathbf{u}_H, \mathbf{x}_H - \mathbf{x}_{H,0} \rangle \right\}, \tag{4.8}$$

where $\mathbf{u}_H := \mathbf{R}\nabla f_h(\mathbf{x}_{h,k}) - \nabla f_H(\mathbf{x}_{H,0})$ and $f_H : \mathbb{R}^n \to \mathbb{R}$. Note that the above

objective function is not just $f_H(\mathbf{x}_H)$, but, in order for coarse model to be *first-order coherent*, the quantity $\langle \mathbf{u}_H, \mathbf{x}_H - \mathbf{x}_{H,0} \rangle$ is added, i.e.,

$$\nabla \psi_H(\mathbf{x}_{H,0}) = \mathbf{R} \nabla f_h(\mathbf{x}_{h,k}).$$

We shall mention that this idea is typical when constructing the coarse model and it has been followed by many authors, see for instance [39, 40, 38]. In addition to the first-order coherency condition, we assume that the coarse model is also *second-order coherent*

$$\mathbf{R} \nabla^2 f_h(\mathbf{x}_{h,k}) \mathbf{P} = \nabla^2 \psi_H(\mathbf{x}_{H,0}).$$

In later section we discuss how the so called *Galerkin* model satisfies both first- and second-order coherency conditions.

To this end, the philosophy behind multilevel algorithms is to make use of the solution $\mathbf{x}_H^*$ obtained by the coarse model (4.8) to provide the search direction. Such direction is called *coarse direction*. Later we will show that, in order to ensure the descent nature of our algorithm, we need to alternate between the coarse and the fine search directions. For this reason, if the search direction is computed by the solution of the fine model shall be called *fine direction*.

4.2.3 A Universal Multilevel Method

In this section we provide a description of the general multilevel method. Recall, that we consider a two-level algorithm based on the fine and the coarse models. Using coarse model (4.8) we derive the coarse direction as follows: first we compute

$$\hat{\mathbf{d}}_{H,k} := \mathbf{x}_H^* - \mathbf{x}_{H,0}, \tag{4.9}$$

where $\mathbf{x}_H^*$ is the minimizer of (4.8), and then we apply the prolongation operator to obtain the coarse direction, i.e.,

$$\hat{\mathbf{d}}_{h,k} := \mathbf{P}(\mathbf{x}_H^* - \mathbf{x}_{H,0}). \tag{4.10}$$

Note that the difference in the subscripts in the above definitions is because $\hat{\mathbf{d}}_{H,k} \in \mathbb{R}^n$, and $\hat{\mathbf{d}}_{h,k} \in \mathbb{R}^N$. We would like further to clarify that $\mathbf{d}_{h,k}$, i.e., the "hat" is omitted, will refer to the fine direction. We can now use $\hat{\mathbf{d}}_{h,k}$ to obtain the next iteration

$$\mathbf{x}_{h,k+1} = \mathbf{x}_{h,k} + t_{h,k}\hat{\mathbf{d}}_{h,k}, \tag{4.11}$$

where $t_{h,k} > 0$ is the stepsize.

The authors in [39] have proved that the coarse direction is a descent direction. However, this result does not suffice for $\hat{\mathbf{d}}_{h,k}$ to always lead to reduction in value function. It is easy for one to see that whenever $\nabla f_h(\mathbf{x}_{h,k}) \neq 0$ and $\nabla f_h(\mathbf{x}_{h,k}) \in \text{null}(\mathbf{R})$ (i.e., $\mathbf{R}\nabla f_h(\mathbf{x}_{h,k}) = 0$), we have that $\hat{\mathbf{d}}_{h,k} = 0$, which, clearly, implies no progress for the multilevel scheme (4.11).

One way to overcome this issue, which we do not consider in this work, is to use multiple prolongation and restriction operators. Specifically, it has been shown in [40] that, if $\mathbf{R}_i$, $i = 0, 1, \ldots, m$, operators are selected, then at least one will yield $\mathbf{R}_j\nabla f_h(\mathbf{x}_{h,k}) \neq 0$ and thus $\hat{\mathbf{d}}_{h,k} \neq 0$, so that the coarse direction will be effective.

Another approach to alleviate the issue of the ineffective coarse direction is by replacing it with the fine direction $\mathbf{d}_{h,k}$. Examples of such directions include search directions arising from Newton, quasi-Newton and gradient descent methods. We shall mention that this approach is very common in the multigrid literature, especially for PDE optimization problems, see for example [39, 66, 38]. In PDE problems, alternation between the coarse and the fine direction is necessary for obtaining the optimal solution. Later we will discuss how to choose the prolongation operator such that the fine direction need not be taken. The following conditions, proposed in [39], determine whether or not the fine direction should be employed, i.e., we use $\mathbf{d}_{h,k}$ when

$$\|\mathbf{R}\nabla f_h(\mathbf{x}_{h,k})\|_2 \leq \rho_1 \|\nabla f_h(\mathbf{x}_{h,k})\|_2 \quad \text{or} \quad \|\mathbf{R}\nabla f_h(\mathbf{x}_{h,k})\|_2 \leq \epsilon \tag{4.12}$$

where $\rho_1 \in (0, \min(1, \|\mathbf{R}\|_2))$. Note that the first condition prevents using the

coarse direction when $\mathbf{R}\nabla f_h(\mathbf{x}_{h,k}) = 0$ while $\nabla f_h(\mathbf{x}_{h,k}) \neq 0$, and the second one when $\mathbf{x}_{H,0}$ is sufficiently close to the solution $\mathbf{x}_H^*$ according to some tolerance $\epsilon \in (0, 1)$. In our analysis, we follow the same idea but, as we will discuss later, for analysis purposes, we use the definitions of the Newton and approximate decrements instead of the standard Euclidean norm.

4.2.4 Coarse Model and Variable Metric Methods

In this section we discuss the relation between the multilevel and the classical variable metric methods, see also [40]. Recall that we can derive the descent direction of a standard variable metric method by explicitly solving

$$\begin{aligned} \mathbf{d}_{h,k} &= \arg\min_{\mathbf{d}\in\mathbb{R}^N} \left\{ \frac{1}{2}\|\mathbf{Q}^{1/2}\mathbf{d}\|_2^2 + \langle \nabla f_h(\mathbf{x}_{h,k}), \mathbf{d}\rangle \right\} \\ &= -\mathbf{Q}^{-1}\nabla f_h(\mathbf{x}_{h,k}), \end{aligned} \tag{4.13}$$

where $\mathbf{Q} \in \mathbb{R}^{N\times N}$ is a positive definite matrix. For example, if $\mathbf{Q} = \nabla^2 f_h(\mathbf{x}_{h,k})$ is selected, we obtain the Newton method. If $\mathbf{Q}$ is chosen to be the identity matrix, then we obtain the steepest descent method.

Consider now that f_H is chosen as

$$f_H(\mathbf{x}_H) = \frac{1}{2}\left\|\mathbf{Q}_H^{1/2}(\mathbf{x}_H - \mathbf{x}_{H,0})\right\|_2^2$$

where $\mathbf{x}_{H,0} = \mathbf{R}\mathbf{x}_{h,k}$, and $\mathbf{Q}_H \in \mathbb{R}^{n\times n}$ is a positive definite matrix. Replacing this definition into the coarse model (4.8) we take

$$\min_{\mathbf{x}_H\in\mathbb{R}^n} \psi_H(\mathbf{x}_H) = \min_{\mathbf{x}_H\in\mathbb{R}^n} \left\{ \frac{1}{2}\left\|\mathbf{Q}_H^{1/2}(\mathbf{x}_H - \mathbf{x}_{H,0})\right\|_2^2 + \langle \mathbf{R}\nabla f_h(\mathbf{x}_{h,k}), \mathbf{x}_H - \mathbf{x}_{H,0}\rangle \right\}. \tag{4.14}$$

From the definition (4.9), $\mathbf{d}_H = \mathbf{x}_H - \mathbf{x}_{H,0}$, and thus

$$\begin{aligned} \hat{\mathbf{d}}_{H,k} &= \arg\min_{\mathbf{d}_H\in\mathbb{R}^n} \left\{ \frac{1}{2}\|\mathbf{Q}_H^{1/2}\mathbf{d}_H\|_2^2 + \langle \nabla f_h(\mathbf{x}_{h,k}), \mathbf{d}_H\rangle \right\} \\ &= -\mathbf{Q}_H^{-1}\mathbf{R}\nabla f_h(\mathbf{x}_{h,k}). \end{aligned} \tag{4.15}$$

Further, by construction of the coarse direction and its definition in (4.10) we conclude that

$$\hat{\mathbf{d}}_{h,k} = -\mathbf{P}\mathbf{Q}_H^{-1}\mathbf{R}\nabla f_h(\mathbf{x}_{h,k}). \tag{4.16}$$

Therefore, note that, using the above definition of $f_H(\mathbf{x}_H)$, the descent direction in equation (4.16) is almost identical with the one in (4.13). Specifically, one can see that if we naively set $n = N$ and $\mathbf{R} = \mathbf{I}_{N\times N}$, then, we obtain exactly equation (4.13).

4.2.5 The Galerkin Model

In this section we study the properties of the Galerkin model, which will be later used to provide improved convergence results. It is worth mentioning that the Galerkin model was introduced by Gratton et al. [39] in multilevel community for solving a trust-region optimization problem and was found to produce competitive numerical results.

The Galerkin model can be considered as a special case of the coarse model (4.14) under a specific choice of the matrix $\mathbf{Q}_H$. In particular, we define $\mathbf{Q}_H$ to be

$$\mathbf{Q}_H(\mathbf{x}_{h,k}) := \mathbf{R}\nabla^2 f_h(\mathbf{x}_{h,k})\mathbf{P}. \tag{4.17}$$

Before we present the Galerkin model let us show the positive-definiteness of matrix $\mathbf{Q}_H(\mathbf{x}_{h,k})$.

Proposition 4.2.2. *Let $f_h : \mathbb{R}^N \to \mathbb{R}$ be a strictly convex self-concordant function. Then, the matrix $\mathbf{Q}_H(\mathbf{x}_h)$ is positive definite.*

Proof. This is a direct result of linear algebra using Assumptions 4.2.1 and $\nabla^2 f_h(\mathbf{x}_h) \succ 0$. □

Now, using the definition (4.17) into the coarse model (4.14) we obtain the Galerkin

model, i.e.,

$$\min_{\mathbf{x}_H \in \mathbb{R}^n} \psi_H(\mathbf{x}_H) = \min_{\mathbf{x}_H \in \mathbb{R}^n} \left\{ \frac{1}{2} \left\| [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{1/2} (\mathbf{x}_H - \mathbf{x}_{H,0}) \right\|_2^2 + \right.$$
$$\left. + \langle \mathbf{R}\nabla f_h(\mathbf{x}_{h,k}), \mathbf{x}_H - \mathbf{x}_{H,0} \rangle \right\}.$$

Since, by Proposition 4.2.2, $[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1}$ is well-defined, using similar arguments, as in (4.15) and (4.16), we can derive $\hat{\mathbf{d}}_{H,k}$ and $\hat{\mathbf{d}}_{h,k}$. Thus,

$$\hat{\mathbf{d}}_{H,k} = -[\mathbf{R}\nabla^2 f_h(\mathbf{x}_{h,k})\mathbf{P}]^{-1}\mathbf{R}\nabla f_h(\mathbf{x}_{h,k}) = -[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1}\mathbf{R}\nabla f_h(\mathbf{x}_{h,k}), \quad (4.18)$$

and then we prolongate the direction $\hat{\mathbf{d}}_{H,k}$ to obtain the coarse direction, that is

$$\hat{\mathbf{d}}_{h,k} = -\mathbf{P}\hat{\mathbf{d}}_{H,k} = -\mathbf{P}[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1}\mathbf{R}\nabla f_h(\mathbf{x}_{h,k}). \quad (4.19)$$

In addition, observe that (4.18) is equivalent to solving the following linear system of equations

$$\mathbf{Q}_H(\mathbf{x}_{h,k})\mathbf{d}_H = -\mathbf{R}\nabla f_h(\mathbf{x}_{h,k}),$$

which, by positive-definiteness of $\mathbf{Q}_H(\mathbf{x}_{h,k})$, has a unique solution.

4.2.6 Technical Results for Self-Concordant Functions

We end this section by collecting and proving some general results that will be required throughout the convergence analysis. In the end of this section, we state the alternative conditions which prevent the use of an ineffective coarse direction $\hat{\mathbf{d}}_{h,k}$.

We begin by defining the *YAWN decrement* (or approximate decrement), a quantity analogous to the Newton decrement in (4.3),

$$\hat{\lambda}_{f_h}(\mathbf{x}_{h,k}) := \left[(\mathbf{R}\nabla f_h(\mathbf{x}_{h,k}))^T [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1} \mathbf{R}\nabla f_h(\mathbf{x}_{h,k}) \right]^{1/2}. \quad (4.20)$$

We clarify that from this point and through the rest of this chapter, un-

less specified differently, we denote $\mathbf{d}_{h,k}$ be the Newton direction, $\mathbf{d}_{h,k} = -[\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1}\nabla f_h(\mathbf{x}_{h,k})$, and in addition, for simplification, we omit the subscript f_h from both YAWN and Newton decrements.

Proposition 4.2.3. *For the approximate decrement in (4.20) we have that*

(i) $\hat{\lambda}(\mathbf{x}_{h,k})^2 = -\nabla f_h(\mathbf{x}_{h,k})^T \hat{\mathbf{d}}_{h,k}$,

(ii) $\hat{\lambda}(\mathbf{x}_{h,k})^2 = \hat{\mathbf{d}}_{h,k}^T \nabla^2 f_h(\mathbf{x}_{h,k}) \hat{\mathbf{d}}_{h,k} = \|\hat{\mathbf{d}}_{h,k}\|^2_{\mathbf{x}_{h,k}}$,

(iii) $\hat{\lambda}(\mathbf{x}_{h,k})^2 = \hat{\mathbf{d}}_{h,k}^T \nabla^2 f_h(\mathbf{x}_{h,k}) \mathbf{d}_{h,k}$,

where $\hat{\mathbf{d}}_{h,k}$ *is defined in (4.19),* $\mathbf{d}_{h,k}$ *is the Newton direction and* $\|\cdot\|_{\mathbf{x}_{h,k}}$ *in (4.2).*

Proof. The results can be immediately showed by direct replacement of the definitions of $\hat{\mathbf{d}}_{h,k}$ and $\mathbf{d}_{h,k}$ respectively. □

Next, using the update rule in (4.11) we derive some useful bounds for Hessian $\nabla^2 f_h(\mathbf{x}_{h,k})$.

Proposition 4.2.4. *Let* $f_h : \mathbb{R}^N \to \mathbb{R}$ *be a strictly convex self-concordant function. By scheme (4.11) we have that*

(i) $\nabla^2 f_h(\mathbf{x}_{h,k+1}) \preceq \frac{1}{(1-t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k}))^2} \nabla^2 f_h(\mathbf{x}_{h,k})$,

(ii) $[\nabla^2 f_h(\mathbf{x}_{h,k+1})]^{-1/2} \preceq \frac{1}{1-t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k})} [\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1/2}$,

where $\hat{\lambda}(\mathbf{x}_{h,k}) < 1/t_{h,k}$.

Proof. Consider the case *(i)*. From the upper bound in (4.6), arising from self-concordant functions, we have that

$$\begin{aligned}\nabla^2 f_h(\mathbf{x}_{h,k+1}) &\preceq \frac{1}{(1 - t_{h,k}\|\hat{\mathbf{d}}_{h,k}\|_{\mathbf{x}_{h,k}})^2} \nabla^2 f_h(\mathbf{x}_{h,k}) \\ &= \frac{1}{(1 - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k}))^2} \nabla^2 f_h(\mathbf{x}_{h,k}),\end{aligned}$$

which holds for $\hat{\lambda}(\mathbf{x}_{h,k}) < 1/t_{h,k}$, as claimed. As for the case *(ii)*, we make use of the lower bound in (4.6), and thus, for $\hat{\lambda}(\mathbf{x}_{h,k}) < 1/t_{h,k}$,

$$\nabla^2 f_h(\mathbf{x}_{h,k+1}) \succeq (1 - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k}))^2 \nabla^2 f_h(\mathbf{x}_{h,k}).$$

Since, further, f_h is strictly convex we take

$$[\nabla^2 f_h(\mathbf{x}_{h,k+1})]^{-1/2} \preceq \frac{1}{1 - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k})} [\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1/2},$$

which concludes the proof.

□

In addition, we can obtain analogous bounds for the matrix $\mathbf{Q}_H(\mathbf{x}_{h,k})$.

Proposition 4.2.5. *Let $f_h : \mathbb{R}^N \to \mathbb{R}$ be a strictly convex self-concordant function. By scheme (4.11) we have that*

(i) $\mathbf{Q}_H(\mathbf{x}_{h,k+1}) \preceq \frac{1}{(1-t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k}))^2} \mathbf{Q}_H(\mathbf{x}_{h,k})$,

(ii) $[\mathbf{Q}_H(\mathbf{x}_{h,k+1})]^{-1/2} \preceq \frac{1}{1-t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k})} [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2}$,

where $\hat{\lambda}(\mathbf{x}_{h,k}) < 1/t_{h,k}$.

Proof. We already know that

$$\nabla^2 f_h(\mathbf{x}_{h,k+1}) \preceq \frac{1}{(1 - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k}))^2} \nabla^2 f_h(\mathbf{x}_{h,k}).$$

By strict convexity and Assumption 4.2.1 we see that

$$\mathbf{Q}_H(\mathbf{x}_{h,k+1}) \preceq \frac{1}{(1 - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k}))^2} \mathbf{Q}_H(\mathbf{x}_{h,k}),$$

which is exactly the bound in case *(i)*. Next, recall that

$$\nabla^2 f_h(\mathbf{x}_{h,k+1}) \succeq (1 - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k}))^2 \nabla^2 f_h(\mathbf{x}_{h,k}),$$

and, again, by strict convexity and Assumption 4.2.1 we have that

$$\mathbf{Q}_H(\mathbf{x}_{h,k+1}) \succeq (1 - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k}))^2 \mathbf{Q}_H(\mathbf{x}_{h,k}).$$

In addition, using Proposition 4.2.2 we can exactly obtain the bound in case *(ii)*. Finally, note that, by relation (4.6), all the above bounds hold for $\hat{\lambda}(\mathbf{x}_{h,k}) < 1/t_{h,k}$, which concludes the proof. □

In our analysis, we will further make use of two general bounds that hold for univariate self-concordant functions. We only mention the following results since they are already proved in [45].

Proposition 4.2.6. *[45] Let $\phi : \mathbb{R} \to \mathbb{R}$ be a strictly convex self-concordant function. Then,*

(i) $\phi(t) \leq \phi(0) + t\phi'(0) - t\phi''(0)^{1/2} - \log(1 - t\phi''(0)^{1/2}), \quad t \leq \phi''(0)^{-1/2},$

(ii) $\phi(t) \geq \phi(0) + t\phi'(0) + t\phi''(0)^{1/2} - \log(1 + t\phi''(0)^{1/2}), \quad t \geq 0.$

Proof. Both inequalities can be proved using relation (4.5), for details see [45]. □

Fine Search Direction

As discussed in previous section, in the multilevel literature, condition (4.12) guarantees the progress of the update rule (4.11) by using the fine direction $\mathbf{d}_{h,k}$ in place of the coarse direction $\hat{\mathbf{d}}_{h,k}$ when the latter appears to be ineffective. In this work, for preventing the use of an ineffective $\hat{\mathbf{d}}_{h,k}$, we adopt the same idea but now instead of the standard Euclidean norm we use the norms defined by the matrices $\mathbf{Q}_H(\mathbf{x}_{h,k})$ and $\nabla^2 f_h(\mathbf{x}_{h,k})$.

Note that the YAWN and Newton decrements can be rewritten as

$$\hat{\lambda}(\mathbf{x}_{h,k}) := \|\mathbf{R}\nabla f_h(\mathbf{x}_{h,k})\|_{[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1}} \quad \text{and} \quad \lambda(\mathbf{x}_{h,k}) := \|\nabla f_h(\mathbf{x}_{h,k})\|_{[\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1}}, \tag{4.21}$$

respectively, where in addition, by positive-definiteness of $\mathbf{Q}_H(\mathbf{x}_{h,k})$ and $\nabla^2 f_h(\mathbf{x}_{h,k})$, we have that both norms are well-defined. Thus, the fine direction $\mathbf{d}_{h,k}$ is taken when

$$\hat{\lambda}(\mathbf{x}_{h,k}) \leq \rho_1 \lambda(\mathbf{x}_{h,k}) \quad \text{or} \quad \hat{\lambda}(\mathbf{x}_{h,k}) \leq \epsilon. \tag{4.22}$$

Let us now derive a bound for ρ_1.

Proposition 4.2.7. *Let $\hat{\lambda}(\mathbf{x}_{h,k})$ and $\lambda(\mathbf{x}_{h,k})$ be as in (4.21). It holds that $\hat{\lambda}(\mathbf{x}_{h,k}) \leq c\lambda(\mathbf{x}_{h,k})$, where*

$$c = \left[\frac{\lambda_{max}(\nabla^2 f_h(\mathbf{x}_{h,k}))}{\lambda_{min}(\nabla^2 f_h(\mathbf{x}_{h,k}))}\right]^{1/2} \| (\mathbf{RP})^{-1/2} \|_2 \|\mathbf{R}\|_2,$$

$\lambda_{max}(\nabla^2 f_h(\mathbf{x}_{h,k}))$ and $\lambda_{min}(\nabla^2 f_h(\mathbf{x}_{h,k}))$ are the largest and smallest eigenvalues of the Hessian $\nabla^2 f_h(\mathbf{x}_{h,k})$, respectively.

Proof. Note that for the approximate decrement we have that

$$\begin{aligned}
\hat{\lambda}(\mathbf{x}_{h,k}) &= \left\| [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2} \mathbf{R} \nabla f_h(\mathbf{x}_{h,k}) \right\|_2 \\
&\leq \left\| [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2} \mathbf{R} [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \right\|_2 \left\| [\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1/2} \nabla f_h(\mathbf{x}_{h,k}) \right\|_2 \\
&= \left\| [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2} \mathbf{R} [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \right\|_2 \lambda(\mathbf{x}_{h,k}).
\end{aligned} \tag{4.23}$$

Next, for the Hessian $\nabla^2 f_h(\mathbf{x}_{h,k})$ it holds that

$$\lambda_{min}(\nabla^2 f_h(\mathbf{x}_{h,k}))\mathbf{I}_{N\times N} \preceq \nabla^2 f_h(\mathbf{x}_{h,k}) \preceq \lambda_{max}(\nabla^2 f_h(\mathbf{x}_{h,k}))\mathbf{I}_{N\times N} \tag{4.24}$$

where $\mathbf{I}_{N\times N}$ is the $N \times N$ identity matrix, and thus

$$[\lambda_{min}(\nabla^2 f_h(\mathbf{x}_{h,k}))]^{1/2}\mathbf{I}_{N\times N} \preceq [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \preceq [\lambda_{max}(\nabla^2 f_h(\mathbf{x}_{h,k}))]^{1/2}\mathbf{I}_{N\times N}.$$

From relation (4.24) we can further obtain

$$[\lambda_{min}(\nabla^2 f_h(\mathbf{x}_{h,k}))]^{1/2} (\mathbf{RP})^{1/2} \preceq [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{1/2} \preceq [\lambda_{max}(\nabla^2 f_h(\mathbf{x}_{h,k}))]^{1/2} (\mathbf{RP})^{1/2}.$$

The above bound yields

$$[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2} \preceq \left(\frac{1}{\lambda_{min}(\nabla^2 f_h(\mathbf{x}_{h,k}))}\right)^{1/2} (\mathbf{RP})^{-1/2},$$

and also note that, by Assumption 4.2.1, $(\mathbf{RP})^{-1/2}$ is well-defined. Finally, putting all the above together, inequality (4.23) becomes

$$\hat{\lambda}(\mathbf{x}_{h,k}) \leq c\lambda(\mathbf{x}_{h,k}),$$

as claimed. □

Therefore, we employ the fine search direction when $\hat{\lambda}(\mathbf{x}_{h,k}) \leq \rho_1 \lambda(\mathbf{x}_{h,k})$, where $\rho_1 \in (0, \min(1, c))$. We can further simplify the latter bound. In particular, we will show that $c \geq 1$ which implies $\rho_1 \in (0, 1)$.

Consider the Singular Value Decomposition (SVD) of $\mathbf{R}$, that is $\mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U} \in \mathbb{R}^{N\times N}$, $\mathbf{V} \in \mathbb{R}^{n\times n}$ are unitary matrices and $\mathbf{\Sigma} = [\mathbf{\Sigma}_n \ \ \mathbf{0}] \in \mathbb{R}^{N\times n}$ is the singular matrix such that $\sigma_1 \geq \cdots \geq \sigma_n > 0 = \sigma_{n+1} = \cdots = \sigma_N$. Then we have

$$\begin{aligned}
\left\|(\mathbf{RP})^{-1/2}\right\|_2 &= \left\|\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\right)^{-1/2}\right\|_2 \\
&= \left\|\mathbf{U}\left(\mathbf{\Sigma}\mathbf{\Sigma}^T\right)^{-1/2}\mathbf{U}^T\right\|_2 \\
&= \left\|\left(\mathbf{\Sigma}\mathbf{\Sigma}^T\right)^{-1/2}\right\|_2 \\
&= \left\|(\mathbf{\Sigma}_n)^{-1}\right\|_2 = \frac{1}{\sigma_n}.
\end{aligned}$$

This implies

$$c = \frac{\sigma_1}{\sigma_n}\left[\frac{\lambda_{max}(\nabla^2 f_h(\mathbf{x}_{h,k}))}{\lambda_{min}(\nabla^2 f_h(\mathbf{x}_{h,k}))}\right]^{1/2} \geq 1.$$

Algorithm 4.1 YAWN

Input: $\rho_1 \in (0,1)$, $\alpha \in (0, 0.5)$, $\beta \in (0,1)$, $\epsilon \in (0, 0.68^2)$, $\mathbf{R} \in \mathbb{R}^{n \times N}$
Initialize: $\mathbf{x}_{h,0} \in \mathbb{R}^N$
for $k = 0, 1, \ldots$ **do**
  Compute the direction as

$$\mathbf{d} := \begin{cases} \hat{\mathbf{d}}_{h,k} \text{ from } (4.19) & \text{if } \hat{\lambda}(\mathbf{x}_{h,k}) > \rho_1 \lambda(\mathbf{x}_{h,k}) \text{ and } \hat{\lambda}(\mathbf{x}_{h,k}) > \epsilon \\ \mathbf{d}_{h,k} \text{ from } (4.13) & \text{otherwise,} \end{cases}$$

  if $\lambda(\mathbf{x}_{h,k})^2 \leq \epsilon$ **then**
    quit
  end if
  while $f_h(\mathbf{x}_{h,k} + t_k \mathbf{d}) > f_h(\mathbf{x}_{h,k}) + \alpha t_{h,k} \nabla f_{h,k}^T(\mathbf{x}_{h,k})\mathbf{d}$, $t_{h,k} \leftarrow 1$ **do**
    $t_{h,k} \leftarrow \beta t_{h,k}$
  end while
  Update

$$\mathbf{x}_{h,k+1} := \mathbf{x}_{h,k} + t_{h,k}\mathbf{d}$$

end for
return $\mathbf{x}_{h,k}$

Therefore we conclude that, for $\rho_1 \in (0,1)$, the first inequality in (4.22) prevents the use of the coarse direction $\hat{\mathbf{d}}_{h,k}$ when $\mathbf{R}\nabla f_h(\mathbf{x}_{h,k}) = 0$ while $\nabla f_h(\mathbf{x}_{h,k}) \neq 0$, and the second one when $\mathbf{x}_{H,0}$ is sufficiently close to the solution $\mathbf{x}_H^*$ according to some tolerance ϵ.

4.3 YAWN: Convergence Analysis

In this section, we analyze YAWN for strictly convex self-concordant functions. The pseudo-code of YAWN is stated in Algorithm 4.1 —see Remark 4.4.7 in Section 4.4.3 for practical implementation of the algorithm. We begin by proving that Algorithm 4.1 can achieve a worse-case sublinear convergence rate. Recall that, in each step, the descent direction is computed according to the condition (4.22) which guarantees the reduction of the value function. We emphasize that the choice of the fine search direction need not necessarily be the Newton search direction but any search direction arising from descent methods, such as steepest descent,

gradient descent or quasi-Newton (e.g., L-BFGS search direction was used as the fine direction in [39]). Further, for both fine and coarse direction steps, an inexact backtracking line search is employed to compute step length (see [45]). For the analysis of Newton method with self-concordant functions, the sub-optimality for the current update $\mathbf{x}_{h,k}$ is bounded by the Newton decrement as

$$f_h(\mathbf{x}_{h,k}) - f_h(\mathbf{x}_h^*) \leq \lambda(\mathbf{x}_{h,k})^2, \tag{4.25}$$

which holds for some $\lambda(\mathbf{x}_{h,k}) \leq 0.68$, and thus, $\lambda(\mathbf{x}_{h,k})$ can be used as exit condition. Later, in Lemma 4.5.1 we show that the same bound can be used as stopping criterion for Algorithm 4.1. We proceed by showing quadratic convergence rate of the coarse model and the general super-linear rate of YAWN method. We would like to emphasize that our results below do not depend on any unknown problem parameters and more importantly they are global, as opposed to the classical analysis for strongly convex functions.

4.3.1 Sublinear Convergence Rate

In this section we show that Algorithm 4.1 can, at least, achieve a sub-linear convergence rate. We show, similar to the classical Newton method, that the convergence of YAWN is split into two phases. The only difference in the sketch of the proof is that $\hat{\lambda}(\mathbf{x}_{h,k})$ is used in place of the Newton decrement. We take advantage of the self-concordance assumption to show that the results depend only on the known constants α and β of the line search condition. Specifically, we will prove that there exist some positive η and γ, with $\eta \leq 0.6$, such that

- if $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$, then

$$f_h(\mathbf{x}_{h,k+1}) - f_h(\mathbf{x}_{h,k}) \leq -\gamma.$$

- if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, then
$$f_h(\mathbf{x}_{h,k}) - f_h(\mathbf{x}_h^*) \leq \frac{1}{k}.$$

We begin by proving reduction of the value function for using the coarse step whilst the line search condition is satisfied. Define $\phi(t_{h,k}) = f_h(\mathbf{x}_{h,k} + t_{h,k}\hat{\mathbf{d}}_{h,k})$, where $\phi : \mathbb{R} \to \mathbb{R}$ preserves the self-concordant properties as an affine transformation of variables. With simple computations we can derive the following equalities

$$\phi'(0) = -\hat{\lambda}(\mathbf{x}_k)^2, \quad \phi''(0) = \hat{\lambda}(\mathbf{x}_k)$$

The idea of the proofs of the following lemmas is parallel with the one in [45].

Lemma 4.3.1. *Let $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$ for some $\eta > 0$. Then, there exists $\gamma > 0$ such that the coarse direction $\hat{\mathbf{d}}_{h,k}$ will yield reduction in value function*

$$f_h(\mathbf{x}_{h,k} + t_{h,k}\hat{\mathbf{d}}_{h,k}) - f_h(\mathbf{x}_{h,k}) \leq -\gamma,$$

for any $k > 0$.

Proof. By Proposition 4.2.6*(i)* we have that

$$\begin{aligned}\phi(t_{h,k}) &\leq \phi(0) + t_{h,k}\phi'(0) - t_{h,k}\phi''(0)^{1/2} - \log\left(1 - t_{h,k}\phi''(0)^{1/2}\right) \\ &= \phi(0) - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k})^2 - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k}) - \log\left(1 - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k})\right) = h(t_{h,k}),\end{aligned}$$

which is valid for $t_{h,k} < \frac{1}{\hat{\lambda}(\mathbf{x}_{h,k})}$. Note that $h(t_{h,k})$ is minimized at $t_h^* = \frac{1}{1+\hat{\lambda}(\mathbf{x}_{h,k})}$ and thus

$$\begin{aligned}\phi(t_h^*) &\leq \phi(0) - \frac{\hat{\lambda}(\mathbf{x}_{h,k})^2}{1+\hat{\lambda}(\mathbf{x}_{h,k})} - \frac{\hat{\lambda}(\mathbf{x}_{h,k})}{1+\hat{\lambda}(\mathbf{x}_{h,k})} - \log\left(1 - \frac{\hat{\lambda}(\mathbf{x}_{h,k})^2}{1+\hat{\lambda}(\mathbf{x}_{h,k})}\right) \\ &= \phi(0) - \hat{\lambda}(\mathbf{x}_{h,k}) + \log\left(1 + \hat{\lambda}(\mathbf{x}_{h,k})\right).\end{aligned}$$

Using the inequality

$$-x + \log(1+x) \leq -\frac{x^2}{2(1+x)},$$

for any $x > 0$, we obtain the following upper bound for $\phi(t_h^*)$

$$\begin{aligned}\phi(t_h^*) &\leq \phi(0) - \frac{\hat{\lambda}(\mathbf{x}_{h,k})^2}{2(1+\hat{\lambda}(\mathbf{x}_{h,k}))} \\ &\leq \phi(0) - \alpha t_h^* \hat{\lambda}(\mathbf{x}_{h,k})^2 \\ &= \phi(0) + \alpha t_h^* \nabla f_h^T(\mathbf{x}_{h,k}) \hat{\mathbf{d}}_{h,k},\end{aligned}$$

which satisfies the condition of the line search and hence it will always return a step size $t_{h,k} > \beta/(1+\hat{\lambda}(\mathbf{x}_{h,k}))$. Therefore,

$$f_h(\mathbf{x}_{h,k} + t_{h,k}\hat{\mathbf{d}}_{h,k}) - f_h(\mathbf{x}_{h,k}) \leq -\alpha\beta \frac{\hat{\lambda}(\mathbf{x}_{h,k})^2}{1+\hat{\lambda}(\mathbf{x}_{h,k})}.$$

Additionally, since $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$ and using the fact that the function $x \to \frac{x^2}{1+x}$ is monotone increasing for any $x > 0$, we have that

$$f_h(\mathbf{x}_{h,k} + t_{h,k}\hat{\mathbf{d}}_{h,k}) - f_h(\mathbf{x}_{h,k}) \leq -\alpha\beta \frac{\eta^2}{1+\eta}.$$

which concludes the proof by setting $\gamma = \alpha\beta\eta^2/(1+\eta)$. □

We proceed by estimating the functional gap $f_h(\mathbf{x}_{h,k}) - f_h(\mathbf{x}_h^*)$. In particular, we argue that this gap can also be bounded in terms of the decrement $\hat{\lambda}(\mathbf{x}_k)$, as opposed to the classical one in (4.25).

Lemma 4.3.2. *Let $\hat{\lambda}(\mathbf{x}_{h,k}) < 1$. Then,*

$$f_h(\mathbf{x}_{h,k}) - f_h(\mathbf{x}_h^*) \leq \omega_*(\hat{\lambda}(\mathbf{x}_{h,k})),$$

where the mapping ω_* is defined in (4.4).

Proof. Using now Proposition 4.2.6*(ii)* we can obtain the following bound

$$\begin{aligned}\phi(t_{h,k}) &\geq \phi(0) + t_{h,k}\phi'(0) + t_{h,k}\phi''(0)^{1/2} - \log(1 + t_{h,k}\phi'(0)^{1/2}) \\ &= \phi(0) - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k})^2 + t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k}) - \log(1 + t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k})) = g(t_{h,k}),\end{aligned}$$

which is true for any $t_{h,k} \geq 0$. Moreover, the function $g(t_{h,k})$ is minimized at $t_h^* = 1/(1 - \hat{\lambda}(\mathbf{x}_{h,k}))$, and we take

$$\begin{aligned}\inf_{t_{h,k}\geq 0}\{\phi(t_{h,k})\} &\geq \phi(0) - \frac{\hat{\lambda}(\mathbf{x}_{h,k})^2}{1-\hat{\lambda}(\mathbf{x}_{h,k})} + \frac{\hat{\lambda}(\mathbf{x}_{h,k})}{1-\hat{\lambda}(\mathbf{x}_{h,k})} - \log(1 + \frac{\hat{\lambda}(\mathbf{x}_{h,k})}{1-\hat{\lambda}(\mathbf{x}_{h,k})}) \\ &= \phi(0) + \hat{\lambda}(\mathbf{x}_{h,k}) + \log(1 - \hat{\lambda}(\mathbf{x}_{h,k})),\end{aligned}$$

which is valid since, by assumption, $\hat{\lambda}(\mathbf{x}_{h,k}) < 1$. Similar to the analysis in [45], since $\hat{\mathbf{d}}_{h,k}$ is a descent direction

$$f_h(\mathbf{x}_h^*) \geq f_h(\mathbf{x}_{h,k}) - \omega_*(\hat{\lambda}(\mathbf{x}_{h,k})),$$

which concludes the proof. □

In order to complete the proof of our theorem we state the following lemma for nonnegative real sequences, introduced in [67].

Lemma 4.3.3 ([67])**.** *Let $\{\mathcal{B}_n\}_{n\geq 0}$ be a sequence of real nonnegative numbers with $n \in \mathbb{Z}$. Further, suppose that there exist positive constants μ and c such that*

$$\mathcal{B}_n - \mathcal{B}_{n+1} \geq \mu\mathcal{B}_n^2 \quad \textit{and} \quad \mathcal{B}_0 \leq \frac{1}{\mu c}.$$

Then,

$$\mathcal{B}_n \leq \frac{1}{\mu(n+c)}.$$

Proof. See [67], Lemma 3.5. □

Combining all the above let us now derive the worst-case convergence rate of Algorithm 4.1.

Theorem 4.3.4. *Suppose that the sequence $\{\mathbf{x}_{h,k}\}_k$ with $k = 0, 1, 2, \ldots$, is generated by Algorithm 4.1. There exists $\eta \in (0, 0.6)$ such that if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, then*

$$f_h(\mathbf{x}_{h,k}) - f(\mathbf{x}_h^*) \leq \frac{1}{k + 1/\lambda(\mathbf{x}_{h,0})r(\mathbf{x}_{h,0})},$$

where $r(\mathbf{x}_{h,0}) = \|\mathbf{x}_{h,0} - \mathbf{x}^*\|_{\mathbf{x}_{h,0}}$.

Proof. Denote $\mathcal{B}_k = f_h(\mathbf{x}_{h,k}) - f(\mathbf{x}_h^*)$. Recall that from Lemma 4.3.1 we have that

$$f_h(\mathbf{x}_{h,k}) - f_h(\mathbf{x}_{h,k+1}) \geq \hat{\lambda}(\mathbf{x}_{h,k}) - \log\left(1 + \hat{\lambda}(\mathbf{x}_{h,k})\right).$$

The above bound together with the following inequality

$$x - \log(1+x) \geq x^4, \quad x \in [0, 0.6]$$

implies that for $\hat{\lambda}(\mathbf{x}_{h,k}) \leq 0.6$ we obtain

$$f_h(\mathbf{x}_{h,k}) - f(\mathbf{x}_h^*) - (f_h(\mathbf{x}_{h,k+1}) - f(\mathbf{x}_h^*)) \geq \hat{\lambda}(\mathbf{x}_{h,k})^4. \tag{4.26}$$

Next, it holds that, for any $x \in [0, 0.68]$

$$-x - \log(1-x) \geq x^2,$$

and thus, the result in Lemma 4.3.2 in conjunction with the above inequality yields

$$\hat{\lambda}(\mathbf{x}_{h,k})^2 \geq f_h(\mathbf{x}_{h,k}) - f(\mathbf{x}_h^*), \quad \hat{\lambda}(\mathbf{x}_{h,k}) \leq 0.68. \tag{4.27}$$

Combining (4.26) and (4.27) we get

$$\mathcal{B}_k - \mathcal{B}_{k+1} \geq \mathcal{B}_k^2. \tag{4.28}$$

which is valid for $\hat{\lambda}(\mathbf{x}_{h,k}) \leq 0.6$. Recall that $|\langle \mathbf{u}, \mathbf{v} \rangle| \leq \|\mathbf{u}\|_{\mathbf{x}}^* \|\mathbf{v}\|_{\mathbf{x}}$, and hence by convexity we obtain

$$\begin{aligned} \mathcal{B}_0 = f_h(\mathbf{x}_{h,0}) - f(\mathbf{x}_h^*) &\leq \langle \nabla f_h(\mathbf{x}_{h,0}), \mathbf{x}_{h,0} - \mathbf{x}_h^* \rangle \\ &\leq \|\nabla f_h(\mathbf{x}_{h,0})\|_{\mathbf{x}_{h,0}}^* \|\mathbf{x}_{h,0} - \mathbf{x}_h^*\|_{\mathbf{x}_{h,0}} \\ &= \lambda(\mathbf{x}_{h,0}) r(\mathbf{x}_{h,0}). \end{aligned} \tag{4.29}$$

Note that inequalities (4.28) and (4.29) imply that the conditions of Lemma 4.3.3 are fulfilled for the positive constants $\mu = 1$ and $c = 1/\lambda(\mathbf{x}_{h,0})r(\mathbf{x}_{h,0})$, respectively. Therefore, we conclude the proof by replacing both values in the result of Lemma 4.3.3.

□

To this end, Theorem 4.3.4 provides us with the description of the convergence of Algorithm 4.1 as given in the beginning of this section: for $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$ the value function is reduced as in Lemma 4.3.1 and when $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$ Algorithm 4.1 achieves a sub-linear convergence rate. Note that the region of the sub-linear convergence phase is given explicitly. Interestingly, we see the above result depends only on the distance of the initial point $\mathbf{x}_{h,0}$ from the solution $\mathbf{x}_h^*$. This fact equips us with a simplified version of the sub-linear phase which is easier to interpret.

4.3.2 Quadratic Convergence Rate of the Coarse Model

In this section we show that the coarse model can achieve a quadratic convergence rate. We start with the next lemma in which we examine the required condition for Algorithm 4.1 to accept the unit step.

Lemma 4.3.5. *Suppose that the coarse direction, $\hat{\mathbf{d}}_{h,k}$, is employed. If*

$$\hat{\lambda}(\mathbf{x}_{h,k}) \leq \frac{1}{2}(1 - 2\alpha),$$

where $\alpha \in (0, 1/2)$, then Algorithm 4.1 accepts the unit step, $t_{h,k} = 1$.

Proof. Recall the inequality below, from Lemma 4.3.1

$$\phi(t_{h,k}) \leq \phi(0) - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k})^2 - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k}) - \log\left(1 - t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k})\right)$$

valid for $\hat{\lambda}(\mathbf{x}_{h,k}) < 1/t_{h,k}$. Setting $t_{h,k} = 1$ we have that

$$\phi(1) \leq \phi(0) - \hat{\lambda}(\mathbf{x}_{h,k})^2 - \hat{\lambda}(\mathbf{x}_{h,k}) - \log\left(1 - \hat{\lambda}(\mathbf{x}_{h,k})\right)$$

with $\hat{\lambda}(\mathbf{x}_{h,k}) < 1$. Further, as in [45], making use of the inequality

$$-x - \log(1-x) \leq \frac{1}{2}x^2 + x^3, \quad x \in [0, 0.81]$$

we get

$$\phi(1) \leq \phi(0) - \frac{1}{2}\hat{\lambda}(\mathbf{x}_{h,k})^2 + \hat{\lambda}(\mathbf{x}_{h,k})^3 = \phi(0) - \frac{1}{2}\left(1 - 2\hat{\lambda}(\mathbf{x}_{h,k})\right)\hat{\lambda}(\mathbf{x}_{h,k})^2$$

which holds for $\hat{\lambda}(\mathbf{x}_{h,k}) \leq 0.81$. Setting $\alpha \leq \frac{1}{2}(1 - 2\hat{\lambda}(\mathbf{x}_{h,k}))$ we obtain

$$f_h(\mathbf{x}_{h,k} + \hat{\mathbf{d}}_{h,k}) \leq f_h(\mathbf{x}_{h,k}) - \alpha\hat{\lambda}(\mathbf{x}_{h,k})^2, \tag{4.30}$$

which satisfies the backtracking line search condition for $t_{h,k} = 1$ and for $\hat{\lambda}(\mathbf{x}_{h,k}) \leq \min\{\frac{1}{2}(1-2\alpha), 0.81\}$. Since $\alpha \in (0, 1/2)$, inequality (4.30) holds for $\hat{\lambda}(\mathbf{x}_{h,k}) \leq \frac{1}{2}(1-2\alpha)$, which concludes the proof. □

Using the above lemma we shall now prove quadratic convergence of the coarse model.

Theorem 4.3.6. *Suppose that the sequence $\{\mathbf{x}_{h,k}\}_k$ with $k = 0, 1, 2, \ldots$, is generated by Algorithm 4.1 and $t_{h,k} = 1$. Suppose also that the coarse direction, $\hat{\mathbf{d}}_{h,k}$, is employed. Then,*

$$\hat{\lambda}(\mathbf{x}_{h,k+1}) \leq \left(\frac{\hat{\lambda}(\mathbf{x}_{h,k})}{1 - \hat{\lambda}(\mathbf{x}_{h,k})}\right)^2.$$

Proof. By the definition of the approximate decrement we have that

$$\hat{\lambda}(\mathbf{x}_{h,k}) = \left\| [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2} \mathbf{R} \nabla f_h(\mathbf{x}_{h,k}) \right\|_2,$$

where, by Proposition 4.2.2, $[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2}$ is well defined and unique. In addition, from Proposition 4.2.5*(ii)*, and since $t_{h,k} = 1$, we get

$$[\mathbf{Q}_H(\mathbf{x}_{h,k+1})]^{-\frac{1}{2}} \preceq \frac{1}{1 - \hat{\lambda}(\mathbf{x}_{h,k})} [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-\frac{1}{2}},$$

which holds since, by assumption, $\hat{\lambda}(\mathbf{x}_{h,k}) < 1$. Using this relation into the definition of $\hat{\lambda}(\mathbf{x}_{h,k+1})$ above, we have that

$$\begin{aligned}\hat{\lambda}(\mathbf{x}_{h,k+1}) &= \left\|[\mathbf{Q}_H(\mathbf{x}_{h,k+1})]^{-1/2}\mathbf{R}\nabla f_h(\mathbf{x}_{h,k+1})\right\|_2 \\ &\leq \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})}\left\|[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2}\mathbf{R}\nabla f_h(\mathbf{x}_{h,k+1})\right\|_2 .\end{aligned}$$

Further, observe that $\nabla f_h(\mathbf{x}_{h,k+1}) = \int_0^1 \nabla^2 f_h(\mathbf{x}_{h,k} + y\hat{\mathbf{d}}_{h,k})\hat{\mathbf{d}}_{h,k}\, dy + \nabla f_h(\mathbf{x}_{h,k})$ and thus

$$\hat{\lambda}(\mathbf{x}_{h,k+1}) \leq \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})}\left\|\mathbf{A}_1 + \mathbf{A}_2\right\|_2 , \tag{4.31}$$

where we denote $\mathbf{A}_1 = [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2}\mathbf{R}\int_0^1 \nabla^2 f_h(\mathbf{x}_{h,k} + y\hat{\mathbf{d}}_{h,k})\hat{\mathbf{d}}_{h,k}\, dy$ and $\mathbf{A}_2 = [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2}\mathbf{R}\nabla f_h(\mathbf{x}_{h,k})$. By the definitions of the coarse step in (4.19) and (4.18), we have that $\hat{\mathbf{d}}_{h,k} = \mathbf{P}\hat{\mathbf{d}}_{H,k}$, and thus, using simple algebra, $\mathbf{A}_1$ and $\mathbf{A}_2$ become

$$\begin{aligned}\mathbf{A}_1 &= [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2}\int_0^1 \mathbf{Q}_H(\mathbf{x}_{h,k} + y\hat{\mathbf{d}}_{h,k})\hat{\mathbf{d}}_{H,k}\, dy \\ &= \int_0^1 [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2}\mathbf{Q}_H(\mathbf{x}_{h,k} + y\hat{\mathbf{d}}_{h,k})[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1/2}\, dy\, [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{1/2}\hat{\mathbf{d}}_{H,k},\end{aligned}$$

and

$$\begin{aligned}\mathbf{A_2} &= [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{1/2}[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{-1}\mathbf{R}\nabla f_h(\mathbf{x}_{h,k}) \\ &= -[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{1/2}\hat{\mathbf{d}}_{H,k},\end{aligned}$$

respectively. From Proposition 4.2.5*(i)*, we take $\mathbf{Q}_H(\mathbf{x}_{h,k} + y\hat{\mathbf{d}}_{h,k}) \preceq \frac{1}{(1-y\hat{\lambda}(\mathbf{x}_{h,k}))^2}\mathbf{Q}_H(\mathbf{x}_{h,k})$, which is valid since $y\hat{\lambda}(\mathbf{x}_{h,k}) < 1$, and so $\mathbf{A}_1$ can be bounded as follows

$$\mathbf{A}_1 \preceq \int_0^1 \frac{1}{(1-y\hat{\lambda}(\mathbf{x}_{h,k}))^2} dy [\mathbf{Q}_H(\mathbf{x}_{h,k})]^{1/2}\hat{\mathbf{d}}_{H,k}.$$

Putting this all together, inequality (4.31) becomes

$$\begin{aligned}\hat{\lambda}(\mathbf{x}_{h,k+1}) &\leq \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})}\left\|\int_0^1 \frac{1}{(1-y\hat{\lambda}(\mathbf{x}_{h,k}))^2}dy\,[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{1/2}\hat{\mathbf{d}}_{H,k}-\right.\\ &\qquad\left.-[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{1/2}\hat{\mathbf{d}}_{H,k}\right\|_2\\ &= \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})}\left\|\int_0^1\left(\frac{1}{(1-y\hat{\lambda}(\mathbf{x}_{h,k}))^2}\mathbf{I}_{n\times n}-\mathbf{I}_{n\times n}\right)dy\,[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{1/2}\hat{\mathbf{d}}_{H,k}\right\|_2\\ &\leq \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})}\left\|\int_0^1\left(\frac{1}{(1-y\hat{\lambda}(\mathbf{x}_{h,k}))^2}-1\right)dy\right\|_2\left\|[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{1/2}\hat{\mathbf{d}}_{H,k}\right\|_2,\end{aligned}$$

where $\mathbf{I}_{n\times n}$ denotes the $n\times n$ identity matrix. Note that

$$\int_0^1\left(\frac{1}{(1-y\hat{\lambda}(\mathbf{x}_{h,k}))^2}-1\right)dy=\frac{\hat{\lambda}(\mathbf{x}_{h,k}))}{(1-\hat{\lambda}(\mathbf{x}_{h,k}))},$$

and also that

$$\begin{aligned}\left\|[\mathbf{Q}_H(\mathbf{x}_{h,k})]^{1/2}\hat{\mathbf{d}}_{H,k}\right\|_2 &= \left(\left(\mathbf{P}\hat{\mathbf{d}}_{H,k}\right)^T\nabla^2 f_h(\mathbf{x}_{h,k})\mathbf{P}\hat{\mathbf{d}}_{H,k}\right)^{\frac{1}{2}}\\ &= \left(\hat{\mathbf{d}}_{h,k}^T\nabla^2 f_h(\mathbf{x}_{h,k})\hat{\mathbf{d}}_{h,k}\right)^{\frac{1}{2}}\\ &= \hat{\lambda}(\mathbf{x}_{h,k}),\end{aligned}$$

which concludes the proof of the theorem by directly replacing both equalities into the last inequality of $\hat{\lambda}(\mathbf{x}_{h,k+1})$. □

According to Theorem 4.3.6, we can infer the following about the convergence rate of the coarse model: first, note that the root of $\lambda/(1-\lambda)^2=1$ can be found at $\lambda\approx 0.38$. Hence, we come up with an explicit expression about the region of quadratic convergence, that is, when $\hat{\lambda}(\mathbf{x}_{h,k})<0.38$, we can guarantee that $\hat{\lambda}(\mathbf{x}_{h,k+1})<\hat{\lambda}(\mathbf{x}_{h,k})$ and specifically that this process converges quadratically with

$$\hat{\lambda}(\mathbf{x}_{h,k+1})\leq\frac{\delta}{(1-\delta)^2}\hat{\lambda}(\mathbf{x}_{h,k}),$$

for some $\delta\in(0,\lambda)$. However, bear in mind that this result provides us with a

description about the convergence of $\|\nabla f_h(\mathbf{x}_{h,k})\|$ onto the space spanned by the rows of $\mathbf{R}$. As such, in the next section we attempt to examine the convergence of $\|\nabla f_h(\mathbf{x}_{h,k})\|$ on the entire space $\mathbb{R}^N$.

4.3.3 Super-linear Convergence Rate of the Fine Model

In this section, we study the convergence of Algorithm 4.1 on $\mathbb{R}^N$ and we show that it can achieve a super-linear rate. Taking advantage of the self-concordant assumption, our results are independent of unknown problem parameters, such as Lipschitz constants. Moreover, it is important to mention that, unlike the classical analysis of the Newton method where convergence is local (i.e., $\mathbf{x}_0$ sufficiently close to $\mathbf{x}^*$), our results are global. To achieve our purpose, we proceed by analyzing the decrement $\lambda(\mathbf{x}_{h,k})$. Again, we follow the same philosophy with that of the Newton method for self-concordant functions by showing that the convergence of our algorithm is split into two phases according to the magnitude of $\hat{\lambda}(\mathbf{x}_{h,k})$. The following lemma constitutes the core of our theorem.

Lemma 4.3.7. *Suppose that the coarse direction, $\hat{\mathbf{d}}_{h,k}$, is employed and, in addition, that the line search selects $t_{h,k} = 1$. Then,*

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{\varepsilon_{\rho_1} + \hat{\lambda}(\mathbf{x}_{h,k})}{\left(1 - \hat{\lambda}(\mathbf{x}_{h,k})\right)^2}\lambda(\mathbf{x}_{h,k}),$$

where $\varepsilon_{\rho_1} = \sqrt{1-\rho_1^2}$ and $\rho_1 \in (0,1)$.

Proof. By the definition of Newton decrement we have that

$$\lambda(\mathbf{x}_{h,k+1}) = \left\|[\nabla^2 f_h(\mathbf{x}_{h,k+1})]^{-1/2}\nabla f_h(\mathbf{x}_{h,k+1})\right\|_2.$$

Since $t_{h,k} = 1$, from Proposition 4.2.4*(ii)*, it holds that

$$[\nabla^2 f_h(\mathbf{x}_{h,k+1})]^{-1/2} \preceq \frac{1}{1 - \hat{\lambda}(\mathbf{x}_{h,k})}[\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1/2},$$

and in conjunction with the above definition we have the following bound

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})} \left\| [\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1/2} \nabla f_h(\mathbf{x}_{h,k+1}) \right\|_2 . \tag{4.32}$$

Denote $\mathbf{Z} = [\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1/2} \nabla f_h(\mathbf{x}_{h,k+1})$. Using the fact that

$$\nabla f_h(\mathbf{x}_{h,k+1}) = \int_0^1 \nabla^2 f_h(\mathbf{x}_{h,k} + y\hat{\mathbf{d}}_{h,k}) \hat{\mathbf{d}}_{h,k} \, dy + \nabla f_h(\mathbf{x}_{h,k})$$

we see that

$$\begin{aligned} \mathbf{Z} &= [\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1/2} \left(\int_0^1 \nabla^2 f_h(\mathbf{x}_{h,k} + y\hat{\mathbf{d}}_{h,k}) \hat{\mathbf{d}}_{h,k} \, dy + \nabla f_h(\mathbf{x}_{h,k}) \right) \\ &= [\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1/2} \int_0^1 \nabla^2 f_h(\mathbf{x}_{h,k} + y\hat{\mathbf{d}}_{h,k}) \hat{\mathbf{d}}_{h,k} \, dy - [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \mathbf{d}_{h,k}, \end{aligned}$$

where $\mathbf{d}_{h,k}$ is the Newton direction. By Proposition 4.2.4*(i)* and since, by assumption, $y\hat{\lambda}(\mathbf{x}_{h,k}) < 1$, we obtain the following bound

$$\mathbf{Z} \preceq \int_0^1 \frac{1}{\left(1 - y\hat{\lambda}(\mathbf{x}_{h,k})\right)^2} dy \; [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \hat{\mathbf{d}}_{h,k} - [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \mathbf{d}_{h,k}.$$

Next, adding and subtracting the quantity $\int_0^1 \frac{1}{\left(1-y\hat{\lambda}(\mathbf{x}_{h,k})\right)^2} dy \; [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \mathbf{d}_{h,k}$ in the above relation it follows that

$$\mathbf{Z} \preceq \mathbf{Z_1} + \mathbf{Z_2},$$

where we further denote $\mathbf{Z_1} = \int_0^1 \frac{1}{\left(1-y\hat{\lambda}(\mathbf{x}_{h,k})\right)^2} dy \; [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \left(\hat{\mathbf{d}}_{h,k} - \mathbf{d}_{h,k} \right)$ and $\mathbf{Z_2} = \int_0^1 \left(\frac{1}{\left(1-y\hat{\lambda}(\mathbf{x}_{h,k})\right)^2} - 1 \right) dy \; [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \mathbf{d}_{h,k}$. Putting this all together,

inequality (4.32) becomes

$$\begin{aligned}\lambda(\mathbf{x}_{h,k+1}) &\leq \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})} \left\|\mathbf{Z_1}+\mathbf{Z_2}\right\|_2 \\ &\leq \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})} \left(\left\|\mathbf{Z_1}\right\|_2+\left\|\mathbf{Z_2}\right\|_2\right).\end{aligned}$$

We complete the proof by estimating the above norms. For the first one, we have that

$$\begin{aligned}\left\|\mathbf{Z_1}\right\|_2 &= \left\| \int_0^1 \frac{1}{\left(1-y\hat{\lambda}(\mathbf{x}_{h,k})\right)^2} dy\, [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \left(\hat{\mathbf{d}}_{h,k}-\mathbf{d}_{h,k}\right) \right\|_2 \\ &= \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})} \left\| [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \left(\hat{\mathbf{d}}_{h,k}-\mathbf{d}_{h,k}\right) \right\|_2 \\ &= \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})} \left[\left(\hat{\mathbf{d}}_{h,k}-\mathbf{d}_{h,k}\right)^T \nabla^2 f_h(\mathbf{x}_{h,k}) \left(\hat{\mathbf{d}}_{h,k}-\mathbf{d}_{h,k}\right) \right]^{\frac{1}{2}} \\ &= \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})} \left[\|\hat{\mathbf{d}}_{h,k}^T\|^2_{\mathbf{x}_{h,k}} + \|\mathbf{d}_{h,k}^T\|^2_{\mathbf{x}_{h,k}} - 2\hat{\mathbf{d}}_{h,k}^T \nabla^2 f_h(\mathbf{x}_{h,k}) \mathbf{d}_{h,k} \right]^{\frac{1}{2}},\end{aligned}$$

where $\|\cdot\|_{\mathbf{x}_{h,k}}$ is defined in (4.2). Using now the results from Proposition 4.2.3, we obtain

$$\left\|\mathbf{Z_1}\right\|_2 \leq \frac{1}{1-\hat{\lambda}(\mathbf{x}_{h,k})} \left[\lambda(\mathbf{x}_{h,k})^2 - \hat{\lambda}(\mathbf{x}_{h,k})^2\right]^{\frac{1}{2}}.$$

Recall that, by assumption (4.22), the coarse direction is taken when $\hat{\lambda}(\mathbf{x}_{h,k}) > \rho_1 \lambda(\mathbf{x}_{h,k})$ and that $\varepsilon_{\rho_1} = \sqrt{1-\rho_1^2}$. Then,

$$\left\|\mathbf{Z_1}\right\|_2 \leq \frac{\varepsilon_{\rho_1}}{1-\hat{\lambda}(\mathbf{x}_{h,k})} \lambda(\mathbf{x}_{h,k}),$$

with $\varepsilon_{\rho_1} \in (0,1)$ since $\rho_1 \in (0,1)$. Next, the second norm implies that

$$\begin{aligned}\left\|\mathbf{Z}_2\right\|_2 &= \left\| \int_0^1 \left(\frac{1}{\left(1-y\hat{\lambda}(\mathbf{x}_{h,k})\right)^2} - 1 \right) dy\, [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \mathbf{d}_{h,k} \right\|_2 \\ &= \frac{\hat{\lambda}(\mathbf{x}_{h,k})}{1-\hat{\lambda}(\mathbf{x}_{h,k})} \lambda(\mathbf{x}_{h,k}).\end{aligned}$$

Combining all the above we conclude that

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{\varepsilon_{\rho_1} + \hat{\lambda}(\mathbf{x}_{h,k})}{\left(1 - \hat{\lambda}(\mathbf{x}_{h,k})\right)^2}\lambda(\mathbf{x}_{h,k}),$$

as claimed. □

We now use this result to obtain the two phases of the convergence of Algorithm 4.1. Precisely, we show that the region of super-linear convergence is governed by $\eta = \frac{3-\sqrt{9-4(1-\varepsilon_{\rho_1})}}{2}$.

Theorem 4.3.8. *Suppose that the sequence $\{\mathbf{x}_{h,k}\}_k$ with $k = 0, 1, 2, \ldots$, is generated by Algorithm 4.1 and that the coarse direction, $\hat{\mathbf{d}}_{h,k}$, is employed. For any $\rho_1 \in (0, 1)$, there exist constants $\gamma > 0$ and $\eta \in (0, 0.38)$ such that*

(i) if $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$, then

$$f_h(\mathbf{x}_{h,k+1}) - f_h(\mathbf{x}_{h,k}) \leq -\gamma$$

(ii) if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, then Algorithm 4.1 selects the unit step and

$$\hat{\lambda}(\mathbf{x}_{h,k+1}) < \hat{\lambda}(\mathbf{x}_{h,k}) \tag{4.33}$$

$$\lambda(\mathbf{x}_{h,k+1}) < \lambda(\mathbf{x}_{h,k}). \tag{4.34}$$

Proof. Notice that case *(i)* is already proved in Lemma 4.3.1 and in particular it holds with

$$\gamma = \alpha\beta\frac{\eta^2}{1+\eta}.$$

Hence, it remains to prove case *(ii)*. Obviously, from Lemma 4.3.5, we have that for $\eta \in (0, 0.38)$ Algorithm 4.1 selects the unit step. Now, from Theorem 4.3.6

$$\hat{\lambda}(\mathbf{x}_{h,k+1}) \leq \left(\frac{\hat{\lambda}(\mathbf{x}_{h,k})}{1 - \hat{\lambda}(\mathbf{x}_{h,k})}\right)^2.$$

The above bound implies that if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, then we guarantee that $\hat{\lambda}(\mathbf{x}_{h,k+1}) < \hat{\lambda}(\mathbf{x}_{h,k})$; in particular, see the discussion followed Theorem 4.3.6. Thus, inequality (4.33) is proved.

Recall, from Lemma 4.3.7, that

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{\varepsilon_{\rho_1} + \hat{\lambda}(\mathbf{x}_{h,k})}{\left(1 - \hat{\lambda}(\mathbf{x}_{h,k})\right)^2} \lambda(\mathbf{x}_{h,k}).$$

By assumption, $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$ and since the function $x \to \frac{\varepsilon_{\rho_1}+x}{(1-x)^2}$ is monotone increasing we have that

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{\varepsilon_{\rho_1} + \eta}{(1-\eta)^2} \lambda(\mathbf{x}_{h,k}).$$

Note that the root of $(\varepsilon_{\rho_1}+\eta)/(1-\eta)^2 = 1$ is attained at $\eta = \frac{3-\sqrt{9-4(1-\varepsilon_{\rho_1})}}{2}$. Therefore, $\rho_1 \in (0,1)$ implies that $\eta \in (0, 0.38)$ and thus inequality (4.34) is proved, which concludes the proof of the theorem. □

Theorem 4.3.8 shows that Algorithm 4.1 can achieve super-linear convergence rate. In particular, we come up with with the following description of the region of the convergence:

- **First Phase:** By Theorem 4.3.8*(i)*, if $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$, we can derive the maximum number of iterations of this stage, i.e.,

$$M_1 \leq \frac{1}{\gamma}[f_h(\mathbf{x}_{h,0}) - f_h(\mathbf{x}_h^*)].$$

- **Second Phase:** By Theorem 4.3.8*(ii)*, for any $\rho_1 \in (0,1)$ there exists $\eta \in (0, 0.38)$ such that: if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, then Algorithm 4.1 enters to its super-linear phase and it holds that

$$\lambda(\mathbf{x}_{h,k+1}) < \lambda(\mathbf{x}_{h,k}).$$

We shall emphasize that the above phases are affected by the user-defined constant ρ_1. That is, the region of convergence is proportional ρ_1. Specifically, as $\rho_1 \to 0$, we see that η approaches zero yielding a restricted region of the super-linear phase and thus lower convergence. On the other hand, when $\rho_1 \to 1$ we obtain higher convergence. However, bear in mind that larger values of ρ_1 may result in fewer number of coarse direction steps. Hence, we conclude that there is a trade-off between the number of coarse correction steps and the region of the super-linear phase. Since this result is general, i.e., holds with the mildest of assumptions, we cannot say much about the region of super-linear rate and how it is affected by the choice of ρ_1. Typically, this region can take any values in $(0, 0.38)$. In the next section, we consider structured problems and we discuss concepts where the value of ρ_1 does not influence the rate of convergence.

4.4 Low-Rank Approximation of the Galerkin Model

In this section we consider the case where the Hessian matrix accepts a low-rank approximation. We show how the Galerkin model can be developed using classical decomposition methods. Based on this framework, we are able to show that our method enjoys super-linear and quadratic convergence rates with very cheap per-iteration cost. We start by discussing the basic setting of low-rank approximation.

Let $\mathbf{A} \in \mathbb{R}^{N \times N}$ be a positive-definite matrix. We consider matrices $\mathbf{A}$ in which the following holds: there exists a large gap between the p and $p+1$ eigenvalues and, in addition, all eigenvalues below λ_p are sufficiently small positive real numbers, i.e.,

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \gg \lambda_{p+1} \geq \cdots \geq \lambda_N.$$

Then, we say that $\mathbf{A}$ admits a low-rank approximation and the idea is to retain the first p-eigenvalues which, in general, are the most informative. We can obtain

$\mathbf{A}_p \approx \mathbf{A}$, with $\mathrm{rank}(\mathbf{A}_p) = p < N$, by solving the following optimization problem

$$\begin{aligned} &\underset{\mathbf{A}_p \in \mathbb{R}^{N\times N}}{\text{minimize}} \quad \|\mathbf{A} - \mathbf{A}_p\| \\ &\text{subject to } \mathrm{rank}(\mathbf{A}_p) = p. \end{aligned} \tag{4.35}$$

It is known, that this problem can be solved by Singular Value Decomposition (SVD) algorithms, where SVD, by positive-definiteness of $\mathbf{A}$, coincides with the eigenvalue decomposition. Among others, the Truncated SVD (T-SVD) yields

$$\mathbf{A} \approx \mathbf{A}_p = \mathbf{U}_p \mathbf{\Sigma}_p \mathbf{U}_p^T,$$

where $\mathbf{\Sigma}_p \in \mathbb{R}^{p\times p}$ is a diagonal matrix containing the p-largest eigenvalues and $\mathbf{U}_p \in \mathbb{R}^{N\times p}$ the corresponding eigenvectors. In this context, T-SVD can now be viewed as, initially, performing an approximate eigenvalue decomposition and then truncating it, so that the eigenvalue matrix $\mathbf{\Sigma}_N$ retains the first p-eigenvalues and, in addition, the last $(N-p)$-eigenvalues are replaced with zero. Computing $[\mathbf{U}_p, \mathbf{\Sigma}_p]$ using a deterministic solver is of $\mathcal{O}(N^2 p)$ order, nevertheless, one can find in the literature randomized algorithms with cheaper per-iteration cost, such as $\mathcal{O}(N^2 \log(p))$, see for instance Halko et al in [49].

Throughout this section we specify the restriction and prolongation operators, defined in Section 4.2.2, to be as follows

Definition 4.4.1. *Let $S_N = \{1, 2, \ldots, N\}$ and denote $S_n \subset S_N$, with the property that the $n < N$ elements are uniformly selected by the set S_N without replacement. Further, assume that s_i is the i^{th} element of S_n. Then the prolongation operator $\mathbf{P}$ is generated as follows: The i^{th} column of $\mathbf{P}$ is the s_i column of $\mathbf{I}_{N\times N}$ and, further, it holds that $\mathbf{R} = \mathbf{P}^T$.*

Clearly, by the above definition, Assumption 4.2.1 remains true. Further, using the above definition, a more sophisticated solution of the optimization problem in (4.35) can be given via the Nyström method. In particular, the Nyström method builds a

rank-n, $n < N$, approximation of $\mathbf{A}$ as

$$\mathbf{A} \approx \mathbf{A}_p = \mathbf{A}\mathbf{P}(\mathbf{P}^T\mathbf{A}\mathbf{P})^{-1}(\mathbf{A}\mathbf{P})^T \tag{4.36}$$

with $\mathbf{P}$ as in Definition 4.4.1 (for more details on Nyström method see [53, 49]). Interestingly, since the right-hand side of the above relation is a low-rank approximation of $\mathbf{A}$, we can perform classical decomposition methods, such as T-SVD, to obtain $\mathbf{A}_p$, see [49].

4.4.1 SVD on the Hessian of the Fine Model

It is very common in machine learning problems that the first few eigenvalues and eigenvectors concentrate the most important second-order information while the rest are small and hence non-informative. Under this regime, small eigenvalues provide poor approximations. We aim to overcome this issue by performing Truncated SVD. Specifically, we adopt the idea presented in [68] where the $(N - n)$-eigenvalues (below the threshold), instead of being replaced by zero, are treated as sufficiently small and almost equal to each other. Therefore, we make use of the following assumption

Assumption 4.4.2. *The Hessian $\nabla^2 f(\mathbf{x}_{h,k}) \in \mathbb{R}^{N\times N}$ admits an approximate low-rank factorization of size $n < N$ and, in particular, for its eigenvalues it holds that*

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \gg \lambda_{n+1} \approx \lambda_{n+2} \approx \cdots \approx \lambda_N.$$

Using low-rank approximation of the Hessian we can show connection between the coarse direction $\hat{\mathbf{d}}_{h,k}$ and the Newton direction $\mathbf{d}_{h,k}$. Consider the low-rank approximation as presented in Nyström method in (4.36) and let $\mathbf{A}$ be the Hessian matrix. Then

$$\nabla^2 f(\mathbf{x}_{h,k}) \approx \nabla^2 f(\mathbf{x}_{h,k})\mathbf{P}\left(\mathbf{R}\nabla^2 f(\mathbf{x}_{h,k})\mathbf{P}\right)^{-1}\mathbf{R}\nabla^2 f(\mathbf{x}_{h,k}). \tag{4.37}$$

Hence, multiplying right and left with $[\nabla^2 f(\mathbf{x}_{h,k})]^{-1}$, respectively, the above rela-

tion yields

$$[\nabla^2 f(\mathbf{x}_{h,k})]^{-1} \approx \tilde{\mathbf{Q}}_{h,k} = \mathbf{P}(\mathbf{R}\nabla^2 f(\mathbf{x}_{h,k})\mathbf{P})^{-1}\mathbf{R}. \tag{4.38}$$

Note that, by Definition 4.4.1 and Assumption 4.2.1, $\tilde{\mathbf{Q}}_{h,k}$ is well-defined. Next, we observe that

$$\mathbf{d}_{h,k} \approx \hat{\mathbf{d}}_{h,k} = -\tilde{\mathbf{Q}}_{h,k}\nabla f(\mathbf{x}_{h,k}). \tag{4.39}$$

Therefore, the coarse direction can be viewed as an approximation of the Newton direction with $[\nabla^2 f(\mathbf{x}_{h,k})]^{-1} \approx \tilde{\mathbf{Q}}_{h,k}$ and, importantly, $\tilde{\mathbf{Q}}_{h,k}$ can be calculated by the T-SVD. Suppose further that Assumption 4.4.2 holds. We build $\tilde{\mathbf{Q}}_{h,k}$ in the following manner: compute the $(n+1)^{\text{th}}$ T-SVD of $\nabla^2 f(\mathbf{x}_{h,k})$ to obtain $[\mathbf{U}_{n+1}, \mathbf{\Sigma}_{n+1}]$ and form

$$\tilde{\mathbf{Q}}_{h,k} := \lambda_{n+1}^{-1}\mathbf{I}_{N\times N} + \mathbf{U}_n(\mathbf{\Sigma}_n^{-1} - \lambda_{n+1}^{-1}\mathbf{I}_{n\times n})\mathbf{U}_n^T. \tag{4.40}$$

In other words, relation (4.40) replaces all the eigenvalues below λ_n, of the full eigenvalue matrix $\mathbf{\Sigma}_N \in \mathbb{R}^{N\times N}$, with λ_{n+1}, i.e., $\lambda_{n+1} = \cdots = \lambda_N$, and hence we have

$$\mathbf{\Sigma}_N^{-1} = \operatorname{diag}\left(\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_n}, \frac{1}{\lambda_{n+1}}, \ldots, \frac{1}{\lambda_{n+1}}\right).$$

Obviously, by relation (4.39) and the definition of $\tilde{\mathbf{Q}}_{h,k}$ in (4.40), $-\tilde{\mathbf{Q}}_{h,k}\nabla f(\mathbf{x}_{h,k})$ is a descent direction and note that the approximated Hessian matrix is positive-definite. In the next section we provide convergence analysis using the above framework.

4.4.2 Convergence Analysis

In this section we show that YAWN achieves quadratic convergence rate provided Assumptions 4.2.1, 4.4.2 and self-concordance hold. The main idea of the proof continues in the same manner, i.e., convergence is split into two phases according to the magnitude of $\hat{\lambda}(\mathbf{x}_{h,k})$. The following lemma takes the place of Lemma 4.3.7, of the super-linear rate in Section 4.3.3, where, now, the new coarse direction arises from the low-rank approximation of the Hessian. Note that, by Assumption 4.4.2

and construction of $\tilde{\mathbf{Q}}_{h,k}$, the Newton search direction and decrement coincide with their approximations versions, i.e.,

$$\begin{aligned}\hat{\lambda}(\mathbf{x}_{h,k})^2 &= \nabla f_h^T(\mathbf{x}_{h,k})\tilde{\mathbf{Q}}_{h,k}\nabla f_h(\mathbf{x}_{h,k}) \\ &= \nabla f_h^T(\mathbf{x}_{h,k})[\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1}\nabla f_h(\mathbf{x}_{h,k}) = \lambda(\mathbf{x}_{h,k})^2\end{aligned}$$

and

$$\hat{\mathbf{d}}_{h,k} = -\tilde{\mathbf{Q}}_{h,k}\nabla f_h(\mathbf{x}_{h,k}) = -\nabla^2 f_h(\mathbf{x}_{h,k})\nabla f_h(\mathbf{x}_{h,k}) = \mathbf{d}_{h,k}$$

respectively. Therefore, it is only natural for one to expect the quadratic rate of this scheme. However, we prefer to give a more sophisticated and detailed analysis because the following results will later be useful for providing the super-linear rate in the case where the coarse direction and the approximate decrement do not coincide with the Newton ones.

Lemma 4.4.3. *Suppose that the coarse direction, $\hat{\mathbf{d}}_{h,k}$, with $\tilde{\mathbf{Q}}_{h,k}$ as in (4.40), is taken and also that $t_{h,k} = 1$. Further, let Assumption 4.4.2 hold. Then,*

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{\hat{\lambda}(\mathbf{x}_{h,k})}{\left(1 - \hat{\lambda}(\mathbf{x}_{h,k})\right)^2}\lambda(\mathbf{x}_{h,k}).$$

Proof. The proof of the lemma is parallel to Lemma 4.3.7 and remains true since $\hat{\mathbf{d}}_{h,k} = \mathbf{d}_{h,k}$ and $\hat{\lambda}(\mathbf{x}_{h,k}) = \lambda(\mathbf{x}_{h,k})$. For this reason, we only highlight the key parts. Recall that

$$\lambda(\mathbf{x}_{h,k+1}) = \left\|[\nabla^2 f_h(\mathbf{x}_{h,k+1})]^{-1/2}\nabla f_h(\mathbf{x}_{h,k+1})\right\|_2.$$

which by Lemma 4.3.7 yields

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{1}{(1-\hat{\lambda}(\mathbf{x}_{h,k}))^2}\left(\left\|[\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2}\left(\hat{\mathbf{d}}_{h,k} - \mathbf{d}_{h,k}\right)\right\|_2 + \hat{\lambda}(\mathbf{x}_{h,k})\lambda(\mathbf{x}_{h,k})\right).$$

Using the definition of $\hat{\mathbf{d}}_{h,k}$ and $\mathbf{d}_{h,k}$, we obtain

$$\begin{aligned}\hat{\mathbf{d}}_{h,k} - \mathbf{d}_{h,k} &= \left([\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1} - \tilde{\mathbf{Q}}_{h,k}\right) \nabla f_h(\mathbf{x}_{h,k}) \\ &= \left([\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1/2} - \tilde{\mathbf{Q}}_{h,k}[\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2}\right) [\nabla^2 f_h(\mathbf{x}_{h,k})]^{-1/2}\nabla f_h(\mathbf{x}_{h,k})\end{aligned}$$

and thus

$$\begin{aligned}\left\|[\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2}\left(\hat{\mathbf{d}}_{h,k} - \mathbf{d}_{h,k}\right)\right\|_2 \leq \|\mathbf{I}_{N\times N} - \\ -[\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2}\tilde{\mathbf{Q}}_{h,k}[\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2}\Big\|_2 \lambda(\mathbf{x}_{h,k}).\end{aligned}$$

Therefore, we have that

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{c_k + \hat{\lambda}(\mathbf{x}_{h,k})}{\left(1 - \hat{\lambda}(\mathbf{x}_{h,k})\right)^2}\lambda(\mathbf{x}_{h,k})$$

where

$$c_k = \left\|\mathbf{I}_{N\times N} - [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2}\tilde{\mathbf{Q}}_{h,k}[\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2}\right\|_2. \tag{4.41}$$

It remains only to show that $c_k = 0$. Consider the eigenvalue decomposition. We take

$$[\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} = \mathbf{U}_N \mathbf{\Sigma}_N^{1/2}\mathbf{U}_N^T,$$

and

$$\tilde{\mathbf{Q}}_{h,k} = \mathbf{U}_{\tilde{\mathbf{Q}}_{h,k}}\mathbf{\Sigma}_{\tilde{\mathbf{Q}}_{h,k}}\mathbf{U}_{\tilde{\mathbf{Q}}_{h,k}}^T,$$

where, by Assumption 4.4.2 and construction of $\tilde{\mathbf{Q}}_{h,k}$, $\mathbf{U}_N$ and $\mathbf{U}_{\tilde{\mathbf{Q}}_{h,k}}$ coincide. Thus, we can obtain

$$\begin{aligned}&\mathbf{I}_{N\times N} - [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2}\tilde{\mathbf{Q}}_{h,k}[\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} = \\ &\mathbf{U}_N\left(\mathbf{I}_{N\times N} - \mathbf{\Sigma}_N^{1/2}\mathbf{U}_N^T\mathbf{U}_{\tilde{\mathbf{Q}}_{h,k}}\mathbf{\Sigma}_{\tilde{\mathbf{Q}}_{h,k}}\mathbf{U}_{\tilde{\mathbf{Q}}_{h,k}}^T\mathbf{U}_N\mathbf{\Sigma}_N^{1/2}\right)\mathbf{U}_N^T = \\ &\mathbf{U}_N\left(\mathbf{I}_{N\times N} - \mathbf{\Sigma}_N^{1/2}\mathbf{\Sigma}_{\tilde{\mathbf{Q}}_{h,k}}\mathbf{\Sigma}_N^{1/2}\right)\mathbf{U}_N^T.\end{aligned}$$

Denote $\lambda_{i,k}$ be the i^{th}-eigenvalue of the Hessian matrix at iteration k. Therefore, by definition of $\tilde{\mathbf{Q}}_{h,k}$ we obtain

$$\begin{aligned} c_k &= \left\| \mathbf{I}_{N\times N} - \boldsymbol{\Sigma}_N^{1/2} \boldsymbol{\Sigma}_{\tilde{\mathbf{Q}}_{h,k}} \boldsymbol{\Sigma}_N^{1/2} \right\|_2 \\ &= \max\left\{ \left| 1 - \frac{\lambda_{1,k}}{\lambda_{1,k}} \right|, \ldots, \left| 1 - \frac{\lambda_{n+1,k}}{\lambda_{n+1,k}} \right|, \left| 1 - \frac{\lambda_{n+2,k}}{\lambda_{n+1,k}} \right|, \ldots, \left| 1 - \frac{\lambda_{N,k}}{\lambda_{n+1,k}} \right| \right\} \\ &= 1 - \frac{\lambda_{N,k}}{\lambda_{n+1,k}}, \end{aligned}$$

and since, by Assumption 4.4.2, $\lambda_{n+1,k} = \cdots = \lambda_{N,k}$ we take $c_k = 0$ which concludes the proof. □

We emphasize that the above lemma can also hold for any matrix decomposition method, not necessarily for the construction in (4.40), as long as the structure of the Hessian matrix presented in Assumption 4.4.2 is preserved. For example, one could use standard eigenvalue or QR decomposition algorithms to construct the matrix $\tilde{\mathbf{Q}}_{h,k}$. On the other hand, it is easy to see that employing just the T-SVD without further applying the construction in (4.40), will yield $c_k \neq 0$. Using Lemma 4.4.3, we are now in position to present our theorem.

Theorem 4.4.4. *Suppose that the sequence $\{\mathbf{x}_{h,k}\}_k$ with $k = 0, 1, 2, \ldots$, is generated by $\mathbf{x}_{h,k+1} = \mathbf{x}_{h,k} + t_{h,k}\hat{\mathbf{d}}_{h,k}$ and let the conditions in Lemma 4.4.3 hold. There exist $\gamma > 0$ and $\eta \in (0, 0.38)$ such that*

(i) if $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$, then

$$f_h(\mathbf{x}_{h,k+1}) - f_h(\mathbf{x}_{h,k}) \leq -\gamma$$

(ii) if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, then the line search selects the unit step and

$$\lambda(\mathbf{x}_{h,k+1}) < \lambda(\mathbf{x}_{h,k}), \tag{4.42}$$

where, in particular, this process progresses quadratically.

Proof. Note that case *(i)* and the unit step selection in case *(ii)* are already proved in Theorem 4.3.8. As for inequality (4.42), by Lemma 4.4.3 we have that

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{\hat{\lambda}(\mathbf{x}_{h,k})}{\left(1-\hat{\lambda}(\mathbf{x}_{h,k})\right)^2}\lambda(\mathbf{x}_{h,k}).$$

Observe that the root $\hat{\lambda}(\mathbf{x}_{h,k})/(1-\hat{\lambda}(\mathbf{x}_{h,k}))^2 = 1$ is attained at $\hat{\lambda}(\mathbf{x}_{h,k}) = \frac{3-\sqrt{5}}{2} \approx 0.38$. Therefore, we conclude that if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, for some $\eta \in (0, 0.38)$, then $\lambda(\mathbf{x}_{h,k+1}) < \lambda(\mathbf{x}_{h,k})$, and this process converges quadratically. □

Theorem 4.4.4 shows that the YAWN scheme (4.11) with $\hat{\mathbf{d}}_{h,k}$ chosen as in relation (4.39) enjoys quadratic convergence rate. Further, we are provided with the following description and the region of convergence:

First phase: if $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$, for some $\eta \in (0, 0.38)$, we obtain reduction of

$$f_h(\mathbf{x}_{h,k+1}) - f_h(\mathbf{x}_{h,k}) \leq -\gamma,$$

where $\gamma = \alpha\beta\eta^2/(1+\eta)$ and hence, for this stage, the total number of iterations is bounded by

$$\frac{1}{\gamma}\left(f_h(\mathbf{x}_{h,0}) - f_h(\mathbf{x}_h^*)\right).$$

Quadratic phase: if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, for some $\eta \in (0, 0.38)$, then $t_{h,k} = 1$ and the process converges quadratically as

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{\eta}{(1-\eta)^2}\lambda(\mathbf{x}_{h,k}) < \lambda(\mathbf{x}_{h,k}).$$

Note that, as expected, this rate is exactly the rate in [9] but we emphasize that it is accompanied with cheaper per iteration cost, i.e., Newton method requires $\mathcal{O}(N^3)$ iterations for computing the inverse while our method $\mathcal{O}(N^2 n)$ to perform the standard T-SVD (or even $\mathcal{O}(N^2 \log(n))$ when performing randomized decomposition methods), where for cases with n small, the computational cost is significantly reduced (approximately $\mathcal{O}(N^2)$). However, for showing the desired result we assume

equality between the last $N - n$ eigenvalues. This assumptions may be restrictive for many practical problems, nevertheless, identifies an instance where the above scheme converges quadratically with cheap per-iteration cost. If equality between the last eigenvalues does not hold, the process in turn achieves super-linear rate with region depending on

$$\eta = \frac{3 - \sqrt{9 - 4\frac{\lambda_{k,N}}{\lambda_{k,n+1}}}}{2},$$

where $\lambda_{i,k}$ denotes the i^{th}-eigenvalue of the Hessian at iteration k. We summarize this result in the following theorem.

Theorem 4.4.5. *Suppose that the conditions of Theorem 4.4.4 remain true and, further, Assumption 4.4.2 holds with* $\lambda_{n+1} \geq \cdots \geq \lambda_N$ *. There exist* $\gamma > 0$ *and* $\eta \in (0, 0.38)$ *such that*

(i) if $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$*, then*

$$f_h(\mathbf{x}_{h,k+1}) - f_h(\mathbf{x}_{h,k}) \leq -\gamma$$

(ii) if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$*, then the line search selects the unit step and*

$$\lambda(\mathbf{x}_{h,k+1}) < \lambda(\mathbf{x}_{h,k}).$$

Proof. Note that the approximation error for using the inexact search direction (4.39) depends on the $(n + 1)$-eigenvalue of the Hessian matrix (see for instance theorem 2.4.2). Since, by assumption, we consider structures where there exists a large gap between the n^{th} and the $(n + 1)^{\text{th}}$ eigenvalues, we can expect that, at each iteration, λ_{n+1} is small so that the above approximation is good. Therefore, lemmas 4.3.1 and 4.3.5 approximately hold for the current setting. This proves the first case of the theorem (i.e., when $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$) and shows that, if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, the unit step

is accepted. Similarly, from Lemma 4.4.3 we obtain

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{c_k + \hat{\lambda}(\mathbf{x}_{h,k})}{\left(1 - \hat{\lambda}(\mathbf{x}_{h,k})\right)^2}\lambda(\mathbf{x}_{h,k})$$

where, as before,

$$c_k = \left\| \mathbf{I}_{N\times N} - [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2}\tilde{\mathbf{Q}}_{h,k}[\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \right\|_2.$$

We immediately see that $c_k = 1 - \lambda_{N,k}/\lambda_{n+1,k}$ and so

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{1 - \frac{\lambda_{N,k}}{\lambda_{n+1,k}} + \hat{\lambda}(\mathbf{x}_{h,k})}{\left(1 - \hat{\lambda}(\mathbf{x}_{h,k})\right)^2}\lambda(\mathbf{x}_{h,k})$$

By assumption $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$,

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{1 - \frac{\lambda_{N,k}}{\lambda_{n+1,k}} + \eta}{(1-\eta)^2}\lambda(\mathbf{x}_{h,k})$$

with the root of $(1 - \frac{\lambda_{N,k}}{\lambda_{n+1,k}} + \eta)/(1-\eta)^2 = 1$ to be attained at

$$\eta = \frac{3 - \sqrt{9 - 4\frac{\lambda_{k,N}}{\lambda_{k,n+1}}}}{2}.$$

Since $\lambda_{k,N}/\lambda_{k,n+1} \in (0,1)$, we conclude that if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, then $\lambda(\mathbf{x}_{h,k+1}) < \lambda(\mathbf{x}_{h,k})$, where $\eta \in (0, 0.38)$. □

To this end, Theorem 4.4.5 shows that when there exits a large gap between the n^{th} and $(n+1)^{\text{th}}$ eigenvalues of the Hessian matrix, the YAWN method with search direction as in (4.39) achieves a super-linear convergence rate. In particular, the rate is governed by the ratio of the N^{th} and $(n+1)^{\text{th}}$ eigenvalues. When this ratio approaches 1 the method approaches the fast convergence rate of the Newton method with cheaper computational cost.

However, although Theorems 4.4.4 and 4.4.5 indicate very fast convergence rates,

computing the T-SVD on the exact Hessian matrix is expensive, especially when N is too large. For instance, consider a dataset matrix $\mathbf{A} \in \mathbb{R}^{m \times N}$. The above scheme requires $\mathcal{O}(mN^2)$ computations to form the Hessian and $\mathcal{O}(nN^2)$ to perform the T-SVD which is quite restrictive for large-scale optimization problems. In the next section we show how to address this issue.

4.4.3 SVD on the Coarse Grained Model

The bottleneck of the previous procedure arises from the fact that computations are performed over the full Hessian matrix. Thus, the idea is now is to form the reduced Hessian, i.e., $\mathbf{Q}_H(\mathbf{x}_{h,k}) = \mathbf{R}\nabla^2 f(\mathbf{x}_{h,k})\mathbf{P} \in \mathbb{R}^{n \times n}$, and then perform a rank-p T-SVD; this process requires $\mathcal{O}(mn^2)$ and $\mathcal{O}(pn^2)$ operations, respectively, where $p < n$ and typically $p \ll n$. We now abandon the restrictive part of the Assumption 4.4.2 and examine the general case where equality over the last eigenvalues does not hold, i.e.,

Assumption 4.4.6. *The Hessian $\nabla^2 f(\mathbf{x}_{h,k}) \in \mathbb{R}^{N \times N}$ admits an approximate low-rank factorization of size $p' < N$ and, in particular, for its eigenvalues it holds that*

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{p'} \gg \lambda_{p'+1} \geq \lambda_{p'+2} \geq \cdots \geq \lambda_N.$$

In addition to the above assumption, we suppose that the prolongation operator $\mathbf{P}$ is selected according to Definition 4.4.1. Clearly, the matrix $\mathbf{Q}_H(\mathbf{x}_{h,k})$ can be seen as a sampled version of the full Hessian. As a result, since uniform sampling provides unbiased estimators, $\mathbf{Q}_H(\mathbf{x}_{h,k})$ inherits all the properties of the Hessian, i.e., $\mathbf{Q}_H(\mathbf{x}_{h,k}) \in \mathbb{R}^{n \times n}$ *admits an approximate low-rank factorization of size $p < n$ with eigenvalues of the form*

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \gg \lambda_{p+1} \geq \lambda_{p+2} \geq \cdots \geq \lambda_n.$$

Therefore, the natural way forward is to perform low-rank approximation on the

matrix $\mathbf{Q}_H(\mathbf{x}_{h,k})$. Specifically, we aim to use T-SVD to obtain

$$\tilde{\mathbf{Q}}_{H,k} \approx (\mathbf{R}\nabla^2 f(\mathbf{x}_{h,k})\mathbf{P})^{-1}. \tag{4.43}$$

Since the above approximation is valid, it remains to provide connections with the multilevel framework. In fact, this can be achieved through the naive Nyström method using similar arguments as in Section 4.4.1: observe that, if we left and right multiply with $\mathbf{P}$ and $\mathbf{R}$, respectively, relation (4.43) becomes

$$\mathbf{P}\tilde{\mathbf{Q}}_{H,k}\mathbf{R} \approx \mathbf{P}[\mathbf{R}\nabla^2 f(\mathbf{x}_{h,k})\mathbf{P}]^{-1}\mathbf{R},$$

and so, by the naive Nyström (4.36) and relation (4.38), we have that

$$\hat{\mathbf{Q}}_k := \mathbf{P}\tilde{\mathbf{Q}}_{H,k}\mathbf{R} \approx [\nabla^2 f(\mathbf{x}_{h,k})]^{-1}.$$

Hence, we can claim that

$$\hat{\mathbf{d}}_{h,k} = -\hat{\mathbf{Q}}_k \nabla f(\mathbf{x}_{h,k}) \tag{4.44}$$

is an approximation of the Newton direction $\mathbf{d}_{h,k}$ and, in addition, note that the new approximate decrement can be written as

$$\hat{\lambda}(\mathbf{x}_{h,k}) = \left[\nabla f(\mathbf{x}_{h,k})^T \hat{\mathbf{Q}}_k \nabla f(\mathbf{x}_{h,k})\right]^{1/2}. \tag{4.45}$$

Further, we construct $\hat{\mathbf{Q}}_k$ as follows: compute the rank-p T-SVD of the reduced Hessian matrix $\mathbf{Q}_H(\mathbf{x}_{h,k})$ to obtain $[\mathbf{U}_{p+1}, \mathbf{\Sigma}_{p+1}]$ and form

$$\tilde{\mathbf{Q}}_{H,k} := \lambda_{p+1}^{-1}\mathbf{I}_{n\times n} + \mathbf{U}_p(\mathbf{\Sigma}_p^{-1} - \lambda_{p+1}^{-1}\mathbf{I}_{p\times p})\mathbf{U}_p^T. \tag{4.46}$$

where $\mathbf{U}_p \in \mathbb{R}^{n\times p}$ and $\mathbf{\Sigma}_p \in \mathbb{R}^{p\times p}$. Then, we left and right multiply equation (4.46) with $\mathbf{P}$ and $\mathbf{R}$, respectively, to obtain $\hat{\mathbf{Q}}_k$. We call the algorithm that computes the coarse correction step in this way YAWN$_{\text{SVD}}$ and present it in Algorithm 4.2.

Algorithm 4.2 YAWN$_{\text{SVD}}$

Input: $\rho_1 \in (0,1),\ \alpha \in (0, 0.5),\ \beta \in (0,1),\ \ \epsilon \in (0, 0.68^2),\ \mathbf{R} \in \mathbb{R}^{n \times N}$ by Definition 4.4.1, and the desired rank-$p < n$,
Initialize: $\mathbf{x}_{h,0} \in \mathbb{R}^N$
for $k = 0, 1, \ldots$ **do**
  if $\hat{\lambda}(\mathbf{x}_{h,k}) > \rho_1 \lambda(\mathbf{x}_{h,k})$ and $\hat{\lambda}(\mathbf{x}_{h,k}) > \epsilon$ **then**
    Compute $\mathbf{Q}_H(\mathbf{x}_{h,k}) := \mathbf{R}\nabla^2 f(\mathbf{x}_{h,k})\mathbf{P}$
    Obtain $[\mathbf{U}_{p+1}, \mathbf{\Sigma}_{p+1}]$ using $(p+1)$-Truncated SVD on $\mathbf{Q}_H(\mathbf{x}_{h,k})$
    Form $\tilde{\mathbf{Q}}_{H,k}$ from (4.46) and then $\hat{\mathbf{Q}} := \mathbf{P}\tilde{\mathbf{Q}}_{H,k}\mathbf{R}$
    Compute direction $\mathbf{d} = \hat{\mathbf{d}}_{h,k}$ from (4.44)
  else
    Compute direction $\mathbf{d} = \mathbf{d}_{h,k}$ from (4.13)
  end if
  if $\lambda(\mathbf{x}_{h,k})^2 \leq \epsilon$ **then**
    quit
  end if
  while $f_h(\mathbf{x}_{h,k} + t_k\mathbf{d}) > f_h(\mathbf{x}_{h,k}) - \alpha t_{h,k}\hat{\lambda}(\mathbf{x}_{h,k})^2,\ \ t_{h,k} \leftarrow 1$ **do**
    $t_{h,k} \leftarrow \beta t_{h,k}$
  end while
  Update

$$\mathbf{x}_{h,k+1} := \mathbf{x}_{h,k} + t_{h,k}\mathbf{d}$$

end for
return $\mathbf{x}_{h,k}$

Remark 4.4.7. *We make some important remarks regarding the practical implementation of Algorithms 4.1 and 4.2. Firstly, in the general case, we make no assumptions about the coarse model beyond what has already been discussed in previously. For example, the fine model dimension N could be very large, while n could be just a single dimension. This is the reason why we need to specify both a coarse and a fine direction above (in practice we take $n = N/2$). Using the algorithm in Definition 4.4.1, we sample different $\mathbf{R}$ and $\mathbf{P}$ at every iteration, but we drop the dependence on k to simplify the notation. In our numerical experiments, we set ρ_1 small enough so that YAWN always chooses the coarse direction. In Section 4.4 we describe instances where the value of ρ_1 should not affect the rate of YAWN. In Section 4.6 we show using several examples that in practice YAWN can reach solutions with high accuracy without ever using fine correction steps.*

4.4.4 Convergence Analysis

In this section we show that YAWN$_{\text{SVD}}$ enjoys super-linear convergence rate. Using the framework presented in Section 4.4.3, Algorithm 4.2 can be seen as an inexact version of Algorithm 4.1. Our goal is to show that Theorem 4.3.8 holds for the current setting and thus, similar to YAWN method, the region of super-linear convergence is controlled by $\eta = \frac{3-\sqrt{9-4(1-\varepsilon_{\rho_1})}}{2}$, where $\varepsilon_{\rho_1} = \sqrt{1-\rho_1^2}$.

Theorem 4.4.8. *Suppose that the sequence $\{\mathbf{x}_{h,k}\}$ with $k = 0, 1, 2, \ldots$, is generated by Algorithm 4.2 and let Assumption 4.4.6 hold. Further, suppose that the prolongation operator is generated according to Definition 4.4.1. Then, for any $\rho_1 \in (0, 1)$, the result in Theorem 4.3.8 applies with the same constants γ and η.*

Proof. The proof of the theorem is an immediate result of Theorem 4.3.8. By Assumption 4.4.6 we have that the approximation error for using the inexact search direction in (4.44) is small and hence all requirements of Theorem 4.3.8 hold approximately for the current setting which concludes the proof of the theorem. □

We have showed that the inexact Algorithm 4.2 enjoys a super-linear convergence rate. Specifically, this rate is identical with that of Algorithm 4.1 (see also the discussion followed Theorem 4.3.8 for a description of the region of super-linear convergence). We would like to clarify two aspects of YAWN$_{\text{SVD}}$ method. First, the choice of the approximated Hessian need not necessarily be the one in equation (4.46). As long as the new approximated matrix $\tilde{\mathbf{Q}}_{H,k}$ produces a descent direction, one could use any valid low-rank method, for details see [49], to approximate the Hessian. This fact indicates the broadness of our method. For instance, one could employ just the standard T-SVD method or the QR decomposition method, nevertheless, our numerical results indicate a more efficient performance when the construction in equation (4.46) is applied. Second, Assumption 4.4.6 can be removed without affecting the convergence of the method. *It does not constitute a necessary and sufficient condition for the convergence of the algorithm.* In fact, Assumption 4.4.6 ensures that the decrement in (4.45) will be an effective approxi-

mation of the Newton decrement and thus the coarse direction will always be taken. On the other hand, recall that condition (4.22) ensures that no coarse step will be taken when the YAWN$_{\text{SVD}}$ decrement provides poor approximations. Therefore, we could abandon Assumption 4.4.6 but this might imply an increased number of fine steps which in turn yields expensive iterations. To avoid expensive iterations, one could set the user-defined parameter ρ_1 very small so that condition (4.22) is not activated. However, our analysis suggests that smaller values in ρ_1 result a restricted region of super-linear convergence and thus an increased number of total iterations. Our numerical experience, nevertheless, indicates that the bound of total steps (see Section 4.5) is fairly pessimistic and that, in many cases, YAWN$_{\text{SVD}}$ is able to converge in as many iterations as the Newton method (see Section 4.6). Our intuition suggests that this fact occurs since YAWN$_{\text{SVD}}$ captures only the effective information of the exact Hessian matrix. We also discuss this in the next section where we provide analytical complexity bounds of our methods.

4.5 Complexity Analysis: YAWN vs Newton

In this section we derive a complexity bound for the YAWN method. Since the analysis of YAWN applies to YAWN$_{\text{SVD}}$, the following bounds hold for both methods. We measure the effectiveness of both algorithms in terms of the worst-case number of iterations needed to achieve an accuracy ϵ. The next lemma provides a bound on suboptimality.

Lemma 4.5.1. *Let* $\lambda(\mathbf{x}_{h,k}) \leq 0.68$. *Then,* $f_h(\mathbf{x}_{h,k}) - f_h(\mathbf{x}_h^*) \leq \lambda(\mathbf{x}_{h,k})^2$.

Proof. Using inequality (4.7) we obtain that

$$\begin{aligned} f_h(\mathbf{x}_{h,k}) - f_h(\mathbf{x}_h^*) &\leq \omega_*(\|\nabla f_h(\mathbf{x}_{h,k})\|^*_{\mathbf{x}_{h,k}}) \\ &= \omega_*(\lambda(\mathbf{x}_{h,k})), \qquad \lambda(\mathbf{x}_{h,k}) < 1 \\ &\leq \lambda(\mathbf{x}_{h,k})^2, \end{aligned}$$

where the last inequality holds since $-x - \log(1-x) \leq x^2$, for $x \leq 0.68$, and $\omega_*(\cdot)$ is defined in (4.4). □

In the next lemma we attempt to derive a similar region of convergence as the one proved for the Newton method in [45], i.e., there exists $\eta \leq 1/4$. The idea of the proof is similar to that of Theorem 4.3.8, i.e., convergence is split into two phases, conditioning $\hat{\lambda}(\mathbf{x}_{h,k})$, in order, now, to obtain reduction of $\lambda(\mathbf{x}_{h,k+1}) \leq \frac{1}{2}\lambda(\mathbf{x}_{h,k})$. We show that the region of super-linear convergence is governed by

$$\eta = 2 - \sqrt{3 + 2\sqrt{1-\rho_1^2}},$$

where $\rho_1 \in (0.866, 1)$.

Lemma 4.5.2. *Suppose that the coarse direction, $\hat{\mathbf{d}}_{h,k}$, is employed. For $\rho_1 \in (0.866, 1)$ there exist constants $\gamma > 0$ and $\eta \in (0, 1/4)$ such that*

(i) if $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$, then

$$f_h(\mathbf{x}_{h,k+1}) - f_h(\mathbf{x}_{h,k}) \leq -\gamma$$

(ii) if $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, then the line search selects $t_{h,k} = 1$ with

$$\lambda(\mathbf{x}_{h,k+1}) < \frac{1}{2}\lambda(\mathbf{x}_{h,k}). \tag{4.47}$$

Proof. Recall that the first case, *(i)*, is already proved in Lemma 4.3.1 with $\gamma = \alpha\beta\eta^2/(1+\eta)$. As for the second case, *(ii)*, we proceed using similar arguments as in Theorem 4.3.6. By Lemma 4.3.7 and assumption $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$ we have that

$$\lambda(\mathbf{x}_{h,k+1}) \leq \frac{\varepsilon_{\rho_1} + \eta}{(1-\eta)^2}\lambda(\mathbf{x}_{h,k}),$$

where $\varepsilon_{\rho_1} = \sqrt{1-\rho_1^2}$. We see that the root of $(\varepsilon_{\rho_1} + \eta)/(1-\eta)^2 = 1/2$ is attained at $\eta = 2 - \sqrt{3 + 2\varepsilon_{\rho_1}}$. Therefore, assumption $\rho_1 \in (0.866, 1)$ implies: (*a*) that

$\varepsilon_{\rho_1} \in (0, 1/2)$ and (*b*) that $\eta \in (0, 1/4)$, and hence we conclude that

$$\lambda(\mathbf{x}_{h,k+1}) < \frac{1}{2}\lambda(\mathbf{x}_{h,k}),$$

when $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, which completes the proof. □

We combine the above lemmas to obtain the total number of iterations for Algorithm 4.1.

Theorem 4.5.3. *Suppose that the sequence* $\{\mathbf{x}_{h,k}\}_k$ *with* $k = 0, 1, 2, \ldots$, *is generated by Algorithm 4.1 and that the coarse direction,* $\hat{\mathbf{d}}_{h,k}$, *is employed with* $\rho_1 \in (0.866, 1)$. *Then, the total number of iterations, for achieving an* ϵ *approximate solution, do not exceed*

$$M = \frac{1}{\gamma}[f_h(\mathbf{x}_{h,0}) - f_h(\mathbf{x}_h^*)] + \log_2\left(\frac{1}{\epsilon}\right) + \log_2\left[\left(\frac{\eta}{\rho_1}\right)^2\right] - 1,$$

where γ *is defined in Lemma 4.3.1 and* $\eta \in (0, 1/4)$.

Proof. We make use of the results in Lemma 4.5.2, for obtaining a bound on the number of iterations at each phase.

First Phase: It is obvious that, from Lemma 4.5.2*(i)*, the number of iterations is bounded by

$$M_1 \leq \frac{1}{\gamma}[f_h(\mathbf{x}_{h,0}) - f_h(\mathbf{x}_h^*)],$$

when $\hat{\lambda}(\mathbf{x}_{h,k}) \geq \eta$.

Second Phase: Now suppose that at some iteration k the second phase is activated, i.e., $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, and denote with μ the total number of iterations of the algorithm with $\mu \geq k$, thus it holds $\hat{\lambda}(\mathbf{x}_{h,\mu}) < \eta$. From the result in Lemma 4.5.2*(ii)* observe that

$$\lambda(\mathbf{x}_{h,\mu}) \leq \frac{1}{2}\lambda(\mathbf{x}_{h,\mu-1}) \leq \left(\frac{1}{2}\right)^2 \lambda(\mathbf{x}_{h,\mu-2}) \leq \ldots \leq \left(\frac{1}{2}\right)^{\mu-k} \lambda(\mathbf{x}_{h,k})$$

which implies

$$\lambda(\mathbf{x}_{h,\mu})^2 \leq \left(\frac{1}{2}\right)^{M_2+1} \lambda(\mathbf{x}_{h,k})^2, \tag{4.48}$$

where we let $M_2 = \mu - k$. Further, since, by assumption, $\hat{\mathbf{d}}_{h,k}$ is employed (see condition (4.22)) and $\hat{\lambda}(\mathbf{x}_{h,k}) < \eta$, we have that $\lambda(\mathbf{x}_{h,k}) < \hat{\lambda}(\mathbf{x}_{h,k})/\rho_1 < \eta/\rho_1$ and thus inequality (4.48) becomes

$$\lambda(\mathbf{x}_{h,\mu})^2 \leq \left(\frac{1}{2}\right)^{M_2+1} \left(\frac{\eta}{\rho_1}\right)^2.$$

Next, by Lemma 4.5.1, for $\lambda(\mathbf{x}_{h,\mu}) \leq 0.68$ and for all $\mu \geq k$, it holds that

$$f_h(\mathbf{x}_{h,\mu}) - f_h(\mathbf{x}_h^*) \leq \lambda(\mathbf{x}_{h,\mu})^2 \leq \left(\frac{1}{2}\right)^{M_2+1} \left(\frac{\eta}{\rho_1}\right)^2.$$

Therefore, the number of iterations, for obtaining accuracy $f_h(\mathbf{x}_{h,\mu}) - f_h(\mathbf{x}_h^*) \leq \epsilon$, must be at least

$$M_2 = \log_2\left(\frac{1}{\epsilon}\right) + \log_2\left[\left(\frac{\eta}{\rho_1}\right)^2\right] - 1.$$

Finally, combining the results of both phases we conclude that the total number of iterations is bounded by

$$M = M_1 + M_2 = \frac{1}{\gamma}[f_h(\mathbf{x}_{h,0}) - f_h(\mathbf{x}_h^*)] + \log_2\left(\frac{1}{\epsilon}\right) + \log_2\left[\left(\frac{\eta}{\rho_1}\right)^2\right] - 1,$$

as claimed. □

To this end, in general, one should not expect for YAWN methods to converge in fewer steps compared to the Newton method. Specifically, our analysis suggests that we should expect a difference in the number of iterations between YAWN and Newton methods when assuming that the Hessian matrix does not possess any kind of structure (general case). This fact would yield poor approximations of YAWN decrement compared to the Newton one and thus smaller values of ρ_1 will suffice for the coarse direction to be taken. On the other hand, if the Hessian matrix possesses a structure as in Assumption 4.4.6 and, further, the valuable second-order information

is concentrated in the first few eigenvalues, we should expect YAWN$_{\text{SVD}}$ to provide effective approximations and thus, for any $\rho_1 \in (0, 1)$, no fine step will be taken. This means that, even if ρ_1 approaches 1, YAWN methods will always perform coarse correction steps and thus the bound in Theorem 4.5.3 approaches the Newton one in [45]. Therefore, we come up with a method with a rate that approaches the quadratic rate of Newton method with much cheaper cost per-iteration. Specifically, the total iteration cost, when only the coarse direction is taken, is $\mathcal{O}(mN + (p + m)n^2)$. In practical problems we, typically, have $p \ll n$ so that the total cost per-iteration approximately is $\mathcal{O}((N+n^2)m)$ which, in such cases, indicates a very fast performance for YAWN$_{\text{SVD}}$. This behavior is illustrated in the next section through numerical experiments.

4.6 Numerical Results

In this section we verify our convergence results on extensive numerical results. To validate the efficacy of our algorithms we consider the ℓ_2-regularized logistic regression example

$$\min_{\mathbf{x}_{k,h}\in\mathbb{R}^N} \frac{1}{m}\sum_{i=1}^{m} \log(1+\exp(-b_i \mathbf{x}_{k,h}^T \mathbf{a}_i)) + \ell\|\mathbf{x}_{k,h}\|_2^2,$$

Table 4.1: Datasets and Algorithms used in the experiments, available from `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/` and `http://archive.ics.uci.edu/ml/index.php`

Algorithms	References
Newton	[45]
Sub-Newton	[43]
NewSamp	[68]

Datasets	m	N	ℓ	p
MNIST	$60,000$	784	$1/m$	60
BlogFeedback	$52,396$	280	$1/m$	60
CT Slices	$53,500$	385	$1/m$	60
HAR	$7,352$	561	$1/m$	60
HAPT	$7,767$	561	$1/m$	60
GISETTE	$6,000$	$5,000$	$1/m$	360
Epsilon Normalized	$100,000$	$2,000$	0	200

Table 4.2: Comparison of optimization algorithms over various datasets.

MNIST		
Algorithms	Iterations	CPU Time(sec)
YAWN_{SVD}	22	26.14
YAWN	22	26.65
Newton	22	36.82
Sub-Newton	150	271.75
NewSamp	34	62.07

BlogFeedback		
Algorithms	Iterations	CPU Time(sec)
YAWN_{SVD}	97	29.18
YAWN	118	41.54
Newton	201	137.85
Sub-Newton	89	50.37
NewSamp	102	65.79

CT Slices		
Algorithms	Iterations	CPU Time(sec)
YAWN_{SVD}	24	8.71
YAWN	23	9.34
Newton	22	15.70
Sub-Newton	30	17.03
NewSamp	21	14.02

HAR		
Algorithms	Iterations	CPU Time(sec)
YAWN_{SVD}	22	2.02
YAWN	24	1.85
Newton	17	2.17
Sub-Newton	19	1.81
NewSamp	17	2.20

HAPT		
Algorithms	Iterations	CPU Time(sec)
YAWN_{SVD}	23	2.33
YAWN	24	2.21
Newton	18	3.26
Sub-Newton	32	4.21
NewSamp	17	3.06

GISETTE		
Algorithms	Iterations	CPU Time(sec)
YAWN_{SVD}	15	22.76
YAWN	27	41.81
Newton	13	67.55
Sub-Newton	-	-
NewSamp	-	-

Epsilon Normalized, $\epsilon = 10^{-3}$		
Algorithms	Iterations	CPU Time(sec)
YAWN_{SVD}	10	42.71
NewSamp	6	35.10

Epsilon Normalized, $\epsilon = 10^{-7}$		
Algorithms	Iterations	CPU Time(sec)
YAWN_{SVD}	41	137.54
NewSamp	23	150.07

where $\ell > 0$ is the regularization parameter and $\{\mathbf{a}_i, b_i\}$ the training set, with $\mathbf{a}_i \in \mathbb{R}^N$ and $b_i \in \mathbb{R}$. The gradient and Hessian of the above example can be written explicitly

$$\nabla f(\mathbf{x}_{k,h}) = \frac{1}{m} \sum_{i=1}^{m} b_i \mathbf{a}_i \left(p_i(\mathbf{x}_{k,h}) - 1 \right),$$

and

$$\nabla^2 f(\mathbf{x}_{k,h}) = \frac{1}{m} \sum_{i=1}^{m} b_i^2 p_i(\mathbf{x}_{k,h}) \left(1 - p_i(\mathbf{x}_{k,h}) \right) \mathbf{a}_i \mathbf{a}_i^T,$$

respectively, where $p_i(\mathbf{x}_{k,h}) = 1/\left(1 + \exp(-b_i \mathbf{x}_{k,h} \mathbf{a}_i)\right)$.

Table 4.1 presents the optimization methods to be compared with YAWN and YAWN_{SVD} and the set-up parameters for each example we consider. Specifically, for YAWN, the Armijo step-size rule is used, and the user-defined parameter ρ_1, which controls the number of coarse and fine steps, is taken equal to 0.1 to ensure

Table 4.3: Comparison of YAWN of different values in ρ_1 on Gisette dataset.

YAWN	Total Iterations	Coarse Iterations	CPU Time(sec)
$\rho_1 = 0.1$	27	27	33
$\rho_1 = 0.3$	27	27	33
$\rho_1 = 0.5$	27	27	33
$\rho_1 = 0.7$	18	11	50
$\rho_1 = 0.9$	14	0	66

that in all experiments no fine step will be taken. Next, the prolongation operator $\mathbf{P}$ is selected according to the Definition 4.4.1 with $N = n/2$. For the sub-sampled Newton methods $m/2$ samples are taken to form the Hessian. We used a tolerance of $\epsilon = 10^{-5}$.

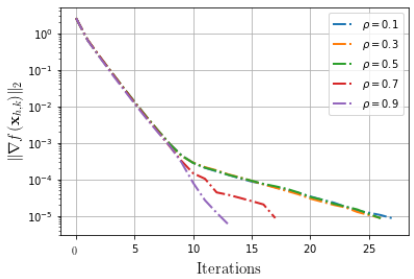
The total performance between the optimization methods can be found in Table 4.2 (see also Figure 4.2 for the convergence behavior). The results were obtained using a standard desktop computer using a Python implementation. In all cases both YAWN and YAWN$_{\text{SVD}}$ outperform their counterparts. In particular, we observe that even in the cases where $m \gg N$, where sub-sampled Newton methods are particularly well suited for, there are instances that both YAWN variants provide an improvement of more than 50% in CPU time. The only case where NewSamp was found to be comparable with YAWN is on the Epsilon Normalized dataset when low accuracy is required. We note that for many applications in machine learning such as background extraction from video and face recognition, much higher accuracy is required (see [8]). When high accuracy is required, YAWN$_{\text{SVD}}$ is faster even for the Epsilon Normalized dataset. On the other hand, when $m \gg N$ does not hold, the performance of state-of-the-art sub-sampled Newton methods is poor even compared to the standard Newton method. In fact, Sub-Newton and NewSamp failed for the GISETTE dataset because they could not reach the required tolerance.

Furthermore, note that even in the cases where YAWN methods require more iterations to converge, they are still better against their competitors since the computational bottleneck arises from the size of the Hessian. For the example we consider here, i.e., $m/2$ samples, Sub-Newton and NewSamp require $\mathcal{O}(mN^2/2 + N^3)$ and

$\mathcal{O}(mN^2/2 + pN^2)$, respectively, to form and compute the inverse of the Hessian, while, for $N/2$, YAWN and YAWN$_{\text{SVD}}$ $\mathcal{O}(mN^2/4 + N^3/8)$ and $\mathcal{O}((m+p)N^2/4)$, respectively, which is a clear advantage. Finally, Figure 4.1 illustrates the convergence behavior of YAWN for different values of ρ_1. In particular, observe that for values $\rho_1 < 0.7$ the coarse step is always taken and thus the convergence is identical. This shows that $\hat{\lambda}(\mathbf{x}_k)$ is a good approximation over $\lambda(\mathbf{x}_k)$ such that condition (4.22) is not activated, see Table 4.3 for exact details. As discussed in Sections 4.3.3 and 4.4, this implies large enough region of super-linear convergence which explains the fast behavior of YAWN we see in this section.

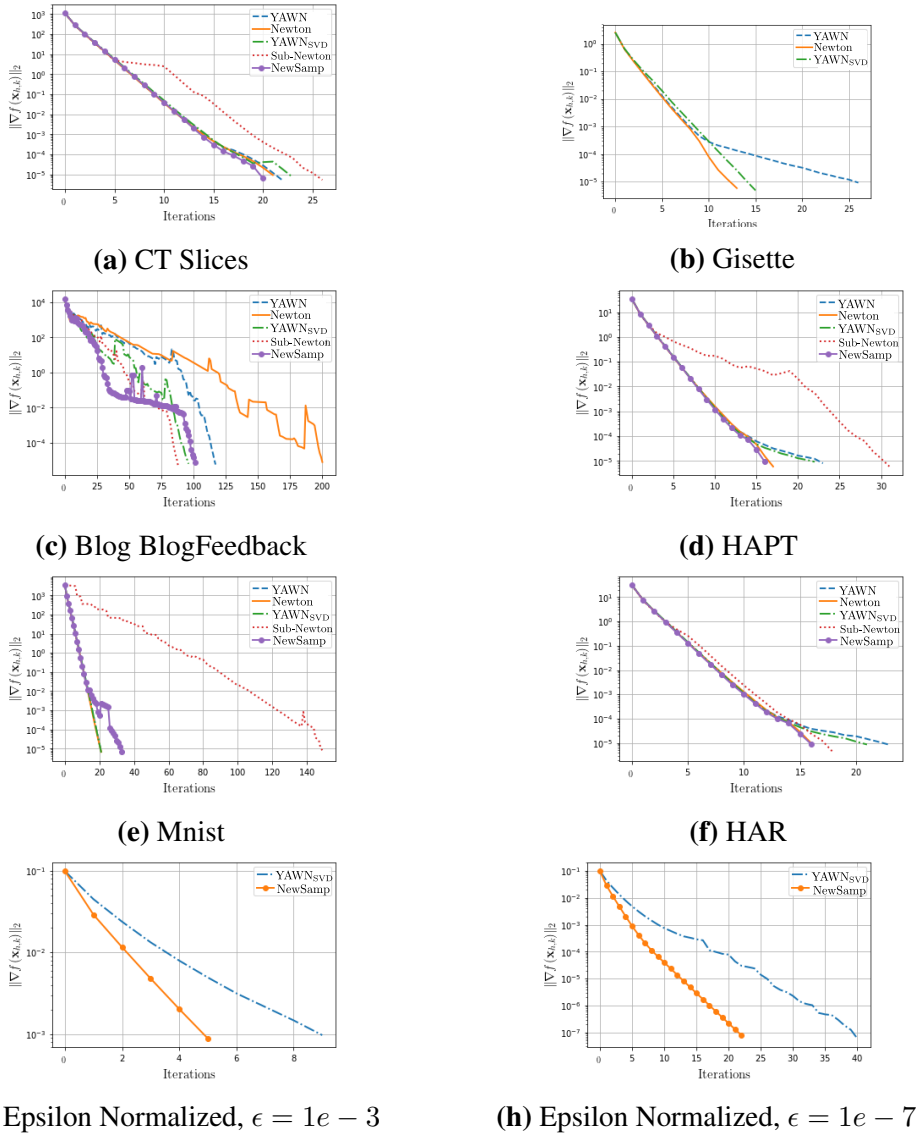
4.7 Conclusion and Perspectives

We proposed YAWN, a second-order variant of the Newton algorithm. We performed the convergence analysis of YAWN with the theory of self-concordant functions. We addressed two significant weaknesses of existing second-order methods for machine learning applications. In particular, the lack of scale-invariant analysis and super-linear convergence rates without restrictive assumptions. Our proof technique draws on insights from a coarse-grained model, called the Galerkin model, from the literature of multi-grid methods. Our primary contribution is the convergence analysis of YAWN. Our analysis closes the theoretical gap between the recent variants of second-order methods for machine learning and the standard Newton method. Beyond the improved theoretical convergence analysis, our prelimi-



(a) Gisette

Figure 4.1: Behavior of YAWN for different values of ρ_1.



(a) CT Slices **(b)** Gisette

(c) Blog BlogFeedback **(d)** HAPT

(e) Mnist **(f)** HAR

(g) Epsilon Normalized, $\epsilon = 1e-3$ **(h)** Epsilon Normalized, $\epsilon = 1e-7$

Figure 4.2: Experiment of different algorithms over various datasets. Error vs Iterations.

nary numerical results suggest that YAWN significantly outperforms state-of-the-art second-order methods.

Recall that our convergence analysis depends on the user-defined parameter ρ_1 which controls the number of fine and coarse iterations to be taken by YAWN. For the general case (see Section 4.3.3), where the problem does not possess any particular structure, one may not expect for the YAWN decrement to be a good approximation of the Newton one, and thus, for always performing the efficient coarse iterations, smaller values in ρ_1 should be selected. As discussed, this fact

may increase the number of steps of the first-phase. In Section 4.4, we discuss how the structure of the problems affects the super-linear rate. In particular, when the Hessian matrix admits a low-rank approximation and, further, the important second-order information is concentrated in the first eigenvalues we should expect that ρ_1 will not affect the region of super-linear rate due to the fact that YAWN decrement will produce effective approximations.

To this end, our primary future goal is to provide a convergence analysis which does not depend on ρ_1, but on parameters arising directly from the structure of the problem —see for instance Theorem 4.4.5 where the rate is governed by λ_{p+1}. This means that, as in Theorem 4.4.5, we are required to estimate the following quantity

$$c_k = \left\| \mathbf{I}_{N\times N} - [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2}\, \hat{\mathbf{Q}}\, [\nabla^2 f_h(\mathbf{x}_{h,k})]^{1/2} \right\|_2 .$$

Thus, the main task would be: select the coarse correction step $\hat{\mathbf{d}}_{h,k}$ with some choice of matrix $\hat{\mathbf{Q}}$ such that $c_k < \epsilon$, where $0 < \epsilon < 1$. Hopefully, using statistical theory (e.g., concentration bounds), this task can be accomplished by making assumptions on the structure of the Hessian matrix and then by carefully selecting the approximated matrix $\hat{\mathbf{Q}}$ to obtain an efficient bound ϵ. Finally, an interesting direction would be to perform sub-sampling techniques within the multilevel framework. That is, assuming that the objective function is written as a sum of functions, sample data points and then build the Galerking (coarse) model by Nyström method. This yields an algorithm with much cheaper iterations. However, the analysis of such a method might be a difficult task.

Chapter 5

Discussion

In this thesis we study optimization algorithms for solving structured problems that arise in machine learning applications. We discuss how such structures can be exploited such that they now lead to improved convergence rates and highly accurate solutions in prediction problems.

In the third chapter, we concentrate on solving the Temporal Difference (TD) learning problem in off-line Reinforcement Learning (RL). In this domain, as our numerical results also suggest, current state-of-the-art algorithms (e.g. LARS-TD) have been unable to provide accurate solutions in the context of policy evaluation and improvement; this is mainly due to the fact that the TD optimization problem (ℓ_1-regularized fixed-point problem) does not reduce to a convex optimization problem. We propose ADMM-TD for solving the ℓ_1-regularized fixed-point problem. We discuss how the proposed method can take advantage of the structure of the problem (separability) which leads to an efficient algorithm in terms of, both, time complexity and accuracy. In particular, our preliminary numerical results indicate an accurate and stable performance even when aiming for optimal policies.

In the fourth chapter, we concentrate on unconstrained convex programs. Building upon the multilevel framework we propose a general optimization method (YAWN), variant of the Newton method, with analysis undertaken using the theory of self-

concordant functions. In particular, we address the following issues that arise in the analysis of randomized-based variants of Newton method (such as sketching and sub-sampling): (a) the analysis of the iterates is not scale-invariant, and, (b) lack of global fast convergence rates without making assumptions on the input data. Therefore, we argue that with the analysis undertaken in this chapter we close the theoretical gap between the recent variants of second-order methods for machine learning and the classical Newton method. In addition, we discuss how typical spectral structures of the Hessian matrix can be exploited, i.e., we show connections between Singular Value Decomposition (SVD) methods and the multilevel framework and we demonstrate how YAWN$_{\text{SVD}}$ can capture such structures (effective rank) to further improve the convergence rates. Finally, our preliminary numerical results suggest that the proposed method is several times faster compared the state-of-the-art methods.

5.1 Future Work

There are many research direction to be considered regarding this thesis. Here, we discuss the main future goals:

In the third chapter, as discussed earlier, the underlying optimization problem in TD learning reduces to a non-convex problem which implies that a proof of convergence of ADMM-TD is difficult task. For this reason, we have not yet been able to come up with a complete analysis of our proposed method. Therefore, our primary goal is to accomplish this task. The satisfying numerical performance of ADMM-TD in the context of policy evaluation and improvement suggests that we can possibly establish a proof of convergence without restrictive assumptions. Further, we aim to test the efficiency of ADMM-TD on machine learning problems using real datasets.

In the fourth chapter, we show the super-linear convergence rate of YAWN method. Specifically, recall the existence of the user-defined parameter ρ_1 in the analysis of YAWN which, as has been discussed, plays a crucial role for the behavior of the

proposed method. Our main goal is to establish an analysis which is independent of the parameter ρ_1. This will help providing more meaningful interpretations about the region of the super-linear convergence. Next, we aim to develop YAWN to incorporate sub-sampling techniques. This can be accomplished by assuming that the objective function can be written as a sum of functions. Therefore, the idea is to sample data points and then build coarse models. It is evident that the computational cost of a method with such iterates will be significantly decreased.

In conclusion, we believe that YAWN can be efficiently applied to various structured problems. In particular, we would like to draw connections between the ℓ_1-regularized fixed-point problem in RL and the YAWN$_{\text{SVD}}$ algorithm. Recall that for the ℓ_1-regularized fixed-point in the context of policy iteration, the positive-definiteness assumption of the matrices does not hold. This is the main drawback in this domain and results in non-convergent algorithms. However, recall that YAWN$_{\text{SVD}}$ developed in this thesis for capturing the effective rank of the second-order information by replacing all the last $(n - p)$ eigenvalues with a small positive eigenvalue. To this end, we conjecture that, when the matrices which arise in the ℓ_1-regularized fixed-point problem exhibit such structure, YAWN$_{\text{SVD}}$ can be efficiently applied to solve the TD learning problem. We argue that YAWN$_{\text{SVD}}$ will always be convergent in the policy iteration context since it will replace the ill-conditioned matrices with positive-definite ones. Therefore, our primary goal is to perform initial numerical experiments to test the behavior of YAWN$_{\text{SVD}}$ in the context of policy iteration.

Bibliography

[1] Csaba Szepesvári. Algorithms for reinforcement learning. *Morgan and Claypool*, 2009.

[2] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[3] Manuel Loth, Manuel Davy, and Philippe Preux. Sparse temporal difference learning using lasso. In *IEEE international symposium on approximate dynamic programming and reinforcement learning*, 2007.

[4] Alejandro J Weinstein. *Inference and learning in high-dimensional spaces*. PhD thesis, Colorado School of Mines. Arthur Lakes Library, 2013.

[5] S Sra, S Nowozin, and SJ Wright. Optimization for machine learning. neural information processing series, 2012.

[6] Jean-Luc Prigent. *Portfolio optimization and performance analysis*. Chapman and Hall/CRC, 2007.

[7] Alice Yalaoui, Hicham Chehade, Farouk Yalaoui, and Lionel Amodeo. *Optimization of logistics*. John Wiley & Sons, 2012.

[8] Vahan Hovhannisyan, Panos Parpas, and Stefanos Zafeiriou. MAGMA: multilevel accelerated gradient mirror descent algorithm for large-scale convex composite minimization. *SIAM J. Imaging Sci.*, 9(4):1829–1857, 2016.

[9] Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, Boston, MA, 2004. A basic course.

[10] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[11] Neal Parikh and Stephen Boyd. Proximal algorithms, in foundations and trends in optimization, 2013.

[12] Jonathan Eckstein and Dimitri P Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

[13] Junfeng Yang and Yin Zhang. Alternating direction algorithms for ℓ_1-problems in compressive sensing. *SIAM journal on scientific computing*, 33(1):250–278, 2011.

[14] Alexander Jung, Gabor Hannak, and Norbert Goertz. Graphical lasso based model selection for time series. *IEEE Signal Processing Letters*, 22(10):1781–1785, 2015.

[15] Joao FC Mota, Joao MF Xavier, Pedro MQ Aguiar, and Markus Puschel. D-admm: A communication-efficient distributed algorithm for separable optimization. *IEEE Transactions on Signal Processing*, 61(10):2718–2723, 2013.

[16] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

[17] Darrin C Bentivegna, Ales Ude, Christopher G Atkeson, and Gordon Cheng. Humanoid robot learning and game playing using pc-based vision. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2449–2454. IEEE, 2002.

[18] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pages 363–372. Springer, 2006.

[19] Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[20] Charles W Anderson. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, 9(3):31–37, 1989.

[21] Wei Zhang and Thomas G Dietterich. A reinforcement learning approach to job-shop scheduling. In *IJCAI*, volume 95, pages 1114–1120. Citeseer, 1995.

[22] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[23] John N Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, pages 1075–1081, 1997.

[24] Justin A Boyan. Least-squares temporal difference learning. In *ICML*, pages 49–56, 1999.

[25] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003.

[26] Steven J Bradtke and Andrew G Barto. Linear least-squares algorithms for temporal difference learning. *Machine learning*, 22(1-3):33–57, 1996.

[27] Amir M Farahmand, Mohammad Ghavamzadeh, Shie Mannor, and Csaba Szepesvári. Regularized policy iteration. In *Advances in Neural Information Processing Systems*, pages 441–448, 2009.

[28] Matthieu Geist and Bruno Scherrer. ℓ_1-penalized projected Bellman residual. In *European Workshop on Reinforcement Learning*, pages 89–101. Springer, 2011.

[29] Zhiwei Qin, Weichang Li, and Firdaus Janoos. Sparse reinforcement learning via convex optimization. In *International Conference on Machine Learning*, pages 424–432, 2014.

[30] Jeffrey Johns, Christopher Painter-Wakefield, and Ronald Parr. Linear complementarity for regularized policy evaluation and improvement. In *Advances in neural information processing systems*, pages 1009–1017, 2010.

[31] J Zico Kolter and Andrew Y Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 521–528. ACM, 2009.

[32] Nikos Tsipinakis and James DB Nelson. Sparse temporal difference learning via alternating direction method of multipliers. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pages 220–225. IEEE, 2015.

[33] Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. *Multigrid*. Elsevier, 2000.

[34] P Wesseling. An introduction to multigrid methods. sl wiley. *New York*, 1992.

[35] William L Briggs, Steve F McCormick, et al. *A multigrid tutorial*, volume 72. Siam, 2000.

[36] Antony Jameson. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. In *10th Computational Fluid Dynamics Conference*, page 1596, 1991.

[37] Stephen G Nash. A multigrid approach to discretized optimization problems. *Optimization Methods and Software*, 14(1-2):99–116, 2000.

[38] Serge Gratton, Annick Sartenaer, and Philippe L Toint. Recursive trust-region methods for multiscale nonlinear optimization. *SIAM Journal on Optimization*, 19(1):414–444, 2008.

[39] Zaiwen Wen and Donald Goldfarb. A line search multigrid method for large-scale nonlinear optimization. *SIAM J. Optim.*, 20(3):1478–1503, 2009.

[40] Chinpang Ho. Multilevel algorithms for the optimization of structured problems. 2016.

[41] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization Methods for Large-Scale Machine Learning. *SIAM Rev.*, 60(2):223–311, 2018.

[42] Mert Pilanci and Martin J. Wainwright. Newton sketch: a near linear-time optimization algorithm with linear-quadratic convergence. *SIAM J. Optim.*, 27(1):205–245, 2017.

[43] Albert S Berahas, Raghu Bollapragada, and Jorge Nocedal. An investigation of Newton-sketch and subsampled Newton methods. *arXiv preprint arXiv:1705.06211*, 2017.

[44] Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled Newton methods i: globally convergent algorithms. *arXiv preprint arXiv:1601.04737*, 2016.

[45] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, 2004.

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

[47] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

[48] Heinz H Bauschke, Patrick L Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

[49] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[50] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, second edition, 2013.

[51] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[52] Jonas Ballani and Daniel Kressner. Matrices with hierarchical low-rank structures. In *Exploiting hidden structure in matrix computations: algorithms and applications*, pages 161–209. Springer, 2016.

[53] Petros Drineas and Michael W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *J. Mach. Learn. Res.*, 6:2153–2175, 2005.

[54] Alex A Gittens. *Topics in randomized numerical linear algebra*. PhD thesis, California Institute of Technology, 2013.

[55] Christopher KI Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

[56] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[57] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[58] Mohammad Ghavamzadeh, Alessandro Lazaric, Rémi Munos, and Matt Hoffman. Finite-sample analysis of Lasso-TD. In *International Conference on Machine Learning*, 2011.

[59] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[60] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[61] Elaine T Hale, Wotao Yin, and Yin Zhang. Fixed-point continuation for ℓ_1-minimization: Methodology and convergence. *SIAM Journal on Optimization*, 19(3):1107–1130, 2008.

[62] Christopher Painter-Wakefield, Ronald Parr, and NC Durham. L1 regularized linear temporal difference learning. *Technical report: Department of Computer Science, Duke University, Durham, NC, TR-2012–01*, 2012.

[63] Serge Gratton, Mélodie Mouffe, Annick Sartenaer, Philippe L. Toint, and Dimitri Tomanos. Numerical experience with a recursive trust-region method for multilevel nonlinear bound-constrained optimization. *Optim. Methods Softw.*, 25(3):359–386, 2010.

[64] Yossi Arjevani and Ohad Shamir. Oracle complexity of second-order methods for finite-sum problems. *arXiv preprint arXiv:1611.04982*, 2016.

[65] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13 of *SIAM Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994.

[66] Robert Michael Lewis and Stephen G Nash. Model problems for the multigrid optimization of systems governed by differential equations. *SIAM Journal on Scientific Computing*, 26(6):1811–1837, 2005.

[67] Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization*, 23(4):2037–2060, 2013.

[68] Murat A Erdogdu and Andrea Montanari. Convergence rates of sub-sampled Newton methods. In *Proceedings of the 28th International Conference on*

Neural Information Processing Systems-Volume 2, pages 3052–3060. MIT Press, 2015.