2017

Limited Memory Steepest Descent Methods for Nonlinear Optimization

Wei Guo
Lehigh University

Limited Memory Steepest Descent Methods for Nonlinear Optimization

by

Wei Guo

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Industrial and Systems Engineering

Lehigh University

May 2017

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Date

Dr. Frank E. Curtis, Dissertation Director

Accepted Date

Committee Members:

Dr. Frank E. Curtis, Committee Chair

Dr. Yu-Hong Dai

Dr. Daniel P. Robinson

Dr. Katya Scheinberg

Dr. Martin Takáč

Acknowledgements

I would like to thank first, and above all, my academic advisor, Professor Frank E. Curtis. It is my honor and privilege to have had him as my advisor. His extraodinary expertise, vision, and patience have guided me during my PhD study, especially his knowledgable insights on how to conduct meaningful and influential research, how to overcome various difficulties and challenges along the way, as well as how to work more efficiently and make good use of time. Besides his professional knowledge and thoughts, I have also received valuable suggestions on how to improve my technical writing skills and polish my communication and presentation skills, which not only have benefitted me in the past years, but also will make my future career more successful and fruitful.

I also thank the remaining members of my dissertation committee — Professor Yu-Hong Dai, Professor Daniel P. Robinson, Professor Katya Scheinberg, and Professor Martin Takáč. Professor Yu-Hong Dai's pioneering explorations and deep understanding in nonlinear optimization have inspired the work in my dissertation a lot. Professor Daniel P. Robinson has made thoughtful suggestions and comments on our papers. Professor Katya Scheinberg has explained her knowledge and understanding in topics of optimization and machine learning through the courses she taught. Professor Martin Takáč has selflessly shared with me many of his helpful experiences and has given me guidance on research topic selections as well as presentation strategies. I also appreciate Professor Frank E. Curtis and Yu-Hong Dai for providing valuable suggestions and comments on my dissertation.

I am also grateful to many people around me, including those warm-hearted aunts and uncles from the church and my close friends, who have played an important part of my

life and have helped me without hesitation when I am in need. Finally, most of all, I owe my deepest gratitude to my parents and grandparents. I could never have gone this far without their unconditional love, support, and encouragement.

Contents

List of Tables

List of Figures

Abstract

This dissertation concerns the development of limited memory steepest descent (LMSD) methods for solving unconstrained nonlinear optimization problems. In particular, we focus on the class of LMSD methods recently proposed by Fletcher, which he has shown to be competitive with well-known quasi-Newton methods such as L-BFGS. However, in the design of such methods, much work remains to be done. First of all, Fletcher only showed a convergence result for LMSD methods when minimizing strongly convex quadratics, but no convergence rate result. In addition, his method mainly focused on minimizing strongly convex quadratics and general convex objectives, while when it comes to nonconvex objectives, open questions remain about how to effectively deal with nonpositive curvature. Furthermore, Fletcher's method relies on having access to exact gradients, which can be a limitation when computing exact gradients is too expensive. The focus of this dissertation is the design and analysis of algorithms intended to solve these issues.

In the first part of the new results in this dissertation, a convergence rate result for an LMSD method is proved. For context, we note that a basic LMSD method is an extension of the Barzilai-Borwein "two-point stepsize" strategy for steepest descent methods for solving unconstrained optimization problems. It is known that the Barzilai-Borwein strategy yields a method with an R-linear rate of convergence when it is employed to minimize a strongly convex quadratic. Our contribution is to extend this analysis for LMSD, also for strongly convex quadratics. In particular, it is shown that, under reasonable assumptions, the method is R-linearly convergent for any choice of the history length parameter. The results of numerical experiments are also provided to illustrate behaviors of the method that are revealed through the theoretical analysis.

The second part proposes an LMSD method for solving unconstrained *nonconvex* optimization problems. As a steepest descent method, the step computation in each iteration only requires the evaluation of a gradient of the objective function and the calculation of a scalar stepsize. When employed to solve certain convex problems, our method reduces to a variant of LMSD method proposed by Fletcher, which means that, when the history length parameter is set to one, it reduces to a steepest descent method inspired by that proposed by Barzilai and Borwein. However, our method is novel in that we propose new algorithmic features for cases when nonpositive curvature is encountered. That is, our method is particularly suited for solving nonconvex problems. With a nonmonotone line search, we ensure global convergence for a variant of our method. We also illustrate with numerical experiments that our approach often yields superior performance when employed to solve nonconvex problems.

In the third part, we propose a limited memory stochastic gradient (LMSG) method for solving optimization problems arising in machine learning. As a start, we focus on problems that are strongly convex. When the dataset is too large such that the computation of full gradients is too expensive, our method computes stepsizes and iterates based on (mini-batch) stochastic gradients. Although in stochastic gradient (SG) methods, a best-tuned fixed stepsize or diminishing stepsize is most widely used, it can be inefficient in practice. Our method adopts a cubic model and always guarantees a positive meaningful stepsize, even when nonpositive curvature is encountered (which can happen when using stochastic gradients, even when the problem is convex). Our approach is based on the LMSD method with cubic regularization proposed in the second part of this dissertation. With a projection of stepsizes, we ensure convergence to a neighborhood of the optimal solution when the interval is fixed and convergence to the optimal solution when the interval is diminishing. We also illustrate with numerical experiments that our approach can outperform an SG method with a fixed stepsize.

Chapter 1

Introduction

Mathematical optimization, where one models a real-world problem to minimize an objective function over a set of variables that might need to satisfy certain constraints, has been a formal subject of research for decades and plays a significant role in many areas of engineering and applied mathematics. It arises in an abundant number of applications in multiple areas, including machine learning [7, 48], control [32, 30, 51], compressed sensing [20, 9, 19], image processing [31, 36], robust optimization [4], finance [25, 37], optics [6], distance geometry [23], and many others. There are a variety of well-developed algorithm classes for solving optimization problems built, e.g., on steepest descent or (quasi-)Newton methodologies.

Optimization problems can be very large in size. In particular, one manner in which an optimization problem can be large is if it involves a large number of variables, say in the millions or more. For problems of such sizes, popular algorithms such as Newton's method—a second-order method—might not be efficient for the following reasons.

- Second order derivatives (i.e., Hessian matrices) are expensive to compute.
- There may not be large enough space to store the Hessian matrices, even if they can be computed.
- They involve solving linear systems of equations, which can have high cost.

By contrast, steepest descent methods—which are first-order methods—are popular

when solving large-scale optimization problems. Steepest descent methods have multiple advantages.

- Function and gradient evaluations are relatively cheap.
- It is beneficial to "move quickly" compared with Newton's methods.
- Steepest descent methods are easily generalized for solving constrained problems (in that they are not complicated, e.g., by indefinite Hessian matrices).

Besides the advantages mentioned above, we can also combine (limited memory) quasi-Newton ideas with steepest descent methods to arrive at effective methods such as L-BFGS [40]. By applying these strategies, we do not only make computation during each iteration cheap but also save a lot of storage cost.

There are also other ways that an optimization problem can be large. For example, in machine learning, optimization problems arise when one wants to minimize an objective defined by a given dataset. There is the number of "samples" composing the dataset, and there is the number of optimization variables, which is often equal to the number of "features" contained in each sample. A challenge in "big data" problems is that either or both of the following can be true:

- **The number of features can be very large.**
- **The number of samples can be very large.**

In some situations, when not only the number of features, but also the number of samples is very large, the design of optimization method is complicated by the fact that the computation of a "full" gradient can be very expensive. When this is the case, stochastic gradient (SG) and its variants have been the main approaches for solving such problems. For example, stochastic gradient methods are actively used in machine learning, including support vector machine [52], linear and logistic regression [50], deep neural networks [1]—well known supervised learning models [39] for prediction and classfication.

The goal in this dissertation is to conduct research on limited memory steepest descent (LMSD) methods for solving nonlinear optimization problems. In particular, the class of

algorithms we explore is based on the work of Fletcher in [22], which in turn can be seen as an extension of the Barzilai-Borwein "two-point stepsize" strategy for steepest descent methods for solving unconstrained optimization problems [2].The extension can be quantified using a so-called history length parameter, for which we use the letter m. A BB method corresponds to $m = 1$, whereas LMSD involves any integer $m \in [1, n]$.

For BB methods, there are now well-known properties when it is employed to minimize an n-dimensional strongly convex quadratic objective function. Such objective functions are interesting in their own right, but one can argue that such analyses also characterize the behavior of the method in the neighborhood of a strong local minimizer of any smooth objective function. In the original work (i.e., [2]), it is shown that the method converges R-superlinearly when $n = 2$. In [44], it is shown that the method converges from any starting point for any natural number n, and in [15] it is shown that the method converges R-linearly for any such n. In [22], it is shown that the proposed LMSD method converges from any starting point when it is employed to minimize a strongly convex quadratic.

However, to the best of our knowledge, the convergence rate of the method for $m > 1$ has not yet been analyzed. Our first contribution is to show that Fletcher's LMSD method converges R-linearly when employed to minimize such a function. Our analysis builds upon the analyses in [22] and [15]. We also present the results of numerical experiments that illustrate our convergence theory and demonstrate that the practical performance of LMSD can be even better than the theory suggests.

Once the rate of convergence properties of LMSD method for strongly convex quadratics are analyzed, in the second part, we focus on designing efficient algorithms within a steepest descent framework for solving general unconstrained optimization problems whose objective functions are continuously differentiable. The motivation of the work in this part is that, when it comes to solving nonconvex problems—which are not rare in practice—many methods falter and do not exhibit the same level of performance as when convex problems are considered. This also happens to the original BB method and LMSD method, which drives us to explore some alternative approaches for handling nonconvex problems.

The main contribution of the new methodology we devise is that it provides a novel

strategy for computing stepsizes when solving nonconvex optimization problems. In particular, when nonpositive curvature (as defined later) is encountered, our method adopts a local cubic model of the objective function in order to determine a stepsize (for $m = 1$) or sequence of stepsizes (for $m \geq 1$). As in the case of the original BB methods and the LMSD method of Fletcher, our basic algorithm does not enforce sufficient decrease in the objective in every iteration. However, as is commonly done for variants of BB methods, we remark that, with a nonmonotone line search, a variant of our algorithm attains global convergence guarantees under weak assumptions. Our method also readily adopts the convergence rates attainable by a BB or LMSD method if/when it reaches a neighborhood of the solution in which the objective is strictly convex.

The motivation of the work in the third part is to solve the type of problems with not only a very large number of features but also a very large number of samples such that the computation of full gradients is very expensive. We focus on designing algorithms of the stochastic gradient (SG) variety. However, one of the major challenges in stochastic gradient (SG) methods is how to choose an appropriate stepsize while running the algorithm. Since traditional line search techniques are not readily applied in stochastic optimization methods, the common practice in SG is either to tune a stepsize by hand, or to tune a diminishing stepsize sequence, which can be time consuming and inefficient in practice.

Our contribution in this part is that we extend the BB ($m = 1$) and LMSD methods ($m \geq 1$) adopting cubic models using full gradient proposed in the second part to Barzilai-Borwein stochastic gradient method with cubic regularization (BBSG) and limited memory stochastic gradient method with cubic regularization (LMSG). We compute meaningful positive stepsizes from BBSG when $m = 1$ or LMSG when $m \geq 1$ instead of using fixed or diminishing stepsizes. We also prove convergence (rate) properties for LMSG with fixed interval projection and diminishing interval projection for strongly convex objectives under mild assumptions, respectively. Furthermore, we conduct numerical experiments for LMSG on solving logistic regression problems. The numerical results show that LMSG can outperform SG with a fixed stepsize.

This dissertation is organized in the following manner. We first review, in Chapter 2,

some mathematical bakground, including convex analysis, (numerical) linear algebra and basic optimization theory. We follow this by introducing literature on gradient-based methods (full gradient) for optimization, including steepest descent (SD) methods, BB methods and LMSD methods, their variants for computing stepsizes together with their theoretical properties. Next, we discuss stochastic gradient (SG) methods. Chapter 3 illustrates the R-linear convergence rate of LMSD for strongly convex quadratics, which extends the convergence rate analysis of BB methods. In Chapter 4, we propose an LMSD method for solving unconstrained optimization problems, especially when nonpositive curvature is encountered. In Chapter 5, we design BBSG and LMSG methods within a stochastic gradient (SG) framework to solve "big data" problems. Final remarks and comments on all of the methods in this dissertation are presented in Chapter 6.

Chapter 2

Background and Literature Review

In this chapter, we first cover background on convex analysis, some fundamental (numerical) linear algebra and basic optimization theory. With this understanding of mathematical background, we will introduce our first method, steepest descent (SD), which belongs to the class of gradient-based methods. This is followed by a literature review on Barzilai-Borwein (BB) methods, including algorithm descriptions and theoretical convergence results. Next, we will show how to incorporate line search techniques to BB methods as well as some BB methods variants. Followed by BB methods, we discuss limited memory steepest descent (LMSD) methods, a generalization of BB methods. Finally, unlike all the methods mentioned above which apply full gradient during each iteration, we will explain in detail the methods that use a porportion of the full gradient, namely stochastic gradient (SG) methods.

We always assume matrix A (used in this chapter) is symmetric and thus all its eigenvalues, $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, are real.

2.1 Convex Analysis

In this Chapter, we provide some basic definitions and properties related to (strong) convexity and Lipschitz continuity.

2.1.1 Convexity

A real-valued function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if we have

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \text{ for all } (x,y) \in \mathbb{R}^n \times \mathbb{R}^n \text{ and } \alpha \in [0,1]. \quad (2.1)$$

A function f is strictly convex if for $x \neq y$ inequality (2.1) holds strictly. A convex function has the following properties.

- (Continuity) If f is convex, then it is continuous.
- (First order condition) If f is continuously differentiable, then we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \text{ for all } (x,y) \in \mathbb{R}^n \times \mathbb{R}^n.$$

- (Monotone mapping) f is differentiable and convex if and only if

$$(\nabla f(x) - \nabla f(y))^T (x - y) \geq 0 \text{ for all } (x,y) \in \mathbb{R}^n \times \mathbb{R}^n.$$

2.1.2 Lipschitz Continuity

A function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous if there exists a scalar $L > 0$ such that

$$||f(x) - f(y)||_2 \leq L||x - y||_2 \text{ for all } (x,y) \in \mathbb{R}^n \times \mathbb{R}^n. \quad (2.2)$$

Here are some properties related to Lipschitz continuity.

- f is convex if and only if $\frac{L}{2}x^T x - f(x)$ is convex.
- Suppose ∇f is Lipschitz continuous with parameter L, then $\frac{L}{2}x^T x - f(x)$ is convex.

- (Quadratic upper bound) Suppose ∇f is Lipschitz continuous with parameter L, then convexity of g means that we can derive this upper bound for f:

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}||y-x||_2^2 \text{ for all } (x,y) \in \mathbb{R}^n \times \mathbb{R}^n. \tag{2.3}$$

2.1.3 Strong Convexity

A function $f : \mathbb{R}^n \to \mathbb{R}$ is strongly convex with parameter $\sigma > 0$ if $f(x) - \frac{\sigma}{2}x^T x$ is convex. A strongly convex function has the following properties.

- (First order condition)

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\sigma}{2}||y-x||_2^2 \text{ for all } (x,y) \in \mathbb{R}^n \times \mathbb{R}^n. \tag{2.4}$$

- If f is strongly convex and x_* is the unique minimizer, then

$$\frac{\sigma}{2}||x-x_*||_2^2 \leq f(x) - f(x_*) \leq \frac{1}{2\sigma}||\nabla f(x)||_2^2 \text{ for all } x \in \mathbb{R}^n. \tag{2.5}$$

2.2 (Numerical) Linear Algebra

2.2.1 Symmetric Positive Definite (SPD) Matrix

A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is positive definite ($A \succ 0$) if and only if for all $x \in \mathbb{R}^n$ and $x \neq 0$, we have

$$x^T A x > 0.$$

Here are some basic properties of symmetric positive definite matrices.

- All A's eigenvalues are real and positive.

$$0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n.$$

- Principle submatrices of A are also positive definite.

- A^{-1} exists and is also symmetric positive definite ($A^{-1} \succ 0$).
- The diagonal elements of matrix A are positive, $a_{ii} > 0$.
- Eigenvalue decomposition.

  There exists $n \times n$ orthogonal matrix $U \in \mathbb{R}^{n \times n}$, where $U^T U = U U^T = I$ and diagonal matrix Λ, such that

  $$U^T A U = \Lambda.$$

  where Λ contains all eigenvalues of A.

- (Quadratic forms) A strongly convex quadratic function $f : \mathbb{R}^n \to \mathbb{R}$ can be formulated as $x^T A x$ where x is the column vector and A is symmetric positive definite.

2.2.2 Krylov Subspace and Krylov Sequence

In linear algebra, the order m ($m \leq n$) Krylov subspace generated by $A \in \mathbb{R}^{n \times n}$ and a vector $g \in \mathbb{R}^n$ is the linear subspace spanned by the images of g under the first m powers of A (starting from $A^0 = I$), that is,

$$\mathcal{K}_m(A, g) = \text{ span } \{g, Ag, A^2 g, \cdots, A^{m-1} g\}.$$

And $A^i g,\ i = 0, 1,\ \cdots,\ m-1$, the basis of Krylov subspace, are called Krylov sequence initiated from g. More properties related to Krylov sequence will be discussed in steepest descent methods, i.e., Chapter 2.4.2 below.

2.2.3 Rayleigh Quotient and (Harmonic) Ritz Value

For the m-dimensional Krylov subspace

$$\mathcal{K}_m(A, g) = \text{ span } \{g, Ag, A^2 g, \cdots, A^{m-1} g\},$$

starting from

$$q_1 = \frac{g}{||g||},$$

we can generate orthonormal basis $Q = \{q_1, q_2, \cdots, q_m\}$ for Krylov subspace $\mathcal{K}_m(A, g)$, where $Q \in \mathbb{R}^{n\times m}$ is orthogonal matrix and $Q^TQ = I$.

Sometimes the eigenvalues of matrix $A \in \mathbb{R}^{n\times n}$ can be expensive to compute especially when n is very large, one basic idea is to formulate a smaller dimensional matrix $T \in \mathbb{R}^{m\times m}$ and compute its m eigenvalues as approximations of the original n eigenvalues. A typical way is to project matrix A to the m dimensional subspace $\mathcal{K}_m(A, g)$, i.e.

$$T = Q^TAQ,$$

and compute the m eigenvalues of matrix T.

When $m = 1$, $Q = q_1 \in \mathbb{R}^{n\times 1}$ with $||q_1||_2^2 = 1$, the matrix T becomes a number, we define this number as Rayleigh quotient $R(A, q_1)$, where

$$R(A, q_1) = q_1^TAq_1.$$

The more general definition of Rayleigh quotient is

$$R(A, q_1) = \frac{q_1^TAq_1}{q_1^Tq_1}.$$

with arbitrary $q_1 \in \mathbb{R}^{n\times 1}$. In addition, the range of $R(A, q_1)$ is

$$\lambda_1 \leq R(A, q_1) \leq \lambda_n.$$

When we have general $m \geq 1$ and obtain matrix $T \in \mathbb{R}^{m\times m}$, we define m eigenvalues of matrix T, i.e.

$$\theta_m \leq \theta_{m-1} \leq \cdots \leq \theta_1,$$

as Ritz values of matrix A, where

$$T = Q^TAQ.$$

We also define m eigenvalues of matrix $(P^{-1}T)^{-1}$, i.e.,

$$\mu_m \leq \mu_{m-1} \leq \cdots \leq \mu_1,$$

as harmonic Ritz values of matrix A, where

$$P = Q^T A^2 Q.$$

We have T as a tridiagonal matrix and P as a pentadiagonal matrix when A is symmetric. Here is the contrast of the computation of Ritz values and harmonic Ritz values. We write $\Theta = diag(\theta_i)$, $\Xi = diag(\mu_i)$ and have the following eigensystems

$$(Q^T A Q)X = (Q^T Q)X\Theta.$$

If we were to include an extra A in the innerproducts of $Q^T Q$ and $Q^T AQ$, we would obtain a generalized eigensystem

$$(Q^T A^2 Q)X = (Q^T AQ)X\Xi.$$

There are also alternative ways of computing matrices both T and P when A is unavailable, which will be explained in detail in Chapter 4.2.2.

Here is also a well known theorem revealing important interlacing relations between the eigenvalues of T, i.e. Ritz values and the eigenvalues of A. The theorem is known as the *Cauchy Interlacing Theorem.*

Theorem 2.2.1. *The eigenvalues of T $(= Q^T AQ$ where $Q^T Q = I)$ satisfy*

$$\theta_j \in [\lambda_{m+1-j}, \lambda_{n+1-j}] \quad \textit{for all} \quad j \in [m].$$

2.2.4 Lanczos Method

A well known method of computing matrix T from matrix A is the Lanczos method. Here is the algorithm framework.

Algorithm 1 Lanczos Method

input $q_1 \leftarrow \frac{g}{||g||},\ q_0 \leftarrow 0,\ \beta_1 \leftarrow 0$

for $j = 1, 2 \cdots, m-1$ **do**

  $w'_j \leftarrow Aq_j$

  $\alpha_j \leftarrow w'_j q_j$

  $w_j \leftarrow w'_j - \alpha_j q_j - \beta_j q_{j-1}$

  $\beta_{j+1} \leftarrow ||w_j||$

  $q_{j+1} \leftarrow \frac{w_j}{\beta_{j+1}}$

end for

$w_m \leftarrow Aq_m$

$\alpha_m \leftarrow w'_m q_m$

From Lanczos method, we can get $T \in \mathbb{R}^{m \times m}$, where

$$T = \begin{pmatrix} \alpha_1 & \beta_2 & & & & 0 \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \ddots & & \\ & & \ddots & \ddots & \beta_{m-1} & \\ & & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ 0 & & & & \beta_m & \alpha_m \end{pmatrix}.$$

2.2.5 QR Factorization

Any real square matrix $A \in \mathbb{R}^{n \times n}$ (not necessarily symmetric) may be decomposed as

$$A = QR,$$

where Q is an orthogonal matrix ($Q^T Q = I$) and R is an upper triangular matrix. If A is invertible, then the factorization is unique if we require that the diagonal elements of R be positive.

More generally, we can factor an $A \in \mathbb{R}^{n \times m}$, with $n \geq m$, as the product of an orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ and an upper triangular matrix $R \in \mathbb{R}^{n \times m}$. As the bottom

$(n-m)$ rows of an $n \times m$ upper triangular matrix consist entirely of zeros, it is often useful to partition R, or both R and Q:

$$A = QR = Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = [Q_1,\ Q_2] \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1,$$

where $R_1 \in \mathbb{R}^{m \times m}$ is an upper triangular matrix, $0 \in \mathbb{R}^{(n-m) \times m}$ is a zero matrix, $Q_1 \in \mathbb{R}^{n \times m}$, $Q_2 \in \mathbb{R}^{n \times (n-m)}$, and Q_1 and Q_2 both have orthogonal columns.

2.2.6 Cholesky Factorization

When $A \in \mathbb{R}^{n \times n}$ is symmetric positive definite, we have

$$A = LL^T,$$

where $L \in \mathbb{R}^{n \times n}$ is a lower triangular matrix.

A closely related variant of the classical Cholesky decomposition is the LDL decomposition,

$$A = LDL^T,$$

where $L \in \mathbb{R}^{n \times n}$ is a lower triangular matrix and $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix. This decomposition is related to the classical Cholesky decomposition, of the form LL^T

$$A = LDL^T = LD^{\frac{1}{2}} D^{\frac{1}{2}T} L^T = (LD^{\frac{1}{2}})(LD^{\frac{1}{2}})^T.$$

2.3 Basic Optimization Theory

Now that we have reviewed some knowledge of (numerical) linear algebra, we introduce some basic optimization theory focusing on unconstrained optimization. Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \ f(x). \tag{2.6}$$

In this chapter, we discuss situations when $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable (i.e., $f \in \mathcal{C}$), or convex (with, perhaps, $f \notin \mathcal{C}$) and f is bounded below.

2.3.1 Global and Local Minima

Ideal minima are those that minimize a function globally over its domain.

Definition 2.3.1 (Global minimum)**.** *A vector x_* is a global minimum of f if*

$$f(x_*) \leq f(x) \quad \textit{for all} \quad x \in \mathbb{R}^n.$$

Commonly, however, we are satisfied with a weaker form of minimum.

Definition 2.3.2 (Local minimum)**.** *A vector x_* is a local minimum of f if there exists $\epsilon > 0$ such that*

$$f(x_*) \leq f(x) \quad \textit{for all} \quad x \in \mathbb{B}(x_*, \epsilon) := \{x \in \mathbb{R}^n : \|x - x_*\|_2 \leq \epsilon\}.$$

We also characterize certain types of global and/or local minima:

- x_* is a strict global/local minimizer if the inequality holds strictly for $x \neq x_*$.
- x_* is an isolated global/local minimizer if, for some $\epsilon' > 0$, it is the only local minimizer in the neighborhood $\mathbb{B}(x_*, \epsilon')$.

An isolated minimum is a strict minimum, but (typically only for some pathological examples) the reverse is not always true.

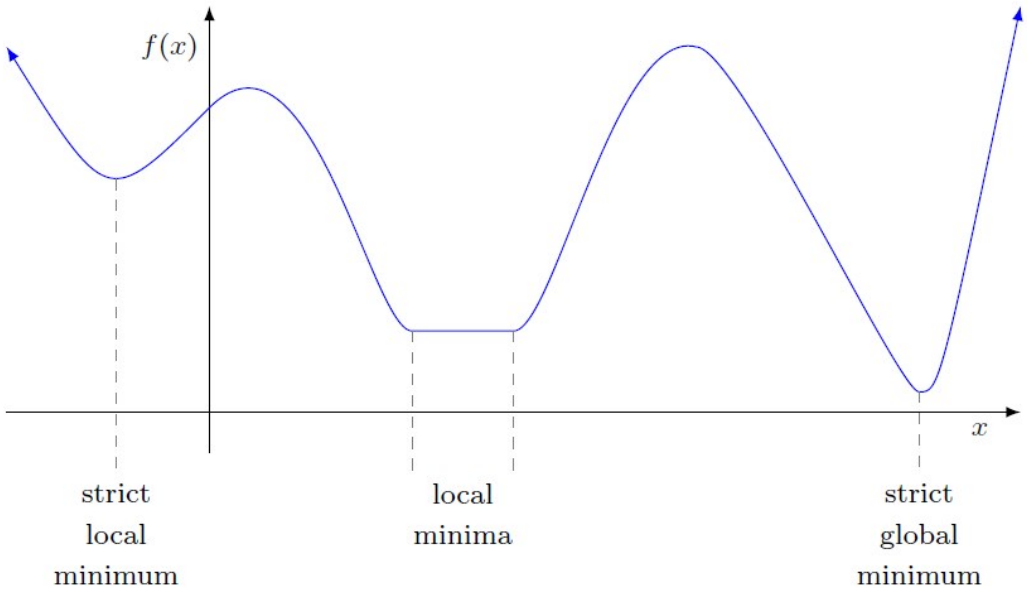

Figure 2.1: Global and local minima

A special fact in convex optimization is that all local minima are global minima.

Theorem 2.3.3. *If $f : \mathbb{R}^n \to \mathbb{R}$ is convex, then a local minimum of f is a global minimum of f. If f is strictly convex, then there exists at most one global minimum of f.*

Unfortunately, for nonconvex optimization, the conditions in the definitions of global and local minima are not entirely useful, we rarely have global information about f, and so have no way to verify if a point is a global minimizer. Thus, in nonconvex optimization, we often focus on finding a local minimizer. Using calculus, we can derive local optimality conditions that aid in determining if a point is a local minimizer.

2.3.2 Optimality Condition

Theorem 2.3.4 (First-order necessary condition)**.** *If $f \in \mathcal{C}$ and x_* is a local minimizer of f, then $\nabla f(x_*) = 0$.*

We can limit our search to points where $\nabla f(x_*) = 0$. However, $\nabla f(x_*) = 0$ does not imply that we have a local minimizer.

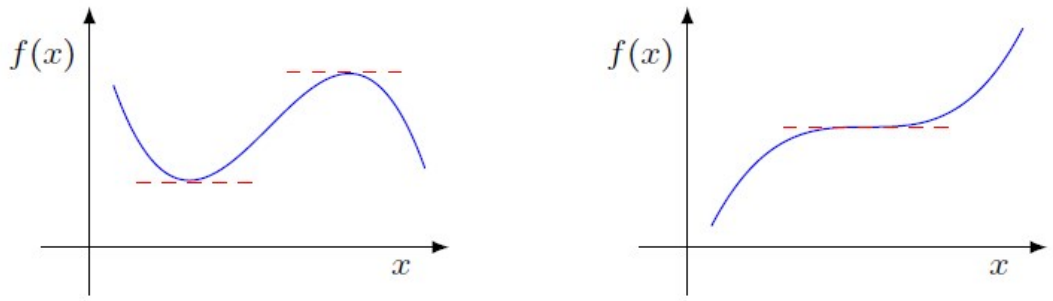

Figure 2.2: Stationary Point

Definition 2.3.5 (Stationary point)**.** *A point $x \in \mathbb{R}^n$ is a stationary point for $f \in \mathcal{C}$ if* $\nabla f(x) = 0$.

2.3.3 Rates of Convergence

There are two types of rates of convergence of sequences, one prefixed by Q (for "quotient") and the other prefixed by R (for "root"). Let $\{x_k\} \subseteq \mathbb{R}^n$ be a sequence that converges to $\{x_*\} \subseteq \mathbb{R}^n$ and let $||\cdot||$ be a vector norm on $\mathbb{R}^n$. There are different rates of convergence.

- The convergence of $\{x_k\}$ to x_* is Q-sublinear if

$$\lim_{k\to\infty} = \frac{||x_{k+1} - x_*||}{||x_k - x_*||} = 1.$$

- The convergence of $\{x_k\}$ to x_* is Q-linear if there exists $c \in (0, 1)$ such that

$$\frac{||x_{k+1} - x_*||}{||x_k - x_*||} \leq c,$$

  for all sufficiently large k. The constant c indicates the rate of linear convergence.

- The convergence of $\{x_k\}$ to x_* is Q-superlinear if

$$\lim_{k\to\infty} = \frac{||x_{k+1} - x_*||}{||x_k - x_*||} = 0.$$

  It is easily seen that any sequence that converges Q-superlinearly also converges Q-linearly.

We now distinguish between various types of Q-superlinear convergence.

- The convergence of $\{x_k\}$ to x_* is Q-quadratic if there exists $c > 0$ such that

$$\frac{||x_{k+1} - x_*||}{||x_k - x_*||^2} \leq c,$$

  for all sufficiently large k.

- The Q-order of convergence of $\{x_k\}$ to x_* is $p > 1$ if there exists $c > 0$ such that

$$\frac{||x_{k+1} - x_*||}{||x_k - x_*||^p} \leq c,$$

  for all sufficiently large k. This leads to the definitions of Q-cubic ($p = 3$) and Q-quartic ($p = 4$) convergence.

The above definitions only focus on the sequences decreasing monotonically, however, they do not cover the situation when certain sequences converge reasonably quickly but approach the limit point nonmonotonically. For such cases, we define a slightly weaker form of convergence known as R-order convergence. If $\{\epsilon_k\}$ converges to 0 and

$$||x_k - x_*|| \leq \epsilon_k \text{ for all } k, \tag{2.7}$$

then the R-order of convergence of $\{x_k\}$ to x_* is said to be that of the Q-order of convergence of $\{\epsilon_k\}$ to 0. If $\{\epsilon_k\}$ converges Q-sublinearly to 0 and (2.7) holds, then $\{x_k\}$ converges R-sublinearly to x_*; if $\{\epsilon_k\}$ converges Q-linearly to 0 and (2.7) holds, $\{x_k\}$ converges R-linearly to x_*; and so on.

2.3.4 Basic Iterations

Typically, we apply iterative methods for solving optimization problems, where the basic iterations have the form

$$x_{k+1} = x_k + \alpha_k d_k. \tag{2.8}$$

In (2.8), x_k is the current iteration point (at kth iteration), in order to reach the next iteration point x_{k+1} (at $(k+1)$th iteration), we need two quantities:

- Search direction d_k, which direction to go.
- Stepsize α_k, how far to go.

There are various ways of selecting each search direction and stepsize. For the choice of search direction, we mainly focus on negative gradient direction, where

$$d_k = -g_k. \tag{2.9}$$

More will be discussed in Chapter 2.4. And for the computation of stepsize, we will introduce two well-known methods in Chapter 2.5 and 2.9. For the purpose of guaranteeing algorithm convergence, sometimes a line search is also carried out to modify the original stepsize at each iteration, which will be explained in detail in Chapter 2.7.

We can also define $s_{k+1} := \alpha_k d_k$ as the step and thus

$$x_{k+1} = x_k + s_{k+1}, \tag{2.10}$$

this notation is frequently used in Chapter 2.5.

2.4 Steepest Descent Methods

Following the discussion of some basic optimization theory, we will introduce one of the simplest and most widely used techniques for solving unconstrained nonlinear optimization problems, which is steepest descent (SD).

2.4.1 Introduction

First, let us consider unconstrained optimization problem 2.6. The simplest gradient-based method for solving this problem is a steepest descent method, which is the term we

use to describe any iterative method of the form

$$x_{k+1} \leftarrow x_k - \alpha_k g_k \quad \text{for all} \quad k \in \mathbb{N}. \tag{2.11}$$

Here, $x_0 \in \mathbb{R}^n$ is a given initial point and, for all $k \in \mathbb{N}$, the scalar $\alpha_k > 0$ is the kth stepsize. In the classical steepest descent method of Cauchy, each stepsize is obtained by an exact line search (see [10]), i.e., assuming f is bounded below along the ray from x_k along $-g_k$, one sets

$$\alpha_k \in \arg\min_{\alpha \geq 0} f(x_k - \alpha g_k).$$

However, in modern variants of steepest descent, alternative stepsizes that are cheaper to compute are employed to reduce per-iteration (and typically overall) computational costs.

Algorithm 2 Steepest Descent Framework

1: input $x_0 \in \mathbb{R}^n$
2: **for** $k = 0, 1, 2 \cdots$ **do**
3: compute $g_k \leftarrow \nabla f(x_k)$
4: choose $\alpha_k \in (0, \infty)$
5: set $x_{k+1} \leftarrow x_k - \alpha_k g_k$
6: **end for**

2.4.2 Krylov Sequence Generated from Steepest Descent Methods

Here is a property related to Krylov sequences. Given a strongly convex quadratic function with SPD matrix A, suppose we generate the consecutive iteration sequence $\{x_{k-m}, x_{k-m+1}, \cdots, x_{k-1}\}$ by steepest descent methods and store the corresponding gradients in a matrix G, where

$$G = [g_{k-m}, g_{k-m+1}, \cdots, g_{k-1}].$$

At x_k, the displacement of the current iterate x_k from any back value x_{k-m} lies in the span of the Krylov sequence initiated from g_{k-m}, where

$$x_k - x_{k-m} = \text{ span } \{g_{k-m}, Ag_{k-m}, A^2 g_{k-m}, \cdots, A^{m-1} g_{k-m}\}.$$

Followed by this, multiplying by A and noting that $g_k = Ax_k - b$, we also have

$$g_k - g_{k-m} = \text{ span } \{Ag_{k-m}, A^2 g_{k-m}, A^3 g_{k-m}, \cdots, A^m g_{k-m}\}.$$

2.5 Barzilai-Borwein Methods

The "two-point stepsize" method proposed by Barzilai and Borwein has two variants, which differ only in the formulas used to compute the stepsizes. We derive these formulas simultaneously now for reference. During iteration $k \in \mathbb{N}_+ := \{1, 2, \dots\}$, defining the displacement vectors

$$s_k := x_k - x_{k-1} \quad \text{and} \quad y_k := g_k - g_{k-1}, \tag{2.12}$$

the classical secant equation is given by $H_k s_k = y_k$ where H_k represents an approximation of the Hessian of f at x_k. In quasi-Newton methods of the Broyden class (such as the BFGS method (see [8, 21, 24, 47])), a Hessian approximation $H_k \succ 0$ is chosen such that the secant equation is satisfied and the kth search direction is set as $-H_k^{-1} g_k$ (see [41]). However, the key idea in BB methods is to maintain a steepest descent framework by approximating the Hessian by a scalar multiple of the identity matrix in such a way that the secant equation is only satisfied in a least-squares sense. In particular, consider

$$\min_{\bar{q} \in \mathbb{R}} \tfrac{1}{2} \|(\bar{q} I) s_k - y_k\|_2^2 \quad \text{and} \quad \min_{\hat{q} \in \mathbb{R}} \tfrac{1}{2} \|s_k - (\hat{q}^{-1} I) y_k\|_2^2.$$

Assuming that $s_k^T y_k > 0$ (which is guaranteed, e.g., when $s_k \neq 0$ and f is strictly convex), the solutions of these one-dimensional problems are, respectively,

$$\bar{q}_k := \frac{s_k^T y_k}{s_k^T s_k} \quad \text{and} \quad \hat{q}_k := \frac{y_k^T y_k}{s_k^T y_k}. \tag{2.13}$$

That is, $\bar{q}_k I$ and $\hat{q}_k I$ represent simple approximations of the Hessian of f along the line segment $[x_{k-1}, x_k]$, meaning that if one minimizes the quadratic model of f at x_k along $-g_k$ given by

$$f(x_k - \alpha g_k) \approx f_k - \alpha \|g_k\|_2^2 + \tfrac{1}{2}\alpha^2 q_k \|g_k\|_2^2,$$

respectively for $q_k = \bar{q}_k$ and $q_k = \hat{q}_k$, then one obtains two potential values for the stepsize α_k, namely

$$\bar{\alpha}_k := \frac{s_k^T s_k}{s_k^T y_k} \quad \text{and} \quad \hat{\alpha}_k := \frac{s_k^T y_k}{y_k^T y_k}. \tag{2.14}$$

(Further discussion on the difference between these stepsizes and their corresponding Hessian approximations is given in Chapter 4.3.1.) Overall, the main idea in such an approach is to employ a two-point approximation to the secant equation in order to construct a simple approximation of the Hessian of f at x_k, which in turn leads to a quadratic model of f at x_k that can be minimized to determine the stepsize α_k.

2.6 Convergence Results

BB methods and enhancements to them have been a subject of research for over two decades. In their original work (see [2]), Barzilai and Borwein proved that either of their two stepsize choices leads to global convergence and an R-superlinear local convergence rate when (2.11) is applied to minimize a two-dimensional strictly convex quadratic. Raydan (see [44]) extended these results to prove that such methods are globally convergent when applied to minimize any finite-dimensional strictly convex quadratic. Dai and Liao (see [15]) also extended these results to show that, on such problems, BB methods attain an R-linear rate of convergence. In general, however, in order to have a globally convergent algorithm for solving general nonlinear objective functions, one typically employs a line search approach. We consider such methods in Chapter 2.7.

2.7 Line Search Techniques

An interesting feature of BB methods, even when applied to minimize strictly convex quadratics, is that they are not guaranteed to yield monotonic decreases in the objective function or a stationarity measure for problem (2.6). That is, when they converge to a minimizer of f, neither the sequence of function values $\{f_k\}$ nor the sequence of gradient norms $\{\|g_k\|\}$ is guaranteed to decrease monotonically. Hence, a variety of extensions of the original BB methods have been designed that ensure convergence when minimizing general continuously differentiable objective functions by incorporating well-established line search techniques.

2.7.1 Monotone Line Search

A line search is a common strategy for ensuring global convergence for an optimization algorithm. For example, in our context of solving problem (2.6), a well-known line search strategy is to compute α_k satisfying the following pair of so-called Wolfe conditions; here, $g : \mathbb{R}^n \to \mathbb{R}^n$ is the gradient function of f, d_k represents the search direction in each iteration, and γ and ξ are user-specified parameters:

$$\begin{aligned} f(x_k + \alpha_k d_k) &\leq f_k + \gamma \alpha_k g_k^T d_k, \qquad && \gamma \in (0, 1), \\ g(x_k + \alpha_k d_k)^T d_k &\geq \xi g_k^T d_k, && \xi \in (\gamma, 1). \end{aligned}$$

Another common approach is to use the first of these two conditions, known as the sufficient decrease or Armijo condition, in the context of a backtracking line search.

2.7.2 Nonmonotone Line Search

In the context of BB methods, one typically tries to avoid having to modify the BB stepsizes by employing a nonmonotone line search such as the one proposed by Grippo, Lampariello, and Lucidi (see [29]). The condition used in this strategy has the form

$$f(x_k + \alpha_k d_k) \leq \max_{0 \leq j \leq m} f(x_{k-j}) + \gamma \alpha_k g_k^T d_k, \qquad \gamma \in (0, 1),$$

where m is a nonnegative integer. Clearly, this condition has a similar form as the Armijo condition, except that the function value at the current iterate is replaced by the largest function value at the most recent m iterates. By employing such a condition, the objective function values are not forced to decrease monotonically, though it can be shown that the sequence of function values defined by the most recent m values does decrease monotonically, which can be used to ensure global convergence.

Raydan (see [45]) proposed a globally convergent BB method (GBB) using this nonmonotone line search for general nonlinear objecive function and showed some significant reduction in the number of line searches and also in the number of gradient evaluations.

More recently, another nonmonotone line search was proposed by Zhang and Hager (see [56]), which requires decreases in a moving average of successive function values. In this approach, the stepsize α_k is chosen to satisfy the condition

$$f(x_k + \alpha_k d_k) \leq C_k + \gamma \alpha_k g_k^T d_k, \qquad \gamma \in (0, 1),$$

where C_k is defined by the recursions

$$\begin{aligned} Q_{k+1} &= \eta_k Q_k + 1, \quad \text{with } Q_0 = 1, \\ C_{k+1} &= \frac{\eta_k Q_k C_k + f(x_{k+1})}{Q_{k+1}}, \quad \text{with } C_0 = f(x_0), \end{aligned}$$

where $\eta_k \in [\eta_{min}, \eta_{max}] \subset [0, \infty)$ is a parameter that controls the degree of nonmonotonicity. If $\eta_k = 0$ for each k, then the line search is the usual monotone Armijo line search. If $\eta_k = 1$ for each k, then $C_k = A_k$, where $A_k = \frac{1}{k+1} \sum_{i=0}^{k} f(x_i)$.

With this nonmonotone line search and some suitable assumptions, we can have global convergence results as described in Theorem 2.2 of [56].

Theorem 2.7.1. *Suppose $f(x)$ is bounded from below and the searching direction d_k at kth iteration satisfies*

$$g_k^T d_k \leq -c_1 ||g_k||^2,$$

$$||d_k|| \leq c_2 ||g_k||,$$

where c_1 and c_2 are positive constants. Moreover, if the Wolfe conditions are used, we

assume that ∇f is Lipschitz continuous, with Lipschitz constant L, on the level set

$$\mathcal{L} = \{x \in \mathbb{R}^n : \ f(x) \leq f(x_0)\}.$$

Let $\overline{\mathcal{L}}$ denote the collection of $x \in \mathbb{R}^n$ whose distance to $\mathcal{L}$ is at most $\mu d_{\max}$, $(\mu > 0)$, where $d_{\max} = \sup_k ||d_k||$. If the Armijo conditions are used, we assume that ∇f is Lipschitz continuous, with Lipschitz constant L, on $\overline{\mathcal{L}}$. Then the iteates x_k by the nonmonotone line search algorithm have the property that

$$\liminf_{k\to\infty} \|g_k\|_2 = 0.$$

Moreover, if $\eta_{max} \in [0, 1)$, then

$$\lim_{k\to\infty} \|g_k\|_2 = 0.$$

Hence, every convergent subsequence of the iterates approaches a point x_*, where

$$\nabla f(x_*) = 0.$$

2.7.3 Summary

Overall, a typical line search strategy involves a condition of the form

$$f(x_k + \alpha_k d_k) \leq f_k^r + \gamma \alpha_k g_k^T d_k, \qquad \gamma \in (0, 1),$$

where f_k^r is a reference function value. If $f_k^r = +\infty$, there is no line search. If $f_k^r = f_k$, then the above reduces to the condition in a (monotone) Armijo line search. The two nonmonotone line search strategies above can be derived by setting $f_k^r = \max_{0\leq j\leq m} f(x_{k-j})$ and $f_k^r = C_k$, respectively.

2.8 Barzilai-Borwein Methods Variants

In BB methods, Barzilai and Borwein proposed two quadratic models for convex quadratic functions and generated two types of stepsizes. These ideas have been extended in various ways with the goal of producing better stepsizes.

2.8.1 Alternative Step (AS) Gradient Method

Dai (see [13]) proposed some approaches alternatively using exact line searches (i.e., Cauchy stepsizes) and BB stepsizes in the iterative sequence known as alternate step (AS) gradient method to solve convex quadratic problems. The motivation is that when minimizing convex quadratics, the SD method produces zigzags and the zigzagging phenomenon will not occur if one of the two SD steps is replaced with a BB step. This motivates the consideration of the gradient method that chooses its stepsize as follows:

$$\alpha_k = \begin{cases} \alpha_k^{SD}, & \text{for odd } k, \\ \alpha_k^{BB}, & \text{for even } k. \end{cases}$$

For convex quadratic case, we have

$$\begin{aligned} \alpha_k^{SD} =& \frac{g_k^T g_k}{g_k^T A g_k}, \\ \alpha_k^{BB} =& \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}} = \frac{g_{k-1}^T g_{k-1}}{g_{k-1}^T A g_{k-1}}, \end{aligned}$$

from which we can see that

$$\alpha_{k-1}^{SD} = \alpha_k^{BB}.$$

We can also consider

$$\alpha_{mk+i} = \begin{cases} \alpha_{mk+i}^{SD}, & \text{for } i = 1, \cdots, m-1, \\ \alpha_{mk+m}^{BB}, & \text{for } i = m, \end{cases}$$

where $m \geq 1$ and obviously the previous one is a special case with $m = 2$.

The theoretical and numerical analyses show that the AS method is a promising alternative or even better than the BB method. A two-step $(3-\epsilon)$-Q-superlinear convergence result $||g_{k+2}|| = O(||g_k||^{3-\epsilon})$ is established for $n = 2$ and an R-linear convergence result for the general case.

2.8.2 Cyclic BB (CBB) Method

Dai (see [13]) and Dai et al. (see [14]) also proposed the method of repeatedly using one BB stepsize in the next m (history length) steps, which is known as cyclic BB (CBB) method for solving unconstrained optimization problems. The motivation for the cyclic BB method comes from the superior performance of cyclic SD compared to the ordinary SD, where in cyclic SD, we have

$$\alpha_{mk+i} = \alpha^{SD}_{mk+1}, \text{ for } i = 1, \cdots, m.$$

In fact, the AS method described in Chapter 2.8.1 is also a special case of cyclic SD with $m = 2$, since for convex quadratic function, the BB step at iteration k is the SD step at iteration $k-1$.

Similarly, in cyclic BB method, we have

$$\alpha_{mk+i} = \alpha^{BB}_{mk+1}, \text{ for } i = 1, \cdots, m.$$

CBB method is locally linearly convergent at a local minimizer with positive definite Hessian. Numerical evidence in Dai et al. (see [14]) indicates that when $m > n/2 \geq 3$, where n is the problem dimension, CBB method is locally superlinearly convergent. In the special case $m = 3$ and $n = 2$, it is proved that the convergence rate is no better than linear.

The reason that the CBB approach is better than the BB gradient algorithm is that it requires less computation. In addition, like in the conjugate gradient method, there is also a quadratic termination result in gradient methods, for convex quadratic function with Hessian matrix A, if we have stepsize $\{\alpha_1^{-1}, \cdots, \alpha_n^{-1}\} = \{\lambda_1, \cdots, \lambda_n\}$, which is the

set of all the eigenvalues of A, the steepest descent method gives the exact solution in at most $n+1$ iterations. We will prove such a finite termination result in the next chapter.

From the numerical experiments in Dai (see [13]) and Dai et al. (see [14]), the stepsizes generated by the cyclic methods are usually closer to the inverse eigenvalues of A than those by the BB method.

Theoretically, under suitable assumptions, the Rayleigh quotient

$$\frac{g_{k+l}^T A g_{k+l}}{g_{k+l}^T g_{k+l}},$$

generated by the gradient method with constant stepsizes

$$x_{k+l} = x_{k+l-1} - \alpha g_{k+l-1}, \quad l = 1, 2, \cdots,$$

converges to some eigenvalue of the matrix A if $l \to \infty$. Thus, it is reasonable to assume that repeated use of a BB stepsize leads to good approximations of eigenvalues of A.

An implementation of the CBB method combines a non-monotone line search and an adaptive choice for the cycle length m performs better than the existing BB gradient algorithm, while it is competitive with the well-known conjugate gradient algorithm.

2.8.3 Alternate Minimization (AM) Method

Dai and Yuan (see [16]) proposed alternate minimization method whose stepsizes alternately minimize the function value and the gradient norm along the line of steepest descent for convex quadratic function, where

$$\alpha_k^{AM} = \begin{cases} \alpha_k^{MG} = \frac{g_k^T A g_k}{g_k^T A^2 g_k}, & \text{for odd k,} \\ \alpha_k^{SD} = \frac{g_k^T g_k}{g_k^T A g_k}, & \text{for even k.} \end{cases}$$

The motivation of this method is that suppose the dimension of the problem is very large, so each algorithm can compute only a few iterations. Then one should prefer a monotonic algorithm to a nonmonotonic one (like BB method) since the objective is to minimize the

function. Therefore the question whether there also exists a monotonic gradient algorithm that is much faster than the SD method rises and one basic idea is to alternately minimize the gradient norm and the function value.

Dai et al. (see [57]) also modified this AM method as

$$\alpha_k = \begin{cases} \alpha_k^{MG}, & \text{if } \frac{\alpha_k^{MG}}{\alpha_k^{SD}} > \kappa, \\ \alpha_k^{SD}, & \text{otherwise,} \end{cases}$$

where $\kappa \in (0, 1)$ and close to 0.5.

For convex quadratics, the AM method is proved to be Q-superlinearly convergent in two dimensions, and Q-linearly convergent in any dimension. Numerical results suggest that the AM method is much better than the SD method and comparable with the BB and AS methods. It can also be extended to unconstrained optimization with a line search.

2.8.4 Adaptive BB (ABB) Method

Dai et al. (see [57]) and Yuan (see [55]) also proposed the alternative BB method which alternatively uses two BB stepsizes for convex quadratic function.

$$\alpha_k = \begin{cases} \alpha_k^{BB2}, & \text{if } \frac{\alpha_k^{BB2}}{\alpha_k^{BB1}} < \kappa, \\ \alpha_k^{BB1}, & \text{otherwise,} \end{cases}$$

where $\kappa \in (0, 1)$. Obviously we have

$$\alpha_k^{BB1} = \alpha_{k-1}^{SD},$$
$$\alpha_k^{BB2} = \alpha_{k-1}^{MG}.$$

Here is the intuition of this method, when we have

$$\frac{\alpha_k^{BB2}}{\alpha_k^{BB1}} = \frac{\alpha_{k-1}^{MG}}{\alpha_{k-1}^{SD}} \approx 0,$$

the MG methods performs poorly at point x_{k-1} and there is little reduction in $||g(x)||_2$, then choose the smaller step-size α_k^{BB2}, othwewise, choose the larger step-size α_k^{BB1}. This is also similar to the trust-region approach, which uses a somewhat similar strategy while choosing the trust-region radius.

For convex quadratics, the ABB method is proved to be R-linearly convergent. Numerical results suggest that the ABB method is comparable and in general preferable to the BB, AS, and AM methods. Particularly, the ABB method is a good option if the coefficient matrix is very ill-conditioned and a high precision is required. And it outperforms the linear CG method when a low precision is required.

This method itself requires no line searches for general functions and therefore might be able to save a lot of computational work while solving unconstrained optimization problems. To ensure global convergence, we could combine the ABB method with the non-monotone line search.

2.8.5 Cubic Interpolation Model

Dai, Yuan, and Yuan (see [12]) employed higher-order models and proposed interpolation techniques to derive a few alternative stepsizes; they use interpolation to recover the original BB stepsizes and employ a cubic model to derive alternatives. Defining the model

$$f(x_{k+1} - \alpha s_k) \approx m_{k+1}(\alpha) := f_{k+1} - \alpha g_{k+1}^T s_k + \tfrac{1}{2} q_k \alpha^2 \|s_k\|_2^2 - \tfrac{1}{6} c_k \alpha^3 \|s_k\|_2^3,$$

they automatically have

$$m_{k+1}(0) = f_{k+1} \quad \text{and} \quad \nabla m_{k+1}(0) = g_{k+1}^T s_k,$$

and, in addition, consider the potential interpolation conditions

$$m_{k+1}(1) = f_k \quad \text{and} \quad \nabla m_{k+1}(1) = g_k^T s_k.$$

Enforcing one, the other, or both of these conditions to determine the pair (q_k, c_k), Dai, Yuan, and Yuan recovered one of the basic BB stepsizes along with two alternative stepsize options:

$$\alpha_k = s_{k-1}^T s_{k-1}/[2(f_{k-1} - f_k + g_k^T s_{k-1})],$$
$$\alpha_k = s_{k-1}^T s_{k-1}/[6(f_{k-1} - f_k) + 4g_k^T s_{k-1} + 2g_{k-1}^T s_{k-1}].$$

Their methods are also globalized by the line search of Grippo *et al.*. Numerical results suggest they require fewer numbers of function and gradient evaluations and perform better compared with some known algorithms.

2.8.6 Modified Secant Equation Model

There has also been recent work by Xiao, Wang, and Wang (see [53]) that proposed alternative stepsizes using an alternative secant equation motivated by better approximating $B_k s_k$. They use the formula

$$B_k s_{k-1} = \tilde{y}_{k-1} = y_{k-1} + \tilde{\gamma} s_{k-1},$$
$$\tilde{\gamma} = \frac{3(g_k + g_{k-1})^T s_{k-1} + 6(f_{k-1} - f_k)}{||s_{k-1}||^2}.$$

and

$$B_k s_{k-1} = \overline{y}_{k-1} = y_{k-1} + \overline{\gamma} s_{k-1},$$
$$\overline{\gamma} = \frac{(g_k + g_{k-1})^T s_{k-1} + 2(f_{k-1} - f_k)}{||s_{k-1}||^2}.$$

They derived two new stepsize formulae

$$\tilde{\alpha}_k = \frac{s_{k-1}^T \tilde{y}_{k-1}}{\tilde{y}_{k-1}^T \tilde{y}_{k-1}},$$
$$\overline{\alpha}_k = \frac{s_{k-1}^T \overline{y}_{k-1}}{\overline{y}_{k-1}^T \overline{y}_{k-1}}.$$

Together with nonmonotone line search by Zhang and Hager (see [56]), they showed these proposed methods are globally convergent as well as efficient numerical results.

2.8.7 Alternative Modified Secant Equation Model

There is recent work by Kafaki and Fatemi (see [33]) that modifies a BB stepsize using a similar strategy as the modified BFGS method proposed by Li and Fukushima (see [35]) for general nonlinear functions. In particular, they modify the BB stepsize $\alpha_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}$ when there is a negative curvature ($s_{k-1}^T y_{k-1} < 0$). The method is:

$$B_k s_{k-1} = \overline{y}_{k-1}, \quad \overline{y}_{k-1} = y_{k-1} + h_{k-1}||g_{k-1}||^r s_{k-1},$$

where we have

$$r > 0 \text{ and } h_{k-1} = C + \max\{-\frac{s_{k_1}^T y_{k-1}}{\|s_{k-1}\|^2}, 0\}\|g_{k-1}\|^{-r},$$

with some constant C.

We observe that $s_{k-1}^T \overline{y}_{k-1} > 0$ regardless of the sign of $s_{k-1}^T y_{k-1}$. This method solves the negative curvature issue by using the modified BB step

$$\alpha_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T \overline{y}_{k-1}}.$$

Together with nonmonotone line search, it is globally convergent and the numerical experiments illustrate good performances.

2.8.8 Other Stepsizes and Models

Besides the above stepsizes and models, Yuan (see [54]) proposed the incorporation of Cauchy stepsizes into the iterative process to improve the efficiency of the algorithm, a technique later extended by De Asmundis, Serafino, and Toraldo (see [17]), motivated by work in [18] with their collaborator Riccio in a monotone gradient scheme. There has also been work by Biglari and Solimanpur (see [5]) that proposes alternative stepsizes derived by fourth-order interpolation models. These articles employ the nonmonotone line search of Zhang and Hager.

2.9 Fletcher's Limited Memory Steepest Descent (LMSD) Method

The foundation of Fletcher's method is a strategy for attaining finite convergence when minimizing a convex quadratic function. Fletcher's LMSD method represents an alterative to the original BB methods that is entirely different than those described in Chapter 2.5. Rather than attempt to compute a better stepsize based on information that can be extracted only from the previous iteration, his approach involves the storage and exploitation of information from m previous iterations, with which a sequence of m stepsizes—to be employed in the subsequent m iterations—are computed. To be more precise, consider iteration $k \geq m$ for some user-specified integer parameter $m \geq 1$ and suppose that a matrix of gradients (computed at previous iterates), namely

$$G_k := \begin{bmatrix} g_{k-m} & \cdots & g_{k-1} \end{bmatrix}, \tag{2.15}$$

is available. The key idea underlying Fletcher's proposed method is that, in the case of minimizing the quadratic function $\frac{1}{2}x^T Ax$ for some $A \succ 0$, a reasonable set of stepsizes can be obtained by computing the reciprocals of the eigenvalues of the symmetric tridiagonal matrix

$$T_k := Q_k^T A Q_k,$$

where Q_k is the orthogonal matrix obtained in the (thin) QR-factorization of the matrix G_k (see [26, Theorem 5.2.2]). In fact, if one has $m = n$, then choosing stepsizes in this manner leads to finite termination of the algorithm in n steps. We formalize this result as the following theorem.

Theorem 2.9.1 (Finite Termination of an LMSD method for minimizing $\frac{1}{2}x^T Ax$ from an arbitrary starting point with $m = n$)**.** *Suppose the matrix A has n distinct eigenvalues*

$$0 < \lambda_1 < \lambda_2 < \cdots < \lambda_n.$$

If $\alpha_{k-1} \leftarrow \lambda_i^{-1}$ for $i \in \{1, 2, \ldots n\}$, then $g_n = 0$.

Furthermore, the matrix T_k can be obtained without access to the matrix A, such as through the partially extended Cholesky factorization $G_k^T[G_k \; g_k] = R_k^T[R_k \; r_k]$ to obtain

$$T_k = \begin{bmatrix} R_k & r_k \end{bmatrix} J_k R_k^{-1}, \tag{2.16}$$

where R_k is the upper triangular matrix obtained in the (thin) QR-factorization of G_k (meaning that it is the upper triangular Cholesky factor of $G_k^T G_k$ (see [26, Theorem 5.2.2])) and

$$J_k := \begin{bmatrix} \alpha_{k-m}^{-1} & & \\ -\alpha_{k-m}^{-1} & \ddots & \\ & \ddots & \alpha_{k-1}^{-1} \\ & & -\alpha_{k-1}^{-1} \end{bmatrix}. \tag{2.17}$$

Exploiting this latter representation, Fletcher extends his approach to the minimization of general objective functions. In particular, by storing G_k and computing T_k in a manner similar to (2.16), he outlines a "Ritz Sweep" algorithm that, in his experiments, performs as well as the well-known L-BFGS method (see Nocedal [40]). In this extension to a more general setting (i.e., nonquadratic functions), Fletcher incorporates line searches and other features to overcome certain issues that may arise and to promote convergence. Some of his procedures are discussed below, but the reader should refer to his article for a more complete discussion.

It should be noted that in the case of minimizing a strictly convex quadratic and with $m = 1$, the formula (2.16) yields $T_k = \bar{q}_k$ (recall (2.13)), which reveals that choosing stepsizes as the reciprocals of the eigenvalues of T_k corresponds to the first BB alternative. Fletcher also remarks that a similar strategy can be designed corresponding to the second BB stepsize. In particular, defining

$$\begin{bmatrix} R_k & r_k \\ 0 & \rho_k \end{bmatrix}^T \begin{bmatrix} R_k & r_k \\ 0 & \rho_k \end{bmatrix} \quad \text{as the Cholesky factorization of} \quad \begin{bmatrix} G_k & g_k \end{bmatrix}^T \begin{bmatrix} G_k & g_k \end{bmatrix},$$

he defines the corresponding pentadiagonal matrix

$$P_k := R_k^{-T} J_k^T \begin{bmatrix} R_k & r_k \\ 0 & \rho_k \end{bmatrix}^T \begin{bmatrix} R_k & r_k \\ 0 & \rho_k \end{bmatrix} J_k R_k^{-1}. \tag{2.18}$$

He explains that, in the case of minimizing a strictly convex quadratic function, appropriate stepsizes are given by the eigenvalues of $P_k^{-1}T_k$; in particular, with $m = 1$, the formulas (2.16) and (2.18) yield $P_k^{-1}T_k = \hat{\alpha}_k$ (recall (2.14)). While he refers to the eigenvalues of T_k as Ritz values, he refers to the reciprocals of the eigenvalues of $P_k^{-1}T_k$ as harmonic Ritz values [42].

2.10 Stochastic Gradient (SG) Methods

2.10.1 Introduction

When it comes to optimization methods for machine learning, the goal is to minimize the sum of cost functions over samples from a finite training set, which has the form

$$\min_{x \in \mathbb{R}^n} F(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x), \tag{2.19}$$

where n is the sample size, and each $f_i : \mathbb{R}^d \to \mathbb{R}$ is the cost function corresponding to the i-th sample data. If the objective is prediction, a classical example is linear regression and the cost function is least squares

$$f_i(x) := (a_i^T x - b_i)^2,$$

where $a_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$ are the data samples associated with a linear regression problem. If the objective is classification, two important examples are logistic regression and support vector machine (SVM). As for logistic regression, the cost function has the form

$$f_i(x) := \log(1 + \exp(-b_i a_i^T x)).$$

While for squared hinge loss SVM, we can have the following as cost function

$$f_i(x) = ([1 - b_i a_i^T x]_+)^2,$$

where $a_i \in \mathbb{R}^d$ and $b_i \in \{-1, 1\}$ in logistic regression and SVM models are the data samples associated with a binary classification problem. Note that unlike Chapter 2.9, in Chapter 2.10, we use n to refer to sample size and d for problem size.

2.10.2 Stochastic Gradient (SG) Methods

The optimization methods for solving problem (2.19) fall into two broad categories as explained in [7]. We refer to them as batch and stochastic. The methods introduced previously (including BB methods, LMSD methods, etc.) all employ a full gradient during each iteration, which are considered as batch / full gradient methods, they use the iterations of the form

$$x_{k+1} \leftarrow x_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} \nabla f_i(x_k), \tag{2.20}$$

where α_k is a positive stepsize on iteration k.

A key challenge arising in (2.19) is that the number of data points n (also known as training examples) can be extremely large such that computing of $\nabla F(x)$ for given x can be very expensive. That triggers the methods which belong to the other category, i.e., stochastic gradient (SG) methods (prototype described in [46]). Stochastic gradient (SG) methods and their variants have been the main approaches for solving (2.19). The iterations of stochastic gradient (SG) methods are defined as

$$x_{k+1} \leftarrow x_k - \alpha_k \nabla f_{i_k}(x_k), \tag{2.21}$$

where i_k is randomly chosen from $\{1, \cdots, n\}$ and α_k is a positive stepsize on iteration k. Here is the framework of stochastic gradient (SG) methods.

Algorithm 3 Stochastic Gradient (SG) Framework

input $x_0 \in \mathbb{R}^n$

for $kin\mathbb{N}$ **do**

    randomly choose an index i_k from $\{1, \cdots, n\}$

    compute a stochastic gradient $\nabla f_{i_k}(x_k)$

    choose $\alpha_k \in (0, \infty)$

    set $x_{k+1} \leftarrow x_k - \alpha_k \nabla f_{i_k}(x_k)$

end for

The rate of convergence of a batch gradient method is faster than a basic stochastic method. If $F(x)$ is strongly convex, then the batch method iterates defined by (2.20) has R-linear convergence rate and the SG iterates defined by (2.21) satisfy the sublinear convergence property if each i_k is drawn uniformly from $\{1, \cdots, n\}$, while on the other side, the per-iteration cost of (2.20) is more expensive than the one of (2.21). We combine the best properties of both approaches. Instead of employing information from one sample point or whole sample points per iteration, one can employ a mini-batch approach in which a small subset of samples, call it $\mathcal{S}_k \subseteq \{1, \cdots, n\}$, is chosen randomly in each iteration, leading to

$$x_{k+1} \leftarrow x_k - \frac{\alpha_k}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} \nabla f_i(x_k). \tag{2.22}$$

Such an approach falls under the framework set out by [46].

As for stepsizes in stochastic gradient (SG) methods, people usually use a best-tuned fixed stepsize $\alpha_k = C$, or a diminishing stepsize, for example $\alpha_k = 1/(k+1)$. For a strongly convex function, SG with fixed stepsize will converge to a neighborhood of the optimal solution, SG with diminishing stepsize will converge to the optimal solution.

Chapter 3

R-Linear Convergence of Limited Memory Steepest Descent

The limited memory steepest descent method (LMSD) proposed by Fletcher is an extension of the Barzilai-Borwein "two-point stepsize" strategy for steepest descent methods for solving unconstrained optimization problems. It is known that the Barzilai-Borwein strategy yields a method with an R-linear rate of convergence when it is employed to minimize a strongly convex quadratic. This part extends this analysis for LMSD, also for strongly convex quadratics. In particular, it is shown that, under reasonable assumptions, the method is R-linearly convergent for any choice of the history length parameter. The results of numerical experiments are also provided to illustrate behaviors of the method that are revealed through the theoretical analysis.

3.1 Introduction

For solving unconstrained nonlinear optimization problems, one of the simplest and most widely used techniques is *steepest descent* (SD). This refers to any strategy in which, from any solution estimate, a productive step is obtained by moving some distance along the negative gradient of the objective function, i.e., the direction along which function descent is steepest.

While SD methods have been studied for over a century and employed in numerical software for decades, a unique and powerful instance came about relatively recently in the work by [2], where a "two-point stepsize" strategy is proposed and analyzed. The resulting SD method, commonly referred to as the BB method, represents an effective alternative to other SD methods that employ an exact or inexact line search when computing the stepsize in each iteration.

The theoretical properties of the BB method are now well-known when it is employed to minimize an n-dimensional strongly convex quadratic objective function. Such objective functions are interesting in their own right, but one can argue that such analyses also characterize the behavior of the method in the neighborhood of a strong local minimizer of any smooth objective function. In the original work (i.e., [2]), it is shown that the method converges R-superlinearly when $n = 2$. In [44], it is shown that the method converges from any starting point for any natural number n, and in [15] it is shown that the method converges R-linearly for any such n.

In each iteration of the BB method, the stepsize is determined by a computation involving the displacement in the gradient of the objective observed between the current iterate and the previous iterate. As shown in [22], this idea can be extended to a *limited memory steepest descent* (LMSD) method in which a *sequence* of m stepsizes is computed using the displacements in the gradient over the previous m steps. This extension can be motivated by the observation that these displacements lie in a Krylov subspace determined by a gradient previously computed in the algorithm, which in turn yields a computationally efficient strategy for computing m distinct eigenvalue estimates of the Hessian (i.e., matrix of second derivatives) of the objective function. The reciprocals of these eigenvalue estimates represent reasonable stepsize choices. Indeed, if the eigenvalues are computed exactly, then the algorithm terminates in a finite number of iterations; e.g., see [34], [22], and §3.2.

In [22], it is shown that the proposed LMSD method converges from any starting point when it is employed to minimize a strongly convex quadratic function. However, to the best of our knowledge, the convergence rate of the method for $m > 1$ has not yet been

analyzed. The main purpose of this chapter is to show that, under reasonable assumptions, this LMSD method converges R-linearly when employed to minimize such a function. Our analysis builds upon the analyses in [22] and [15].

We mention at the outset that numerical evidence has shown that the practical performance of the BB method is typically much better than known convergence proofs suggest; in particular, the empirical rate of convergence is often Q-linear with a contraction constant that is better than that observed for a basic SD method. Based on such evidence, we do not claim that the convergence results proved in this chapter fully capture the practical behavior of LMSD methods. To explore this claim, we present the results of numerical experiments that illustrate our convergence theory and demonstrate that the practical performance of LMSD can be even better than the theory suggests. We conclude with a discussion of possible explanations of why this is the case for LMSD, in particular by referencing a known finite termination result for a special (computationally expensive) variant of the algorithm.

Organization In §3.2, we formally state the problem of interest, notation to be used throughout the chapter, Fletcher's LMSD algorithm, and a finite termination property for it. In §3.3, we prove that the LMSD algorithm is R-linearly convergent for any history length. The theoretical results proved in §3.3 are demonstrated numerically in §3.4 and concluding remarks are presented in §3.6.

Notation The set of real numbers (i.e., scalars) is denoted as $\mathbb{R}$, the set of nonnegative real numbers is denoted as $\mathbb{R}_+$, the set of positive real numbers is denoted as $\mathbb{R}_{++}$, and the set of natural numbers is denoted as $\mathbb{N} := \{1, 2, \dots\}$. A natural number as a superscript is used to denote the vector-valued extension of any of these sets—e.g., the set of n-dimensional real vectors is denoted as $\mathbb{R}^n$—and a Cartesian product of natural numbers as a superscript is used to denote the matrix-valued extension of any of these sets—e.g., the set of $n \times n$ real matrices is denoted as $\mathbb{R}^{n\times n}$. A finite sequence of consecutive positive integers of the form $\{1, \dots, n\} \subset \mathbb{N}$ is denoted using the shorthand $[n]$. Subscripts are used to refer to a specific element of a sequence of quantities, either fixed or generated by

an algorithm. For any vector $v \in \mathbb{R}^n$, its Euclidean (i.e., ℓ_2) norm is denoted by $\|v\|$.

3.2 Fundamentals

In this Chapter, we state the optimization problem of interest along with corresponding definitions and concepts to which we will refer throughout the remainder of the chapter. We then state Fletcher's LMSD algorithm and prove a finite termination property for it, as is done in [34] and [22].

3.2.1 Problem Statement

Consider the problem to minimize a strongly convex quadratic function $f : \mathbb{R}^n \to \mathbb{R}$ defined by a symmetric positive definite matrix $A \in \mathbb{R}^{n \times n}$ and vector $b \in \mathbb{R}^n$, namely,

$$\min_{x \in \mathbb{R}^n} \; f(x), \quad \text{where} \quad f(x) = \tfrac{1}{2} x^T A x - b^T x. \tag{3.1}$$

Formally, we make the following assumption about the problem data.

Assumption 3.2.1. *The matrix A in problem (3.1) has $r \leq n$ distinct eigenvalues denoted by*

$$\lambda_{(r)} > \cdots > \lambda_{(1)} > 0. \tag{3.2}$$

Consequently, this matrix yields the eigendecomposition $A = Q \Lambda Q^T$, where

$$\begin{aligned} Q = \begin{bmatrix} q_1 & \cdots & q_n \end{bmatrix} \quad & \textit{is orthogonal} \\ \textit{and} \;\; \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \quad & \textit{with} \;\; \lambda_n \geq \cdots \geq \lambda_1 > 0 \\ & \textit{and} \;\; \lambda_i \in \{\lambda_{(1)}, \ldots, \lambda_{(r)}\} \;\; \textit{for all} \;\; i \in [n]. \end{aligned} \tag{3.3}$$

The eigendecomposition of A defined in Assumption 3.2.1 plays a crucial role in our analysis. In particular, we will make extensive use of the fact that any gradient of the objective function computed in the algorithm, a vector in $\mathbb{R}^n$, can be written as a linear combination of the columns of the orthogonal matrix Q. This will allow us to analyze the behavior of the algorithm componentwise according to the weights in these linear

combinations corresponding to the sequence of computed objective gradients. Such a strategy has been employed in all of the aforementioned articles on BB and LMSD.

3.2.2 Limited memory steepest descent (LMSD) method

Fletcher's limited memory steepest descent method is stated as Algorithm LMSD. The iterate update in the algorithm is the standard update in an SD method: each subsequent iterate is obtained from the current iterate minus a multiple of the gradient of the objective function evaluated at the current iterate. With this update at its core, Algorithm LMSD operates in cycles. At $x_{k,1} \in \mathbb{R}^n$ representing the initial point of the kth cycle, a sequence of m positive stepsizes $\{\alpha_{k,j}\}_{j\in[m]}$ are selected to be employed in an inner cycle composed of m updates, the result of which is set as the initial point for cycle $k+1$.

Once such an inner cycle has been performed, the stepsizes to be employed in the next cycle are computed as the reciprocals of Ritz values of A, i.e., estimates of eigenvalues of A that are contained in the spectrum of A in a certain desirable sense (e.g., see Lemma 3.3.6 in §3.3.2). [22] describes how these can be obtained in one of three ways, all offering the same estimates (in exact arithmetic). The most intuitive definition is that, for cycle $k+1$, the estimates come as the eigenvalues of $T_k := Q_k^T A Q_k$, where $Q_k \in \mathbb{R}^{n\times m}$ satisfying $Q_k^T Q_k = I$ is defined by a thin QR factorization of the matrix of kth cycle gradients, i.e., for some upper triangular matrix $R_k \in \mathbb{R}^{m\times m}$, such a factorization satisfies the equation

$$Q_k R_k = G_k := \begin{bmatrix} g_{k,1} & \cdots & g_{k,m} \end{bmatrix}. \tag{3.4}$$

(For now, let us assume that G_k has linearly independent columns, in which case the matrix R_k in (3.4) is nonsingular. For a discussion of situations when this is not the case, see Remark 3.2.2 later on.) Practically, however, obtaining T_k in this manner requires multiplications with A as well as storage of the n-vectors composing the columns of Q_k. Both can be avoided in the following manner. First, it can be shown from the iterate update in Step 7 of Algorithm LMSD (e.g., see the proof of Lemma 3.2.3 in §3.2.3) that $g_{k,j+1} = g_{k,j} - \alpha_{k,j} A g_{k,j}$ for all $(k,j) \in \mathbb{N}\times[m]$. This means that, with the gradient at the initial point of cycle $k+1$, namely $g_{k+1,1} \equiv g_{k,m+1}$, and the matrix of kth-cycle reciprocal

stepsizes, namely

$$J_k \leftarrow \begin{bmatrix} \alpha_{k,1}^{-1} & & \\ -\alpha_{k,1}^{-1} & \ddots & \\ & \ddots & \alpha_{k,m}^{-1} \\ & & -\alpha_{k,m}^{-1} \end{bmatrix}, \tag{3.5}$$

one has $AG_k = \begin{bmatrix} G_k & g_{k,m+1} \end{bmatrix} J_k$, which in turn means that

$$G_k^T A G_k = G_k^T \begin{bmatrix} G_k & g_{k,m+1} \end{bmatrix} J_k. \tag{3.6}$$

Hence, by computing (upper triangular) R_k and r_k from the partially extended Cholesky factorization

$$G_k^T \begin{bmatrix} G_k & g_{k,m+1} \end{bmatrix} = R_k^T \begin{bmatrix} R_k & r_k \end{bmatrix}, \tag{3.7}$$

one can see (by plugging (3.7) into (3.6) and using $G_k = Q_k R_k$) that T_k can be computed by

$$T_k \leftarrow \begin{bmatrix} R_k & r_k \end{bmatrix} J_k R_k^{-1}. \tag{3.8}$$

Fletcher's third approach, which also avoids multiplications with A, is to compute

$$T_k \leftarrow \begin{bmatrix} R_k & Q_k^T g_{k,m+1} \end{bmatrix} J_k R_k^{-1}. \tag{3.9}$$

However, this is less efficient than using (3.8) due to the need to store Q_k and since the QR factorization of G_k requires $\sim m^2 n$ flops, as opposed to the $\sim \frac{1}{2} m^2 n$ flops required for (3.8); see [22].

Algorithm LMSD Limited Memory Steepest Descent Method

choose an initial point $x_{1,1} \in \mathbb{R}^n$, history length $m \in [n]$, and termination tolerance $\epsilon \in \mathbb{R}_+$
choose stepsizes $\{\alpha_{1,j}\}_{j\in[m]} \subset \mathbb{R}_{++}$
compute $g_{1,1} \leftarrow \nabla f(x_{1,1})$
if $\|g_{1,1}\| \leq \epsilon$, **then return** $x_{1,1}$
for $k \in \mathbb{N}$ **do**
    for $j \in [m]$ **do**
        set $x_{k,j+1} \leftarrow x_{k,j} - \alpha_{k,j} g_{k,j}$
        compute $g_{k,j+1} \leftarrow \nabla f(x_{k,j+1})$
        if $\|g_{k,j+1}\| \leq \epsilon$, **then return** $x_{k,j+1}$
    end for
    set $x_{k+1,1} \leftarrow x_{k,m+1}$ and $g_{k+1,1} \leftarrow g_{k,m+1}$
    set G_k by (3.4) and J_k by (3.5)
    compute R_k and r_k to satisfy (3.7) and set T_k by (3.8)
    set $\{\theta_{k,j}\}_{j\in[m]} \subset \mathbb{R}_{++}$ as the eigenvalues of T_k in decreasing order
    set $\{\alpha_{k+1,j}\}_{j\in[m]} \leftarrow \{\theta_{k,j}^{-1}\}_{j\in[m]} \subset \mathbb{R}_{++}$
end for

The choice to order the eigenvalues of T_k in decreasing order is motivated by [22]. In short, this ensures that the stepsizes in cycle $k+1$ are ordered from smallest to largest, which improves the likelihood that the objective function and the norm of the objective gradient decrease monotonically, at least initially, in each cycle. This ordering is not essential for our analysis, but is a good choice for any implementation of the algorithm; hence, we state the algorithm to employ this ordering.

One detail that remains for a practical implementation of the method is how to choose the initial stepsizes $\{\alpha_{1,j}\}_{j\in[m]} \subset \mathbb{R}_{++}$. This choice has no effect on the theoretical results proved in this chapter, though our analysis does confirm the fact that the practical performance of the method can be improved if one has the knowledge to choose one or more stepsizes exactly equal to reciprocals of eigenvalues of A; see §3.2.3. Otherwise, one can

either provide a full set of m stepsizes or carry out an initialization phase in which the first few cycles are shorter in length, dependent on the number of objective gradients that have been observed so far; see [22] for further discussion on this matter.

Remark 3.2.2. *In (3.4), if G_k for some $k \in \mathbb{N}$ does not have linearly independent columns, then R_k is singular and the formulas (3.8) and (3.9) are invalid, meaning that the employed approach is not able to provide m eigenvalue estimates for cycle k. As suggested in [22], an implementation of the method can address this by iteratively removing "older" columns of G_k until the columns form a linearly independent set of vectors, in which case the approach would be able to provide $\tilde{m} \leq m$ stepsizes for the subsequent (shortened) cycle. We advocate such an approach in practice and, based on the results proved in this chapter, conjecture that the convergence rate of the algorithm would be R-linear. However, the analysis for such a method would be extremely cumbersome given that the number of iterations in each cycle might vary from one cycle to the next within a single run of the algorithm. Hence, in our analysis in §3.3, we assume that G_k has linearly independent columns for all $k \in \mathbb{N}$. In fact, we go further and assume that $\|R_k^{-1}\|$ is bounded proportionally to the reciprocal of the norm of the objective gradient at the first iterate in cycle k (meaning that the upper bound diverges as the algorithm converges to the minimizer of the objective function). These norms are easily computed in an implementation of the algorithm; hence, we advocate that a procedure of iteratively removing "older" columns of G_k would be based on observed violations of such a bound. See the discussion following Assumption 3.3.4 in §3.3.*

3.2.3 Finite Termination Property of LMSD

If, for some $k \in \mathbb{N}$ and $j \in [m]$, the stepsizes in Algorithm LMSD up through iteration $(k, j) \in \mathbb{N} \times [m]$ include the reciprocals of all of the $r \leq n$ distinct eigenvalues of A, then the algorithm terminates by the end of iteration (k, j) with $x_{k,j+1}$ yielding $\|g_{k,j+1}\| = 0$. This is shown in the following lemma and theorem, which together demonstrate and extend the arguments made, e.g., in §2 of [22].

Lemma 3.2.3. *Under Assumption 3.2.1, for each $(k,j) \in \mathbb{N} \times [m]$, there exist weights $\{d_{k,j,i}\}_{i\in[n]}$ such that $g_{k,j}$ can be written as a linear combination of the columns of Q in (3.3), i.e.,*

$$g_{k,j} = \sum_{i=1}^{n} d_{k,j,i} q_i. \tag{3.10}$$

Moreover, these weights satisfy the recursive property

$$d_{k,j+1,i} = (1 - \alpha_{k,j}\lambda_i) d_{k,j,i} \quad \text{for all} \quad (k,j,i) \in \mathbb{N} \times [m] \times [n]. \tag{3.11}$$

Proof. Since $g_{k,j} = Ax_{k,j} - b$ for all $(k,j) \in \mathbb{N} \times [m]$, it follows that

$$\begin{aligned}
& x_{k,j+1} = x_{k,j} - \alpha_{k,j} g_{k,j}, \\
\implies \quad & Ax_{k,j+1} = Ax_{k,j} - \alpha_{k,j} Ag_{k,j}, \\
\implies \quad & g_{k,j+1} = g_{k,j} - \alpha_{k,j} Ag_{k,j}, \\
\implies \quad & g_{k,j+1} = (I - \alpha_{k,j} A) g_{k,j}, \\
\implies \quad & g_{k,j+1} = (I - \alpha_{k,j} Q\Lambda Q^T) g_{k,j},
\end{aligned}$$

from which one obtains that

$$\sum_{i=1}^{n} d_{k,j+1,i} q_i = \sum_{i=1}^{n} d_{k,j,i} (I - \alpha_{k,j} Q\Lambda Q^T) q_i = \sum_{i=1}^{n} d_{k,j,i} (q_i - \alpha_{k,j}\lambda_i q_i) = \sum_{i=1}^{n} d_{k,j,i} (1 - \alpha_{k,j}\lambda_i) q_i.$$

The result then follows since the columns of Q form an orthogonal basis of $\mathbb{R}^n$. □

Theorem 3.2.4. *Suppose that Assumption 3.2.1 holds and that Algorithm LMSD is run with termination tolerance $\epsilon = 0$. If, for some $(k,j) \in \mathbb{N} \times [m]$, the set of computed stepsizes up through iteration (k,j) includes all of the values $\{\lambda_{(l)}^{-1}\}_{l\in[r]}$, then, at the latest, the algorithm terminates finitely at the end of iteration (k,j) with $x_{k,j+1}$ yielding $\|g_{k,j+1}\| = 0$.*

Proof. Consider any $(k,j) \in \mathbb{N} \times [m]$ such that the stepsize is equal to the reciprocal of an eigenvalue of A, i.e., $\alpha_{k,j} = \lambda_{(l)}^{-1}$ for some $l \in [r]$. By Lemma 3.2.3, it follows that

$$d_{k,j+1,i} = (1 - \alpha_{k,j}\lambda_i) d_{k,j,i} = (1 - \lambda_{(l)}^{-1}\lambda_i) d_{k,j,i} = 0 \quad \text{for all} \quad i \in [n] \quad \text{such that} \quad \lambda_i = \lambda_{(l)}.$$

Along with the facts that Lemma 3.2.3 also implies

$$d_{k,j,i} = 0 \implies d_{k,j+1,i} = 0 \quad \text{for all} \quad (k,j) \in \mathbb{N} \times [m]$$

and $x_{k+1,1} \leftarrow x_{k,m+1}$ (and $g_{k+1,1} \leftarrow g_{k,m+1}$) for all $k \in \mathbb{N}$, the desired conclusion follows.

□

Remark 3.2.5. *Theorem 3.2.4 implies that Algorithm LMSD will converge finitely by the end of the second cycle if $m \geq r$ and the eigenvalues of T_1 include all eigenvalues $\{\lambda_{(l)}\}_{l \in [r]}$. This is guaranteed, e.g., when the first cycle involves $m = n$ steps and G_1 has linearly independent columns.*

3.3 R-Linear Convergence Rate of LMSD

Our primary goal in this Chapter is to prove that Algorithm LMSD converges R-linearly for any choice of the history length parameter $m \in [n]$. For context, we begin by citing two known convergence results that apply to Algorithm LMSD, then turn our attention to our new convergence rate results.

3.3.1 Known Convergence Properties of LMSD

In the Appendix of [22], the following convergence result is proved for Algorithm LMSD. The theorem is stated slightly differently here only to account for our different notation.

Theorem 3.3.1. *Suppose that Assumption 3.2.1 holds and that Algorithm LMSD is run with termination tolerance $\epsilon = 0$. Then, either $g_{k,j} = 0$ for some $(k,j) \in \mathbb{N} \times [m]$ or the sequences $\{g_{k,j}\}_{k=1}^{\infty}$ for each $j \in [m]$ converge to zero.*

As a consequence of this result, we may conclude that if Algorithm LMSD does not terminate finitely, then, according to the relationship (3.10), the following limits hold:

$$\lim_{k \to \infty} g_{k,j} = 0 \quad \text{for each} \quad j \in [m] \quad \text{and} \tag{3.12a}$$

$$\lim_{k \to \infty} d_{k,j,i} = 0 \quad \text{for each} \quad (j,i) \in [m] \times [n]. \tag{3.12b}$$

Fletcher's result, however, does not illuminate the rate at which these sequences converge to zero. Only for the case of $m = 1$ in which Algorithm LMSD reduces to a BB method do the following results from [15] (see Lemma 2.4 and Theorem 2.5 therein) provide a convergence rate guarantee.

Lemma 3.3.2. *Suppose that Assumption 3.2.1 holds and that Algorithm LMSD is run with history length $m = 1$ and termination tolerance $\epsilon = 0$. Then, there exists $K \in \mathbb{N}$, dependent only on (λ_1, λ_n), such that*

$$\|g_{k+K,1}\| \leq \tfrac{1}{2}\|g_{k,1}\| \quad \text{for all} \quad k \in \mathbb{N}.$$

Theorem 3.3.3. *Suppose that Assumption 3.2.1 holds and that Algorithm LMSD is run with history length $m = 1$ and termination tolerance $\epsilon = 0$. Then, either $g_{k,1} = 0$ for some $k \in \mathbb{N}$ or*

$$\|g_{k,1}\| \leq c_1 c_2^k \|g_{1,1}\| \quad \text{for all} \quad k \in \mathbb{N},$$

where, with $K \in \mathbb{N}$ from Lemma 3.3.2, the constants are defined as

$$c_1 := 2\left(\frac{\lambda_n}{\lambda_1} - 1\right)^{K-1} \quad \text{and} \quad c_2 := 2^{-1/K} \in (0, 1).$$

Overall, the computed gradients vanish R-linearly with constants that depend only on (λ_1, λ_n).

3.3.2 R-Linear Convergence Rate of LMSD for Arbitrary $m \in [n]$

Our goal in this Chapter is to build upon the proofs of the results stated in the previous Chapter (as given in the cited references) to show that, under reasonable assumptions, Algorithm LMSD possesses an R-linear rate of convergence for any $m \in [n]$. More precisely, our goal is to show that the gradients computed by the algorithm vanish R-linearly with constants that depend only on the spectrum of the data matrix A. One of the main challenges in this pursuit is the fact, hinted at by Lemma 3.3.2 for the case of $m = 1$, that the gradients computed in Algorithm LMSD might not decrease monotonically in norm.

This is one reason why the analysis in [15] is so remarkable, and, not surprisingly, it is an issue that must be overcome in our analysis as well. But our analysis also overcomes new challenges. In particular, the analysis in [15] is able to be more straightforward due to the fact that, in a BB method, a stepsize computation is performed after *every* iterate update. In particular, this means that, in iteration $k \in \mathbb{N}$, the current gradient $g_{k,1}$ plays a role in the computation of $\alpha_{k,1}$. In LMSD, on the other hand, a set of stepsizes are computed and employed in sequence, meaning that multiple iterate updates are performed until the next set of stepsizes are computed. This means that, in each cycle, iterate updates are performed using stepsizes computed using *old* gradient information. Another challenge that our analysis overcomes is the fact that the computed stepsizes cannot all be characterized in the same manner; rather, as revealed later in Lemma 3.3.7, each set of stepsizes is *spread* through distinct intervals in the spectrum of A. Our analysis overcomes all of these challenges by keeping careful track of the affects of applying each sequence of stepsizes vis-à-vis the weights in (3.10) for all $(k, j) \in \mathbb{N} \times [m]$. In particular, we show that even though the gradients might not decrease monotonically in norm and certain weights in (3.10) might increase within each cycle and from one cycle to the next, the weights ultimately vanish in a manner that corresponds to R-linear vanishing of the gradients for any $m \in [n]$.

Formally, for simplicity and brevity in our analysis, we make the following standing assumption throughout this Chapter.

Assumption 3.3.4. *Assumption 3.2.1 holds, as do the following:*

(i) Algorithm LMSD is run with $\epsilon = 0$ and $g_{k,j} \neq 0$ for all $(k, j) \in \mathbb{N} \times [m]$.

(ii) For all $k \in \mathbb{N}$, the matrix G_k has linearly independent columns. Further, there exists a scalar $\rho \geq 1$ such that, for all $k \in \mathbb{N}$, the nonsingular matrix R_k satisfies $\|R_k^{-1}\| \leq \rho \|g_{k,1}\|^{-1}$.

Assumption 3.3.4(i) is reasonable since, in any situation in which the algorithm terminates finitely, all of our results hold for the iterations prior to that in which the algorithm terminates. Hence, by proving that the algorithm possesses an R-linear rate of convergence

for cases when it does not terminate finitely, we claim that it possesses such a rate in all cases. As for Assumption 3.3.4(*ii*), first recall Remark 3.2.2. In addition, the bound on the norm of the inverse of R_k is reasonable since, in the case of $m = 1$, one finds that $Q_k R_k = G_k = g_{k,1}$ has $Q_k = g_{k,1}/\|g_{k,1}\|$ and $R_k = \|g_{k,1}\|$, meaning that the bound holds with $\rho = 1$. (This means that, in practice, one might choose $\rho \geq 1$ and iteratively remove columns of G_k for the computation of T_k until one finds $\|R_k^{-1}\| \leq \rho \|g_{k,1}\|^{-1}$, knowing that, in the extreme case, there will remain one column for which this condition is satisfied. However, for the reasons already given in Remark 3.2.2, we make Assumption 3.3.4, meaning that G_k always has m columns.)

Remark 3.3.5. *Our analysis hinges on properties of the stepsizes computed in Steps 12–15 of Algorithm LMSD as they relate to the spectrum of the matrix A. These properties do not necessarily hold for the initial set of stepsizes $\{\alpha_{1,j}\}_{j\in[m]}$, which are merely restricted to be in $\mathbb{R}_{++}$. However, for ease of exposition in our analysis, rather than distinguish between the stepsizes in the initial cycle (i.e., $k = 1$) versus all subsequent cycles (i.e., $k \geq 2$), we proceed under the assumption that all properties that hold for $k \geq 2$ also hold for $k = 1$. (One could instead imagine that an "initialization cycle" is performed corresponding to $k = 0$, in which case all of our subsequent results are indeed true for all $k \in \mathbb{N}$.) We proceed in this manner, without stating it as a formal assumption, since our main conclusion (see Theorem 3.3.13) remains true whether or not one counts the computational effort in the initial cycle.*

We begin by stating two results that reveal important properties of the eigenvalues (corresponding to the elements of $\{T_k\}$) computed by the algorithm, which in turn reveal properties of the stepsizes. The first result is a direct consequence of the *Cauchy Interlacing Theorem*. Since this theorem is well-known—see, e.g., [43]—we state the lemma without proof.

Lemma 3.3.6. *For all $k \in \mathbb{N}$, the eigenvalues of T_k $(= Q_k^T A Q_k$ where $Q_k^T Q_k = I)$ satisfy*

$$\theta_{k,j} \in [\lambda_{m+1-j}, \lambda_{n+1-j}] \quad \text{for all} \quad j \in [m].$$

The second result provides more details about how the eigenvalues computed by the algorithm at the end of iteration $k \in \mathbb{N}$ relate to the weights in (3.10) corresponding to k for all $j \in [m]$.

Lemma 3.3.7. *For all $(k, j) \in \mathbb{N} \times [m]$, let $q_{k,j} \in \mathbb{R}^m$ denote the unit eigenvector corresponding to the eigenvalue $\theta_{k,j}$ of T_k, i.e., the vector satisfying $T_k q_{k,j} = \theta_{k,j} q_{k,j}$ and $\|q_{k,j}\| = 1$. Then, defining*

$$D_k := \begin{bmatrix} d_{k,1,1} & \cdots & d_{k,m,1} \\ \vdots & \ddots & \vdots \\ d_{k,1,n} & \cdots & d_{k,m,n} \end{bmatrix} \quad \text{and} \quad c_{k,j} := D_k R_k^{-1} q_{k,j}, \tag{3.13}$$

it follows that, with the diagonal matrix of eigenvalues (namely, Λ) defined in Assumption 3.2.1,

$$\theta_{k,j} = c_{k,j}^T \Lambda c_{k,j} \quad \text{and} \quad c_{k,j}^T c_{k,j} = 1. \tag{3.14}$$

Proof. For any $k \in \mathbb{N}$, it follows from (3.13) and Lemma 3.2.3 (in particular, (3.11)) that $G_k = QD_k$ where Q is the orthogonal matrix defined in Assumption 3.2.1. Then, since $G_k = Q_k R_k$ (recall (3.4)), it follows that $Q_k = QD_k R_k^{-1}$, according to which one finds

$$T_k = Q_k^T A Q_k = R_k^{-T} D_k^T Q^T A Q D_k R_k^{-1} = R_k^{-T} D_k^T \Lambda D_k R_k^{-1}.$$

Hence, for each $j \in [m]$, the first equation in (3.14) follows since

$$\theta_{k,j} = q_{k,j}^T T_k q_{k,j} = q_{k,j}^T R_k^{-T} D_k^T \Lambda D_k R_k^{-1} q_{k,j} = c_{k,j}^T \Lambda c_{k,j}.$$

In addition, since $G_k = QD_k$ and the orthogonality of Q imply that $D_k^T D_k = G_k^T G_k$, and since $Q_k = G_k R_k^{-1}$ with Q_k having orthonormal columns (i.e., with Q_k satisfying $Q_k^T Q_k = I$), it follows that

$$c_{k,j}^T c_{k,j} = q_{k,j}^T R_k^{-T} D_k^T D_k R_k^{-1} q_{k,j} = q_{k,j}^T R_k^{-T} G_k^T G_k R_k^{-1} q_{k,j} = q_{k,j}^T Q_k^T Q_k q_{k,j} = q_{k,j}^T q_{k,j} = 1,$$

which yields the second equation in (3.14). □

The implications of Lemma 3.3.7 are seen later in our analysis. For now, combining Lemma 3.3.6, Lemma 3.2.3 (in particular, (3.11)), and the fact that (3.10) implies

$$\|g_{k,j}\|^2 = \sum_{i=1}^{n} d_{k,j,i}^2 \quad \text{for all} \quad (k,j) \in \mathbb{N} \times [m], \tag{3.15}$$

one is lead to the following result pertaining to recursive properties of the weights in (3.10).

Lemma 3.3.8. *For each $(k,j,i) \in \mathbb{N} \times [m] \times [n]$, it follows that*

$$|d_{k,j+1,i}| \leq \delta_{j,i} |d_{k,j,i}| \quad \textit{where} \quad \delta_{j,i} := \max\left\{ \left|1 - \frac{\lambda_i}{\lambda_{m+1-j}}\right|, \left|1 - \frac{\lambda_i}{\lambda_{n+1-j}}\right| \right\}. \tag{3.16}$$

Hence, for each $(k,j,i) \in \mathbb{N} \times [m] \times [n]$, it follows that

$$|d_{k+1,j,i}| \leq \Delta_i |d_{k,j,i}| \quad \textit{where} \quad \Delta_i := \prod_{j=1}^{m} \delta_{j,i}. \tag{3.17}$$

Furthermore, for each $(k,j,p) \in \mathbb{N} \times [m] \times [n]$, it follows that

$$\sqrt{\sum_{i=1}^{p} d_{k,j+1,i}^2} \leq \hat{\delta}_{j,p} \sqrt{\sum_{i=1}^{p} d_{k,j,i}^2} \quad \textit{where} \quad \hat{\delta}_{j,p} := \max_{i \in [p]} \delta_{j,i}, \tag{3.18}$$

while, for each $(k,j) \in \mathbb{N} \times [m]$, it follows that

$$\|g_{k+1,j}\| \leq \Delta \|g_{k,j}\| \quad \textit{where} \quad \Delta := \max_{i \in [n]} \Delta_i. \tag{3.19}$$

Proof. Recall that, for any given $(k,j,i) \in \mathbb{N} \times [m] \times [n]$, Lemma 3.2.3 (in particular, (3.11)) states

$$d_{k,j+1,i} = (1 - \alpha_{k,j} \lambda_i) d_{k,j,i}.$$

The relationship (3.16) then follows due to Lemma 3.3.6, which, in particular, shows that

$$\alpha_{k,j} \in \left[\frac{1}{\lambda_{n+1-j}}, \frac{1}{\lambda_{m+1-j}} \right] \subseteq \left[\frac{1}{\lambda_n}, \frac{1}{\lambda_1} \right] \quad \text{for all} \quad (k,j) \in \mathbb{N} \times [m].$$

The consequence (3.17) then follows by combining (3.16) for all $j \in [m]$ and recalling that Step 11 yields $g_{k+1,1} \leftarrow g_{k,m+1}$ for all $k \in \mathbb{N}$. Now, from (3.16), one finds that

$$\sum_{i=1}^{p} d_{k,j+1,i}^2 \leq \sum_{i=1}^{p} \delta_{j,i}^2 d_{k,j,i}^2 \leq \hat{\delta}_{j,p}^2 \sum_{i=1}^{p} d_{k,j,i}^2 \quad \text{for all} \quad (k,j,p) \in \mathbb{N} \times [m] \times [n],$$

yielding the desired conclusion (3.18). Finally, combining (3.17) and (3.15), one obtains that

$$\|g_{k+1,j}\|^2 = \sum_{i=1}^{n} d_{k+1,j,i}^2 \leq \sum_{i=1}^{n} \Delta_i^2 d_{k,j,i}^2 \leq \Delta^2 \sum_{i=1}^{n} d_{k,j,i}^2 = \Delta^2 \|g_{k,j}\|^2 \quad \text{for all} \quad (k,j) \in \mathbb{N} \times [m],$$

yielding the desired conclusion (3.19). □

A consequence of the previous lemma is that if $\Delta_i \in [0,1)$ for all $i \in [n]$, then $\Delta \in [0,1)$, from which (3.19) implies that, for each $j \in [m]$, the gradient norm sequence $\{\|g_{k,j}\|\}_{k\in\mathbb{N}}$ vanishes Q-linearly. For example, such a situation occurs when $\lambda_n < 2\lambda_1$. However, as noted in [15], this is a highly uncommon case that should not be assumed to hold widely in practice. A more interesting and widely relevant consequence of the lemma is that for any $i \in [n]$ such that $\Delta_i \in [0,1)$, the sequences $\{|d_{k,j,i}|\}_{k\in\mathbb{N}}$ for each $j \in [m]$ vanish Q-linearly. For example, this is *always* true for $i = 1$, where

$$\delta_{j,1} = \max\left\{1 - \frac{\lambda_1}{\lambda_{m+1-j}}, 1 - \frac{\lambda_1}{\lambda_{n+1-j}}\right\} \in [0,1) \quad \text{for all} \quad j \in [m],$$

from which it follows that

$$\Delta_1 = \prod_{j=1}^{m} \delta_{j,1} \in [0,1). \tag{3.20}$$

The following is a crucial consequence that one can draw from this observation.

Lemma 3.3.9. *If $\Delta_1 = 0$, then $d_{1+\hat{k},\hat{j},1} = 0$ for all $(\hat{k},\hat{j}) \in \mathbb{N} \times [m]$. Otherwise, if $\Delta_1 > 0$, then:*

(i) for any $(k,j) \in \mathbb{N} \times [m]$ such that $d_{k,j,1} = 0$, it follows that $d_{k+\hat{k},\hat{j},1} = 0$ for all $(\hat{k},\hat{j}) \in \mathbb{N} \times [m]$;

(ii) for any $(k,j) \in \mathbb{N} \times [m]$ *such that* $|d_{k,j,1}| > 0$ *and any* $\epsilon_1 \in (0,1)$*, it follows that*

$$\frac{|d_{k+\hat{k},\hat{j},1}|}{|d_{k,j,1}|} \leq \epsilon_1 \quad \textit{for all} \quad \hat{k} \geq 1 + \left\lceil \frac{\log \epsilon_1}{\log \Delta_1} \right\rceil \quad \textit{and} \quad \hat{j} \in [m].$$

Proof. If $\Delta_1 = 0$, then the desired conclusion follows from Lemma 3.3.8; in particular, it follows from the inequality (3.17) for $i = 1$. Similarly, for any $(k,j) \in \mathbb{N} \times [m]$ such that $d_{k,j,1} = 0$, the conclusion in part (i) follows from the same conclusion in Lemma 3.3.8, namely, (3.17) for $i = 1$. Hence, let us continue to prove part (ii) under the assumption that $\Delta_1 \in (0,1)$ (recall (3.20)).

Suppose that the given condition holds with $j = 1$, i.e., consider $k \in \mathbb{N}$ such that $|d_{k,1,1}| > 0$. Then, it follows by Lemma 3.3.8 (in particular, (3.17) for $j = 1$ and $i = 1$) that

$$\frac{|d_{k+\hat{k},1,1}|}{|d_{k,1,1}|} \leq \Delta_1^{\hat{k}} \quad \text{for any} \quad \hat{k} \in \mathbb{N}. \tag{3.21}$$

Since $\Delta_1 \in (0,1)$, taking the logarithm of the term on the right-hand side with $\hat{k} = \lceil \log \epsilon_1 / \log \Delta_1 \rceil$ yields

$$\left\lceil \frac{\log \epsilon_1}{\log \Delta_1} \right\rceil \log \Delta_1 \leq \left(\frac{\log \epsilon_1}{\log \Delta_1} \right) \log \Delta_1 = \log (\epsilon_1) \,. \tag{3.22}$$

Since $\log(\cdot)$ is nondecreasing, the inequalities yielded by (3.22) combined with (3.21) and (3.17) yield the desired result for $j = 1$. On the other hand, if the conditions of part (ii) hold for some other $j \in [m]$, then the desired conclusion follows from a similar reasoning, though an extra cycle may need to be completed before the desired conclusion holds for all points in the cycle, i.e., for all $\hat{j} \in [m]$. This is the reason for the addition of 1 to $\lceil \log \epsilon_1 / \log \Delta_1 \rceil$ in the general conclusion. □

One may conclude from Lemma 3.3.9 and (3.10) that, for any $(k,j) \in \mathbb{N} \times [m]$ and $\epsilon_1 \in (0,1)$, one has

$$\frac{|d_{k+\hat{k},\hat{j},1}|}{\|g_{k,j}\|} \leq \epsilon_1 \quad \text{for all} \quad \hat{k} \geq K_1 \quad \text{and} \quad \hat{j} \in [m]$$

for some $K_1 \in \mathbb{N}$ that depends on the desired contraction factor $\epsilon_1 \in (0,1)$ and the problem-dependent constant $\Delta_1 \in (0,1)$, but does *not* depend on the iteration number pair (k,j). Our goal now is to show that if a similar, but looser conclusion holds for a squared sum of the weights in (3.10) up through $p \in [n-1]$, then the squared weight corresponding to index $p+1$ eventually becomes sufficiently small in a number of iterations that is independent of the iteration number k. (For this lemma, we fix $j = \hat{j} = 1$ so as to consider only the first gradient in each cycle. This choice is somewhat arbitrary since our concluding theorem will confirm that a similar result holds for any $j \in [m]$ and $\hat{j} = j$.) For the lemma, we define the following constants that depend only on p, the spectrum of A (which, in particular, yields the bounds and definitions in Lemma 3.3.8), and the scalar constant $\rho \geq 1$ from Assumption 3.3.4:

$$\hat{\delta}_p := \left(1 + \hat{\delta}_{1,p}^2 + \hat{\delta}_{1,p}^2\hat{\delta}_{2,p}^2 + \cdots + \prod_{j=1}^{m-1} \hat{\delta}_{j,p}^2\right) \in [1,\infty), \tag{3.23a}$$

$$\hat{\Delta}_{p+1} := \max\left\{\frac{1}{3}, 1 - \frac{\lambda_{p+1}}{\lambda_n}\right\}^m \in (0,1), \tag{3.23b}$$

$$\text{and} \quad \hat{K}_p := \left\lceil \frac{\log\left(2\hat{\delta}_p\rho\epsilon_p\Delta_{p+1}^{-(K_p+1)}\right)}{\log \hat{\Delta}_{p+1}} \right\rceil. \tag{3.23c}$$

Lemma 3.3.10. *For any $(k,p) \in \mathbb{N} \times [n-1]$, if there exists $(\epsilon_p, K_p) \in (0, \frac{1}{2\hat{\delta}_p\rho}) \times \mathbb{N}$ independent of k with*

$$\sum_{i=1}^{p} d_{k+\hat{k},1,i}^2 \leq \epsilon_p^2 \|g_{k,1}\|^2 \quad \textit{for all} \quad \hat{k} \geq K_p, \tag{3.24}$$

then one of the following holds:

(i) $\Delta_{p+1} \in [0,1)$ and there exists $K_{p+1} \geq K_p$ dependent only on ϵ_p, ρ, and the spectrum of A with

$$d_{k+K_{p+1},1,p+1}^2 \leq 4\hat{\delta}_p^2\rho^2\epsilon_p^2\|g_{k,1}\|^2; \tag{3.25}$$

(ii) $\Delta_{p+1} \in [1,\infty)$ and, with $K_{p+1} := K_p + \hat{K}_p + 1$, there exists $\hat{k}_0 \in \{K_p, \ldots, K_{p+1}\}$

with

$$d^2_{k+\hat{k}_0,1,p+1} \leq 4\hat{\delta}_p^2\rho^2\epsilon_p^2\|g_{k,1}\|^2. \tag{3.26}$$

Proof. By Lemma 3.3.8 (in particular, (3.17) with $j = 1$ and $i = p+1$) and (3.15), it follows that

$$d^2_{k+\hat{k},1,p+1} \leq \left(\Delta^{\hat{k}}_{p+1} d_{k,1,p+1}\right)^2 = \Delta^{2\hat{k}}_{p+1} d^2_{k,1,p+1} \leq \Delta^{2\hat{k}}_{p+1}\|g_{k,1}\|^2 \quad \text{for all} \quad \hat{k} \in \mathbb{N}. \tag{3.27}$$

If $\Delta_{p+1} \in [0,1)$, then (3.27) immediately implies the existence of K_{p+1} dependent only on ϵ_p, ρ, and the spectrum of A such that (3.25) holds. Hence, let us continue under the assumption that $\Delta_{p+1} \geq 1$, where one should observe that $\rho \geq 1$, $\hat{\delta}_p \geq 1$, $\epsilon_p \in (0, \frac{1}{2\hat{\delta}_p\rho})$, $K_p \in \mathbb{N}$, and $\Delta_{p+1} \geq 1$ imply $2\hat{\delta}_p\rho\epsilon_p\Delta_{p+1}^{-K_p} \in (0,1)$, meaning that $\hat{K}_p \in \mathbb{N}$. To prove the desired result, it suffices to show that if

$$d^2_{k+\hat{k},1,p+1} > 4\hat{\delta}_p^2\rho^2\epsilon_p^2\|g_{k,1}\|^2 \quad \text{for all} \quad \hat{k} \in \{K_p, \ldots, K_{p+1}-1\}, \tag{3.28}$$

then (3.26) holds at the beginning of the next cycle (i.e., when $\hat{k}_0 = K_{p+1}$). From Lemma 3.3.7, Lemma 3.3.8 (in particular, (3.18)), (3.24), and (3.28), it follows that with $\{c_{k+\hat{k},j,i}\}_{i=1}^n$ representing the elements of the vector $c_{k+\hat{k},j}$ and the matrix $D_{k+\hat{k},p}$ representing the first p rows of $D_{k+\hat{k}}$, one finds

$$\begin{aligned}
\sum_{i=1}^{p} c^2_{k+\hat{k},j,i} &\leq \|D_{k+\hat{k},p}\|_2^2\|R^{-1}_{k+\hat{k}}\|^2\|q_{k+\hat{k},j}\|^2 \\
&\leq \left(1 + \hat{\delta}^2_{1,p} + \hat{\delta}^2_{1,p}\hat{\delta}^2_{2,p} + \cdots + \prod_{j=1}^{m-1}\hat{\delta}^2_{j,p}\right)\left(\sum_{i=1}^{p} d^2_{k+\hat{k},1,i}\right)\rho^2\|g_{k+\hat{k},1}\|^{-2} \\
&\leq \hat{\delta}^2_p(\epsilon_p^2\|g_{k,1}\|^2)\rho^2(4\hat{\delta}_p^2\rho^2\epsilon_p^2)^{-1}\|g_{k,1}\|^{-2} \\
&\leq \tfrac{1}{4} \quad \text{for all} \quad \hat{k} \in \{K_p, \ldots, K_{p+1}-1\} \quad \text{and} \quad j \in [m].
\end{aligned}$$

Along with Lemma 3.3.7, this implies that

$$\theta_{k+\hat{k},j} = \sum_{i=1}^{n} \lambda_i c^2_{k+\hat{k},j,i} \geq \tfrac{3}{4}\lambda_{p+1} \quad \text{for all} \quad \hat{k} \in \{K_p, \ldots, K_{p+1}-1\} \quad \text{and} \quad j \in [m]. \tag{3.29}$$

Together with Lemma 3.2.3 (see (3.11)) and $\alpha_{k+\hat{k}+1,j} = \theta^{-1}_{k+\hat{k},j}$ for all $j \in [m]$, the bound (3.29) implies

$$d^2_{k+\hat{k}+2,1,p+1} = \left(\prod_{j=1}^{m}\left(1-\alpha_{k+\hat{k}+1,j}\lambda_{p+1}\right)^2\right) d^2_{k+\hat{k}+1,1,p+1}$$
$$\leq \hat{\Delta}^2_{p+1} d^2_{k+\hat{k}+1,1,p+1} \quad \text{for all} \quad \hat{k} \in \{K_p, \ldots, K_{p+1}-1\}. \tag{3.30}$$

Applying this bound recursively, it follows with $K_{p+1} = K_p + \hat{K}_p + 1$ and (3.27) for $\hat{k} = K_{p+1}$ that

$$d^2_{k+K_{p+1},1,p+1} \leq \hat{\Delta}^{2\hat{K}_p}_{p+1} d^2_{k+K_p+1,1,p+1} \leq \hat{\Delta}^{2\hat{K}_p}_{p+1} \Delta^{2(K_p+1)}_{p+1} \|g_{k,1}\|^2 \leq 4\hat{\delta}^2_p r^2 \epsilon^2_p \|g_{k,1}\|^2,$$

where the last inequality follows by the definition of $\hat{K}_p$ in (3.23c). □

We have shown that small squared weights in (3.10) associated with indices up through $p \in [n-1]$ imply that the squared weight associated with index $p+1$ eventually becomes small. The next lemma shows that these latter squared weights also remain sufficiently small indefinitely.

Lemma 3.3.11. *For any $(k,p) \in \mathbb{N} \times [n-1]$, if there exists $(\epsilon_p, K_p) \in (0, \frac{1}{2\hat{\delta}_p \rho}) \times \mathbb{N}$ independent of k such that (3.24) holds, then, with $\epsilon^2_{p+1} := (1 + 4\max\{1, \Delta^4_{p+1}\}\hat{\delta}^2_p \rho^2)\epsilon^2_p$ and $K_{p+1} \in \mathbb{N}$ from Lemma 3.3.10,*

$$\sum_{i=1}^{p+1} d^2_{k+\hat{k},1,i} \leq \epsilon^2_{p+1}\|g_{k,1}\|^2 \quad \textit{for all} \quad \hat{k} \geq K_{p+1}. \tag{3.31}$$

Proof. For the same reasons as in the proof of Lemma 3.3.10, the result follows if $\Delta_{p+1} \in [0,1)$. Hence, we may continue under the assumption that $\Delta_{p+1} \geq 1$ and define $\hat{\Delta}_{p+1} \in (0,1)$ and $\hat{K}_p \in \mathbb{N}$ as in (3.23). By Lemma 3.3.10, there exists $\hat{k}_0 \in \{K_p, \ldots, K_{p+1}\}$ such that

$$d^2_{k+\hat{k},1,p+1} \leq 4\hat{\delta}^2_p \rho^2 \epsilon^2_p \|g_{k,1}\|^2 \quad \text{when} \quad \hat{k} = \hat{k}_0. \tag{3.32}$$

If the inequality in (3.32) holds for all $\hat{k} \geq \hat{k}_0$, then (3.31) holds with $\epsilon^2_{p+1} = (1 + 4\hat{\delta}^2_p \rho^2)\epsilon^2_p$.

Otherwise, let $\hat{k}_1 \in \mathbb{N}$ denote the smallest natural number such that

$$d^2_{k+\hat{k},1,p+1} \leq 4\hat{\delta}^2_p \rho^2 \epsilon^2_p \|g_{k,1}\|^2 \quad \text{for all} \quad \hat{k}_0 \leq \hat{k} \leq \hat{k}_1, \tag{3.33}$$

but

$$d^2_{k+\hat{k}_1+1,1,p+1} > 4\hat{\delta}^2_p \rho^2 \epsilon^2_p \|g_{k,1}\|^2. \tag{3.34}$$

As in the arguments that lead to (3.30) in the proof of Lemma 3.3.10, combining (3.24) and (3.34) implies

$$d^2_{k+\hat{k}_1+3,1,p+1} \leq \hat{\Delta}^2_{p+1} d^2_{k+\hat{k}_1+2,1,p+1}.$$

Generally, this same argument can be used to show that

$$\hat{k} \geq K_p \quad \text{and} \quad d^2_{k+\hat{k}+1,1,p+1} > 4\hat{\delta}^2_p \rho^2 \epsilon^2_p \|g_{k,1}\|^2 \quad \text{imply} \quad d^2_{k+\hat{k}+3,1,p+1} \leq \hat{\Delta}^2_{p+1} d^2_{k+\hat{k}+2,1,p+1}.$$

Since $\hat{\Delta}_{p+1} \in (0,1)$, this fact and (3.34) imply the existence of $\hat{k}_2 \in \mathbb{N}$ such that

$$d^2_{k+\hat{k}+1,1,p+1} > 4\hat{\delta}^2_p \rho^2 \epsilon^2_p \|g_{k,1}\|^2 \quad \text{for all} \quad \hat{k}_1 \leq \hat{k} \leq \hat{k}_2 - 2, \tag{3.35}$$

but

$$d^2_{k+\hat{k}_2,1,p+1} \leq 4\hat{\delta}^2_p \rho^2 \epsilon^2_p \|g_{k,1}\|^2,$$

while, from above,

$$d^2_{k+\hat{k}+3,1,p+1} \leq \hat{\Delta}^2_{p+1} d^2_{k+\hat{k}+2,1,p+1} \quad \text{for all} \quad \hat{k}_1 \leq \hat{k} \leq \hat{k}_2 - 2. \tag{3.36}$$

Moreover, by Lemma 3.3.8 (in particular, (3.17)) and (3.33), it follows that

$$d^2_{k+\hat{k}_1+1,1,p+1} \leq \Delta^2_{p+1} d^2_{k+\hat{k}_1,1,p+1} \leq 4\Delta^2_{p+1} \hat{\delta}^2_p \rho^2 \epsilon^2_p \|g_{k,1}\|^2 \tag{3.37a}$$

$$\text{and} \quad d^2_{k+\hat{k}_1+2,1,p+1} \leq 4\Delta^4_{p+1} \hat{\delta}^2_p \rho^2 \epsilon^2_p \|g_{k,1}\|^2. \tag{3.37b}$$

Combining (3.36) and (3.37b), it follows that

$$d^2_{k+\hat{k}+3,1,p+1} \leq 4\hat{\Delta}^2_{p+1}\Delta^4_{p+1}\hat{\delta}^2_p\rho^2\epsilon^2_p\|g_{k,1}\|^2 \quad \text{for all} \quad \hat{k}_1 \leq \hat{k} \leq \hat{k}_2 - 2.$$

Overall, since (3.23b) ensures $\hat{\Delta}_{p+1} \in (0,1)$, we have shown that

$$d^2_{k+\hat{k},1,p+1} \leq 4\Delta^4_{p+1}\hat{\delta}^2_p\rho^2\epsilon^2_p\|g_{k,1}\|^2 \quad \text{for all} \quad \hat{k} \in \{\hat{k}_0, \ldots, \hat{k}_2\}. \tag{3.38}$$

Repeating this argument for later iterations, we arrive at the desired conclusion. □

The following lemma is a generalization of Lemma 3.3.2 for any $m \in [n]$. Our proof is similar to that of Lemma 2.4 in [15]. We provide it in full for completeness.

Lemma 3.3.12. *There exists $K \in \mathbb{N}$ dependent only on the spectrum of A such that*

$$\|g_{k+K,1}\| \leq \tfrac{1}{2}\|g_{k,1}\| \quad \textit{for all} \quad k \in \mathbb{N}.$$

Proof. By Lemma 3.3.11, if for some $(\epsilon_p, K_p) \in (0, \frac{1}{2\hat{\delta}_p\rho}) \times \mathbb{N}$ independent of k one finds

$$\sum_{i=1}^{p} d^2_{k+\hat{k},1,i} \leq \epsilon^2_p\|g_{k,1}\|^2 \quad \text{for all} \quad \hat{k} \geq K_p, \tag{3.39}$$

then for $\epsilon^2_{p+1} := (1 + 4\max\{1, \Delta^4_{p+1}\}\hat{\delta}^2_p\rho^2)\epsilon^2_p$ and some $K_{p+1} \geq K_p$ independent of k one finds

$$\sum_{i=1}^{p+1} d^2_{k+\hat{k},1,i} \leq \epsilon^2_{p+1}\|g_{k,1}\|^2 \quad \text{for all} \quad \hat{k} \geq K_{p+1}. \tag{3.40}$$

Since Lemma 3.3.9 implies that for any $\epsilon_1 \in (0,1)$ one can find K_1 independent of k such that (3.39) holds with $p = 1$, it follows that, independent of k, there exists a sufficiently small $\epsilon_1 \in (0,1)$ such that

$$\epsilon^2_1 \leq \cdots \leq \epsilon^2_n \leq \tfrac{1}{4}.$$

Hence, for any $k \in \mathbb{N}$, it follows that there exists $K = K_n$ such that

$$\|g_{k+\hat{k},1}\|^2 = \sum_{i=1}^{n} d^2_{k+\hat{k},1,i} \leq \tfrac{1}{4}\|g_{k,1}\|^2 \quad \text{for all} \quad \hat{k} \geq K,$$

as desired. □

We are now prepared to state our final result, the proof of which follows in the same manner as Theorem 3.3.3 follows from Lemma 3.3.2 in [15]. We prove it in full for completeness.

Theorem 3.3.13. *The sequence* $\{\|g_{k,1}\|\}$ *vanishes R-linearly.*

Proof. If $\Delta \in [0,1)$, then it has already been argued (see the discussion following Lemma 3.3.8) that $\{\|g_{k,1}\|\}$ vanishes Q-linearly. Hence, let us continue assuming that $\Delta \geq 1$. By Lemma 3.3.12, there exists $K \in \mathbb{N}$ dependent only on the spectrum of A such that

$$\|g_{1+Kl,1}\| \leq \tfrac{1}{2}\|g_{1+K(l-1),1}\| \quad \text{for all} \quad l \in \mathbb{N}.$$

Applying this result recursively, it follows that

$$\|g_{1+Kl,1}\| \leq (\tfrac{1}{2})^l \|g_{1,1}\| \quad \text{for all} \quad l \in \mathbb{N}. \tag{3.41}$$

Now, for any $k \geq 1$, let us write $k = Kl + \hat{k}$ for some $l \in \{0\} \cup \mathbb{N}$ and $\hat{k} \in \{0\} \cup [K-1]$. It follows that

$$l = k/K - \hat{k}/K \geq k/K - 1.$$

By this fact, (3.19), and (3.41), it follows that for any $k = Kl + \hat{k} \in \mathbb{N}$ one has

$$\|g_{k,1}\| \leq \Delta^{\hat{k}-1}\|g_{1+Kl,1}\| \leq \Delta^{K-1}(\tfrac{1}{2})^{k/K-1}\|g_{1,1}\| \leq c_1 c_2^k \|g_{1,1}\|,$$

where

$$c_1 := 2\Delta^{K-1} \quad \text{and} \quad c_2 := 2^{-1/K} \in (0,1),$$

which implies the desired conclusion. □

3.4 Numerical Demonstrations

The analysis in the previous Chapter provides additional insights into the behavior of Algorithm LMSD beyond its R-linear rate of convergence. In this Chapter, we provide the results of numerical experiments to demonstrate the behavior of the algorithm in a few types of cases. The algorithm was implemented and the experiments were performed in Matlab. It is not our goal to show the performance of Algorithm LMSD for various values of m, say to argue whether the performance improves or not as m is increased. This is an important question for which some interesting discussion is provided by [22]. However, to determine what is a good choice of m for various types of cases would require a larger set of experiments that are outside of the scope of this chapter. For our purposes, our only goal is to provide some simple illustrations of the behavior as shown by our theoretical analysis.

Our analysis reveals that the convergence behavior of the algorithm depends on the spectrum of the matrix A. Therefore, we have constructed five test examples, all with $n = 100$, but with different eigenvalue distributions. For the first problem, the eigenvalues of A are evenly distributed in $[1, 1.9]$. Since this ensures that $\lambda_n < 2\lambda_1$, our analysis reveals that the algorithm converges Q-linearly for this problem; recall the discussion after Lemma 3.3.8. All other problems were constructed so that $\lambda_1 = 1$ and $\lambda_n = 100$, for which one clearly finds $\lambda_n > 2\lambda_1$. For the second problem, all eigenvalues are evenly distributed in $[\lambda_1, \lambda_n]$; for the third problem, the eigenvalues are clustered in five distinct blocks; for the fourth problem, all eigenvalues except one are clustered around λ_1; and for the fifth problem, all eigenvalues except one are clustered around λ_n. Table 3.1 shows the spectrum of A for each problem.

Table 3.1 also shows the numbers of outer and (total) inner iterations required by Algorithm LMSD (indicated by column headers "k" and "j", respectively) when it was run with $\epsilon = 10^{-8}$ and either $m = 1$ or $m = 5$. In all cases, the initial m stepsizes were generated randomly from a uniform distribution over the interval $[\lambda_{100}^{-1}, \lambda_1^{-1}]$. One finds that the algorithm terminated in relatively few outer and inner iterations relative to n, especially when many of the eigenvalues are clustered. This dependence on clustering of

the eigenvalues should not be surprising since, recalling Lemma 3.3.6, clustered eigenvalues makes it likely that an eigenvalue of T_k will be near an eigenvalue of A, which in turn implies by Lemma 3.2.3 that the weights in the representation (3.10) will vanish quickly. On the other hand, for the problems for which the eigenvalues are more evenly spread in $[1, 100]$, the algorithm required relatively more outer iterations, though still not an excessively large number relative to n. For these problems, the performance was mostly better for $m = 5$ versus $m = 1$, both in terms of outer and (total) inner iterations.

In Table 3.1, we also provide the maximum over k of the ratio $\|R_k^{-1}\|/(\|g_{k,1}\|^{-1})$ (indicated by the column header "ρ") observed during the run of the algorithm for each test problem and each m. The purpose of this is to confirm that Assumption 3.3.4 indeed held in our numerical experiments, but also to demonstrate for what value of ρ the assumption holds. As explained following Assumption 3.3.4, for $m = 1$ the ratio was always equal to 1. As for $m = 5$, on the other hand, the ratio was sometimes quite large, though it is worthwhile to remark that the ratio was typically much smaller than this maximum value. We did not observe any predictable behavior about when this maximum value was observed; sometimes it occurred early in the run, while sometimes it occurred toward the end. Overall, the evolution of this ratio depends on the initial point and path followed by the algorithm to the minimizer.

Table 3.1: Spectra of A for five test problems along with outer and (total) inner iteration counts required by Algorithm LMSD and maximum value of the ratio $\|R_k^{-1}\|/(\|g_{k,1}\|^{-1})$ observed during the run of Algorithm LMSD. For each spectrum, a set of eigenvalues in an interval indicates that the eigenvalues are evenly distributed within that interval.

Problem	Spectrum			$m = 1$			$m = 5$		
				k	j	ρ	k	j	ρ
1	$\{\lambda_1, \ldots, \lambda_{100}\}$	$\subset$	$[1, 1.9]$	13	13	1	3	14	$\sim 6 \times 10^3$
2	$\{\lambda_1, \ldots, \lambda_{100}\}$	$\subset$	$[1, 100]$	124	124	1	23	114	$\sim 1 \times 10^4$
3	$\{\lambda_1, \ldots, \lambda_{20}\}$	$\subset$	$[1, 2]$	112	112	1	16	79	$\sim 2 \times 10^5$
	$\{\lambda_{21}, \ldots, \lambda_{40}\}$	$\subset$	$[25, 26]$						
	$\{\lambda_{41}, \ldots, \lambda_{60}\}$	$\subset$	$[50, 51]$						
	$\{\lambda_{61}, \ldots, \lambda_{80}\}$	$\subset$	$[75, 76]$						
	$\{\lambda_{81}, \ldots, \lambda_{100}\}$	$\subset$	$[99, 100]$						
4	$\{\lambda_1, \ldots, \lambda_{99}\}$	$\subset$	$[1, 2]$	26	26	1	4	20	$\sim 2 \times 10^{16}$
	λ_{100}	$=$	100						
5	λ_1	$=$	1	16	16	1	5	25	$\sim 2 \times 10^{10}$
	$\{\lambda_2, \ldots, \lambda_{100}\}$	$\subset$	$[99, 100]$						

As seen in our analysis (inspired by [44], [15], and [22]), a more refined look into the

behavior of the algorithm is obtained by observing the step-by-step magnitudes of the weights in (3.10) for the generated gradients. Hence, for each of the test problems, we plot in Figures 3.1, 3.3, 3.5, 3.7, and 3.9 these magnitudes (on a log scale) for a few representative values of $i \in [n]$. Each figure consists of four sets of plots: the first and third show the magnitudes corresponding to $\{g_{k,1}\}$ (i.e., for the first point in each cycle) when $m = 1$ and $m = 5$, respectively, while the second and fourth show the magnitudes at all outer and inner iterations, again when $m = 1$ and $m = 5$, respectively. In a few of the images, the plot ends before the right-hand edge of the image. This is due to the log of the absolute value of the weight being evaluated as $-\infty$ in Matlab.

The figures show that the magnitudes of the weights corresponding to $i = 1$ always decrease monotonically, as proved in Lemma 3.3.9. The magnitudes corresponding to $i = 2$ also often decrease monotonically, but, as seen in the results for Problem 5, this is not always the case. In any case, the magnitudes corresponding to $i = 50$ and $i = 100$ often do not decrease monotonically, though, as proved in our analysis, one observes that the magnitudes demonstrate a downward trend over a finite number of cycles.

Even further insight into the plots of these magnitudes can be gained by observing the values of the constants $\{\Delta_i\}_{i\in[n]}$ for each problem and history length. Recalling (3.17), these constants bound the increase that a particular weight in (3.10) might experience from one point in a cycle to the same point in the subsequent cycle. For illustration, we plot in Figures 3.2, 3.4, 3.6, 3.8, and 3.10 these constants. Values less than 1 are indicated by a purple bar while values greater than or equal to 1 are indicated by a blue bar. Note that, in Figure 3.8, all values are small for both history lengths except Δ_{100}. In Figure 3.10, Δ_1 is less than one in both figures, but the remaining constants are large for $m = 1$ while being small for $m = 5$.

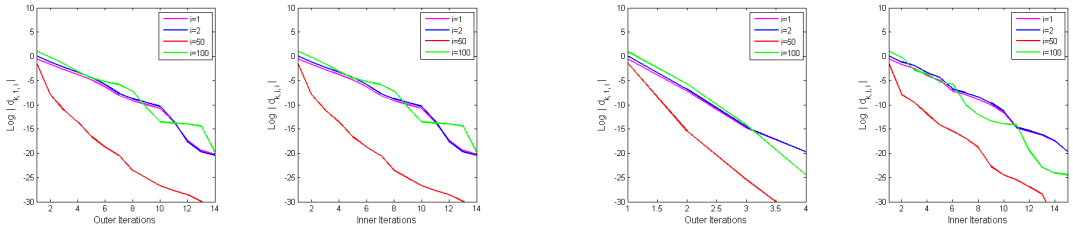

Figure 3.1: Weights in (3.10) for problem 1 with history length $m = 1$ (left two plots) and $m = 5$ (right two plots).

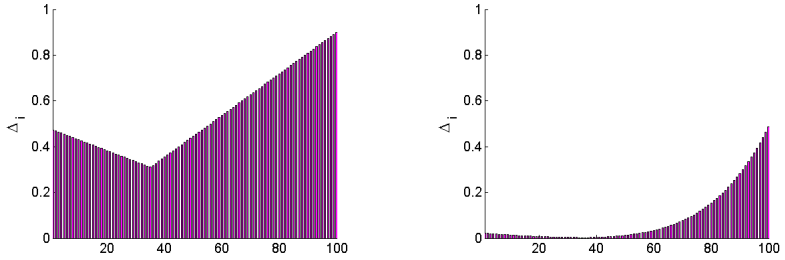

Figure 3.2: Constants in (3.17) for problem 1 with history length $m = 1$ (left plot) and $m = 5$ (right plot).

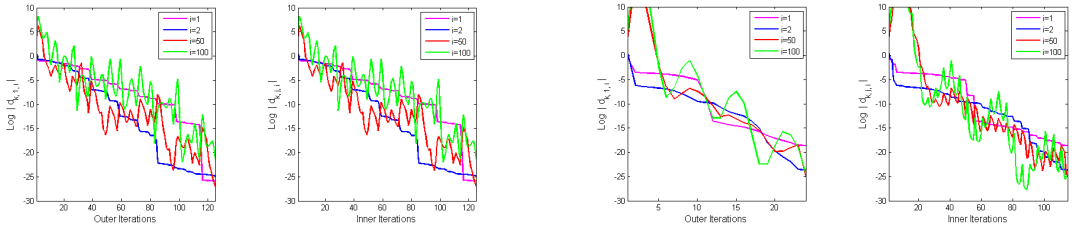

Figure 3.3: Weights in (3.10) for problem 2 with history length $m = 1$ (left two plots) and $m = 5$ (right two plots).

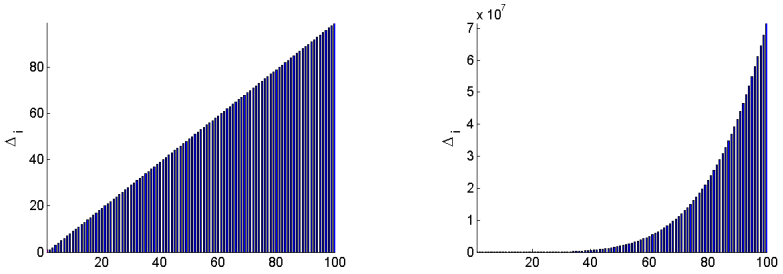

Figure 3.4: Constants in (3.17) for problem 2 with history length $m = 1$ (left plot) and $m = 5$ (right plot).

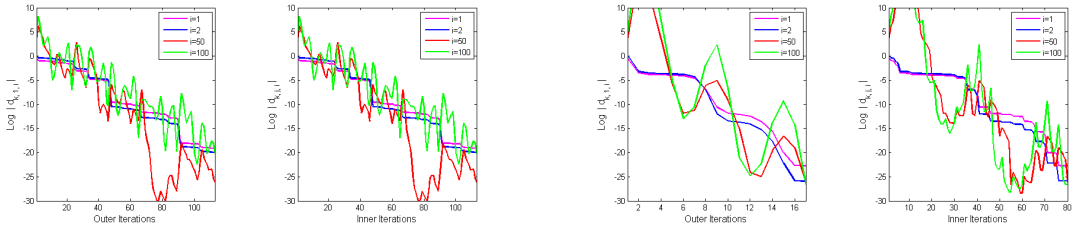

Figure 3.5: Weights in (3.10) for problem 3 with history length $m = 1$ (left two plots) and $m = 5$ (right two plots).

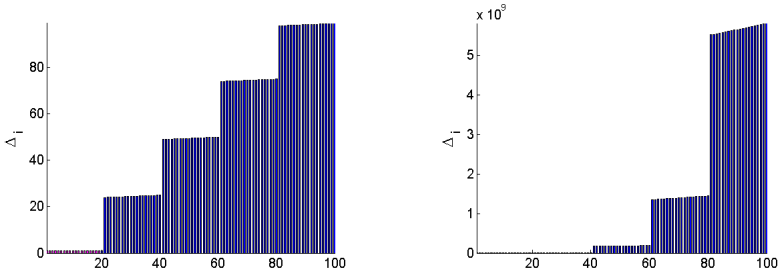

Figure 3.6: Constants in (3.17) for problem 3 with history length $m = 1$ (left plot) and $m = 5$ (right plot).

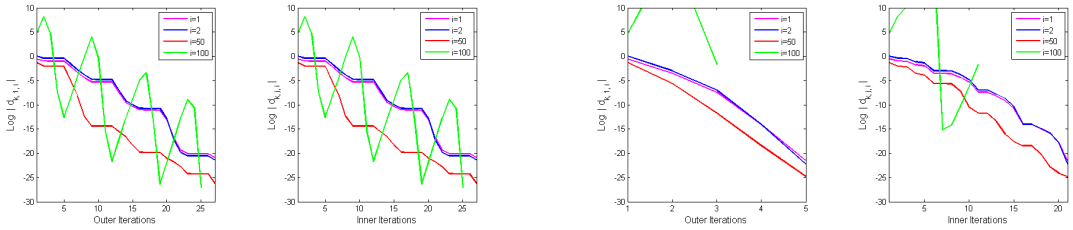

Figure 3.7: Weights in (3.10) for problem 4 with history length $m = 1$ (left two plots) and $m = 5$ (right two plots).

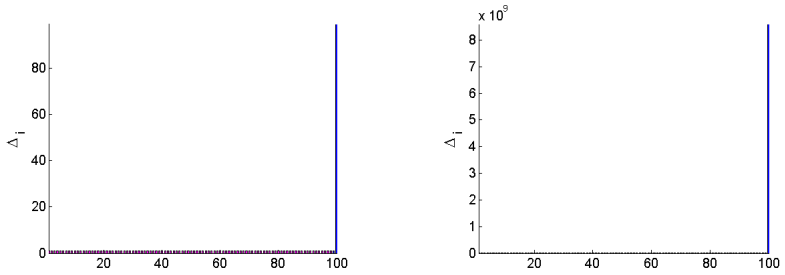

Figure 3.8: Constants in (3.17) for problem 4 with history length $m = 1$ (left plot) and $m = 5$ (right plot).

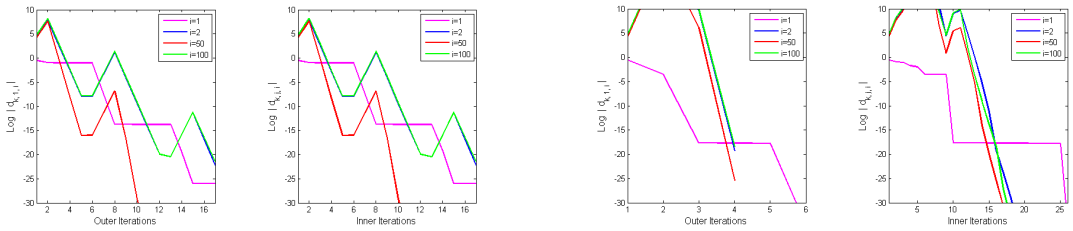

Figure 3.9: Weights in (3.10) for problem 5 with history length $m = 1$ (left two plots) and $m = 5$ (right two plots).

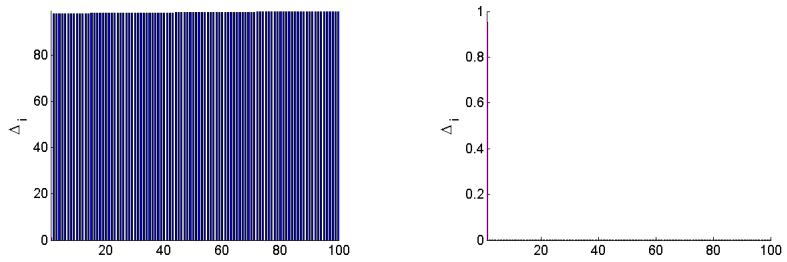

Figure 3.10: Constants in (3.17) for problem 5 with history length $m = 1$ (left plot) and $m = 5$ (right plot).

3.5 LMSD with Harmonic Ritz Values

As explained in §7 of [22], an alternative LMSD method is one that replaces Ritz values of A with harmonic Ritz values; see also [11]. In the case of $m = 1$, this reduces to replacing the "first" with the "second" BB stepsize formula; see (5)–(6) in [2]. In this Chapter, we briefly describe the differences in the computations involved in this alternative approach, then argue that the resulting algorithm is also R-linearly convergent. In fact, much of the analysis in §3.3.2 remains true for this alternative algorithm, so here we only highlight the minor differences.

First, let us briefly review the differences in the computations involved in this alternative algorithm. For this, we follow the description in [22]. Recalling that $T_k = Q_k^T A Q_k$, the Ritz values used in Algorithm LMSD can be viewed as being determined by the eigensystem $(Q_k^T A Q_k)V = (Q_k^T Q_k)V\Theta$, i.e., the solution of this system has $\Theta = \text{diag}(\theta_{k,1}, \ldots, \theta_{k,m})$. Including another instance of A on both sides of this system, one obtains the generalized eigensystem $(Q_k^T A^2 Q_k)V = (Q_k^T A Q_k)V\Theta$, the eigenvalues of which are referred to as harmonic Ritz values of A; see [42]. Defining $P_k := Q_k^T A^2 Q_k$, the eigenvalues are those of $(P_k^{-1} T_k)^{-1}$, which we shall denote as $\{\mu_{k,j}\}_{j\in[m]} \subset \mathbb{R}_{++}$ in decreasing order. The alternative LMSD method is simply Algorithm LMSD with $\{\mu_{k,j}\}_{j\in[m]}$ in place of $\{\theta_{k,j}\}_{j\in[m]}$. As explained in [22], the matrix P_k, like T_k, can be computed with relative little computation and storage, and without explicit access to A. In particular, if G_k has linearly independent columns, one can compute upper triangular $R_k \in \mathbb{R}^{m\times m}$, $r_k \in \mathbb{R}^m$,

and $\xi_k \in \mathbb{R}$ from the Cholesky factorization

$$\begin{bmatrix} G_k & g_{k,m+1} \end{bmatrix}^T \begin{bmatrix} G_k & g_{k,m+1} \end{bmatrix} = \begin{bmatrix} R_k & r_k \\ & \xi_k \end{bmatrix}^T \begin{bmatrix} R_k & r_k \\ & \xi_k \end{bmatrix}, \tag{3.42}$$

then, with J_k again from (3.5), compute

$$P_k \leftarrow R_k^{-T} J_k^T \begin{bmatrix} R_k & r_k \\ & \xi_k \end{bmatrix}^T \begin{bmatrix} R_k & r_k \\ & \xi_k \end{bmatrix} J_k R_k^{-1}. \tag{3.43}$$

One also finds that $P_k = T_k^T T_k + \zeta_k \zeta_k^T$, where $\zeta_k^T = \begin{bmatrix} 0 & \xi_k \end{bmatrix} J_k R_k^{-1}$; see [11].

Let us now argue that this alternative LMSD method is R-linearly convergent. For this, we first show that the harmonic Ritz values satisfy the same property as shown for the Ritz values in Lemma 3.3.6.

Lemma 3.5.1. *Given T_k from (3.8) and P_k from (3.43), the eigenvalues $\{\mu_{k,j}\}_{j\in[m]}$ of $(P_k^{-1}T_k)^{-1}$ satisfy*

$$\mu_{k,j} \in [\lambda_{m+1-j}, \lambda_{n+1-j}] \quad \textit{for all} \quad j \in [m].$$

Proof. One can apply, e.g., Theorem 2.1 from [3] with "K"$= A$, "M"$= I$, and "P"$= Q_k$, the proof of which follows from min-max characterizations of the eigenvalues. □

Given Lemma 3.5.1, one can verify that the results shown in Lemmas 3.3.8 and 3.3.9 also hold for our alternative LMSD method. The result in Lemma 3.3.10 remains true as well, though the argument for this requires a slight addition to the proof. First, we need the following known property that the Ritz and harmonic Ritz values are interlaced; e.g., see Theorem 3.3 in [11].

Lemma 3.5.2. *Given T_k from (3.8) and P_k from (3.43), the eigenvalues $\{\theta_{k,j}\}_{j\in[m]}$ of T_k and the eigenvalues $\{\mu_{k,j}\}_{j\in[m]}$ of $(P_k^{-1}T_k)^{-1}$ are interlaced in the sense that*

$$\mu_{k,1} \geq \theta_{k,1} \geq \mu_{k,2} \geq \theta_{k,2} \geq \cdots \geq \mu_{k,m} \geq \theta_{k,m} > 0.$$

Using this result, let us now argue that Lemma 3.3.10 remains true. Indeed, our previous proof remains valid through the statement of (3.29). Then, combining (3.29) with Lemma 3.5.2, one may conclude that

$$\mu_{k+\hat{k},j} \geq \theta_{k+\hat{k},j} \geq \tfrac{3}{4}\lambda_{p+1} \quad \text{for all} \quad \hat{k} \in \{K_p, \ldots, K_{p+1} - 1\} \quad \text{and} \quad j \in [m].$$

The remainder of the proof then follows as before with $\alpha_{k+\hat{k}+1,j} = \mu^{-1}_{k+\hat{k},j}$ for all $j \in [m]$. A similar modification is needed in the proof of Lemma 3.3.11 since it employs a similar argument as in the proof of Lemma 3.3.10. This way, one can verify that Lemma 3.3.11 remains true for the alternative LMSD method. Finally, as for Lemma 3.3.12 and Theorem 3.3.13, our proofs follow as before without any modifications.

3.6 Conclusion

We have shown that the limited memory steepest descent (LMSD) method proposed by [22] possesses an R-linear rate of convergence for any history length $m \in [n]$ when it is employed to minimize a strongly convex quadratic function. Our analysis effectively extends that in [15], which covers only the $m = 1$ case. We have also provided the results of numerical experiments to demonstrate that the behavior of the algorithm reflects certain properties revealed by the theoretical analysis.

One might wonder whether the convergence rate of LMSD is the same when Ritz values are replaced by harmonic Ritz values; see §7 in [22]. We answer this in the affirmative in Chapter 3.5.

Chapter 4

Handling Nonpositive Curvature in a Limited Memory Steepest Descent Method

We propose a limited memory steepest descent method for solving unconstrained optimization problems. As a steepest descent method, the step computation in each iteration only requires the evaluation of a gradient of the objective function and the calculation of a scalar stepsize. When employed to solve certain convex problems, our method reduces to a variant of the limited memory steepest descent method proposed by Fletcher [22] which means that, when the history length parameter is set to one, it reduces to a steepest descent method inspired by that proposed by Barzilai and Borwein [2]. However, our method is novel in that we propose new algorithmic features for cases when nonpositive curvature is encountered. That is, our method is particularly suited for solving nonconvex problems. With a nonmonotone line search, we ensure global convergence for a variant of our method. We also illustrate with numerical experiments that our approach often yields superior performance when employed to solve nonconvex problems.

4.1 Introduction

Algorithms for finding minimizers of continuously differentiable functions have been the subject of research for centuries. In particular, steepest descent methods—the most basic gradient-based methods—have been the focus of a great deal of work due to their simplicity and effectiveness in many applications. Over the past few decades, great improvements in the practical performance of steepest descent methods have been made simply by the design of clever techniques for choosing the stepsize in each iteration.

In this chapter, we propose a limited memory steepest descent (LMSD) method for solving unconstrained optimization problems whose objective functions are continuously differentiable. Our method is based on the LMSD method recently proposed by Fletcher (see [22]). In a given iteration, this method, by exploiting previously computed gradient information stored as a set of m vectors, computes a sequence of m stepsizes to be employed in a "sweep" over the next m iterations. The calculations involved in determining these stepsizes are motivated by the case of minimizing a strictly convex quadratic form defined by a positive definite matrix A, where with m previously computed gradients one can define a Krylov sequence that provides m estimates of eigenvalues of A. (These estimates, or Ritz values, are contained in the spectrum of A, so Fletcher's method belongs to the class often referred to as spectral gradient descent methods.) In particular, considering the choice of $m = 1$ leads to stepsizes as chosen in the algorithms proposed by Barzilai and Borwein (see [2]), which many consider to be the work responsible for inspiring renewed interest in steepest descent methods.

Many have observed the impressive practical performance of Barzilai-Borwein (BB) methods when solving unconstrained optimization problems. Moreover, in his work, Fletcher illustrates that his approach represents a competitive alternative to the well known limited memory variant of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (see [8, 21, 24, 47]), otherwise known as the L-BFGS method (see [40]). However, in our opinion, these approaches and their proposed enhancements (see §4.2) suffer from the fact that when the objective function is nonconvex, the sophisticated mechanisms designed to compute stepsizes are abandoned, and instead the stepsizes are chosen arbitrarily

(e.g., as prescribed constants). Such choices can lead to poor performance when solving nonconvex problems.

The main contribution of the algorithm proposed in this chapter is that it provides a novel strategy for computing stepsizes when solving nonconvex optimization problems. In particular, when nonpositive curvature (as defined later) is encountered, our method adopts a local cubic model of the objective function in order to determine a stepsize (for $m = 1$) or sequence of stepsizes (for $m > 1$). (As mentioned in §4.2, cubic models have previously been employed in the computation of stepsizes for steepest descent methods. However, in the work that we cite, emphasis was placed on computing stepsizes in convex settings. By contrast, when only positive curvature is encountered, we use a standard quadratic model, as such a choice typically yielded good performance in our experiments. We only employ a cubic model in iterations when nonpositive curvature is present.) As in the case of the original BB methods and the LMSD method of Fletcher, our basic algorithm does not enforce sufficient decrease in the objective in every iteration. However, as is commonly done for variants of BB methods, we remark that, with a nonmonotone line search, a variant of our algorithm attains global convergence guarantees under weak assumptions. Our method also readily adopts the convergence rates attainable by a BB method if/when it reaches a neighborhood of the solution in which the objective is strictly convex (see §4.2).

Overall, our proposed algorithm is designed to strike a balance between multiple (potentially conflicting) goals simultaneously. On one hand, our method preserves the impressive theoretical and practical performance of an LMSD method when nonpositive curvature is not an issue; indeed, when nonpositive curvature is not encountered, our approach reduces to a variant of Fletcher's LMSD method. On the other hand, however, our method is designed to compute and employ meaningful stepsizes when nonpositive curvature is encountered in such a way that (*i*) the cost of the stepsize computation remains negligible, (*ii*) the strategy for handling nonpositive curvature can be generalized to cases when more historical information is maintained and exploited (i.e., when $m > 1$), and, (*iii*) on a diverse set of large-scale test problems, our method yields consistently better perfor-

mance than a method that follows the typical strategy of setting the stepsize to a large prescribed constant when nonpositive curvature is encountered. To achieve these goals, we do not adopt the approach of other authors that attempt to compute accurate higher-order models using, e.g., Hermite interpolation, as such a technique may not shed any light on what may be a reasonable stepsize when nonpositive curvature is encountered, and may not be readily generalized when $m > 1$. Rather, we account for nonpositive curvature in the objective function by observing the difference between two quadratic models obtained using typical LMSD estimation strategies—with which we construct a (bounded below) cubic model for determining a stepsize—in a manner that offers a generalized approach for cases when $m > 1$.

This chapter is organized as follows. In §4.2, we provide a brief summary of the original BB methods and a few of their proposed variants. We also briefly review Fletcher's LMSD algorithm, which can be viewed as another BB method variant/extension. In §4.3, we present the details of our proposed algorithm. We first motivate the ideas underlying our approach by considering the case when, at a given iteration, information from the previous iteration is exploited, and then discuss a generalization of the method for cases when information from any number of previous iterations is maintained and utilized. We discuss the details of an implementation of our method in §4.4, then present the results of numerical experiments in §4.5 which illustrate that our strategies typically yield better performance than some related approaches when solving nonconvex problems. Finally, in §4.6, we present concluding remarks.

The problem that we consider herein is the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{4.1}$$

where $\mathbb{R}^n$ is the set of n-dimensional real vectors (with $\mathbb{R} := \mathbb{R}^1$) and $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable. The algorithms that we discuss are iterative in that, over $k \in \mathbb{N} := \{0, 1, 2, \dots\}$, they produce a sequence $\{x_k\} \subset \mathbb{R}^n$ of iterates, where for each element of the sequence the subscript corresponds to the iteration number in the algorithm. Given an iterate x_k for some $k \in \mathbb{N}$, we define $f_k := f(x_k)$ as the corresponding function

value and $g_k := \nabla f(x_k)$ as the corresponding gradient value. Throughout the chapter, we also apply the subscript k to other quantities that appear in an algorithm during iteration k.

4.2 Literature Review

4.2.1 Barzilai-Borwein Methods

We have described BB methods, their convergence results with and without line search in detail in Chapter 2.5, Chapter 2.6 and Chapter 2.7. An interesting feature of BB methods, even when applied to minimize strictly convex quadratics, is that they are not guaranteed to yield monotonic decreases in the objective function or a typical stationarity measure for problem (4.1). That is, when they converge to a minimizer of f, neither the sequence of function values $\{f_k\}$ nor the sequence of gradient norms $\{\|g_k\|\}$ is guaranteed to decrease monotonically. Hence, a variety of extensions of the original BB methods have been designed that ensure convergence when minimizing general continuously differentiable objective functions by incorporating a nonmonotone line search such as the one proposed by Grippo, Lampariello, and Lucidi (see [29]), or, more recently, the one proposed by Zhang and Hager (see [56]). Extensions of BB methods also typically try to produce better stepsizes by employing higher-order models of f and/or alternating exact line searches (i.e., Cauchy stepsizes) into the iterative sequence (2.11). A few examples are the following. Raydan (see [45]) proposed a globally convergent BB method using the line search of Grippo *et al.*. Dai, Yuan, and Yuan (see [12]) followed this work by proposing interpolation techniques to derive a few alternative stepsizes; they use interpolation to recover the original BB stepsizes and employ a cubic model to derive alternatives. Their methods are also globalized by the line search of Grippo *et al.*. More recently, Yuan (see [54]) proposed the incorporation of Cauchy stepsizes into the iterative process to improve the efficiency of the algorithm, a technique later extended by De Asmundis, Serafino, and Toraldo (see [17]), motivated by work in [18] with their collaborator Riccio) in a monotone gradient scheme. There has also been recent work by Xiao, Wang, and Wang (see [53])

that proposes alternative stepsizes using an alternative secant equation, as well as work by Biglari and Solimanpur (see [5]) that proposes alternative stepsizes derived by fourth-order interpolation models. These later articles employ the nonmonotone line search of Zhang and Hager.

Despite all of the unique features of the BB method variants that have been proposed in the literature, to the best of our knowledge there are no variants that focus on the inefficiencies that may arise when f is nonconvex. (One exception is the recent work by Kafaki and Fatemi (see [33]) that modifies a BB stepsize using a similar strategy as the modified BFGS method proposed by Li and Fukushima (see [35]). However, this strategy is quite different than the strategy proposed in this chapter.) In such cases, the inner product $s_k^T y_k$ may be nonpositive, which must be handled as a special case in all of the algorithms previously described. For example, in [5], [12], [45], and [53], when a nonpositive stepsize is computed, the algorithms revert to setting the stepsize to a large user-defined constant. (In [2], [17], [44], and [54], only convex quadratics are considered, so no strategies are proposed for handling nonpositive curvature.) Such a choice fails to capture any information from the objective function, which may be detrimental to performance.

As a brief illustration of the stepsizes computed in a BB method in which $s_k^T y_k < 0$ implies that one sets α_k to a prescribed positive constant (as in [5], [12], [45], and [53]), consider an arbitrary $k \in \mathbb{N}_+$ and suppose that $g_{k-1} = (-1, 0)$ and $\alpha_{k-1} = 1$ so that $s_k = -\alpha_{k-1} g_{k-1} = (1, 0)$. The contour plots in Figure 4.2.1 illustrate the stepsizes that would be computed as a function of the gradient $g_k \in [-3, 1] \times [-2, 2]$. The plots differ since, on the left (respectively, right), we plot the stepsizes that would be computed when $s_k^T y_k > 0$ implies $\alpha_k \leftarrow 1/\bar{q}_k$ (respectively, $\alpha_k \leftarrow 1/\hat{q}_k$). These plots lead to a few important observations. First, one can observe that when $s_k^T y_k > 0$ and the vectors s_k and y_k are parallel (corresponding to the horizontal axes in the plots), the stepsizes in the two plots are the same since $\bar{q}_k = \hat{q}_k$ in such cases. However, it is interesting to note the stepsizes that result when $s_k^T y_k > 0$ while s_k and y_k are nearly orthogonal: Setting $\alpha_k \leftarrow 1/\bar{q}_k$ leads to extremely large stepsizes, whereas setting $\alpha_k \leftarrow 1/\hat{q}_k$ leads to extremely small

stepsizes. Clearly, the two BB alternatives differ significantly for such g_k. That being said, if one were to employ a globalization mechanism such as a Wolfe line search (see [41]), then a typical strategy would ensure that $s_k^T y_k$ is large in proportion to $\|s_k\|_2^2$. In such an approach, the only values computed in the algorithm would be those illustrated in the regions between the two lines emanating from $(-1, 0)$ drawn in the plots. In these regions, the two BB stepsize alternatives do not reach such extremes, though they still differ substantially for certain values of g_k. Hence, a Wolfe line search can diminish the effect of the differences between these stepsizes, though it should be noted that such a line search can be expensive as it may require many additional function and gradient evaluations. One final (striking) observation about the contours in Figure 4.2.1 is that both strategies fail to exploit any useful information when $s_k^T y_k < 0$. We comment on this further in §4.3.

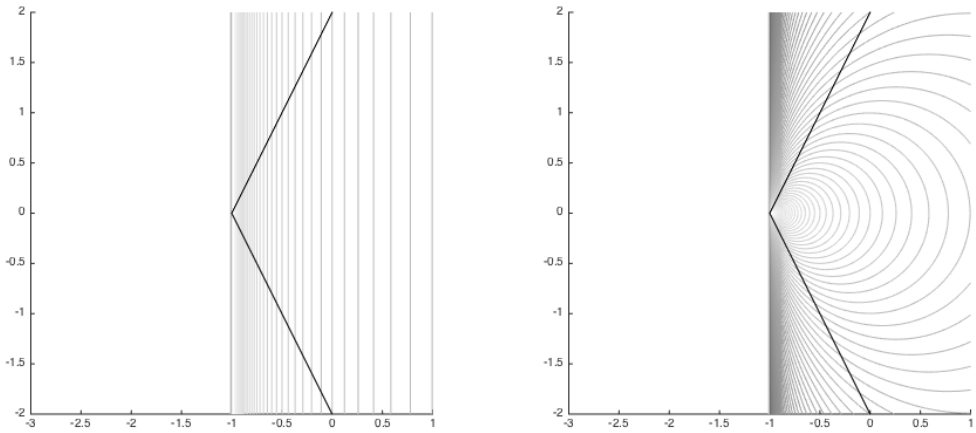

Figure 4.1: Illustration of the stepsizes computed, as a function of the gradient g_k, in an algorithm in which $s_k^T y_k > 0$ implies $\alpha_k \leftarrow 1/\bar{q}_k$ (left) versus one in which $s_k^T y_k > 0$ implies $\alpha_k \leftarrow 1/\hat{q}_k$ (right). In both cases, $s_k^T y_k < 0$ implies α_k is set to a constant; hence, no contour lines appear in the left half of each plot.

4.2.2 Fletcher's limited memory steepest descent (LMSD) method

We described Fletcher's LMSD method in Chapter 2.9, despite the sophisticated mechanisms employed in his stepsize computation procedure, Fletcher admits that his approach leaves unanswered the question of how to handle nonconvexity. (In the case of the two-

point stepsize methods described above, we say that nonpositive curvature is encountered whenever one computes $s_k^T y_k \leq 0$, which can only hold strictly when the objective function is nonconvex. By contrast, in the case of an LMSD method, we say that nonpositive curvature is encountered whenever the matrix whose eigenvalues are used to compute the stepsizes has a nonpositive eigenvalue. It should be noted that nonpositive eigenvalues may arise merely due to non-quadratic features in the objective function f. For ease of exposition, however, we merely refer to the phenomenon of having nonpositive eigenvalues as nonpositive curvature.) In his implementation, Fletcher employs a strategy that carries out a line search whenever a nonpositive stepsize is computed, and then terminates the sweep to effectively throw out previously computed information. By contrast, in our approach, we avoid discarding previously computed information, yet are still able to obtain reasonable stepsizes.

4.3 Algorithm Descriptions

In this Chapter, we present our limited memory steepest descent algorithm. We motivate our method by describing the development of a variant of our approach in which information from only one previous iteration is stored throughout the algorithm. We then present a generalized version of our approach that can exploit information maintained from any number of previous iterations.

4.3.1 An algorithm that stores information from one previous iteration

Suppose that an initial solution estimate x_0 (with $g_0 \neq 0$) and an initial stepsize $\alpha_0 > 0$ are given. Then, after k iterations, we obtain a solution estimate x_k for $k \in \mathbb{N}_+$. At this point, the calculations in the kth iteration of our algorithm are based on the cubic model $m_k : \mathbb{R}^n \to \mathbb{R}$ of f at x_k defined by

$$m_k(s) = f_k + g_k^T s + \tfrac{1}{2} q_k \|s\|_2^2 + \tfrac{1}{6} c_k \|s\|_2^3 \approx f(x_k + s), \tag{4.2}$$

where q_k and c_k are scalars to be determined. In particular, as will be seen in this Chapter, we choose $c_k \geq 0$ in such a way that m_k has a unique minimizer from the origin along its steepest descent direction $-g_k$. As such, we choose the stepsize α_k as an optimal solution of the one-dimensional problem

$$\min_{\alpha \geq 0} \ \phi_k(\alpha), \quad \text{where} \quad \phi_k(\alpha) = m_k(-\alpha g_k) \quad \text{for all} \quad \alpha \in \mathbb{R}. \tag{4.3}$$

A solution of this problem is easily obtained in the cases that will be of interest in our algorithm. In particular, if $q_k > 0$ and $c_k = 0$, then, similar to a basic BB method (recall (2.14)), we have $\alpha_k \leftarrow 1/q_k > 0$; otherwise, if $q_k \leq 0$ and $c_k > 0$, then it is easily verified that (4.3) is solved by setting

$$\alpha_k \leftarrow \frac{2}{q_k + \sqrt{q_k^2 + 2c_k\|g_k\|_2}} > 0. \tag{4.4}$$

We now present our strategies for setting $q_k \in \mathbb{R}$ and $c_k \geq 0$ for a given $k \in \mathbb{N}$. (For simplicity in the majority of our algorithm development, let us suppose that $s_k \neq 0$, $y_k \neq 0$, and $s_k^T y_k \neq 0$; our techniques for handling cases when one or more of these conditions does not hold will be considered later.) First, consider q_k. Defining θ_k as the angle between s_k and y_k, two options for q_k come from (2.13):

$$\begin{aligned} \bar{q}_k &:= \frac{s_k^T y_k}{s_k^T s_k} = \cos(\theta_k)\frac{\|y_k\|_2}{\|s_k\|_2} \\ \text{and} \quad \hat{q}_k &:= \frac{y_k^T y_k}{s_k^T y_k} = \frac{1}{\cos(\theta_k)}\frac{\|y_k\|_2}{\|s_k\|_2}. \end{aligned} \tag{4.5}$$

Through these representations, it is clear that $|\bar{q}_k| \leq |\hat{q}_k|$, and hence the quantities in (2.14) satisfy $|\bar{\alpha}_k| \geq |\hat{\alpha}_k|$. Indeed, even though both $\bar{q}_k I$ and $\hat{q}_k I$ are valid approximations of the Hessian of f along $[x_{k-1}, x_k]$, it can be seen that $\bar{q}_k$ only estimates the curvature of f by observing the change in its gradient along the line segment $[x_{k-1}, x_k]$, whereas $\hat{q}_k$ actually accounts for changes in the gradient along an orthogonal vector as well. To see

this, let

$$u_k := \left(\frac{s_k^T y_k}{s_k^T s_k}\right) s_k \quad \text{and} \quad v_k := y_k - u_k \quad \text{so that} \quad y_k = u_k + v_k,$$

i.e., we define u_k to be the vector projection of y_k onto $\text{span}(s_k)$, which implies that we have $s_k^T y_k = s_k^T u_k$ and $s_k^T v_k = 0$. We then find from (2.13) that

$$\begin{aligned} \bar{q}_k &:= \frac{s_k^T y_k}{s_k^T s_k} = \frac{s_k^T u_k}{s_k^T s_k} \\ \text{and} \quad \hat{q}_k &:= \frac{y_k^T y_k}{s_k^T y_k} = \frac{u_k^T u_k + v_k^T v_k}{s_k^T u_k} = \bar{q}_k + \frac{v_k^T v_k}{s_k^T u_k}, \end{aligned} \tag{4.6}$$

where the last equation follows since $s_k^T u_k / s_k^T s_k = u_k^T u_k / s_k^T u_k$. Through these representations, it is clear that $\bar{q}_k$ is unaffected by v_k (i.e., the component of y_k orthogonal to s_k), whereas $\hat{q}_k$ takes the magnitude of this vector into account. Comparing these representations to those in (4.5) and recalling that u_k is parallel to s_k, one observes that if $v_k = 0$, then $\bar{q}_k = \hat{q}_k$, whereas if $v_k \neq 0$, then $|\bar{q}_k| < |\hat{q}_k|$. Overall, these observations provide a clearer understanding of the differing contour lines illustrated in Figure 4.2.1.[1]

In our approach, we could follow the common strategy of setting $q_k \leftarrow \bar{q}_k$ or $q_k \leftarrow \hat{q}_k$, or even choose randomly between these two options based on some probability distribution. For various reasons, we choose to always set $q_k \leftarrow \hat{q}_k$. This reasoning can be explained by considering the two cases depending on the sign of the inner product $s_k^T y_k$. If $s_k^T y_k > 0$, then we set $q_k \leftarrow \hat{q}_k$ primarily due to the fact that, when $s_k^T y_k \approx 0$, this leads to smaller (i.e., more conservative) stepsizes. Indeed, this will be consistent with our preference for choosing a small stepsize in the extreme case when $s_k^T y_k = 0$ (as explained in our later discussion of handling special cases). On the other hand, when $s_k^T y_k < 0$ and $s_k^T y_k \approx 0$, then setting $q_k \leftarrow \hat{q}_k$ corresponds to an extremely large *negative* quadratic coefficient, which has the potential to cause (4.14) to yield large stepsizes. This would be inconsistent with our choice of having smaller stepsizes when $s_k^T y_k > 0$ and $s_k^T y_k \approx 0$! Hence, when $s_k^T y_k < 0$, we set $q_k \leftarrow \hat{q}_k$, but will rely on a nonzero cubic term to lead the algorithm to

[1] These observations have motivated our choice of notation. In particular, the "bar" quantities are computed based on information *straight* along s_k, whereas the "hat" quantities are computed based on information along s_k plus an orthogonal direction—a vector that can be visualized by the *bent* line in the "hat" over the corresponding quantities.

computing a reasonable stepsize, as explained next.

With q_k fixed at $\hat{q}_k$, consider c_k. If $s_k^T y_k > 0$, then, as mentioned, a reasonable choice is $c_k \leftarrow 0$, since then m_k is strictly convex from the origin along $-g_k$. On the other hand, if $s_k^T y_k < 0$, then we desire an intuitive, meaningful strategy for choosing $c_k > 0$ so that problem (4.3) has a unique minimizer. We examine two possible strategies, both of which lead to a similar conclusion:

- Consider choosing c_k to minimize the least squared error between the gradient of the model m_k at $-s_k$ (corresponding to the previous point x_{k-1}) and the previous gradient g_{k-1}, i.e.,

$$\tfrac{1}{2}\|\nabla m_k(-s_k) - g_{k-1}\|_2^2. \tag{4.7}$$

  Differentiating m_k, we have for all $s \in \mathbb{R}^n$ that

$$\nabla m_k(s) = g_k + q_k s + \tfrac{1}{2} c_k \|s\|_2 s. \tag{4.8}$$

  It can then easily be verified that one minimizes (4.7) by choosing

$$c_k \leftarrow \frac{2}{\|s_k\|_2}(\bar{q}_k - q_k). \tag{4.9}$$

- Consider choosing c_k so that the curvature of m_k at $-s_k$ along s_k is equal to $\bar{q}_k$, i.e., so that

$$s_k^T \nabla^2 m_k(-s_k) s_k = \bar{q}_k \|s_k\|_2^2. \tag{4.10}$$

  This is a reasonable goal since, in a BB method, it is established that $\bar{q}_k I$ is a sensible approximation of the Hessian of f along $[x_{k-1}, x_k]$, and in particular at x_{k-1} (i.e., the point corresponding to m_k evaluated at $-s_k$) along s_k. Differentiating ∇m_k (recall (4.8)), we have for all $s \in \mathbb{R}^n$ that

$$\nabla^2 m_k(s) = q_k I + \frac{1}{2} c_k \left(\|s\|_2 I + \frac{1}{\|s\|_2} s s^T \right).$$

Hence, we obtain (4.10) by setting

$$c_k \leftarrow \frac{1}{\|s_k\|_2} (\bar{q}_k - q_k) . \tag{4.11}$$

The similarity between (4.9) and (4.11) is immediately apparent, as they only differ by a constant factor. Overall, we propose that the cubic term coefficient should be set, for some constant $c > 0$, as

$$c_k \leftarrow \frac{c}{\|s_k\|_2} (\bar{q}_k - q_k) . \tag{4.12}$$

Using the notation of (4.5), if $s_k^T y_k < 0$ and $\cos(\theta_k) \neq 1$, then this formula yields $c_k > 0$. Similarly, using the notation of (4.6), if $s_k^T u_k = s_k^T y_k < 0$ and $v_k^T v_k > 0$, then $c_k > 0$. Overall, one can see that we have taken the curvature that has been captured orthogonal to s_k and have used it to determine an appropriate magnitude of a cubic term so that (4.3) has a unique minimizer. A relatively large discrepancy between $\bar{q}_k$ and $q_k = \hat{q}_k$ indicates a relatively large displacement in the gradient orthogonal to s_k, which in turn suggests a relatively large cubic term coefficient should be used to safeguard the next stepsize. One may also observe that (4.11) is particularly appealing in the sense that it represents a finite-difference approximation of a third-order (directional) derivative using the difference between the two available second-order (directional) derivative estimates for the interval $[x_{k-1}, x_k]$.

We are almost prepared to present a complete description of our algorithm (for $m = 1$), but first we must remark on the special cases that we have ignored until this point. That is, we must specify how the algorithm is to proceed when $s_k = 0$, $y_k = 0$, $s_k^T y_k = 0$, or $s_k^T y_k < 0$ while s_k and y_k are parallel. In fact, as long as the algorithm terminates in any iteration $k \in \mathbb{N}$ for which $g_k = 0$, and otherwise computes a stepsize $\alpha_k > 0$, the algorithm cannot produce $s_k = 0$. Hence, we need only consider the computation of stepsizes when $s_k \neq 0$, so the only special cases that remain to be considered are as follows.

- If $y_k = 0$, then the step from x_{k-1} to x_k has yielded a zero displacement in the gradient of f. Consequently, between the points x_{k-1} and x_k, we have no useful information to approximate the Hessian of f; in fact, based on the relationship

between gradients at these points, f "appears affine" at x_k along $-g_k = -g_{k-1}$. In such cases, we set α_k to a maximum allowable stepsize, call it $\Omega > 0$, in an attempt to aggressively minimize f along the steepest descent direction $-g_k$.

- If $y_k \neq 0$, but $s_k^T y_k = 0$, then the displacement from x_{k-1} to x_k has yielded a nonzero displacement in the gradient of f, but this displacement is orthogonal to $s_k = x_k - x_{k-1}$. Hence, as there has been no displacement of the gradient in a direction parallel to the displacement in the iterate, a two-point stepsize approximation of the Hessian of f at x_k is inadequate. Thus, since the next iteration will involve exploring f along $-g_k \neq -g_{k-1}$, in this "new" direction we conservatively set α_k to a minimum allowable stepsize, call it $\omega > 0$ (with $\omega \leq \Omega$).

- If $y_k \neq 0$, $s_k^T y_k < 0$, and s_k and y_k are parallel, then the displacement from x_{k-1} to x_k has only yielded a nonzero displacement in the gradient of f in the direction of s_k. Consequently, similar to cases when $y_k = 0$, we have no useful information to approximate the Hessian of f in any direction other than s_k; in fact, based on the relationship between the gradients at these points, f "appears affine" at x_k along any direction other than s_k. In such cases, since $-g_k$ is parallel to s_k, we set α_k to the large stepsize $\Omega > 0$ to try to aggressively minimize f along $-g_k$.

We are now prepared to provide a complete description of our first approach, given as Algorithm 5. Along with the safeguards employed in the special cases discussed above, we employ the universal safeguard of projecting any computed stepsize onto the interval $[\omega, \Omega]$. For simplicity in our description, we omit mention of the computation of function and gradient values, as well as of the displacement vectors in (2.12); these are implied whenever a new iterate is computed. Furthermore, we suppress any mention of a termination condition, but remark that any practical implementation of our algorithm would terminate as soon as a gradient is computed that has a norm that is approximately zero. Hence, in the algorithm, we assume for all practical purposes that $g_k \neq 0$ for all $k \in \mathbb{N}$.

Algorithm 5 LMSD Method with Cubic Regularization (for $m = 1$)

choose $(\omega, \Omega) \in \mathbb{R} \times \mathbb{R}$ satisfying $0 < \omega \leq \Omega$ and $c \in \mathbb{R}_+ := \{c \in \mathbb{R} : c > 0\}$
choose $x_0 \in \mathbb{R}^n$ and $\alpha_0 \in [\omega, \Omega]$
set $x_1 \leftarrow x_0 - \alpha_0 g_0$ and $k \leftarrow 1$
loop
    if $y_k = 0$ or $s_k^T y_k = -\|s_k\|_2 \|y_k\|_2 < 0$ **then**
        set $\alpha_k \leftarrow \Omega$
    else if $s_k^T y_k = 0$ **then**
        set $\alpha_k \leftarrow \omega$
    else
        set $\bar{q}_k \leftarrow s_k^T y_k / s_k^T s_k$ and $q_k \leftarrow y_k^T y_k / s_k^T y_k$
        if $q_k > 0$ **then** set $c_k \leftarrow 0$ **else** set $c_k \leftarrow c(\bar{q}_k - q_k)/\|s_k\|_2$
        if $q_k > 0$ **then** set $\alpha_k \leftarrow 1/q_k$ **else** set $\alpha_k \leftarrow 2/(q_k + \sqrt{q_k^2 + 2c_k\|g_k\|_2})$
        replace α_k by its projection onto the interval $[\omega, \Omega]$
    end if
    set $x_{k+1} \leftarrow x_k - \alpha_k g_k$ and $k \leftarrow k + 1$
end loop

We close this Chapter by providing an illustration of the types of stepsizes computed in Algorithm 5, which may be compared to those illustrated in Figure 4.2.1. Using the same set-up as for Figure 4.2.1 and with $c \leftarrow 1$, we plot in Figure 4.3.1 the stepsizes computed via Algorithm 5 as a function of the gradient $g_k \in [-3, 1] \times [-2, 2]$. In this plot, it is clear that, when $s_k^T y_k > 0$, Algorithm 5 computes stepsizes that are equal to those in the plot on the right in Figure 4.2.1. More important, however, is that Algorithm 5 computes reasonable stepsizes that exploit problem information even when $s_k^T y_k < 0$. In particular, when $s_k^T y_k < 0$ and $s_k^T y_k \approx 0$, the algorithm computes stepsizes consistent with those computed in "nearby cases" when $s_k^T y_k > 0$. On the other hand, if $s_k^T y_k < 0$ while s_k and y_k are (nearly) parallel, then the algorithm computes very large stepsizes, which has been motivated in the third special case above.

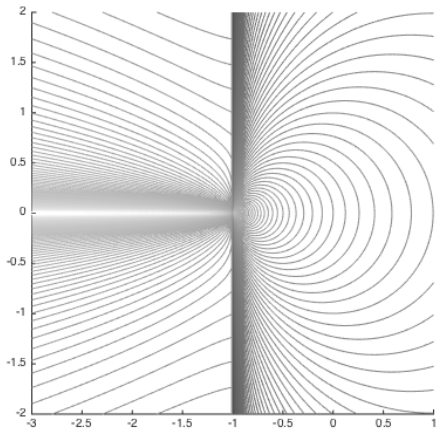

Figure 4.2: Illustration of the stepsizes computed, as a function of the current gradient g_k, by Algorithm 5.

4.3.2 An algorithm that stores information from $m \geq 1$ previous iteration(s)

We have presented our algorithm for $m = 1$ as one that, at x_k, computes a stepsize based on minimizing a cubic model of the objective from x_k along the steepest descent direction $-g_k$. On the other hand, in §4.2, we described Fletcher's LMSD method for $m \geq 1$ as one that, at x_k, computes m stepsizes to be employed in the next m iterations by computing (reciprocals of) the eigenvalues of an $m \times m$ matrix. On the surface, these approaches are quite different. Therefore, in order to extend our approach to cases when $m \geq 1$ and have it reduce to (a variant of) Fletcher's LMSD method, we must explain how Fletcher's stepsize computation procedure can be understood in terms of minimizing a local model of f at x_k. This can be done with the use of a quadratic model, but for our purposes of generalizing the approach, we employ a cubic model (and refer to his approach as one in which the cubic term is zero).

For a given $j \in \{0, \ldots, m-1\}$, consider the cubic model $m_{k+j} : \mathbb{R}^n \to \mathbb{R}$ of f at x_{k+j} defined by

$$m_{k+j}(s) = f_{k+j} + g_{k+j}^T s + \tfrac{1}{2} q_{k+j} \|s\|_2^2 + \tfrac{1}{6} c_{k+j} \|s\|_2^3 \approx f(x_{k+j} + s), \tag{4.13}$$

where q_{k+j} and c_{k+j} are scalars to be determined. Furthermore, consider the problem

$$\min_{\alpha \geq 0} \ \phi_{k+j}(\alpha), \quad \text{where} \quad \phi_{k+j}(\alpha) := m_{k+j}(-\alpha g_{k+j}) \quad \text{for all} \quad \alpha \in \mathbb{R}. \tag{4.14}$$

It is easily seen that when $q_{k+j} > 0$ and $c_{k+j} = 0$, the solution of problem (4.14) is given by $\alpha_{k+j} = q_{k+j}^{-1}$, while if $q_{k+j} \leq 0$ and $c_{k+j} > 0$, then the solution is given by a formula similar to (4.4).

Supposing that iteration k represents the beginning of a "sweep," Fletcher's approach can be understood in terms of minimizing the local model of f at x_{k+j} given by (4.13) *simultaneously* for all $j \in \{0, \ldots, m-1\}$, despite the fact that, for $j > 0$, the point x_{k+j} and gradient g_{k+j} are unknown at the beginning of iteration k. In particular, his approach involves the construction of an $m \times m$ matrix, call it M_k, based on information obtained from the m iterations prior to, and including, iteration k. He computes the eigenvalues of M_k, say composing $\{q_k, \ldots, q_{k+m-1}\}$, the reciprocals of which, say composing $\{\alpha_k, \ldots, \alpha_{k+m-1}\}$, are to be used as the stepsizes in the next m iterations. In fact, Fletcher orders these stepsizes from smallest to largest before employing them in the algorithm.

Our approach proceeds in a similar manner, but with some notable differences. Specifically, at the beginning of a "sweep" occurring from iteration k, we compute scalars that may be used for q_{k+j} for all $j \in \{0, \ldots, m-1\}$. However, we do not simply use the reciprocals of these values as the stepsizes in the subsequent m iterations, especially since some or all of these values may be negative. Instead, we iterate in the usual manner through iterations $k+j$ for all $j \in \{0, \ldots, m-1\}$, where for each such j we compute $c_{k+j} \geq 0$ such that m_{k+j} is bounded below over $-g_{k+j}$ and (4.14) yields a unique minimizer.

For computing the quadratic term coefficients, we follow the approach of Fletcher and assume that, at the beginning of iteration k, we have available an invertible symmetric tridiagonal matrix $\tilde{T}_k \in \mathbb{R}^{m \times m}$ and an invertible symmetric pentadiagonal matrix $\tilde{P}_k \in \mathbb{R}^{m \times m}$. (It is possible that our formulas for these matrices, provided later, yield matrices that are not invertible, but we handle these as special cases later on.) We then define the

following sets, each ordered from largest to smallest:

$$\begin{aligned} &\{\bar{q}_k, \ldots, \bar{q}_{k+m-1}\} \text{ as the eigenvalues of } \tilde{T}_k \\ \text{and } &\{\hat{q}_k, \ldots, \hat{q}_{k+m-1}\} \text{ as the eigenvalues of } (\tilde{P}_k^{-1}\tilde{T}_k)^{-1}. \end{aligned} \tag{4.15}$$

At iteration $k+j$ for $j \in \{0, \ldots, m-1\}$, we follow the strategy of the $m = 1$ case and set $q_{k+j} \leftarrow \hat{q}_{k+j}$. If $\hat{q}_{k+j} > 0$, then we set $c_{k+j} \leftarrow 0$, but otherwise we again follow the strategy of the $m = 1$ case and set

$$c_{k+j} \leftarrow \frac{c}{\|s_{k+j}\|_2}(\bar{q}_{k+j} - q_{k+j}). \tag{4.16}$$

Observe that when $m = 1$, the approach described in the previous paragraph reduces to that in §4.3.1, and in that case we have $\bar{q}_{k+j} - q_{k+j} > 0$ for $j = 0$ whenever $s_{k+j}^T y_{k+j} < 0$ and y_{k+j} is not parallel to s_{k+j}. However, for $m > 1$, we must ensure that (4.16) yields $c_{k+j} \geq 0$. In fact, this is guaranteed by our approach for constructing $\tilde{T}_k$ and $\tilde{P}_k$, but, before describing this construction, we remark the following.

- If $m = 1$, $\tilde{T}_k = \bar{q}_k$, and $\tilde{P}_k = \|y_k\|_2^2/\|s_k\|_2^2$, then $(\tilde{P}_k^{-1}\tilde{T}_k)^{-1} = \hat{q}_k$ and (4.16) reduces to (4.12).

- If $m \geq 1$ and $f(x) = \frac{1}{2}x^T Ax$ for some $A \succ 0$, then with $\tilde{T}_k \leftarrow T_k$ from (2.16) and $\tilde{P}_k \leftarrow P_k$ from (2.18), we have that $\tilde{T}_k$ is symmetric tridiagonal and $\tilde{P}_k$ is symmetric pentadiagonal. Furthermore, in this case, we have $T_k \succ 0$ and $(P_k^{-1}T_k)^{-1} \succ 0$, meaning that our approach for computing stepsizes reduces to Fletcher's method when harmonic Ritz values are employed. On the other hand, when $\tilde{T}_k$ and $\tilde{P}_k$ are set in the same manner but $A \not\succ 0$, the eigenvalues of $\tilde{T}_k$ and $(\tilde{P}_k^{-1}\tilde{T}_k)^{-1}$ will be interlaced. In such a case, (4.16) involves $\bar{q}_{k+j} - q_{k+j} \geq 0$ for any $j \in \{0, \ldots, m-1\}$ with $\bar{q}_{k+j} < 0$.

A critical feature of our algorithm is how we choose $\tilde{T}_k$ and $\tilde{P}_k$ when f is not quadratic. In Fletcher's work, he remarks that, in the nonquadratic case, the matrix T_k in (2.16) will be upper Hessenberg, but not necessarily tridiagonal. He handles this by constructing $\tilde{T}_k$,

which is set to T_k except that its strict upper triangle is replaced by the transpose of its strict lower triangle, thus creating a tridiagonal matrix. We employ the same strategy in our algorithm. Then, observing from (2.18) that

$$P_k = T_k^T T_k + \zeta_k \zeta_k^T, \quad \text{where} \quad \zeta_k^T := \begin{bmatrix} 0 & \rho_k \end{bmatrix} J_k R_k^{-1}, \tag{4.17}$$

we set $\tilde{P}_k$ as $\tilde{T}_k^T \tilde{T}_k + \zeta_k \zeta_k^T$, i.e., we use the same expression as in (4.17), but with T_k replaced by $\tilde{T}_k$.

The strategy described in the previous paragraph is well-defined if T_k in (2.16) is well-defined, and (4.16) ensures $c_{k+j} \geq 0$ for all $k \in \mathbb{N}_+$ and $j \in \{0, \ldots, m-1\}$ if the eigenvalues of $\tilde{T}_k$ and $(\tilde{P}_k^{-1} \tilde{T}_k)^{-1}$ are interlaced. For these cases, we provide the following theorems with proofs.

Theorem 4.3.1. *The matrix T_k in (2.16) is well-defined if and only if the columns of G_k are linearly independent. Furthermore, with $\alpha_k > 0$ for all $k \in \mathbb{N}$, the matrix T_k is invertible if and only if the columns of G_k are linearly independent and the elements of the vector $R_k^{-1} Q_k^T g_k$ do not sum to one.*

Proof of Theorem 4.3.1. The matrix R_k in the (thin) QR-factorization of G_k is invertible if and only if the columns of G_k are linearly independent (see [26]), which is the first part of the theorem. Now, for the remainder of the proof, we may proceed under the assumption that the columns of G_k are linearly independent (which implies that R_k is inverible). Since R_k is invertible, we have

$$\begin{aligned} T_k &= \begin{bmatrix} R_k & r_k \end{bmatrix} J_k R_k^{-1} \\ &= \begin{bmatrix} Q_k^T Q_k R_k & Q_k^T Q_k r_k \end{bmatrix} J_k R_k^{-1} \\ &= Q_k^T \begin{bmatrix} G_k & g_k \end{bmatrix} J_k R_k^{-1} \\ &= R_k^{-T} G_k^T \begin{bmatrix} G_k & g_k \end{bmatrix} J_k R_k^{-1}, \end{aligned}$$

which implies that T_k is invertible if and only if $G_k^T[G_k\ g_k]J_k$ is invertible. We find that

$$G_k^T\begin{bmatrix}G_k & g_k\end{bmatrix}J_k = G_k^T\begin{bmatrix}\alpha_{k-m}^{-1}(g_{k-m+1}-g_{k-m}) & \cdots & \alpha_{k-1}^{-1}(g_k-g_{k-1})\end{bmatrix},$$

which, along with the fact that $\alpha_k > 0$ for all $k \in \mathbb{N}$, implies that $G_k^T[G_k\ g_k]J_k$ is invertible if and only if

$$G_k^T\begin{bmatrix}g_k-g_{k-m} & \cdots & g_k-g_{k-1}\end{bmatrix} = G_k^T(\begin{bmatrix}g_k & \cdots & g_k\end{bmatrix} - G_k) \tag{4.18}$$

is invertible. Since G_k has linearly independent columns, we have by properties of determinants that

$$\begin{aligned}\det(G_k^T(\begin{bmatrix}g_k & \cdots & g_k\end{bmatrix} - G_k)) &= \det(G_k^T\begin{bmatrix}g_k & \cdots & g_k\end{bmatrix} - G_k^TG_k)\\ &= \det(G_k^TG_k)\det((G_k^TG_k)^{-1}G_k^T\begin{bmatrix}g_k & \cdots & g_k\end{bmatrix} - I),\end{aligned}$$

from which it follows that (4.18) is invertible if and only if

$$1 \neq \operatorname{trace}((G_k^TG_k)^{-1}G_k^T\begin{bmatrix}g_k & \cdots & g_k\end{bmatrix}) = \operatorname{trace}(R_k^{-1}Q_k^T\begin{bmatrix}g_k & \cdots & g_k\end{bmatrix}),$$

which is true if and only if the elements of $R_k^{-1}Q_k^Tg_k$ do not sum to one. □

Theorem 4.3.2. *Suppose that T_k in (2.16) is well-defined. Then, let $\tilde{T}_k$ be set equal to T_k, except that its strict upper triangle is replaced by the transpose of its strict lower triangle, and let $\tilde{P}_k \leftarrow \tilde{T}_k^T\tilde{T}_k + \zeta_k\zeta_k^T$ where ζ_k is defined in (4.17). Then, $\tilde{T}_k$ is symmetric tridiagonal and $\tilde{P}_k$ is symmetric pentadiagonal.*

Proof of Theorem 4.3.2. Letting $R_k^{(j)}$ denote the jth column of R_k for all $j \in \{1,\ldots,m\}$, we have

$$\begin{bmatrix}R_k & r_k\end{bmatrix}J_k = \begin{bmatrix}\alpha_{k-m}^{-1}(R_k^{(1)}-R_k^{(2)}) & \cdots & \alpha_{k-2}^{-1}(R_k^{(m-1)}-R_k^{(m)}) & \alpha_{k-1}^{-1}(R_k^{(m)}-r_k)\end{bmatrix}. \tag{4.19}$$

Since R_k is upper triangular, it follows that (4.19) is upper Hessenberg and that R_k^{-1} is

upper triangular. Then, since the product of an upper Hessenberg and an upper triangular matrix is upper Hessenberg, it follows that T_k is upper Hessenberg. Thus, our construction of $\tilde{T}_k$ ensures that it is symmetric tridiagonal.

Now consider $\tilde{P}_k = \tilde{T}_k^T \tilde{T}_k + \zeta_k \zeta_k^T$ where $\zeta_k^T := \begin{bmatrix} 0 & \rho_k \end{bmatrix} J_k R_k^{-1}$. For any pair of indices $\{i, j\} \subseteq \{1, \ldots, m\}$ such that $|i - j| > 2$, we have $[\tilde{T}_k^T \tilde{T}_k]_{i,j} = 0$, from which it follows that $\tilde{T}_k^T \tilde{T}_k$ is symmetric pentadiagonal. Moreover, by the structure of J_k and the fact that R_k^{-1} is upper triangular, it follows that $\zeta_k \zeta_k^T$ is zero except for its (m, m) entry, which overall implies that $\tilde{P}_k$ is symmetric pentadiagonal. □

Theorem 4.3.3. *Suppose that T_k in (2.16) is well-defined and that the matrices $\tilde{T}_k$ and $\tilde{P}_k$, constructed as described in Theorem 4.3.2, are invertible. Then, the eigenvalues in (4.15) satisfy*

$$|\bar{q}_{k+j}| \leq |\hat{q}_{k+j}| \quad \text{for all} \quad j \in \{0, \ldots, m-1\}.$$

In particular, if for some $j \in \{0, \ldots, m-1\}$ one has $\hat{q}_{k+j} < 0$, then (4.16) yields $c_{k+j} \geq 0$.

Proof of Theorem 4.3.3. For ease of exposition in this proof, we drop the tilde and iteration subscript from all quantities of interest; in particular, we let $T = \tilde{T}_k$, $P = \tilde{P}_k$, and $\zeta = \zeta_k$. Since T is invertible, there exists an orthogonal matrix V such that $T = V \Lambda V^T$ where Λ is a diagonal matrix whose diagonal elements are the (nonzero) eigenvalues of T; in particular, for some nonnegative integers p and q with $p + q = m$, we have $\Lambda = \operatorname{diag}(a, -b)$ for some positive vectors $a \in \mathbb{R}^p$ and $b \in \mathbb{R}^q$. Without loss of generality, we assume that $a_1 \geq \cdots \geq a_p > 0$ and $b_1 \geq \cdots \geq b_q > 0$. Letting $w = V^T \zeta$, we have

$$\begin{aligned} V^T (P^{-1} T)^{-1} V &= V^T (T^{-1} (T^T T + \zeta \zeta^T)) V \\ &= \Lambda + V^T T^{-1} \zeta \zeta^T V \\ &= \Lambda + \Lambda^{-1} V^T \zeta \zeta^T V \\ &= \Lambda + \Lambda^{-1} w w^T. \end{aligned}$$

Thus, denoting by $|\Lambda|$ the diagonal matrix whose diagonal elements are the absolute values

of the elements of Λ, we have for some vectors $c \in \mathbb{R}^p$ and $d \in \mathbb{R}^m$ that

$$\begin{aligned} V^T(P^{-1}T)^{-1}V &= |\Lambda|^{-1/2}\Lambda|\Lambda|^{1/2} + |\Lambda|^{-1/2}(|\Lambda|^{1/2}\Lambda^{-1}ww^T|\Lambda|^{-1/2})|\Lambda|^{1/2} \\ &= |\Lambda|^{-1/2}\left(\begin{bmatrix} \mathrm{diag}(a) & 0 \\ 0 & \mathrm{diag}(-b) \end{bmatrix} + \begin{bmatrix} c \\ -d \end{bmatrix}\begin{bmatrix} c \\ d \end{bmatrix}^T\right)|\Lambda|^{1/2} \\ &= |\Lambda|^{-1/2}\left(\begin{bmatrix} \mathrm{diag}(a) + cc^T & cd^T \\ -dc^T & \mathrm{diag}(-b) - dd^T \end{bmatrix}\right)|\Lambda|^{1/2}. \end{aligned}$$

It follows that the eigenvalues of $(P^{-1}T)^{-1}$ are the same as those for

$$\Omega := \begin{bmatrix} \mathrm{diag}(a) + cc^T & cd^T \\ -dc^T & \mathrm{diag}(-b) - dd^T \end{bmatrix}.$$

Now, for $i \in \{1, \ldots, p\}$, let e_j denote the jth unit vector in $\mathbb{R}^p$ and note that, for all $j \in \{1, \ldots, i\}$,

$$\begin{bmatrix} e_j \\ 0 \end{bmatrix}^T (\Omega - a_i I)\begin{bmatrix} e_j \\ 0 \end{bmatrix} = (a_j + c_j^2) - a_i \geq 0.$$

We may conclude from this fact that, for all $i \in \{1, \ldots, p\}$, the matrix $\Omega - a_i I$ has at least i positive eigenvalue(s). Similarly, for a given $i \in \{1, \ldots, q\}$ and with $\bar{e}_j$ denoting the jth unit vector in $\mathbb{R}^q$, we have for all $j \in \{1, \ldots, i\}$ that

$$\begin{bmatrix} 0 \\ \bar{e}_j \end{bmatrix}^T (\Omega + b_i I)\begin{bmatrix} 0 \\ \bar{e}_j \end{bmatrix} = (-b_j - d_j^2) + b_i \leq 0,$$

from which it follows that $\Omega + b_i I$ has at least i negative eigenvalue(s). Recalling that $(a, -b)$ are the eigenvalues of T, we conclude that the eigenvalues of $(P^{-1}T)^{-1}$, call them $(\bar{a}, -\bar{b})$, satisfy

$$-\bar{b}_1 \leq -b_1 \leq \cdots \leq -\bar{b}_q \leq -b_q < 0 < a_p \leq \bar{a}_p \leq \cdots \leq a_1 \leq \bar{a}_1,$$

as desired. □

Overall, like Fletcher's method, our strategy reduces to using Ritz and harmonic Ritz values in the quadratic case, and otherwise manipulates the matrices in (2.16) and (2.18) to obtain matrices with similar structure as would be obtained automatically in the quadratic case.

We are almost prepared to discuss our main algorithm, but first we must discuss the special cases that must be considered for our algorithm to be well-defined.

- Suppose that the columns of G_k are linearly dependent, or the columns of G_k are linearly independent while the elements of $R_k^{-1}Q_k^T g_k$ sum to one. In such cases, T_k is not well-defined, so our desired procedure for constructing $\tilde{T}_k$ and $\tilde{P}_k$ also is not well-defined. To handle this, we iteratively consider fewer previous gradients (where, at each stage, the eldest member is ignored in favor of newer gradients) until the set of considered previous gradients consists of linearly independent vectors for which $R_k^{-1}Q_k^T g_k$ has elements that do not sum to one. (Note that the situation in which G_k has linearly dependent columns did not need to be considered when $m = 1$ since, in that case, $G_k = [g_{k-1}]$ having a linearly dependent column corresponds to $g_{k-1} = 0$, in which case Algorithm 5 would have terminated in iteration $k - 1$. Also note that in the $m = 1$ case, having $1 = R_k^{-1}Q_k^T g_k$ corresponds to the special case of having $y_k = 0$.) In the extreme case when the set of previously computed gradients is reduced to only the most recently computed previous gradient, we compute a stepsize as in Algorithm 5.

- Suppose that T_k is well defined, but for some $q_{k+j} \leq 0$ we obtain $c_{k+j} = 0$. This is similar to the last of the special cases considered when $m = 1$, and so, as in that case, we aggressively minimize f by computing a stepsize as $\Omega > 0$.

Our main approach is presented as Algorithm 6. As for Algorithm 5, we suppress mention of the computation of function values, gradient values, and displacement vectors, and suppress mention of termination checks throughout the algorithm. Correspondingly, we assume for all practical purposes that $g_k \neq 0$ for all $k \in \mathbb{N}$. Also for simplicity in its description, we state Algorithm 6 in such a way that it iterates *in order* through the

sets of eigenvalues in (4.15). Note, however, that during a given iteration k, one may also consider computing all stepsizes that would be obtained by any of the available pair of eigenvalues, and then choosing the pair that leads to the smallest corresponding stepsize. This would be consistent with Fletcher's approach in that, at each point in a "sweep," the eigenvalue yielding the smallest stepsize is chosen. For this reason, this is the strategy that we have adopted in our numerical experiments. (Note that by ordering the eigenvalues from largest to smallest, employing them in order corresponds to choosing stepsizes in increasing order of magnitude *if* all of the eigenvalues are positive.)

In our implementation described in the following Chapter, we incorporated a nonmonotone line search into Algorithm 6, which, for ease of exposition, has also been suppressed in the presentation of the algorithm. In particular, we incorporated the Armijo (i.e., backtracking) line search of [56] with moving average parameter $\eta_k = \eta \in (0, 1)$ for all $k \in \mathbb{N}$, which is employed in every step of the algorithm. With this line search, we claim the following global convergence result for our algorithm; the proof is a special case of [56, Theorem 2.2].

Theorem 4.3.4. *Suppose f is bounded below and ∇f is Lipschitz continuous on*

$$\{x \in \mathbb{R}^n : f(x) \leq f(x_0)\} + \{d \in \mathbb{R}^n : \|d\|_2 \geq \sup_{k \in \mathbb{N}} \|\alpha_k g_k\|_2\}. \tag{4.20}$$

Then, the iterate sequence $\{x_k\}$ generated by Algorithm 6 yields

$$\lim_{k \to \infty} g_k = 0.$$

That is, every convergent subsequence of $\{x_k\}$ approaches a point x_* with $\nabla f(x_*) = 0$.

Algorithm 6 LMSD Method with Cubic Regularization (for $m \geq 1$)

choose $(\omega, \Omega) \in \mathbb{R} \times \mathbb{R}$ satisfying $0 < \omega \leq \Omega$, $c \in \mathbb{R}_+$, and $m \in \mathbb{N}_+$
choose $x_0 \in \mathbb{R}^n$ and $\alpha_j \in [\omega, \Omega]$ for all $j \in \{0, \ldots, m-1\}$
for $j = 0, \ldots, m-1$ **do**
    set $x_{j+1} \leftarrow x_j - \alpha_j g_j$
end for
set $k \leftarrow m$
loop
    loop
        set G_k as in (2.15) and J_k as in (2.17).
        compute the (thin) QR-factorization $G_k = Q_k R_k$.
        if G_k is composed of only one column **then break**
        if R_k is invertible and the elements of $R_k^{-1} Q_k^T g_k$ do not sum to one **then break**
        remove the first column each of G_k and J_k
    end loop
    set $\tilde{m}$ as the number of columns of G_k
    if $\tilde{m} = 1$ **then**
        set α_k as in Algorithm 5 and $x_{k+1} \leftarrow x_k - \alpha_k g_k$
    else
        set T_k as in (2.16) and set $\tilde{T}_k$ and $\tilde{P}_k$ as described in Theorem 4.3.2
        set $\{\bar{q}_k, \ldots, \bar{q}_{k+\tilde{m}-1}\}$ and $\{\hat{q}_k, \ldots, \hat{q}_{k+\tilde{m}-1}\}$ as in (4.15)
        for $j = 0, \ldots, \tilde{m}-1$ **do**
            Set $q_{k+j} \leftarrow \hat{q}_{k+j}$.
            if $q_{k+j} > 0$ **then** set $c_{k+j} \leftarrow 0$ **else** set $c_{k+j} \leftarrow c(\bar{q}_{k+j} - q_{k+j})/\|s_{k+j}\|_2$
            if $q_{k+j} > 0$ **then** set $\alpha_{k+j} \leftarrow 1/q_{k+j}$
            else if $c_{k+j} > 0$ **then** set $\alpha_{k+j} \leftarrow 2/(q_{k+j} + \sqrt{q_{k+j}^2 + 2c_{k+j}\|g_{k+j}\|_2})$
            else if $\bar{q}_{k+j} = 0$ **then** set $\alpha_{k+j} \leftarrow \omega$
            else set $\alpha_{k+j} \leftarrow \Omega$
            set $x_{k+j+1} \leftarrow x_{k+j} - \alpha_{k+j} g_{k+j}$
        end for
    end if
    set $k \leftarrow k + \tilde{m}$
end loop

4.4 Implementation

Algorithm 6 (which includes Algorithm 5 as a special case) was implemented in Matlab along with two other algorithms for comparison purposes. In this Chapter, we describe the implementations of these algorithms along with input parameter settings that were used in our experiments (see §4.5).

We used built-in Matlab functions to compute matrix factorizations and eigenvalues in all of the implemented algorithms. In order to avoid the influence of numerical error and the computation of excessively small or large stepsizes, we removed previously computed gradients (in a similar manner as in the inner **loop** of Algorithm 6) if more than one was currently being held and any of the corresponding computed eigenvalues were smaller than 10^{-12} or larger than 10^{12} in absolute value. Similarly, prior to computing corresponding stepsizes, we projected any computed quadratic term coefficient so that it would have an absolute value at least 10^{-12} and at most 10^{12}, and we projected any computed cubic term coefficient onto the widest possible subset of the positive reals so that the resulting stepsize would be at least 10^{-12} and at most 10^{12}. (For the quadratic term coefficients, this projection was performed to maintain the sign of the originally computed coefficient.) Overall, this implied $\omega = 10^{-12}$ and $\Omega = 10^{12}$ for all algorithms. As described in §4.3, the eigenvalues, once computed, were ordered from largest to smallest, though the implementation of Algorithm 6 potentially used these eigenvalues out-of-order in order to ensure that, in any given iteration, the eigenvalue pair leading to the smallest stepsize was used (after which this pair was removed from the set for subsequent iterations).

The three implemented algorithms only differed in the manner in which stepsizes were computed. Our implementation of Algorithm 6—hereafter referred to as CUBIC—employed the strategy described in §4.3 with $c \leftarrow 1$. The other two algorithms, on the other hand, were two variants of an algorithm derived from the ideas in [22]. In particular, the algorithm we refer to as QUAD-RITZ computes stepsizes as reciprocals of the Ritz values $\{\bar{q}_k, \ldots, \bar{q}_{k+m-1}\}$, whereas the algorithm we refer to as QUAD-HRITZ computes stepsizes as reciprocals of the harmonic Ritz values $\{\hat{q}_k, \ldots, \hat{q}_{k+m-1}\}$ (recall (4.15)). In both QUAD-RITZ and QUAD-HRITZ, the standard approach of handling nonpositive curvature was em-

ployed; i.e., if a computed eigenvalue was negative, then the stepsize was simply set to Ω.

For simplicity and consistency, all algorithms employed the same nonmonotone line search in every step, using the stepsize computed in the main procedure as the initial stepsize for the line search. As mentioned in §4.3, for this purpose, we implemented the Armijo (i.e., backtracking) line search of [56]. This strategy requires a sufficient decrease parameter, for which we used 10^{-12}, a backtracking parameter, for which we used 0.5, and a moving average parameter (see η_k as defined in [56]), for which we used 0.5 for all $k \in \mathbb{N}$. For the history length parameter m, we experimented with values in the set $\{1, 3, 5\}$. (As discussed further in §4.5, results for larger values of m did not lead to improved performance beyond the values considered here. This is consistent with Fletcher's experience with his LMSD method, and in some previous studies of L-BFGS.) All algorithms terminated whenever either the ℓ_∞-norm of a computed gradient was less than or equal to $10^{-8} \max\{1, \|g_0\|_\infty\}$—indicating a successful run—or the maximum iteration count of 10^{10} was reached—indicating a failure.

4.5 Numerical Experiments

We tested the algorithms CUBIC, QUAD-RITZ, and QUAD-HRITZ by employing them to solve unconstrained problems from the CUTEst collection; see [28] (and [27] for information about a previous version of the test set). We resized all unconstrained problems to the largest of their preset sizes, and from that set kept those (a) that had at least 50 variables, so as to have $m \ll n$; (b) for which at least one run of an algorithm required at least 5 seconds, so as to focus on the more difficult problems in the set; (c) for which at least one run of an algorithm involved the computation of a nonpositive quadratic term coefficient, so as to focus on the issue of handling nonpositive curvature; and (d) for which at least one run of an algorithm led to a successful termination. The resulting set of 30 problems and their sizes can be found in the tables of results provided at the end of this chapter.

Over all runs of all algorithms on all of our test problems, we compiled the number of function and gradient evaluations required prior to termination. (Note that the number

of iterations can be considered equal to the number of gradient evaluations.) To compare the results of the experiments, we consider the technique proposed in [38], which proceeds as follows. Letting func_A^j and func_B^j be the number of function evaluations required by algorithms A and B, respectively, on the jth problem in the set, we compute the logarithmic outperforming factor $r_{AB}^j := -\log_2(\text{func}_A^j/\text{func}_B^j)$. For example, $r_{AB}^j = 2$ indicates that algorithm A required 2^{-2} of the function evaluations required by algorithm B. Such a factor is similarly defined for comparing the numbers of gradient evaluations required by two algorithms. By computing all such factors for each problem in the test set, one can compare the performance of two algorithms side-by-side in a bar plot, where positive bars indicate better performance with respect to a particular measure for algorithm A and negative bars indicate better performance for algorithm B. In all cases, we restrict attention to $r_{AB}^j \in [-1, 1]$ since the particular magnitude of a factor beyond this interval is not of great interest; it is sufficient to know that an algorithm performed fewer than half (or more than double) the number of function or gradient evaluations as did another algorithm.

As a first comparison, we consider the performance of the algorithms for $m = 1$, for which we have the bar plots in Figures 4.3 and 4.4, corresponding to function and gradient evaluations, respectively. In each figure, the performance of each pair of algorithms is revealed in a side-by-side comparison with the name of algorithm "A" indicated above the plot and the name of algorithm "B" indicated below the plot. The profiles clearly indicate that the use of harmonic Ritz values led to better performance than the use of Ritz values in that CUBIC and QUAD-HRITZ consistently outperformed QUAD-RITZ with respect to both measures on most (if not all) problems in our test set.

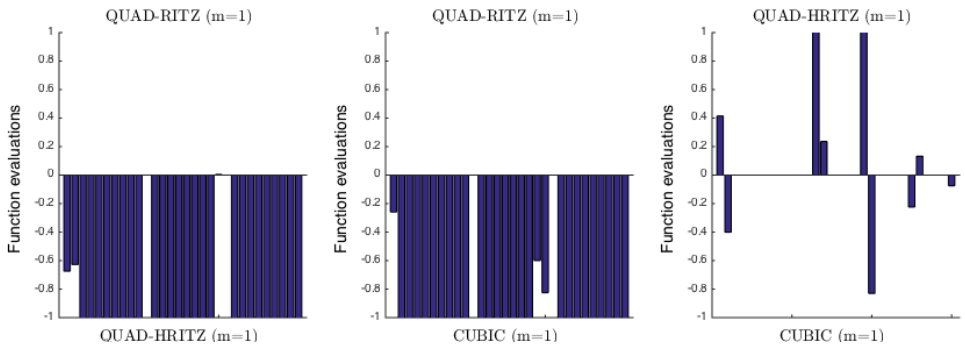

Figure 4.3: Outperforming factors for function evaluations with $m = 1$.

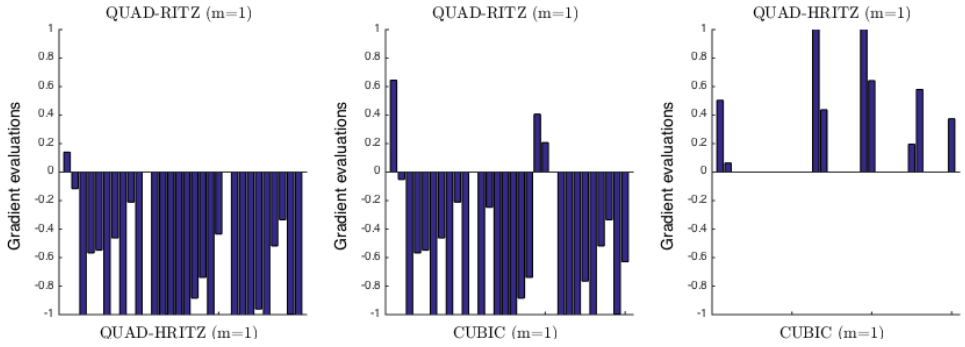

Figure 4.4: Outperforming factors for gradient evaluations with $m = 1$.

One observes in Figures 4.3 and 4.4 that CUBIC and QUAD-HRITZ performed similarly on many problems, with better performance provided by QUAD-HRITZ on some problems in the set. This situation changed, however, when we considered $m = 3$ and $m = 5$, for which we have the bar plots in Figures 4.5–4.8. Particularly in terms of function evaluations and often in terms of gradient evaluations, CUBIC outperformed both QUAD-RITZ and QUAD-HRITZ on a majority of problems in our set.

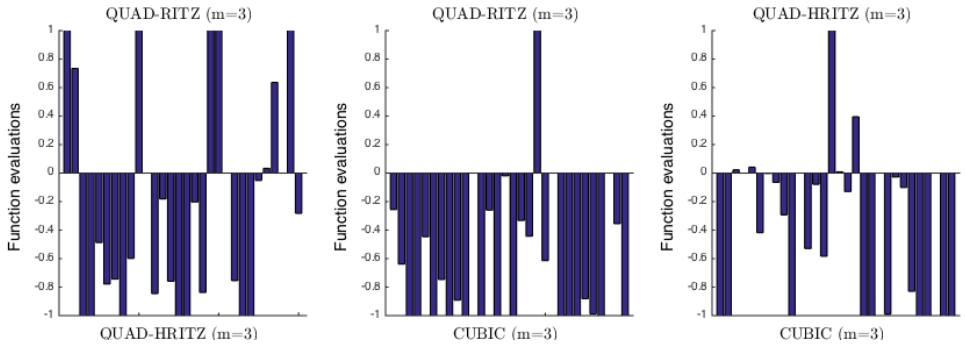

Figure 4.5: Outperforming factors for function evaluations with $m = 3$.

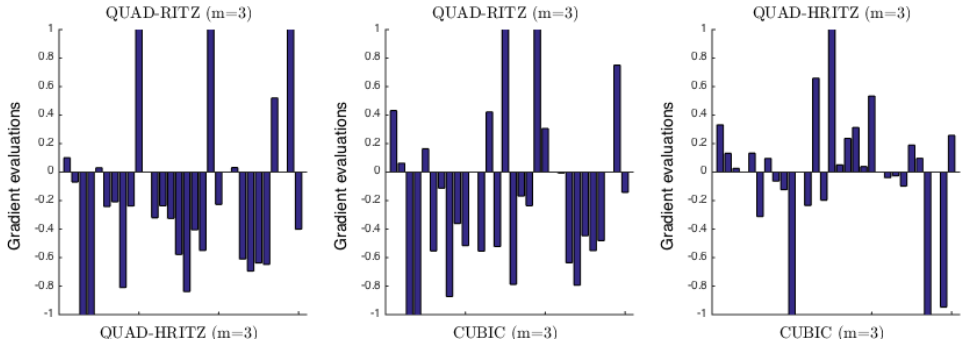

Figure 4.6: Outperforming factors for gradient evaluations with $m = 3$.

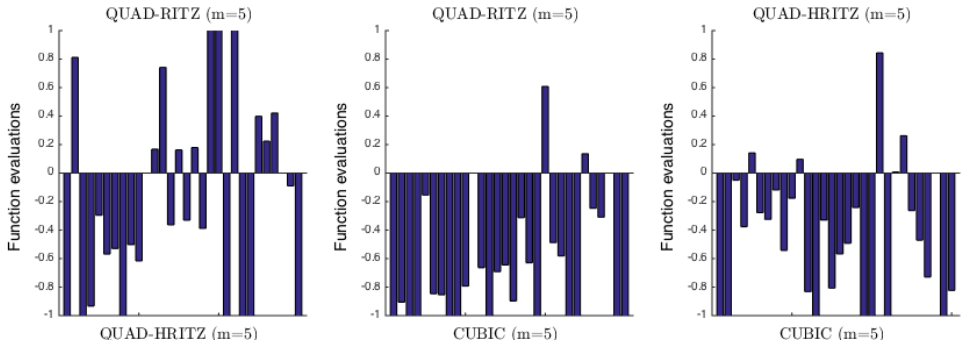

Figure 4.7: Outperforming factors for function evaluations with $m = 5$.

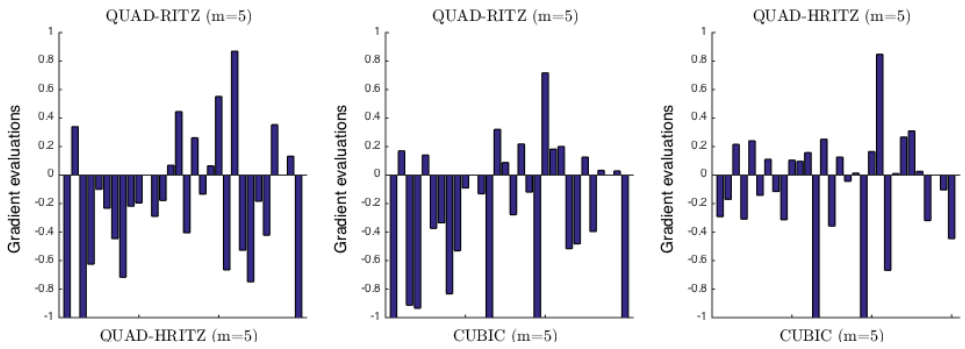

Figure 4.8: Outperforming factors for gradient evaluations with $m = 5$.

As a final comparison, we consider the performance of CUBIC for $m \in \{1, 3, 5\}$. The results are provided in the bar plots in Figures 4.9 and 4.10. Despite mixed results in terms of function evaluations, the factors computed based on gradient evaluations indicate better performance when $m = 5$. Combining this conclusion with those drawn from the results above, we claim that we obtained the best results in our experiments with CUBIC and $m = 5$. (We also experimented with larger values of m, but they did not lead to improved performance over those obtained with $m = 5$. We suspect that this was due to the presence of nonpositive curvature—i.e., nonpositive eigenvalues—much more often than would have been observed with a smaller value of m. This led us to conclude that while one may benefit by confronting a modest amount of nonpositive curvature with our proposed technique, an excessive amount of nonpositive curvature is not easily overcome.)

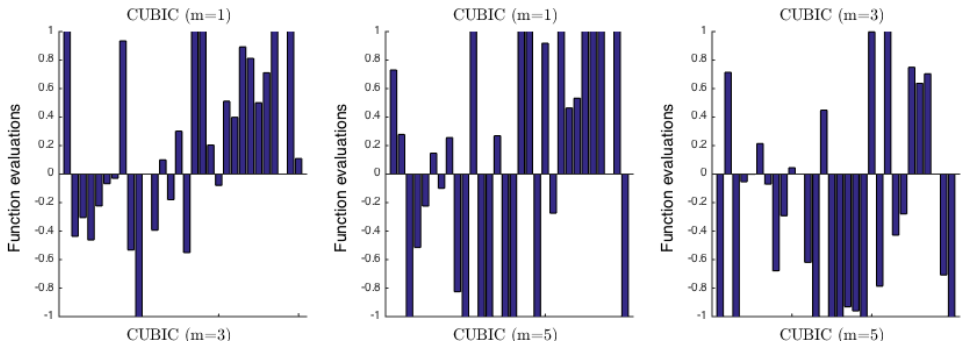

Figure 4.9: Outperforming factors for function evaluations for CUBIC for $m \in \{1, 3, 5\}$.

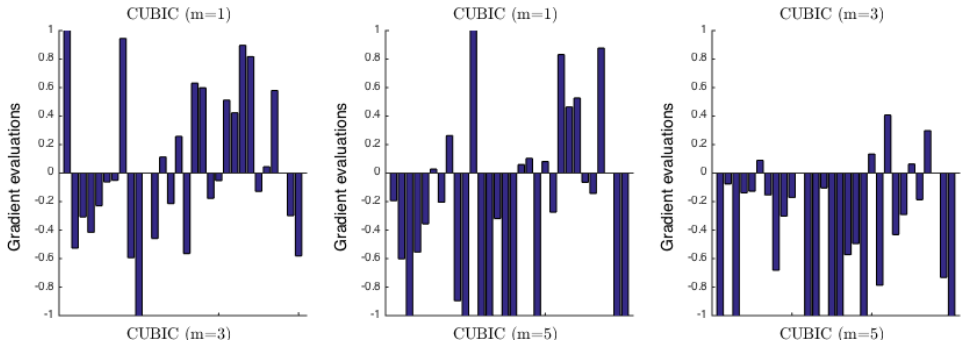

Figure 4.10: Outperforming factors for gradient evaluations for CUBIC for $m \in \{1, 3, 5\}$.

We provide details of the results of our numerical experiments that were summarized in §4.5. The following tables provide function and gradient evaluation counts for the implemented algorithms QUAD-RITZ, QUAD-HRITZ, and CUBIC for the history length parameter values $m \in \{1, 3, 5\}$.

Table 4.1: Results for $m = 1$

		QUAD-RITZ		QUAD-HRITZ		CUBIC	
Name	n	$\#f$	$\#g$	$\#f$	$\#g$	$\#f$	$\#g$
CHAINWOO	10000	809	412	507	454	676	644
CHNROSNB	50	2977	1486	1927	1371	1459	1433
DECONVU	61	276033	126801	23625	23553	23625	23553
DIXMAANE	9000	3597	1705	1205	1151	1205	1151
DIXMAANF	9000	3788	1789	1284	1224	1284	1224
DIXMAANG	9000	3755	1763	791	761	791	761
DIXMAANH	9000	2610	1265	939	918	939	918
DIXMAANI	9000	158013	70930	16912	16764	16912	16764
DIXMAANJ	9000	2402	1156	1033	999	1033	999
DIXMAANK	3000	10612	4851	1275	1238	1275	1238
DIXON3DQ	10000	1907708	865306	222481	222086	222481	222086
EIGENALS	110	20405	9231	3049	2947	3049	2947
EIGENBLS	110	24276	11095	1232	894	10075	9351
EIGENCLS	462	49256	22791	9640	8186	11355	11086
ERRINROS	50	714084	331317	20352	20105	20352	20105
EXTROSNB	1000	1031208	474887	126174	126096	126174	126096
FMINSRF2	15625	6454	3104	1709	1681	1709	1681
FMINSURF	1024	2541	1298	817	778	817	778
GENHUMPS	5000	29372	14564	2135	1109	19378	19305
GENROSE	500	9483	4245	9517	3140	5353	4899
HYDC20LS	99	4614763	2095069	1071208	1070629	1071208	1070629
MODBEALE	2000	1273	527	204	164	204	164
MSQRTALS	529	248340	113968	28812	28688	28812	28688
MSQRTBLS	529	218650	100286	24993	24882	24993	24882
NONCVXU2	10000	39042	18952	13287	9734	11375	11151
NONCVXUN	10000	75236	36361	15263	10987	16727	16418
NONDQUAR	10000	612	368	266	257	266	257
SPMSRTLS	10000	1451	753	632	597	632	597
TQUARTIC	10000	11279	2026	1282	1016	1282	1016
WOODS	10000	2760	960	887	479	842	621

Table 4.2: Results for $m = 3$

		QUAD-RITZ		QUAD-HRITZ		CUBIC	
Name	n	$\#f$	$\#g$	$\#f$	$\#g$	$\#f$	$\#g$
CHAINWOO	10000	139022	72636	440326	77900	116431	97997
CHNROSNB	50	1678	953	2792	908	1078	995
DECONVU	61	101433	57765	18820	18680	19120	19019
DIXMAANE	9000	2782	1822	875	863	875	863
DIXMAANF	9000	1499	932	1069	952	1100	1044
DIXMAANG	9000	1731	1070	1009	905	755	729
DIXMAANH	9000	1544	958	922	829	920	886
DIXMAANI	9000	98504	59120	33824	33723	32322	32270
DIXMAANJ	9000	1325	850	875	721	714	662
DIXMAANK	3000	1053	668	2530	2479	478	467
DIXON3DQ	10000	1779714	1079416	1421808	1421753	1387582	1387527
EIGENALS	110	6016	3150	3348	2522	2320	2144
EIGENBLS	110	12932	7548	11406	6405	10798	10116
EIGENCLS	462	25440	13729	15027	10957	10025	9556
ERRINROS	50	25428	9363	10147	6269	25089	24050
EXTROSNB	1000	265288	147208	85540	82318	86104	85247
FMINSRF2	15625	17509	2928	15208	2211	13905	2605
FMINSURF	1024	8793	1388	4919	948	6470	1178
GENHUMPS	5000	7414	2054	89478	16611	22323	17068
GENROSE	500	7754	3820	19232	3264	5067	4724
HYDC20LS	99	3989081	2379958	1527159	1527100	1527159	1527100
MODBEALE	2000	901	221	534	226	269	220
MSQRTALS	529	134835	83020	54501	54410	53480	53414
MSQRTBLS	529	123556	76020	47057	46951	43885	43854
NONCVXU2	10000	29644	13907	28592	8941	16091	10200
NONCVXUN	10000	54336	24814	55618	15836	27385	16938
NONDQUAR	10000	4286	536	6664	769	1319	384
SPMSRTLS	10000	1307875	729339	607	554	8370	7667
TQUARTIC	10000	6068	491	20204	1593	4746	826
WOODS	10000	2461	458	2023	347	908	415

Table 4.3: Results for $m = 5$

		QUAD-RITZ		QUAD-HRITZ		CUBIC	
Name	n	$\#f$	$\#g$	$\#f$	$\#g$	$\#f$	$\#g$
CHAINWOO	10000	156942	56418	4233	689	1122	563
CHNROSNB	50	3314	839	5818	1062	1769	944
DECONVU	61	12229	7398	4551	3381	4397	3928
DIXMAANE	9000	2089	1498	1094	971	843	784
DIXMAANF	9000	1223	867	996	809	1099	956
DIXMAANG	9000	1575	1006	1062	856	876	776
DIXMAANH	9000	1583	1005	1097	738	876	797
DIXMAANI	9000	53132	35845	21932	21797	20201	20120
DIXMAANJ	9000	1202	776	849	667	583	537
DIXMAANK	3000	854	442	557	386	493	415
DIXON3DQ	10000	1777107	1213356	850208	849833	909144	908761
EIGENALS	110	2392	1076	2687	880	1510	982
EIGENBLS	110	8701	5418	14548	4788	1764	903
EIGENCLS	462	22116	7112	17196	7464	13688	8888
ERRINROS	50	9760	1427	10927	1944	6244	1517
EXTROSNB	1000	22726	10282	18073	7767	12199	8480
FMINSRF2	15625	9062	1506	10264	1806	7293	1752
FMINSURF	1024	5143	908	3931	828	3325	836
GENHUMPS	5000	57998	18114	143813	18945	6110	1393
GENROSE	500	6639	3152	30851	4620	10119	5182
HYDC20LS	99	1241851	780101	493466	492033	885588	885337
MODBEALE	2000	1106	254	2611	464	739	292
MSQRTALS	529	82328	56605	39500	39273	39734	39589
MSQRTBLS	529	72615	50086	30137	29798	36146	35852
NONCVXU2	10000	24633	9768	32486	8602	27062	10662
NONCVXUN	10000	50520	19570	59028	14601	42572	14875
NONDQUAR	10000	2662	461	3564	589	2149	472
SPMSRTLS	10000	3857	947	3462	907	1962	844
TQUARTIC	10000	6336	487	5953	534	2906	497
WOODS	10000	1804	255	377	94	213	69

4.6 Conclusion

We have designed, analyzed, and experimented with a limited memory steepest descent (LMSD) method for solving unconstrained nonconvex optimization problems. The unique feature of our algorithm is a novel approach for handling nonpositive curvature; in particular, we propose that when nonpositive curvature is encountered, stepsizes can be computed by constructing local cubic models of the objective function, for which reasonable values of the quadratic and cubic term coefficients can be derived using previously computed gradient information. Our numerical experiments suggest that our approach yields superior performance in practice compared to algorithms that do not attempt to incorporate problem information when computing stepsizes when nonpositive curvature is encountered.

Chapter 5

A Limited Memory Stochastic Gradient Method

We propose a limited memory stochastic gradient (LMSG) method for optimization problems arising in machine learning. We focus on problems that are strongly convex. When a dataset is too large such that the computation of a full gradient is too expensive, our method computes stepsizes and iterates based on stochastic gradients or mini-batch gradients. Although in stochastic gradient (SG) methods, a best-tuned fixed stepsize or diminishing stepsize is most widely used, it can be inefficient in practice. Our method adopts a cubic model and always guarantees a positive meaningful stepsize, even when nonconvexity / nonpositive curvature is encountered. It reduces to a limited memory steepest descent method with cubic regularization described in Chapter 4.3.2 when full gradients are used. With a projection of stepsizes, we ensure convergence to a neighborhood of the optimal solution when the interval is fixed and convergence to the optimal solution when the interval is diminishing with $1/k$. We also illustrate with numerical experiments that our approach can outperform a stochastic gradient (SG) method with fixed stepsizes.

5.1 Introduction

The following optimization problem, which minimizes the sum of cost functions over samples from a finite training set, appears frequently in machine learning

$$\min_{x\in\mathbb{R}^n} F(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{5.1}$$

where n is the sample size, and each $f_i : \mathbb{R}^d \to \mathbb{R}$ is the cost function corresponding to the i-th sample point. For example, in logistic regression, the cost function has the form

$$f_i(x) := \log(1 + \exp(-b_i a_i^T x)),$$

where $a_i \in \mathbb{R}^d$ and $b_i \in \{-1, 1\}$ are the data samples associated with a binary classification problem.

In this work, we focus on problems where each f_i is convex and smooth and assume F is strongly convex (see (2.4) in Chapter 2.1.3).

One important issue regarding to stochastic algorithms is how to choose an appropriate stepsize α_k while running the algorithm. In a full gradient method, the stepsize is often obtained by carrying out a line search. However, line searches are computationally prohibitive in stochastic gradient methods because only sub-sampled information of gradient is available. As a result, for stochastic gradient (SG), people usually use a best-tuned fixed stepsize $\alpha_k = C$, or a diminishing stepsize, for example $\alpha_k = 1/(k+1)$, which can be time consuming sometimes.

Recently, a stochastic Barzilai-Borwein (BB) method, which borrows the idea of Barzilai-Borwein two point stepsize methods used in full gradient methods, is proposed by [49]. The original BB methods are developed for strongly convex quadratic which employ one previous iteration and gradient information (history length $m = 1$) to compute stepsizes. For full gradient methods, we can always guarantee the positiveness of BB stepsize when $F(x)$ is strongly convex, however, for stochastic gradient (SG) or mini-batch gradient, there appear to be nonpositive curvature / nonconvexity which will result in nonpositive

BB stepsize. In [49], they simply take the absolute value of the original BB stepsize, we have an idea that might be better.

In this Chapter, we provide a novel strategy for computing stepsizes when there is nonpositive curvature. In particular, instead of taking the absolute value of a negative stepsize, we adopt a local cubic model of the objective function in order to determine a positive stepsize (see [11]). In addition, since Fletcher extends BB methods by exploiting information from more than one previous iteration (history length $m \geq 1$), which are known as Limited Memory Steepest Descent (LMSD) methods, we also apply the ideas of LMSD methods to SG. We also adopt a local cubic model in order to determine a sequence of positive stepsizes.

Our contributions in this Chapter are as follows:

- We introduce Barzilai-Borwein Stochastic Gradient (BBSG) Method, apply BB stepsizes to SG and extend the stochastic BB methods proposed in [49]. Instead of simply taking the absolute value, we adopt a cubic model to determine a positive stepize when nonpositive curvature / nonpositive BB stepsizes are encountered.

- We extend BBSG to Limited Memory Stochastic Gradient (LMSG) Method, from exploiting only one previous iteration and gradient information ($m = 1$) to more than one previous information ($m \geq 1$), incorporate LMSD stepsizes into SG and propose a cubic model for nonpositive curvature in order to determine a sequence of positive stepsizes.

- We prove convergence (rate) properties for LMSG with Fixed Interval Projection and Diminishing Interval Projection for strongly convex objective under mild assumptions, respectively.

- We conduct numerical experiments for LMSG on solving logistic regression problems. The numerical results show that LMSG can outperform an SG method with a fixed stepsize.

The rest of this Chapter is organized in the following manner. In Chapter 5.2 and 5.3, we propose BBSG and LMSG methods, including models and algorithm descriptions. We

also prove convergence (rate) properties for LMSG with Fixed Interval Projection and Diminishing Interval Projection for strongly convex objective under mild assumptions, respectively. We conduct numerical experiments for LMSG and show its superior performances in Chapter 5.4. Final conclusions are presented in Chapter 5.5.

5.2 Barzilai-Borwein Stochastic Gradient (BBSG) Method

5.2.1 Motivation

At x_k, if $s_k^T y_k < 0$ then BB stepsize $\alpha_k = (s_k^T s_k)/(s_k^T y_k)$ computed in [49] is negative. We show a concrete example and explain why we think simply taking absolute value of the stepsize is not always the best approach. We assume $s_k^T s_k = 1$, $s_k^T y_k = -1$ and -10, respectively. Since $s_k^T y_k$ represents curvature of the function at x_k, a more negative $s_k^T y_k$ implies a more concave curvature and we can be aggressive by taking a larger stepsize, while on the opposite, a less negative $s_k^T y_k$ implies a less concave curvature and we should be conservative by taking a smaller stepsize. However, in this example, when $s_k^T y_k = -1$, we obtain the stepsize (after taking absolute value) $\alpha_k = 1$ and when $s_k^T y_k = -10$, $\alpha_k = 0.1$, which contradicts the discussion and intuition. However, we will obtain a positive meaningful stepsize by adopting a local cubic model.

5.2.2 Algorithm Description

At x_k, the calculations in the kth iteration of our algorithm are based on the cubic model $m_k : \mathbb{R}^n \to \mathbb{R}$ of f at x_k defined by

$$m_k(s) = f_k + g_k^T s + \tfrac{1}{2} q_k \|s\|_2^2 + \tfrac{1}{6} c_k \|s\|_2^3 \approx f(x_k + s), \tag{5.2}$$

where q_k and c_k are scalars to be determined.

When there is no nonpositive curvature encountered, we would like the cubic model to reduce to the quadratic model described in Chapter 2.5 and hence we set q_k the same as (2.13). While when $q_k < 0$, we require a positive c_k and consider two different options:

- Consider choosing c_k to minimize the least squared error between the gradient of

the model m_k at $-s_k$ and the previous gradient g_{k-1}, i.e., $\frac{1}{2}\|\nabla m_k(-s_k) - g_{k-1}\|_2^2$ and obtain

$$c_k \leftarrow \frac{2}{\|s_k\|_2}(\bar{q}_k - q_k). \tag{5.3}$$

- Consider choosing c_k so that the curvature of m_k at $-s_k$ along s_k is equal to $\bar{q}_k$, i.e., so that $s_k^T \nabla^2 m_k(-s_k) s_k = \bar{q}_k \|s_k\|_2^2$, and we obtain

$$c_k \leftarrow \frac{1}{\|s_k\|_2}\left(\bar{q}_k - q_k\right). \tag{5.4}$$

Overall, we propose that the cubic term coefficient should be set, for some constant $c > 0$, as

$$c_k \leftarrow \frac{c}{\|s_k\|_2}\left(\bar{q}_k - q_k\right). \tag{5.5}$$

As such, we choose the stepsize α_k as an optimal solution of the one-dimensional problem

$$\min_{\alpha \geq 0} \ \phi_k(\alpha), \quad \text{where} \quad \phi_k(\alpha) = m_k(-\alpha g_k) \quad \text{for all} \quad \alpha \in \mathbb{R}. \tag{5.6}$$

It is easily verified that (5.6) is solved by setting

$$\alpha_k \leftarrow \frac{2}{q_k + \sqrt{q_k^2 + 2c_k\|g_k\|_2}} > 0. \tag{5.7}$$

The algorithm is described as follows and g_k can be computed using formula (2.20), (2.22) or (2.21) depending on which method the algorithm uses. Before computing $\bar{q}_k$ and $\hat{q}_k$, we consider some special cases when $\bar{q}_k$ or $\hat{q}_k$ is not well-defined.

Algorithm BBSG Barzilai-Borwein Stochastic Gradient Method

choose $(\omega, \Omega) \in \mathbb{R} \times \mathbb{R}$ satisfying $0 < \omega \leq \Omega$ and $c \in \mathbb{R}_+ := \{c \in \mathbb{R} : c > 0\}$

choose $x_0 \in \mathbb{R}^n$ and $\alpha_0 \in [\omega, \Omega]$

set $x_1 \leftarrow x_0 - \alpha_0 g_0$ and $k \leftarrow 1$

loop

  if $y_k = 0$ or $s_k^T y_k = -\|s_k\|_2 \|y_k\|_2 < 0$ **then**

    set $\alpha_k \leftarrow \Omega$

  else if $s_k^T y_k = 0$ **then**

    set $\alpha_k \leftarrow \omega$

  else

    set $\bar{q}_k \leftarrow s_k^T y_k / s_k^T s_k$ and $q_k \leftarrow y_k^T y_k / s_k^T y_k$

    if $q_k > 0$ **then** set $c_k \leftarrow 0$ **else** set $c_k \leftarrow c(\bar{q}_k - q_k)/\|s_k\|_2$

    if $q_k > 0$ **then** set $\alpha_k \leftarrow 1/q_k$ **else** set $\alpha_k \leftarrow 2/(q_k + \sqrt{q_k^2 + 2c_k\|g_k\|_2})$

    replace α_k by its projection onto the interval $[\omega, \Omega]$

  end if

  set $x_{k+1} \leftarrow x_k - \alpha_k g_k$ and $k \leftarrow k + 1$

end loop

5.3 Limited Memory Stochastic Gradient (LMSG) Method

5.3.1 Algorithm Description

Similar to the cubic model described in Chapter 5.2.2, in LMSD methods, we employ cubic models to generate a sequence of m positive stepsizes. For a given $j \in \{0, \ldots, m-1\}$, consider the cubic model $m_{k+j} : \mathbb{R}^n \to \mathbb{R}$ of f at x_{k+j} defined by

$$m_{k+j}(s) = f_{k+j} + g_{k+j}^T s + \tfrac{1}{2} q_{k+j} \|s\|_2^2 + \tfrac{1}{6} c_{k+j} \|s\|_2^3 \approx f(x_{k+j} + s), \tag{5.8}$$

where q_{k+j} and c_{k+j} are scalars to be determined. We still compute q_k as eigenvalues of T_k and $(P_k^{-1} T_k)^{-1}$. However, when f is not quadratic, T_k will be upper Hessenberg, but not necessarily symmetric. We modify T_k to $\tilde{T}_k$ by setting it equal to T_k, except that

its strict upper triangle is replaced by the transpose of its strict lower triangle, and let $\tilde{P}_k \leftarrow \tilde{T}_k^T \tilde{T}_k + \zeta_k \zeta_k^T$ where $\zeta_k := [0 \rho_k] J_k R_k^{-1}$. Now we obtain symmetric tridiagonal $\tilde{T}_k$, symmetric pentadiagonal $\tilde{P}_k$ and compute the following sets with elements ordered largest to smallest:

$$\{\bar{q}_k, \ldots, \bar{q}_{k+m-1}\} \text{ as the eigenvalues of } \tilde{T}_k$$

$$\text{and } \{\hat{q}_k, \ldots, \hat{q}_{k+m-1}\} \text{ as the eigenvalues of } (\tilde{P}_k^{-1} \tilde{T}_k)^{-1}.$$

If $\hat{q}_{k+j} > 0$, then we set $c_{k+j} \leftarrow 0$, but otherwise we again follow the strategy of the $m = 1$ case and set

$$c_{k+j} \leftarrow \frac{c}{\|s_{k+j}\|_2} (\bar{q}_{k+j} - q_{k+j}). \tag{5.9}$$

Furthermore, consider the problem

$$\min_{\alpha \geq 0} \ \phi_{k+j}(\alpha), \quad \text{where} \quad \phi_{k+j}(\alpha) := m_{k+j}(-\alpha g_{k+j}) \quad \text{for all} \quad \alpha \in \mathbb{R}. \tag{5.10}$$

It is easily seen that when $q_{k+j} > 0$ and $c_{k+j} = 0$, the solution of problem (5.10) is given by $\alpha_{k+j} = q_{k+j}^{-1}$, while if $q_{k+j} \leq 0$ and $c_{k+j} > 0$, then the solution is given by a formula similar to (5.7).

The algorithm is described as follows. Before computing matrix $\tilde{T}_k$ and $\tilde{P}_k$, we consider some special cases when T_k is not well-defined.

Algorithm LMSG Limited Memory Stochastic Gradient Method

choose $(\omega, \Omega) \in \mathbb{R} \times \mathbb{R}$ satisfying $0 < \omega \leq \Omega$, $c \in \mathbb{R}_+$, and $m \in \mathbb{N}_+$
choose $x_0 \in \mathbb{R}^n$ and $\alpha_j \in [\omega, \Omega]$ for all $j \in \{0, \ldots, m-1\}$
for $j = 0, \ldots, m-1$ **do**
    set $x_{j+1} \leftarrow x_j - \alpha_j g_j$
end for
set $k \leftarrow m$
loop
    loop
        set G_k as in (2.15) and J_k as in (2.17).
        compute the (thin) QR-factorization $G_k = Q_k R_k$.
        if G_k is composed of only one column **then break**
        if R_k is invertible and the elements of $R_k^{-1} Q_k^T g_k$ do not sum to one **then break**
        remove the first column each of G_k and J_k
    end loop
    set $\tilde{m}$ as the number of columns of G_k
    if $\tilde{m} = 1$ **then**
        set α_k as in Algorithm BBSG and $x_{k+1} \leftarrow x_k - \alpha_k g_k$
    else
        set T_k as in (2.16) and set $\tilde{T}_k$ and $\tilde{P}_k$ as described in Theorem 4.3.2
        set $\{\bar{q}_k, \ldots, \bar{q}_{k+\tilde{m}-1}\}$ and $\{\hat{q}_k, \ldots, \hat{q}_{k+\tilde{m}-1}\}$ as in (4.15)
        for $j = 0, \ldots, \tilde{m}-1$ **do**
            Set $q_{k+j} \leftarrow \hat{q}_{k+j}$.
            if $q_{k+j} > 0$ **then** set $c_{k+j} \leftarrow 0$ **else** set $c_{k+j} \leftarrow c(\bar{q}_{k+j} - q_{k+j})/\|s_{k+j}\|_2$
            if $q_{k+j} > 0$ **then** set $\alpha_{k+j} \leftarrow 1/q_{k+j}$
            else if $c_{k+j} > 0$ **then** set $\alpha_{k+j} \leftarrow 2/(q_{k+j} + \sqrt{q_{k+j}^2 + 2c_{k+j}\|g_{k+j}\|_2})$
            else if $\bar{q}_{k+j} = 0$ **then** set $\alpha_{k+j} \leftarrow \omega$
            else set $\alpha_{k+j} \leftarrow \Omega$
            set $x_{k+j+1} \leftarrow x_{k+j} - \alpha_{k+j} g_{k+j}$
        end for
    end if
    set $k \leftarrow k + \tilde{m}$
end loop

We can see that Algorithm BBSG is a special case of Algorithm LMSG when $m = 1$.

5.3.2 Convergence Results

Our approach for establishing convergence guarantees for LMSG is built upon the assumptions of smoothness of the objective function, namely, Lipchitz continuity of its gradient function (see (2.2) in Chapter 2.1.2).

From the properties of Lipchitz continuity (2.3) and strong convexity (2.4), we observe that the strong convexity parameter σ and Lipschitz constant L must satisfy $\sigma \leq L$. And the convergence of LMSG also depends on the first and second moment limits of the stochastic gradient, which are described as follows.

Assumption 5.3.1. ***(First and second moment limits).*** *The objective function and the Algorithm LMSG satisfy the following:*

(a) The sequence of iterates $\{x_k\}$ is contained in an open set over which F is bounded below by a scalar F_{inf}.

(b) The stochastic gradient is an unbiased estimation of full gradient ∇F, i.e.

$$\mathbb{E}[\nabla f_{i_k}(x_k)] = \nabla F(x_k) \text{ , where } i_k \text{ is randomly chosen from } \{1, \cdots, n\}.$$

(c) There exist scalars $M_1 > 0$ and $M_2 \geq 1$ such that, for all $k \in \mathbb{N}$,

$$\mathbb{E}[||\nabla f_{i_k}(x_k)||_2^2] \leq M_1 + M_2 ||\nabla F(x_k)||_2^2.$$

The following lemma, which is described in (2.5) in Chapter 2.1.3 and the next two lemmas, which are from [7], are useful in our analysis.

Lemma 5.3.2. *Under Assumption 2.4, the optimality gap at a given point in terms of the squared l_2-norm of the gradient of the objective at that point:*

$$2\sigma(F(x) - F_*) \leq ||\nabla F(x)||_2^2 \quad \text{for all } x \in \mathbb{R}^d.$$

Lemma 5.3.3. *Under Assumption 2.2, the iterates of LMSG (Algorithm LMSG) satisfy*

the following inequality for all $k \in \mathbb{N}$:

$$\mathbb{E}[F(x_{k+1}) - F(x_k)] \leq -\alpha_k^{LMSG} \nabla F(x_k)^T \mathbb{E}[\nabla f_{i_k}(x_k)] + \frac{1}{2}(\alpha_k^{LMSG})^2 L \mathbb{E}[||\nabla f_{i_k}(x_k)||_2^2].$$

Lemma 5.3.4. *Under Assumption 2.2 and 5.3.1, the iterates of LMSG (Algorithm LMSG) satisfy the following inequality for all $k \in \mathbb{N}$:*

$$\mathbb{E}[F(x_{k+1})] - F(x_k) \leq -(1 - \frac{1}{2}\alpha_k^{LMSG} L M_2)\alpha_k^{LMSG} ||\nabla F(x_k)||_2^2 + \frac{1}{2}(\alpha_k^{LMSG})^2 L M_1.$$

In Algorithm LMSG, we project LMSG stepsize to an interval, if the interval is fixed, we can have the algorithm converge to a neighborhood of a solution, if the interval is diminishing with $1/k$, the algorithm will converge. The theorems below show these two situations, respectively. First we discuss the conclusion for fixed interval.

Theorem 5.3.5. ***(LMSG with Fixed Interval Projection).*** *Under Assumptions 2.4 (with $F_{inf} = F_*$), 2.2 and 5.3.1, suppose that LMSG method (Algorithm LMSG) is run with projecting stepsize onto a fixed interval $[A,\ B]$, i.e., $\alpha_k = proj_{[A,\ B]}\left(\alpha_k^{LMSG}\right)$ for all $k \in \mathbb{N}$, satisfying*

$$0 < A \leq \alpha_k^{LMSG} \leq B \leq \frac{1}{LM_2}. \tag{5.11}$$

Then the expected optimality gap satisfies the following inequality for all $k \in \mathbb{N}$:

$$\mathbb{E}[F(x_{k+1}) - F_*] \leq \frac{\alpha_k^{LMSG} L M_1}{2\sigma} + (1 - \alpha_k^{LMSG}\sigma)^{k-1}\left(F(x_1) - F_* - \frac{\alpha_k^{LMSG} L M_1}{2\sigma}\right),$$
$$\lim_{k\to\infty} \mathbb{E}[F(x_{k+1}) - F_*] \leq \frac{\alpha_k^{LMSG} L M_1}{2\sigma} \leq \frac{B L M_1}{2\sigma}.$$

Proof. Using Lemma 5.3.4 with (5.11) and Lemma 5.3.2, we have for all $k \in \mathbb{N}$ that

$$\begin{aligned}
\mathbb{E}[F(x_{k+1})] - F(x_k) &\leq -(1 - \frac{1}{2}\alpha_k^{LMSG} L M_2)\alpha_k^{LMSG} ||\nabla F(x_k)||_2^2 + \frac{1}{2}(\alpha_k^{LMSG})^2 L M_1 \\
&\leq -\frac{1}{2}\alpha_k^{LMSG} ||\nabla F(x_k)||_2^2 + \frac{1}{2}(\alpha_k^{LMSG})^2 L M_1 \\
&\leq -\alpha_k^{LMSG}\sigma(F(x_k) - F_*) + \frac{1}{2}(\alpha_k^{LMSG})^2 L M_1.
\end{aligned}$$

Subtracting F_* from both sides, taking total expectations, and rearranging, this yields

$$\mathbb{E}[F(x_{k+1})] - F(x_*) \leq (1 - \alpha_k^{LMSG}\sigma)\mathbb{E}[F(x_k) - F_*] + \frac{1}{2}(\alpha_k^{LMSG})^2 LM_1.$$

Subtracting the constant $\alpha_k^{LMSG} LM_1/(2\sigma)$ from both sides, one obtains

$$\begin{aligned}
&\mathbb{E}[F(x_{k+1})] - \frac{\alpha_k^{LMSG} LM_1}{2\sigma} \\
\leq &(1 - \alpha_k^{LMSG}\sigma)\mathbb{E}[F(x_k) - F_*] + \frac{1}{2}(\alpha_k^{LMSG})^2 LM_1 - \frac{\alpha_k^{LMSG} LM_1}{2\sigma} \\
\leq &(1 - \alpha_k^{LMSG}\sigma)\left(\mathbb{E}[F(x_k) - F_*] - \frac{\alpha_k^{LMSG} LM_1}{2\sigma}\right).
\end{aligned}$$

Observe that the above inequality is a contraction inequality since

$$0 < \alpha_k^{LMSG}\sigma \leq \frac{1}{LM_2}\sigma = \frac{1}{M_2}\frac{\sigma}{L} \leq 1.$$

We repeatedly apply the above contraction inequality through iteration $k \in \mathbb{N}$ and obtain the results. □

When it comes to interval diminishing with $1/k$, to achieve the desired balance between diminishing the stepsizes sufficiently quickly to ensure convergence, but not so quickly so as to prevent the iterates from reaching the minimizer, one can choose the stepsizes to satisfy

$$\sum_{k=1}^{\infty} \alpha_k = \infty \text{ and } \sum_{k=1}^{\infty} \alpha_k^2 < \infty. \tag{5.12}$$

Hence when we compute the stepsize from LMSG, we project it onto the interval $\left[\frac{A}{\gamma+k}, \frac{B}{\gamma+k}\right]$ in order to satisfy (5.12). Below is the formal statement.

Theorem 5.3.6. ***(LMSG with Diminishing Interval Projection).*** *Under Assumptions 2.4 (with $F_{inf} = F_*$), 2.2 and 5.3.1, suppose that LMSG method (Algorithm LMSG) is run with projecting stepsize onto a diminishing interval $\left[\frac{A}{\gamma+k}, \frac{B}{\gamma+k}\right]$,*

i.e., $\alpha_k = proj_{\left[\frac{A}{\gamma+k},\ \frac{B}{\gamma+k}\right]}(\alpha_k^{LMSG})$ *for all* $k \in \mathbb{N}$*, satisfying*

$$A > \frac{1}{\sigma},\ B \geq A \text{ and } \gamma > 0 \text{ such that } \frac{B}{\gamma+1} \leq \frac{1}{LM_2}. \tag{5.13}$$

Then, for all $k \in \mathbb{N}$*, the expected optimality gap satisfies*

$$\mathbb{E}[F(x_{k+1})] - F(x_*)] \leq \frac{\nu}{\gamma+k}, \tag{5.14}$$

where

$$\nu := \max\left\{\frac{A^2LM_1}{2(A\sigma-1)},\ \frac{B^2LM_1}{2(B\sigma-1)},\ (\gamma+1)(F(x_1)-F_*)\right\}. \tag{5.15}$$

Proof. Using the similar argument of the proof in Theorem 5.3.5, one obtains

$$\mathbb{E}[F(x_{k+1})] - F(x_*) \leq (1-\alpha_k^{LMSG}\sigma)\mathbb{E}[F(x_k)-F_*] + \frac{1}{2}(\alpha_k^{LMSG})^2LM_1.$$

We now prove (5.14) by induction. First, the definition of ν ensures that it holds for $k=1$. Then, assuming (5.14) holds for some $k>1$, now we prove (5.14) holds for $k+1$. Defining $\hat{k} := \gamma + k$, then it follows from the above inequality that

$$\begin{aligned}
\mathbb{E}[F(x_{k+1})] - F(x_*) &\leq (1-\alpha_k^{LMSG}\sigma)\frac{\nu}{\hat{k}} + \frac{1}{2}(\alpha_k^{LMSG})^2LM_1 \\
&= \frac{\hat{k}-\alpha_k^{LMSG}\hat{k}\sigma}{\hat{k}^2}\nu + \frac{1}{2}(\alpha_k^{LMSG})^2LM_1 \\
&= \frac{\hat{k}-1}{\hat{k}^2}\nu + \frac{1-\alpha_k^{LMSG}\hat{k}\sigma}{\hat{k}^2}\nu + \frac{1}{2}(\alpha_k^{LMSG})^2LM_1.
\end{aligned}$$

Now we prove the term $\frac{1-\alpha_k^{LMSG}\hat{k}\sigma}{\hat{k}^2}\nu + \frac{1}{2}(\alpha_k^{LMSG})^2LM_1$ in the above inequality is nonpositive. Defining $t := \alpha_k^{LMSG}\hat{k}$ where $t \in [A,\ \ B]$ and considering the function $h(t) := \frac{t^2}{t\sigma-1}$. Taking the derivative of $h(t)$, we obtain $h'(t) = \frac{t^2\sigma-2t}{(t\sigma-1)^2}$. Letting the derivative be equal to 0 and given that $t \in [A,\ \ B]$ with $A > \frac{1}{\sigma}$, we know $\min h(t) = h(\frac{2}{\sigma})$ and $\max h(t) = \max\{h(A),\ h(B)\}$. From the definition of ν, it is true that

$$\frac{2\nu}{LM_1} \geq \max\{h(A),\ h(B)\} = \max\left\{\frac{A^2}{A\sigma-1},\ \frac{B^2}{B\sigma-1}\right\} \geq h(t) \text{ for all } t \in [A,\ \ B].$$

Replacing t with $\alpha_k^{LMSG}\hat{k}$ and the followings are true:

$$\frac{2(\alpha_k^{LMSG}\sigma - 1)}{\hat{k}^2}\nu \geq (\alpha_k^{LMSG})^2 LM_1,$$
$$\frac{1 - \alpha_k^{LMSG}\hat{k}\sigma}{\hat{k}^2}\nu + \frac{1}{2}(\alpha_k^{LMSG})^2 LM_1 \leq 0.$$

Finally we complete the final step of induction that

$$\mathbb{E}[F(x_{k+1})] - F(x_*) \leq \frac{\hat{k}-1}{\hat{k}^2}\nu \leq \frac{\hat{k}-1}{\hat{k}^2-1}\nu \leq \frac{1}{\hat{k}+1}\nu.$$

□

Extension to Nonconvex Objective. Even though we mainly focus on solving strongly convex objective and obtain convergence (rate) results in this Chapter, Algorithm LMSG is specifically designed to deal with nonpositive curvature and solve nonconvex problems, Algorithm LMSG always guarantees a positive meaningful stepsize.

As for convegence results, with only Assumptions 2.2 and 5.3.1, we have the following conclusions. For LMSG with Fixed Interval Projection, the sum of squared gradients remains finite. For LMSG with Diminishing Interval Projection, the infimum of the expected gradient norms converges to zero. To be more precisely, the weighted average norm of the squared gradients converges to zero.

5.4 Numerical Experiments

5.4.1 Implementation

In this section, we conduct numerical experiments on LMSG (Algorithm LMSG), which includes BBSG (Algorithm BBSG) as a special case. In order to demonstrate its efficiency, we implement LMSG together with fixed stepsize algorithms for comparison purposes. In particular, we apply LMSG to solving a standard testing problems in machine learning: logistic regression. We test LMSG on standard real data sets, which were downloaded from the LIBSVM website. Detailed information of the data sets are given in Table 5.1.

Table 5.1: Dataset Description

Dataset	n	d	Dataset	n	d
a1a	1605	123	heart	270	13
australian	690	14	madelon	2000	500
diabetes	768	8	splice	1000	60
german.numer	1000	24	svmguide3	1243	21

For each dataset, we run 5 epochs. We first implement gradient descent method with different fixed stepsizes and choose the optimal constant $C = 0.1$ from $\{0.01, 0.1, 1, 10, 100\}$. Then we compare the performances of LMSG with history length $m = 1, 3, 5$ with gradient descent method using fixed stepsize $C = 0.1$. In LMSG, once a stepsize is computed (described in Algorithm LMSG), it is projected to the range $[\omega,\ \Omega]$ with $\omega =$ "low" $\times C$ and $\Omega =$ "up" $\times C$, where we choose "low" from $\{1/2, 1/4, 1/8, 1/16, 1/32\}$ and "up" from $\{2, 4, 8, 16, 32\}$.

Besides parameter settings, we conduct tests on different types of gradient descent, including full gradient, mini-batch gradient and single gradient by choosing different proportion (i.e. $\{1, 1/2, 1/4, 1/8, 1/16, 1/32\}$) of the full gradient. When the proportion is not an integer, we take the smallest integer larger than that number.

In order to avoid the influence of numerical error and the computation of excessively small or large stepsizes, we remove previously computed gradients (in a similar manner as in the inner **loop** of Algorithm LMSG) if more than one was currently being held and any of the corresponding computed eigenvalues were smaller than 10^{-4} or larger than 10^4 in absolute value. Similarly, prior to computing corresponding stepsizes, we projected any computed quadratic term coefficient so that it would have an absolute value at least 10^{-4} and at most 10^4, and we projected any computed cubic term coefficient onto the range $[10^{-4},\ 10^4]$. (For the quadratic term coefficients, this projection was performed to maintain the sign of the originally computed coefficient.)

5.4.2 Numerical Results

As a comparison, we show the results that at 1/16 of the full gradient, within the consistent projection range $[\frac{1}{8}\times,\ 16\times]$, for all testing problems, at least two out of three LMSG algorithms ($m = 1, 3, 5$) works better than the one with fixed stepsize. The table below shows the detailed description, note % indicates gradient proportion and $Fave - fix, Fave - 1, Fave - 3, Fave - 5$ refers to the average function value over all 80 (5Epochs $\times$ 16Iterations/Epoch) iterations for fixed stepsize, LMSG with history length $1, 3, 5$, respectively.

Table 5.2: Numerical Results

Dataset	%	C	low	up	$Fave - fix$	$Fave - 1$	$Fave - 3$	$Fave - 5$
a1a	1/16	0.10	1/8	16	0.470	0.431	0.418	0.419
australian	1/16	0.10	1/8	16	0.438	0.370	0.371	0.377
diabetes	1/16	0.10	1/8	16	0.671	0.615	0.636	0.629
german.numer	1/16	0.10	1/8	16	0.638	0.660	0.634	0.615
heart	1/16	0.10	1/8	16	0.460	0.446	0.448	0.406
madelon	1/16	0.10	1/8	16	0.663	0.686	0.652	0.640
splice	1/16	0.10	1/8	16	0.557	0.638	0.551	0.549
svmguide3	1/16	0.10	1/8	16	0.585	0.562	0.566	0.568

We can observe from Table 5.2 that LMSG with history length 5 works best or almost the best for all datasets. The comparison results of four algorithms for each dataset are shown in the following figures. In these figures, we use "Fixed, LMSG-1, LMSG-3, LMSG-5" for stochastic gradient with fixed stepsize, LMSG with history length $1, 3, 5$, respectively.

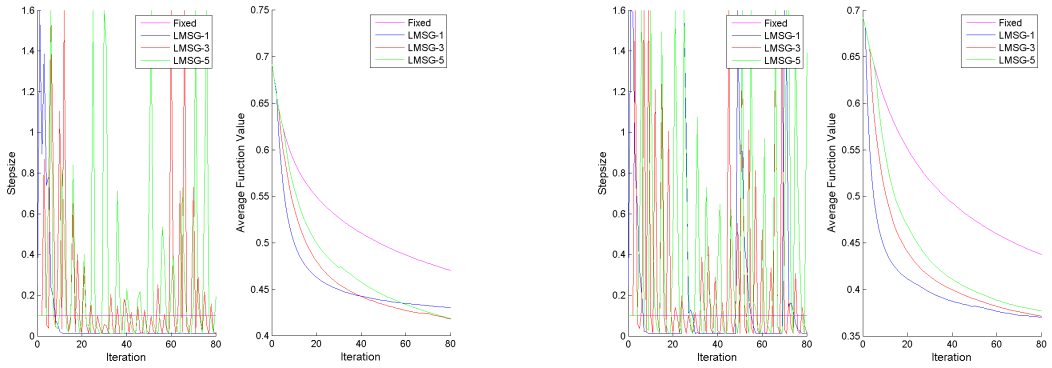

Figure 5.1: Stepsize and average function value for "a1a" (left two plots) and "australian" (right two plots).

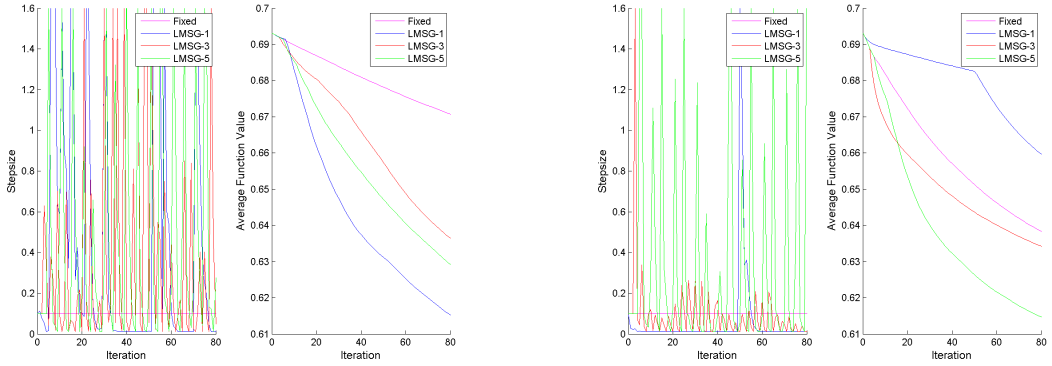

Figure 5.2: Stepsize and average function value for "diabetes" (left two plots) and "german.numer" (right two plots).

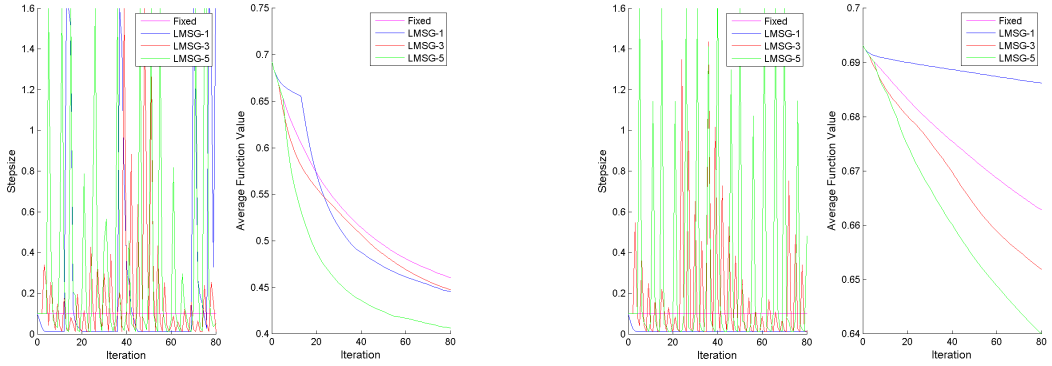

Figure 5.3: Stepsize and average function value for "heart" (left two plots) and "madelon" (right two plots).

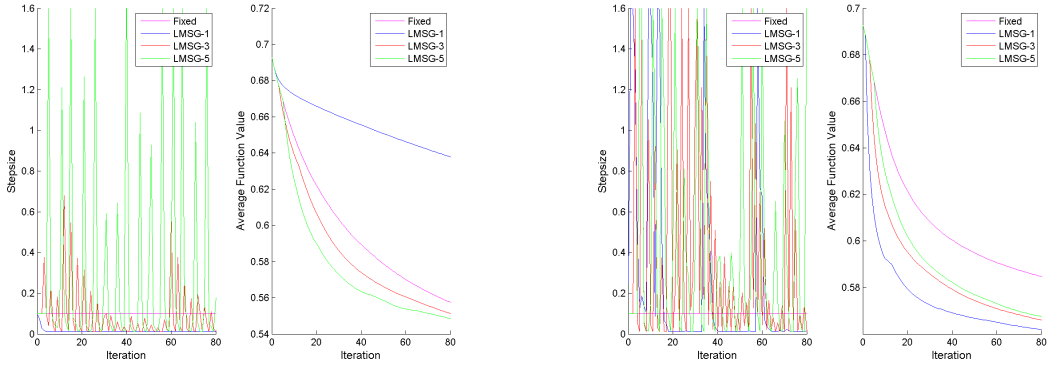

Figure 5.4: Stepsize and average function value for "splice" (left two plots) and "svmguide3" (right two plots).

5.5 Conclusion

In this part we have designed, analyzed, and experimented with a limited memory stochastic gradient (LMSG) method for solving optimization problems arising in machine learning. Our method was specifically designed for large scale optimization problems where the computation of full gradient is too expensive. The method always guarantees a positive meaningful stepsize especially when nonconvexity / nonpositive curvature is encountered. With a projection of stepsizes, we ensure convergence to a neighborhood of the optimal

solution or to the optimal solution depending on whether the projection interval is fixed or diminishing with $1/k$. Our numerical experiments suggest that our approach can outperform an SG method with a fixed stepsize.

Chapter 6

Conclusion

In this dissertation, we focus on the exploration and development of LMSD methods from both theoretical and practical perspectives. First we have shown that the limited memory steepest descent (LMSD) method proposed by [22] possesses an R-linear rate of convergence for any history length $m \in [n]$ when it is employed to minimize a strongly convex quadratic function. Our analysis effectively extends that in [15], which covers only the $m = 1$ case. We have also provided the results of numerical experiments to demonstrate that the behavior of the algorithm reflects certain properties revealed by the theoretical analysis. Besides, we also show that the convergence rate of LMSD is the same when Ritz values are replaced by harmonic Ritz values; see §7 in [22].

Once the thoeretical convergence rate result is analyzed, we designed and experimented with a limited memory steepest descent (LMSD) method for solving unconstrained nonconvex optimization problems. The unique feature of our algorithm is a novel approach for handling nonpositive curvature; in particular, we propose that when nonpositive curvature is encountered, stepsizes can be computed by constructing local cubic models of the objective function, for which reasonable values of the quadratic and cubic term coefficients can be derived using previously computed gradient information. Our numerical experiments suggest that our approach yields superior performance in practice compared to algorithms that do not attempt to incorporate problem information when computing stepsizes when nonpositive curvature is encountered.

Next, we extend this method from deterministic optimization to stochastic optimization by proposing a limited memory stochastic gradient (LMSG) method for solving optimization problems arising in machine learning. Our method was specifically designed for large scale optimization problems where the computation of full gradient is too expensive or even unavailable. The method always guarantees a positive meaningful stepsize even when nonconvexity / nonpositive curvature is encountered. With a projection of stepsizes, we ensure convergence to a neighborhood of the optimal solution or to the optimal solution depending on whether the projection interval is fixed or diminishing with $1/k$. Our numerical experiments suggest that our approach yields comparable and sometimes better results than stochastic gradient (SG) methods with best-tuned fixed stepsize.

Bibliography

[1] S. Amari. *Natural gradient works efficiently in learning. Neural computation.* MIT Press, edition, 1998.

[2] J. Barzilai and J. M. Borwein. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[3] C. Beattie. Harmonic Ritz and Lehmann bounds. *Electronic Transactions on Numerical Analysis*, 7:18–39, 1998.

[4] A. Ben-Tal and A. Nemirovski. Robust Optimization - Methodology and Applications. *Mathematical Programming, Series B*, 92:453–480, 2002.

[5] M. Biglari and M. Solimanpur. Scaling on the Spectral Gradient Method. *Journal of Optimization Theory and Applications*, 158(2):626–635, 2013.

[6] E.G. Birgin, I. Chambouleyron, and J.M. Martinez. Estimation of the optical constants and the thickness of thin films using unconstrained optimization. *Journal of Computational Physics*, 151(2):862–880, 1999.

[7] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *arXiv:1606.04838, 2016.*

[8] C. G. Broyden. The Convergence of a Class of Double-Rank Minimization Algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6(1):76–90, 1970.

[9] E. J. Candes and T. Tao. Near Optimal Signal Recovery from Random Projections:

Universal Encoding Strategies. *IEEE Transactions on Informaton Theory*, 52:5406–5425, 2006.

[10] A. Cauchy. Méthode Générale pour la Résolution des Systèms d'Equations Simultanées. *Compte Rendu des Séances de L'Académie des Sciences*, 25:536–538, 1847.

[11] F. E. Curtis and W. Guo. Handling Nonpositive Curvature in a Limited Memory Steepest Descent Method. *IMA Journal of Numerical Analysis*, 36(2):717–742, 2016.

[12] Y. Dai, J. Yuan, and Y.-X. Yuan. Modified Two-Point Stepsize Gradient Methods for Unconstrained Optimization. *Computational Optimization and Applications*, 22(1):103–109, 2002.

[13] Y.-H. Dai. Alternate Step Gradient Method. *Optimization*, 52(4-5):395–415, 2003.

[14] Y.-H Dai, W. W. Hager, K. Schittkowski, and H. Zhang. The cyclic BarzilaiBorwein method for unconstrained optimization. *IMA Journal of Numerical Analysis*, (26):604627, 2006.

[15] Y.-H. Dai and L.-Z. Liao. *R*-linear Convergence of the Barzilai and Borwein Gradient Method. *IMA Journal of Numerical Analysis*, 22:1–10, 2002.

[16] Y.-H. Dai and Y.-X. Yuan. Alternate minimization gradient method. *IMA Journal of Numerical Analysis*, (23):377–393, 2003.

[17] R. De Asmundis, D. Di Serafino, W. W. Hager, G. Toraldo, and H. Zhang. An Efficient Gradient Method Using the Yuan Steplength. *Computational Optimization and Applications*, 2014.

[18] R. De Asmundis, D. Di Serafino, F. Riccio, and G. Toraldo. On Spectral Properties of Steepest Descent Methods. *IMA Journal of Numerical Analysis*, 33:1416–1435, 2013.

[19] D. L. Donoho. Compressed Sensing. *IEEE Transactions on Informaton Theory*, 52:1289–1306, 2006.

[20] M. Figueiredo, R. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1:586–598, 2007.

[21] R. Fletcher. A New Approach to Variable Metric Algorithms. *Computer Journal*, 13(3):317–322, 1970.

[22] R. Fletcher. A Limited Memory Steepest Descent Method. *Mathematical Programming*, 135(1-2):413–436, 2012.

[23] W. Glunt, T.L. Hayden, and M. Raydan. Molecular conformations from distance matrices. *Journal of Computational Chemistry*, 14(1):114–120, 1993.

[24] D. Goldfarb. A Family of Variable Metric Updates Derived by Variational Means. *Mathematics of Computation*, 24(109):23–26, 1970.

[25] D. Goldfarb and G. Iyengar. Robust Portfolio Selection problems. *Mathematics of Operations Research*, 28:1–38, 2003.

[26] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Third edition, 1996.

[27] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr and SifDec: A Constrained and Unconstrained Testing Environment, Revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394, 2003.

[28] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEst: A Constrained and Unconstrained Testing Environment with Safe Threads. Technical Report RAL-TR-2013-005, Rutherford-Appleton Laboratory, 2013.

[29] L. Grippo, F. Lampariello, and S. Lucidi. A Nonmonotone Line Search Technique. *SIAM Journal on Numerical Analysis*, 23(1986):707–716, 1986.

[30] S. Guenter, M. Sempf, P. Merkel, E. Strumberger, and C. Tichmann. Robust Control of Resistive Wall Modes Using Pseudospectra. *New Journal of Physics*, 11:1–40, 2009.

[31] H. Huang. Efficient reconstruction of 2D images and 3D surfaces. *Ph.D. thesis, University of BC, Vancouver*, 2008.

[32] K. Ito and K. Kunisch. Optimal control of elliptic variational inequalities. *Applied Mathematics and Optimization*, 41(3):343–364, 2000.

[33] S. B. Kafaki and M. Fatemi. A Modified Two-Point Stepsize Gradient Algorithm for Unconstrained Minimization. *Optimization Methods and Software*, 28(5):1040–1050, 2013.

[34] Y. L. Lai. Some Properties of the Steepest Descent Method. *Acta Mathematicae Applicatae Sinica*, 4(2):106–116, 1981.

[35] D.-H. Li and M. Fukushima. A Modified BFGS Method and its Global Convergence in Nonconvex Minimization. *Journal of Computational and Applied Mathematics*, 129(1-2):15–35, 2001.

[36] G. Loosli and S. Canu. *Optimization in Signal and Image Processing, chap. Quadratic Programming and Machine Learning - Large Scale Problems and Sparsity.* Wiley, edition, 2010.

[37] H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[38] J. L. Morales. A Numerical Study of Limited Memory BFGS Methods. *Applied Mathematics Letters*, 15:481–487, 2002.

[39] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. *In Proceedings of the 22nd International Conference on Machine Learning*, pages 625–632, 2005.

[40] J. Nocedal. Updating Quasi-Newton Matrices With Limited Storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[41] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, Second edition, 2006.

[42] C. C. Paige, B. N. Parlett, and H. van der Vorst. Approximate Solutions and Eigenvalue Bounds from Krylov Subspaces. *Numerical Linear Algebra with Applications*, 2(2):115–133, 1995.

[43] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998.

[44] M. Raydan. On the Barzilai and Borwein Choice of Steplength for the Gradient Method. *IMA Journal of Numerical Analysis*, 13(3):321–326, 1993.

[45] M. Raydan. The Barzilai and Borwein Gradient Method for the Large Scale Unconstrained Minimization Problem. *SIAM Journal on Optimization*, 7(1):26–33, 1997.

[46] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[47] D. F. Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24(111):647–656, 1970.

[48] S. Sra, S. Nowozin, and S. J. Wright. *Optimization for Machine Learning. Neural information processing series.* MIT Press, edition, 2012.

[49] Conghui Tan, Shiqian Ma, Yu-Hong Dai, and Yuqiu Qian. Barzilai-borwein step size for stochastic gradient descent.

[50] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, Series B (Methodological):267–288, 1996.

[51] J. Vanbiervliet, K. Verheyden, W. Michiels, and S. Vandewalle. A Nonsmooth Optimisation Approach for the Stabilisation of Time-delay Systems. *ESAIM: Control, Optimization and Calculus of Variations*, 14:478–493, 2008.

[52] V. Vapnik and Cortes C. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

[53] Y. Xiao, Q. Wang, and D. Wang. Notes on the Dai-Yuan-Yuan Modified Spectral Gradient Method. *Journal of Computational and Applied Mathematics*, 234(10):2986–2992, 2010.

[54] Y.-X. Yuan. A New Stepsize for the Steepest Descent Method. *Journal of Computational Mathematics*, 24(2):149–156, 2006.

[55] Y.-X. Yuan. Gradient Methods for Large Scale Convex Quadratic Functions. *Optimization and Regularization for Computational Inverse Problems and Applications*, pages 141–155, 2010.

[56] H. Zhang and W. W. Hager. A Nonmonotone Line Search Technique and its Application to Unconstrained Optimization. *SIAM Journal on Optimization*, 14(4):1043–1056, 2004.

[57] B. Zhou, L. Gao, and Y.-H. Dai. Gradient Methods with Adaptive Step-sizes. *Computational Optimization and Applications*, (35):69–86, 2006.

Biography

Wei Guo received his B.S. in Mathematics and Applied Mathematics from University of Science and Technology of China in 2011. His research interests lie in both theoretical analysis and development of efficient nonlinear optimization algorithms with applications in big data problems arising in machine learning and financial services. He served as the treasurer of INFORMS Student Chapter and Ph.D. representative of the Department of Industrial and Systems Engineering Council and Graduate Student Senate. He is a first prize winner of the 2016 American Express Machine Learning Competition. He will join J.P. Morgan as an Associate in Summer 2017.