FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

RIEMANNIAN BROYDEN FAMILY OF LIMITED-MEMORY QUASI-NEWTON METHODS

By

SHUGUANG ZHANG

A Dissertation submitted to the
Department of Mathematics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2024

Shuguang Zhang defended this dissertation on March 14 2024.
The members of the supervisory committee were:

Kyle A. Gallivan
Professor Co-Directing Dissertation

Wen Huang
Professor Co-Directing Dissertation

Gordon Erlebacher
University Representative

Mark Sussman
Committee Member

Giray Okten
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my advisors, Kyle A. Gallivan and Wen Huang, for their invaluable guidance throughout my PhD journey. I would also like to thank my committee members, Gordon Erlebacher, Mark Sussman and Giray Okten for their contributions and counsel on this dissertation. I would like to express my deep gratitude to the love of my life and to those who love and support me unconditionally, in particular to my parents, Haixia Shu and Ming Zhang, my girlfriend, Bingyu Su.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Quasi-Newton methods have gained popularity across various domains, providing efficient iterative algorithms for finding optimal solutions to unconstrained optimization problems. Their limited-memory variants offer advantages in terms of reduced storage and computational requirements. This dissertation generalizes full Broyden family of limited-memory quasi-Newton optimization methods on Riemannian manifold (LRBroyden). The work encompasses the comprehensive analysis, implementation, application and evaluation of LRBroyden. An essential extension to this research involves the adaptation of these algorithms to stochastic methods tailored for large-scale optimization. This results in the generalization of the stochastic variant of the Riemannian full Broyden family of limited-memory quasi-Newton methods, incorporating with variance reduction (LR-SBroyden-VR). The dissertation provides empirical insights into parameter selection and offers a thorough evaluation of the performance of various methods.

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem

Riemannian optimization, or optimization on Riemannian manifolds is to find an optimum of a real-valued function f defined on a Riemannian manifold, i.e.,

$$\min_{x \in \mathcal{M}} f(x), \tag{1.1}$$

where $\mathcal{M}$ is a Riemannian manifold. Optimization on manifolds can be viewed as unconstrained optimization on a constrained space. If the manifold $\mathcal{M}$ is $\mathbb{R}^n$, then the problem is simply a Euclidean unconstrained optimization problem.

In the last fifty years, there has been significant development in a powerful collection of algorithms designed for the unconstrained optimization of smooth functions. The concept of quasi-Newton methods can be traced back to mid 1950s. These methods, which require only gradient information but achieve superlinear convergence, are highly competitive in the field of optimization. The Broyden class represents a family of quasi-Newton methods that depend on a real parameter ϕ_k, which has the Hessian approximation update formula

$$B_{k+1} = (1 - \phi_k) B_{k+1}^{\mathrm{BFGS}} + \phi_k B_{k+1}^{\mathrm{DFP}},$$

where B_{k+1}^{BFGS} denotes the update of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method and B_{k+1}^{DFP} denotes the update of the Davidon-Fletcher-Powell (DFP) method. The family of restricted Broyden methods is defined with $\phi_k \in [0, 1]$ and has been a topic of much interest over the past few years, owing to its effectiveness and well-developed convergence analysis. Nevertheless, some researchers have advocated for considering a broader range of values for ϕ_k and have suggested that there may be benefits in exploiting other members within the full Broyden family, as discussed in, e.g., [11, 61, 8, 32].

In recent years, the Broyden family of methods have been developed on a Riemannian manifold, evolving into a highly effective optimization algorithm with efficient implementations such as that in ROPTLIB [26]. The concept of optimizing a real-valued function on a manifold dates back

to the early 1970s. In 2008, Absil et al. [1] introduced a practical approach to optimization on manifolds, leveraging the principles of differential geometry. In 2011, Qi [40] proposed a method to generalize BFGS to Riemannian manifolds and conducted convergence analysis in her dissertation. Ring and Wirth [47] presented another approach for implementing BFGS on Riemannian manifolds. More recently, Huang [22] explored quasi-Newton optimization algorithms on Riemannian manifolds in his dissertation, introducing the Riemannian Broyden Family of methods (RBroyden). These developments have significantly advanced the field of optimization on Riemannian manifolds.

In both Euclidean and Riemannian contexts, when dealing with large-scale problems, the Broyden family faces challenges in terms of storage requirements and associated costs due to the use of Hessian approximation. To address this, limited-memory variant of quasi-Newton methods have been developed. The limited-memory BFGS method in Euclidean space was initially proposed by Liu and Nocedal [33]. Reed [46] extended the limited-memory BFGS to the Broyden family. Erway and Marcia [18] applied a compact representation, as introduced by Byrd, Nocedal and Schnabel [9], within the LBroyden family, albeit with constraints on the choice of the Broyden parameter. Deguchy et al. [3] further extended the results from [18] to cover the full Broyden family, allowing the Broyden parameter to take any real values. However, none of these papers delved deeper into the methodology for selecting the Broyden parameters.

On a Riemannian manifold, Huang et al. [27] proposed the Riemannian limited-memory variant of BFGS method, known as LRBFGS. The method is known for its robustness, effectiveness and suitability for solving large problems where computing Hessian matrices is prohibitively expensive. Other than LRBFGS, however, a limited-memory variant of the RBroyden family has not been explored until now. This dissertation aims to address several questions: Can we develop a robust and efficient limited-memory quasi-Newton algorithm by considering other members of the Broyden family? How can we choose from the members of the LRBroyden family for different large-scale problems? Is it possible to efficiently implement the LRBroyden family of methods, similar to LRBFGS? In this work, these questions are answered by proposing an LRBroyden algorithm that accommodates a wider range of Broyden parameters, allowing for different values at each step. The performance of various members in the LRBroyden family is thoroughly investigated.

The idea of the LRBroyden family of methods can also be applied to the stochastic optimization methods. The focused problem is of the following form:

$$\min_{\omega \in \mathcal{M}} \{f(\omega) := \frac{1}{n} \sum_{i=1}^{n} f_i(\omega)\}, \tag{1.2}$$

where f is a smooth real-valued functions on a Riemannian manifold $\mathcal{M}$. This problem type finds widespread applications in machine learning, statistical inference, and image processing, particularly when dealing with large datasets (i.e., when n is very large). Stochastic optimization methods, in this context, introduce randomness either in the objective function or in the optimization algorithm. Since 1951, with the work of Robbins and Monro [48], stochastic gradient descent (SGD) has been a core strategy for addressing such problems. Unlike traditional gradient descent, SGD considers only a sub-sample or a small batch of the loss function, significantly reducing the computational complexity per iteration.

Several adaptive methods have been established based on SGD to enhance its performance, including AdaGrad [17], Adam [31], and AMSGrad [45]. Variance-reduced SGD, proposed by Johnson and Zhang [28], aims to mitigate the issues introduced by the randomness of noisy gradient estimates. In the pursuit of achieving faster convergence, the research on stochastic quasi-Newton methods emerged. Byrd, Hansen, Nocedal, and Singer [7] introduced a stochastic quasi-Newton method, suggesting that it outperforms SGD more effectively than gradient rescaling. Another approach by Moritz et al. [36] demonstrated rapid convergence to high levels of precision and performed well across a wide range of step sizes. These developments offer promising avenues for tackling stochastic optimization problems more efficiently.

In recent years, many of these stochastic methods have been generalized to certain Riemannian settings, including Riemannian SGD [4], Riemannian adaptive gradient methods like RAdaGrad, RAdam, RAMSGrad [3], as well as variance-reduced Riemannian SGD [60]. Moreover, stochastic quasi-Newton methods have been adapted to Riemannian manifolds by researchers like Roychowdhury [49] and Sato et al. [30]. Their focus was on the stochastic LRBFGS, employing variance reduction techniques, and their experiments demonstrated that these algorithms outperformed other state-of-the-art Riemannian stochastic gradient methods. Nevertheless, other members of the Broyden family have not been explored in the context of stochastic algorithms, neither in Euclidean nor Riemannian settings. Motivated by the above considerations, we equip the LRBroyden family of methods with stochastic strategy and investigate its robustness and efficiency in this dissertation.

1.2 Research Overview and Dissertation Statement

The primary objective of this dissertation is to develop, analyze, and assess the limited-memory variant of Riemannian full Broyden family methods. We propose to exploit a strategy for select-

ing ϕ in LRBroyden that is more efficient and robust than LRBFGS; to investigate how to select appropriate stepsizes with varing ϕ values; to propose and analyze the efficient and robust Riemannian stochastic Broyden family of quasi-Newton mtehods; to provide user guidelines on how to choose between various stochastic parameters. We also provide a tool box for LRBroyden and LR-SBroyden-VR. The toolbox relies on ROPTLIB, an object-oriented C++ library for optimization on Riemannian manifolds [26].

Through these objectives, the dissertation aims to enhance the effectiveness and practicality of the limited-memory variant of Riemannian full Broyden family methods, positioning them as valuable tools for optimization tasks. This dissertation asserts that the proposal above can be achieved by the following:

1. the development of LRBroyden, the use of intrinsic representation and compact representation in the implementation, and the convergence analysis of LRBroyden (Chapter 3);
2. the development of methodologies for selecting ϕ in LRBroyden, and the experimental evidence that illustrates robustness and effectiveness of the chosen methodology used in this dissertation (Chapter 3, 4);
3. the experimental evidence that exploits the relationship between ϕ and initial stepsize to enhance performance of LRBroyden, and the numerical results that show the superiority of Davidon's ϕ in specific synthetic problems (Chapter 4);
4. the design of a hybrid strategy between the choices of Davidon's and BFGS's ϕ in LRBroyden, and the numerical results that illustrate the improvements of the hybrid strategy with appropriate initial stepsize selection (Chapter 5);
5. the application of the LRBroyden family to stochastic method, and the development of LR-SBroyden-VR, and the convergence analysis of LR-SBroyden-VR (Chapter 6);
6. the experimental evidence that LR-SBroyden-VR with Davidon's ϕ offers advantages in the selection of stochastic parameters, and the heuristic conclusion on how to determine these parameters based on practical insights (Chapter 7);
7. the experimental evidence that LR-SBroyden-VR is more efficient and more robust compared to the existing stochastic quasi-Newton algorithms (Chapter 7).

1.3 Dissertation Outline

The dissertation is organized as follows.

Chapter 2. This chapter briefly reviews some basic definitions and concepts for Riemannian manifolds and Riemannian optimization. It goes on to introduce the Broyden family of quasi-Newton methods in Euclidean space, covering both non-limited-memory and limited-memory variants. The chapter also includes a complexity comparison with some existing algorithms in the LBroyden family. Finally, it reviews a secant condition under Riemannian settings.

Chapter 3. This chapter generalizes the LBroyden family on a Riemannian manifold, providing implementation details and computational efficiency analysis. It also delves into the methods for selecting the Broyden parameters within the algorithm. The convergence analysis includes global convergence and R-linear convergence for the LRBroyden family of methods.

Chapter 4. This chapter explores the relationship between initial stepsize and Broyden parameter by testing with a constant ϕ. Then LRBroyden is tested with Davidon's choice of ϕ in some specific synthetic problems and found to be superior than LRBFGS.

Chapter 5. This chapter provides a hybrid strategy of choosing ϕ between Davidon and BFGS. With a suitable choice of the initial stepsize, the efficiency and robustness of the strategy are investigated and demonstrated empirically.

Chapter 6. This chapter applies the stochastic LRBroyden family of methods. Inspired by Kasai et al. [30], we generalize an LR-SBroyden-VR algorithm with its implementation details and computational efficiency analysis. Then the global convergence and local convergence rate of the LR-SBroyden-VR method is provided.

Chapter 7. This chapter includes a systematic empirical comparison of LR-SBroyden-VR, existing Riemannian stochastic LRBFGS and Riemannian stochastic variance-reduced gradient descent methods. We set up the experiments with different parameter pairs in order to determine the appropriate values in each situation. Then all the methods are evaluated under the selected parameter settings.

Chapter 8. This chapter gives a summary of completed work.

CHAPTER 2

PRELIMINARIES

This section reviews some important concepts and definitions of Riemannian manifolds that are used in this dissertation. Additionally, the Riemannian optimization algorithms of interest are identified and characterized briefly.

2.1 Riemannian Geometry

2.1.1 Tangent Space

The direction of motion on a manifold must be defined to apply line search algorithms. Consider a smooth mapping γ: $\mathbb{R} \to \mathcal{M}$ which satisfies $\gamma(0) = x$. Given a smooth real-valued function f: $\mathcal{M} \to \mathbb{R}$, we can define a mapping that is the direction at x along γ, also called a tangent vector to the curve γ at $t = 0$:

$$\dot{\gamma}(0)f = (f \circ \gamma)'(0) = \lim_{h \to 0} \frac{f(\gamma(h)) - f(\gamma(0))}{h}.$$

The formal definition of tangent vectors is as follows.

Definition 2.1. *(tangent vector). Let $\mathcal{F}_x(\mathcal{M})$ denote the set of smooth real-valued functions on a neighborhood of x. A tangent vector ξ_x to a manifold $\mathcal{M}$ at a point x is a mapping from $\mathcal{F}_x(\mathcal{M})$ to $\mathbb{R}$ such that there exists a curve γ on $\mathcal{M}$ with $\gamma(0) = x$, satisfying*

$$\xi_x f = \dot{\gamma}(0)f = \frac{df(\gamma(t))}{dt}|_{t=0}$$

for all $f \in \mathcal{F}_x(\mathcal{M})$. The curve γ is said to realize the tangent vector ξ_x. The point x is called the root of the tangent vector ξ_x.

The set of all tangent vectors at x is called the tangent space to $\mathcal{M}$ at x, denoted as $\mathrm{T}_x\mathcal{M}$. This generalizes the idea of direction and allows us to perform the line search on the tangent space. For Riemannian optimization, we work on a tangent space and go back to the manifold using retraction, which is discussed later. The union of all tangent spaces is called the tangent bundle of the manifold and denoted as $\mathrm{T}\mathcal{M}$.

A vector field ξ on a manifold $\mathcal{M}$ is a smooth function from $\mathcal{M}$ to the tangent bundle, which assigns to each point $x \in \mathcal{M}$ a tangent vector $\xi_x \in \mathrm{T}_x\mathcal{M}$.

2.1.2 Riemannian Metric

The tangent space at a point on the manifold provides us with a vector space of tangent vectors that give an idea of direction on the manifold. Endowing the tangent space with an inner product allows us to compute angles and lengths of tangent vectors.

Definition 2.2. *(inner product). Let $\mathcal{M}$ be a smooth manifold and $x \in \mathcal{M}$. An inner product $\langle \cdot, \cdot \rangle_x$ on $\mathrm{T}_x\mathcal{M}$ is a bilinear, symmetric positive-definite form, i.e., $\forall \xi_x, \zeta_x, \eta_x \in \mathrm{T}_x\mathcal{M}$, $a, b \in \mathbb{R}$, $\langle a\xi_x + b\zeta_x, \eta_x \rangle_x = a\langle \xi_x, \eta_x \rangle_x + b\langle \zeta_x, \eta_x \rangle_x$, $\langle \xi_x, \zeta_x \rangle_x = \langle \zeta_x, \xi_x \rangle_x$, and $\langle \xi_x, \xi_x \rangle_x \geq 0$ with $\langle \xi_x, \xi_x \rangle_x = 0 \Leftrightarrow \xi_x = 0$.*

The inner product $\langle \cdot, \cdot \rangle_x$ induces a norm,

$$\|\xi_x\|_x := \sqrt{\langle \xi_x, \xi_x \rangle_x},$$

on $\mathrm{T}_x\mathcal{M}$. A Riemannian metric g is defined on each tangent space $\mathrm{T}_x\mathcal{M}$ as a smoothly varying inner product $g_x : \mathrm{T}_x\mathcal{M} \times \mathrm{T}_x\mathcal{M} \to \mathbb{R}$. Consider $\eta_x, \xi_x \in \mathrm{T}_x\mathcal{M}$, we will use interchangeably the notation

$$g_x(\eta_x, \xi_x) = g(\eta_x, \xi_x) = \langle \eta_x, \xi_x \rangle_x = \langle \eta_x, \xi_x \rangle$$

to denote the inner product of two elements η_x and ξ_x on $\mathrm{T}_x\mathcal{M}$. $\xi_x^\flat$ denotes a function $\mathrm{T}_x\mathcal{M} \to \mathbb{R}$ such that $\xi_x^\flat \eta_x = g_x(\xi_x, \eta_x)$ for all $\eta_x \in \mathrm{T}_x\mathcal{M}$. A Riemannian manifold is the combination $(\mathcal{M}, g)$.

The length of a curve on Riemannian manifold is defined by the the inner product as

$$d(x, y) = \inf_\gamma \{ \int_0^1 \sqrt{g_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t))} dt \} = \inf_\gamma \{ \int_0^1 \|\dot{\gamma}(t)\|_{g_{\gamma(t)}} dt \},$$

where γ is a smooth curve on $\mathcal{M}$ with $\gamma(0) = x$ and $\gamma(1) = y$.

2.1.3 Affine Connections, Geodesics, Exponential Mapping and Parallel Translation

In Euclidean space, straight lines are curves γ with zero acceleration that satisfy $\frac{d^2}{dt^2}\gamma(t) = 0$ for all t. On a Riemannian manifold, we already know $\dot{\gamma}(t)$ is defined to show the direction along the curve. A "straight line", called a geodesic on a Riemannian manifold, is a curve $\gamma(t)$ with zero acceleration. We define first an affine connection, which provides the idea of differentiating tangent vectors.

Definition 2.3. *(Affine Connection). Let and $\mathcal{X}(\mathcal{M})$ denote the set of smooth vector fields on $\mathcal{M}$, and $\mathcal{F}_x(\mathcal{M})$ denote the set of all smooth functions on a neighborhood of x. An affine connection ∇ on a manifold $\mathcal{M}$ is a mapping*

$$\nabla : \mathcal{X}(\mathcal{M}) \times \mathcal{X}(\mathcal{M}) \to \mathcal{X}(\mathcal{M}) : (\xi, \eta) \mapsto \nabla_\xi \eta,$$

that satisfies the following properties:

1.$\mathcal{F}(\mathcal{M})$-linearity in the first argument: $\nabla_{f\eta+g\zeta}\xi = f\nabla_\eta\xi + g\nabla_\zeta\xi$;

2.$\mathbb{R}$-linearity in the second argument: $\nabla_\eta(a\xi + b\zeta) = a\nabla_\eta\xi + b\nabla_\eta\zeta$;

3.Product rule (Leibniz's law): $\nabla_\eta(f\xi) = (\eta f)\xi + f\nabla_\eta\xi$,

where $a, b \in \mathbb{R}, \eta, \xi, \zeta \in \mathcal{X}(\mathcal{M})$ and for any $x \in \mathcal{M}, f, g \in \mathcal{F}_x(\mathcal{M})$.

$\nabla_\xi \eta \in \mathrm{T}_x\mathcal{M}$ is a covariant derivative of η with respect to ξ. There is an infinite number of affine connections for a general manifold. The Levi-Civita connection, also called the Riemannian connection is the one satisfies the following conditions:

1. $(\nabla_\eta\xi - \nabla_\xi\eta)f = \eta(\xi f) - \xi(\eta f)$ (symmetry);
2. $\zeta\langle\eta, \xi\rangle = \langle\nabla_\zeta\eta, \xi\rangle + \langle\eta, \nabla_\zeta\xi\rangle$ (compatibility with the Riemannian metric).

Let γ be a curve in $\mathcal{M}$ with domain $I \subseteq \mathbb{R}$. A vector field on the curve γ smoothly assigns to each $t \in I$ a tangent vector to $\mathcal{M}$ at $\gamma(t)$. The set of all smooth vector fields on γ is denoted by $\mathcal{X}(\gamma)$. It can be shown that there is a unique function $\xi \mapsto \frac{D}{dt}\xi$ from $\mathcal{X}(\gamma)$ to $\mathcal{X}(\gamma)$ such that:

1. $\frac{D}{dt}(a\xi + b\zeta) = a\frac{D}{dt}\xi + b\frac{D}{dt}\zeta$ $(a, b \in \mathbb{R})$;
2. $\frac{D}{dt}(f\xi) = f'\xi + f\frac{D}{dt}\xi$ $(f \in \mathcal{F}(I))$;
3. $\frac{D}{dt}(\eta \circ \gamma)(t) = \nabla_{\dot\gamma(t)}\eta$ $(t \in I,\ \eta \in \mathcal{X}(\mathcal{M}))$.

The acceleration vector field $\frac{D^2}{dt^2}\gamma$ on γ is defined by

$$\frac{D^2}{dt^2}\gamma := \frac{D}{dt}\dot\gamma.$$

The geodesic defined by an affine connection is a curve with zero acceleration that satisfies:

$$\nabla_{\dot\gamma(t)}\dot\gamma(t) := \frac{D^2}{dt^2}\gamma(t) := \frac{D}{dt}\dot\gamma(t) = 0.$$

If the connection is chosen to be the Riemannian connection, one of the geodesics between two points on the manifold is also a minimal length curve. This is consistent with the straight line in Euclidean space.

Given a point $x \in \mathcal{M}$ and a tangent vector $\eta_x \in \mathrm{T}_x\mathcal{M}$, there exists a unique geodesic $\gamma(t; x, \eta_x)$ satisfying $\gamma(0) = x$ and $\dot{\gamma}(0) = \eta_x$. Moreover, we have the homogeneity property $\gamma(t; x, a\eta_x) = \gamma(at; x, \eta_x)$. The mapping

$$\mathrm{Exp}_x : \mathrm{T}_x\mathcal{M} \to \mathcal{M} : \eta_x \to \mathrm{Exp}_x\eta_x = \gamma(1; x, \eta_x)$$

is called the exponential mapping at x. In the Riemannian optimization algorithm, such as line search based algorithm, the exponential mapping allows us to move in the tangent space and then map the resulting tangent vector back to the manifold in a neighborhood of x.

A vector field ξ on a curve γ is called parallel if it satisfies $\frac{D}{dt}\xi = 0$. Given $a \in \mathbb{R}$ in the domain of γ and $\xi_{\gamma(a)} \in \mathrm{T}_{\gamma(a)}\mathcal{M}$, there is a unique parallel vector field ξ on γ such that $\xi(a) = \xi_{\gamma(a)}$. The operator $P_\gamma^{b\leftarrow a}$ sending $\xi(a)$ to $\xi(b)$ is called parallel translation along γ. In other words, we have

$$\frac{D}{dt}(P_\gamma^{t\leftarrow a}\xi(a)) = 0.$$

If ∇ is the Riemannian connection, then the parallel translation is an isometry, i.e.,

$$\langle P_\gamma^{t\leftarrow a}\xi(a), P_\gamma^{t\leftarrow a}\zeta(a)\rangle = \langle \xi(a), \zeta(a)\rangle.$$

2.1.4 Riemannian Gradient and Hessian

The definition of the gradient of a function at a point as the direction of the steepest ascent of an objective function is required by gradient-based optimization, meanwhile Newton's method requires second-order information Hessian. The definition in the Riemannian setting is shown below:

Definition 2.4. *(Riemannian gradient). Let f be a function defined on a Riemannian manifold $(\mathcal{M}, g)$. The Riemannian gradient of f at x, denoted as* $\mathrm{grad}f(x)$*, is the unique tangent vector in $\mathrm{T}_x\mathcal{M}$ satisfying*

$$\langle \mathrm{grad}f(x), \eta_x\rangle_x = \mathrm{D}f(x)[\eta_x], \quad \forall \eta_x \in \mathrm{T}_x\mathcal{M},$$

where $\mathrm{D}f(x)[\eta_x]$ is the directional derivative of f at x along η_x.

This is consistent with the Euclidean gradient such that given a function f defined on $\mathbb{R}^n$, the directional derivative along v is

$$\lim_{\epsilon\to 0} \frac{f(x+\epsilon v) - f(x)}{\epsilon} = \mathrm{grad}f(x)^T v = \langle \mathrm{grad}f(x), v\rangle_2.$$

Definition 2.5. *(Riemannian Hessian). Let f be a function defined on a Riemannian manifold $(\mathcal{M}, g)$. The Riemannian Hessian of f at x in the direction of $\eta_x \in \mathrm{T}_x\mathcal{M}$, denoted by* $\mathrm{Hess}f(x)[\eta_x]$*, is defined by*

$$\mathrm{Hess}f(x)[\eta_x] = \nabla_{\eta_x}\mathrm{grad}f(x),$$

where ∇ is the Riemannian connection. The Riemannian Hessian is the linear mapping from $\mathrm{T}_x\mathcal{M}$ to $\mathrm{T}_x\mathcal{M}$.

This is also consistent with the idea in Euclidean case, that is, the derivative of $\mathrm{grad}f(x)$ along direction v is

$$\lim_{\epsilon\to 0}\frac{\mathrm{grad}f(x+\epsilon v) - \mathrm{grad}f(x)}{\epsilon} = \mathrm{Hess}f(x)v.$$

The Hessian is a self-symmetric(adjoint) operator with respect to the Riemannian metric, for all $\eta_x, \xi_x \in \mathrm{T}_x\mathcal{M}$

$$\langle \mathrm{Hess}f(x)[\eta_x], \xi_x\rangle_x = \langle \eta_x, \mathrm{Hess}f(x)[\xi_x]\rangle_x.$$

2.1.5 Retraction and Vector Transport

For a general line search optimization method on Euclidean space, we can use

$$x_+ = x + \alpha d$$

to find the next iteration, with α stepsize and d a descent direction. However, on a Riemannian manifold, since we work on the tangent space to find a tangent vector to define the next iterate on the manifold, we need a method to map the tangent vector back to the manifold. Retraction allows us to move in the direction of a tangent vector while staying on the manifold.

Definition 2.6. *(Retraction). A retraction on a manifold $\mathcal{M}$ is a smooth mapping R from the tangent bundle $\mathrm{T}\mathcal{M}$ onto $\mathcal{M}$ with the following properties. Let R_x denotes the restriction of R to $\mathrm{T}_x\mathcal{M}$.*

1. $R_x(0_x) = x$, where 0_x denotes the zero element of $\mathrm{T}_x\mathcal{M}$;

2. With the canonical identification $\mathrm{T}_{0_x}\mathrm{T}_x\mathcal{M} \simeq \mathrm{T}_x\mathcal{M}$, R_x satisfies

$$\mathrm{D}R_x(0_x) = \mathrm{id}_{\mathrm{T}_x\mathcal{M}},$$

where $\mathrm{id}_{\mathrm{T}_x\mathcal{M}}$ denotes the identity mapping on $\mathrm{T}_x\mathcal{M}$.

The exponential mapping defined earlier is a special case of retraction. When the exponential mapping is used to map a tangent vector back to the manifold, we move along the geodesic defined by the tangent vector. Retraction provides a critical alternative to the exponential mapping which can often be too expensive to define an efficient Riemannian optimization method.

Some Riemannian optimization methods, in particular, quasi-Newton methods, require combining information in different tangent spaces. Parallel translation provides an idea of moving tangent vectors between tangent spaces, but it is often too expensive since it is based on the idea of the exponential mapping. Vector transport provides an alternative to parallel translation, and is built upon retraction.

Definition 2.7. *(Vector Transport). Vector transport on a manifold $\mathcal{M}$ is a smooth mapping*

$$\mathrm{T}\mathcal{M} \oplus \mathrm{T}\mathcal{M} \to \mathrm{T}\mathcal{M} : (\eta_x, \xi_x) \mapsto \mathcal{T}_{\eta_x}(\xi_x) \in \mathrm{T}\mathcal{M}$$

satisfying the following properties for all $x \in \mathcal{M}$.

1. (Associated retraction) There exists a retraction R, called the retraction associated with $\mathcal{T}$, such that the following diagram commutes

$$\begin{array}{ccc} (\eta_x, \xi_x) & \xrightarrow{\quad\mathcal{T}\quad} & \mathcal{T}_{\eta_x}(\xi_x) \\ \downarrow & & \downarrow \pi \\ \eta_x & \xrightarrow[\quad R \quad]{} & \pi(\mathcal{T}_{\eta_x}(\xi_x)) \end{array}$$

where $\pi(\mathcal{T}_{\eta_x}(\xi_x))$ denotes the foot of the tangent vector $\mathcal{T}_{\eta_x}(\xi_x)$;

2. (Consistency) $\mathcal{T}_{0_x}\xi_x = \xi_x$ for all $\xi_x \in \mathrm{T}_x\mathcal{M}$;

3. (Linearity) $\mathcal{T}_{\eta_x}(a\xi_x + b\zeta_x) = a\mathcal{T}_{\eta_x}(\xi_x) + b\mathcal{T}_{\eta_x}(\zeta_x)$.

Note that $\mathrm{T}\mathcal{M} \oplus \mathrm{T}\mathcal{M}$ indicates the Whitney sum, or direct sum of the tangent bundle with itself. combines the tangent spaces of $\mathcal{M}$ at each point with the tangent spaces of $\mathcal{M}$ at the same points. Vector transport is called isometric if it also satisfies

$$\langle \mathcal{T}_{\eta_x}\xi_x, \mathcal{T}_{\eta_x}\zeta_x \rangle_{R_{\eta_x}} = \langle \xi_x, \zeta_x \rangle_x. \tag{2.1}$$

Vector transport by differentiated retraction is a vector transport given by

$$\mathcal{T}_{\eta_x}\xi_x = \mathrm{D}R(\eta_x)[\xi_x] = \frac{d}{dt}R_x(\eta_x + t\xi_x)|_{t=0}.$$

We use $\mathcal{T}_S$ and $\mathcal{T}_R$ to denote an isometric vector transport and a differentiated retraction of R respectively. For a quasi-Newton algorithms on Riemannian manifold, vector transport is crucial in making use of the information from previous iterations and approximating the action of the Hessian. The choices of retraction and vector transport play important roles in the design of efficient Riemannian optimization algorithms.

2.1.6 Coordinate Expressions

Coordinate expressions allow concepts in a vector space to represent concepts on a manifold. In this dissertation, we use the "hat" notation to denote a coordinate expression when analyzing a problem and building the optimization algorithm.

Let $(\mathcal{U}, \varphi)$ be a chart of a manifold $\mathcal{M}$ and $x \in \mathcal{U}$. Suppose the coordinate expression of x, $\hat{x} \in \mathbb{R}^d$, is defined by $\hat{x} = \phi(x)$. And the i-th coordinate vector field of $(\mathcal{U}, \varphi)$, E_i , is defined by

$$(E_i f)(x) := \partial_i (f \circ \varphi^{-1})(\varphi(x)) = D(f \circ \varphi^{-1})(\varphi(x))[e_i].$$

These coordinate vector fields are smooth and every vector field ξ on $\mathcal{U}$ has a decomposition

$$\xi = \sum_i (\xi \varphi_i) E_i,$$

where $(E_i f)_x, i = 1, \cdots, d$ is a basis of $\mathrm{T}_x\mathcal{M}$ and the coordinate expression $\hat{\xi}_x$ of ξ_x with this basis is $(\xi_x \varphi_1, \cdots, \xi_x \varphi_d)$. We can always use QR decomposition to form a smooth orthonormal vector field and thus the coordinate expression of ξ_x can be obtained with the orthonormal basis.

The coordinate expression of the metric at x is $(G_x)_{ij} = \langle E_i, E_j \rangle_x$ which satisfies $g_x(\eta_x, \xi_x) = \hat{\eta}_x^T G_x \hat{\xi}_x$. When the orthonormal vector fields is used, the matrix expression of G_x is a identity and $\|\eta_x\| = \sqrt{g_x(\eta_x, \eta_x)} = \sqrt{\hat{\eta}_x^T \hat{\eta}_x} = \|\hat{\eta}_x\|_2$ is the Euclidean norm. A linear operator $\mathcal{B}$ and a vector transport $\mathcal{T}$ have the matrix form as coordinate expressions $\hat{\mathcal{B}}$ and $\hat{\mathcal{T}}$ respectively, where these are used in our Riemannian algorithm and analysis part.

2.2 Broyden Family of Quasi-Newton Methods in Euclidean Space

A line search strategy chooses a direction and searches along the direction from the current iterate for a new iterate with a lower objective function value. Newton's method has a local quadratic convergence rate, and is defined by the sequence

$$x_{k+1} = x_k - (\mathrm{Hess} f(x))^{-1} \mathrm{grad} f(x).$$

The quasi-Newton methods are motivated by reducing the computational complexity of each step compared to Newton's method and are defined by the iteration of the form

$$x_{k+1} = x_k - B_k^{-1}\text{grad}f(x).$$

The quasi-Newton search direction is an alternative to Newton direction in that it uses Hessian approximation B_k instead of the true Hessian. In each step, a new Hessian approximation B_{k+1} is chosen to mimic certain properties of the true Hessian. By Taylor's Theorem,

$$\text{grad}f(x_{k+1}) = \text{grad}f(x_k) + \text{Hess}f(x)(x_{k+1} - x_k) + O(\|x_{k+1} - x_k\|^2).$$

Ignoring the high order term, we have

$$\text{grad}f(x_{k+1}) - \text{grad}f(x_k) \approx \text{Hess}f(x)(x_{k+1} - x_k).$$

The quasi-Newton update is required to satisfy the so-called secant condition:

$$\text{grad}f(x_{k+1}) - \text{grad}f(x_k) = B_{k+1}(x_{k+1} - x_k).$$

The matrix B_{k+1} that satisfies the secant condition is not unique. By imposing other conditions, different quasi-Newton methods are obtained including the Broyden family of methods. Let $y_k = \text{grad}f(x_{k+1}) - \text{grad}f(x_k)$ and $s_k = x_{k+1} - x_k$. The simplest of these is the symmetric rank-1(SR1) update:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}, \tag{2.2}$$

which is usually combined with the trust region method because the direction is not guaranteed to be a descent direction. The Davidon-Fletcher-Powell (DFP) update $B_{k+1} = (I - \frac{y_k s_k^T}{y_k^T s_k})B_k(I - \frac{s_k y_k^T}{y_k^T s_k}) + \frac{y_k y_k^T}{y_k^T s_k}$ is obtained by imposing the following condition:

$$\min_B \|B - B_k\|_W \quad \text{subject to } B = B^T, Bs_k = y_k,$$

where $\|A\|_W = \|W^{1/2}AW^{1/2}\|_F$ and W satisfies $Wy_k = s_k$. For the inverse Hessian approximation $H_k = B_k^{-1}$, a similar condition

$$\min_H \|H - H_k\|_W \quad \text{subject to } H = H^T, Hy_k = s_k$$

with $\|A\|_W = \|W^{1/2}AW^{1/2}\|_F$ and W satisfies $Ws_k = y_k$ leads to the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update $B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}$. The search direction generated by both

DFP and BFGS updates are guaranteed to be descent if $s_k^T y_k > 0$. The core family of methods discussed in this dissertation, the Broyden family, is defined by taking a combination of BFGS and DFP updates as in the following:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \phi_k (s_k^T B_k s_k) v_k v_k^T, \tag{2.3}$$

where ϕ_k is a scalar, $y_k = \text{grad} f(x_{k+1}) - \text{grad} f(x_k)$, $s_k = x_{k+1} - x_k$, and

$$v_k = \frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k}.$$

Note that BFGS is defined by setting $\phi_k = 0$ and DFP by setting $\phi_k = 1$. The restricted Broyden family with $\phi_k \in [0, 1]$ yields a convex combination of BFGS and DFP, thus preserves the positive definiteness of Hessian approximation. BFGS is the most successful of these restricted Broyden family of methods in terms of efficiency and robustness.

To address a wide range of optimization problems, there is a significant interest in analyzing the entire Broyden family, particularly when exploring the use of negative parameters. Davidon [11] introduced a strategy for choosing ϕ_k in Euclidean space, which, in some cases, can enhance the overall performance of the Broyden family. Zhang and Tewarson [61] provided numerical evidence indicating that Broyden family methods with negative ϕ_k can be more efficient than BFGS in terms of numbers of iterations. Byrd, Liu and Nocedal [8] demonstrated a theoretically optimal value of ϕ_k that requires the knowledge of $\text{Hess} f(x^*)$ at the solution. This value, however, is not practically usable in a practical algorithm. In the meanwhile, the experiments conducted in [8] illustrated specific scenarios where Davidon's update [11] can outperform the BFGS update. These findings serve as motivation for the experiments conducted in Section 4.4 in this dissertation.

2.3 Broyden Family of Limited-memory Quasi-Newton Methods in Euclidean Space

Limited-memory quasi-Newton methods are useful for solving large-scale problems for which the Hessians cannot be computed or stored at a reasonable cost. Instead of storing fully dense approximation matrices, they save only a few vectors that represent the approximation implicitly. The idea of limited-memory BFGS was first described by Nocedal [37], and a limited-memory variant of BFGS (LBFGS) was first proposed by Liu and Nocedal [33]. The BFGS update for the inverse approximation has the form:

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T, \tag{2.4}$$

where $\rho_k = \frac{1}{y_k^T s_k}$ and $V_k = \rho_k y_k s_k^T$. The LBFGS update formula is derived by repeated application of the above inverse BFGS update:

$$\begin{aligned} H_k =& (V_{k-1}^T \cdots V_{k-L}^T) H_k^0 (V_{k-L} \cdots V_{k-1}) \\ &+ \rho_{k-L} (V_{k-1}^T \cdots V_{k-L+1}^T) s_{k-L} s_{k-L}^T (V_{k-L+1} \cdots V_{k-1}) \\ &+ \rho_{k-L+1} (V_{k-1}^T \cdots V_{k-L+2}^T) s_{k-L+1} s_{k-L+1}^T (V_{k-L+2} \cdots V_{k-1}) \\ &+ \cdots + \rho_{k-1} s_{k-1} s_{k-1}^T. \end{aligned} \tag{2.5}$$

The initial Hessian approximation H_k^0 is allowed to vary from iteration to iteration. For each iterate at x_k, LBFGS only stores the set of vector pairs $\{s_i, y_i\}$ for $i = k-L, \cdots, k-1$, so-called curvature pairs. In the Limited-memory Quasi-Newton Methods, the computation of the product $-H_k \text{grad} f_k$ can be produced efficiently without the need to explicitly compute H_k. The implementation of LBFGS has an efficient two-loop structure (a special case for Algorithm 2 in Euclidean space), which ensures its efficiency for the large-scale problems.

Several researchers have focused on exploring other members of the limited-memory Broyden family of methods, with a primary emphasis on how to compute $-H_k \text{grad} f_k$ efficiently. Reed [46] proposed the two-loop recursive limited-memory Broyden family of methods (consider Algorithm 2 in Euclidean space), but the algorithm exhibited certain limitations indicated in Section 3.6. Reed stated that LBFGS with Shanno scaling [55] remains the most efficient and reliable method in the family. The choices of ϕ for the experiments in Reed's paper [46] include the secondary-BFGS update (see Section 3.6 for details), the Hoshino update [21] and the SR1 update (2.2). Byrd, Nocedal and Schnabel [9] presented compact representations of matrices for BFGS and SR1 updates within the Broyden family. However, they did not include any numerical experiments in their work. Erway and Marcia [18] subsequently developed the compact representation for the restricted Broyden family, building upon the work by Byrd, Nocedal and Schnabel [9]. Deguchy et al. [12] further expanded upon Erway and Marcia's findings [18], applying them to the full Broyden family (consider Algorithm 3 and 4 in Euclidean space). The extension is realized by incorporating a permutation matrix which does not produce extra cost if the initial Hessian approximation is chosen as a constant times identity matrix. The authors of [18] and [12] investigated the accuracy of the B_{k+1} generated by the compact representation compared to the B_{k+1} generated by the full matrix updates.

Consider the limited-memory quasi-Newton algorithms discussed above, denoted as LBFGS [33], LBroyden by Reed [46], Compact-representation LBroyden [18, 12]. These algorithms are

Table 2.1: Complexity comparison between Broyden family of limited-memory quasi-Newton methods in Euclidean space.

	LBFGS [33]	LBroyden by Reed [46]	Compact-representation LBroyden [18, 12]
Complexity of $-H_k \text{grad} f_k$	$8Ln + 2n$	$14Ln + 2n$	$16Ln - 6n + \mathcal{O}(L^2)$

utilized to solve an n-dimensional function while maintaining a memory size of L. There is a disparity in complexity of these algorithms arising from their particular choices when defining the computation of $-H_k \text{grad} f_k$. The complexity differences among three algorithms are summarized in Table 2.1. LBFGS stands out as the most computationally efficient algorithm for computing $-H_k \text{grad} f_k$. This efficiency of LBFGS is justified, considering that the other two algorithms pay for the flexibility of selecting alternative members within the Broyden family. The LBroyden by Reed has lower computational costs than the compact-representation LBroyden when $L > 4$, however, it is important to recognize that the LBroyden by Reed comes with certain limitations, as highlighted in Section 3.6. These limitations restrict the practical usage of the LBroyden by Reed, and thus, a careful consideration of these factors is necessary when selecting an appropriate algorithm for a given problem. Limited-memory quasi-Newton algorithms are designed for solving large-scale problems. The memory size L should be maintained at a small value in practice, making it significantly smaller than n.

2.4 A Secant Condition on a Riemannian manifold

Consider a function $f(x)$ defined on a manifold $\mathcal{M}$. Let $x \in \mathcal{M}$, $\mathcal{V}$ be a normal neighborhood of x, and ζ be a C^1 tangent vector field on $\mathcal{M}$. Taylor's Theorem for a vector field on a manifold rather than for $f(x)$ [1] has the following form, that for all $y \in \mathcal{V}$

$$P_{\gamma}^{0\leftarrow 1}\zeta_y = \zeta_x + \nabla_{\xi}\zeta + \int_0^1 (P_{\gamma}^{0\leftarrow \tau}\nabla_{\gamma'(\tau)}\zeta - \nabla_{\xi}\zeta)d\tau,$$

where γ is the unique minimizing geodesic satisfying $\gamma(0) = x$ and $\gamma(1) = y$, and $\xi = \text{Exp}_x^{-1} y = \gamma'(0)$. By Taylor's Theorem,

$$P_{\gamma_k}^{0\leftarrow 1}\text{grad}f(x_{k+1}) = \text{grad}f(x_k) + \nabla_{\xi}\text{grad}f(x_k) + \int_0^1 (P_{\gamma_k}^{0\leftarrow \gamma}\nabla_{\gamma_k'(\tau)}\text{grad}f(x_k) - \nabla_{\xi}\text{grad}f(x_k))d\tau,$$

where γ_k is the unique minimizing geodesic satisfying $\gamma_k(0) = x_k$ and $\gamma_k(1) = x_{k+1}$, and $\xi = \text{Exp}_{x_k}^{-1} x_{k+1} = \gamma_k'(0)$. Ignoring the integral reminder term, we have

$$P_{\gamma_k}^{0\leftarrow 1}\text{grad}f(x_{k+1}) - \text{grad}f(x_k) \approx \nabla_{\xi}\text{grad}f(x_k) = \text{Hess}f(x_k)\text{Exp}_{x_k}^{-1} x_{k+1}.$$

The above is defined on $\mathrm{T}_{x_k}\mathcal{M}$, the desired Hessian approximation $\mathcal{B}_{k+1}$ defined on $\mathrm{T}_{x_{k+1}}\mathcal{M}$ by applying parallel translation, yields the following Riemannian secant condition:

$$\mathrm{grad}f(x_{k+1}) - P_{\gamma_k}^{1\leftarrow 0}\mathrm{grad}f(x_k) = \mathcal{B}_{k+1}(P_{\gamma_k}^{1\leftarrow 0}\mathrm{Exp}_{x_k}^{-1}x_{k+1}). \tag{2.6}$$

Let s_k denote $P_{\gamma_k}^{1\leftarrow 0}\mathrm{Exp}_{x_k}^{-1}x_{k+1}$ and y_k denote $\mathrm{grad}f(x_{k+1}) - P_{\gamma_k}^{1\leftarrow 0}\mathrm{grad}f(x_k)$. We require $\mathcal{B}_{k+1}$ to be self-adjoint with respect to the Riemannian metric instead of symmetry.

All of the quasi-Newton update formulas below are based on the Riemmanian secant condition (2.6). SR-1 update is generalized to a Riemannian manifold as:

$$\mathcal{B}_{k+1} = \tilde{\mathcal{B}}_k + \frac{(y_k - \tilde{\mathcal{B}}_k s_k)(y_k - \tilde{\mathcal{B}}_k s_k)^\flat}{g(s_k, y_k - \tilde{\mathcal{B}}_k s_k)} \tag{2.7}$$

with $\tilde{\mathcal{B}}_k = P_{\gamma_k}^{1\leftarrow 0}\mathcal{B}_k P_{\gamma_k}^{0\leftarrow 1}$. DFP and BFGS updates are generalized on a Riemannian manifold as the following:

$$\mathrm{DFP}: \ \mathcal{B}_{k+1} = (\mathrm{id} - \frac{y_k s_k^\flat}{y_k^\flat s_k})\tilde{\mathcal{B}}_k(\mathrm{id} - \frac{s_k y_k^\flat}{y_k^\flat s_k}) + \frac{y_k y_k^\flat}{y_k^\flat s_k}, \tag{2.8}$$

$$\mathrm{BFGS}: \ \mathcal{B}_{k+1} = \tilde{\mathcal{B}}_k - \frac{\tilde{\mathcal{B}}_k s_k (\tilde{\mathcal{B}}_k s_k)^\flat}{s_k^\flat \tilde{\mathcal{B}}_k s_k} + \frac{y_k y_k^\flat}{y_k^\flat s_k}. \tag{2.9}$$

As in the Euclidean space, the Broyden family on a Riemannian manifold is defined by taking a combination of the Riemmanian DFP and Riemmanian BFGS operators. The formula for the RBroyden family is given in the next chapter (3.1), and is discussed thoroughly throughout the dissertation. Rather than the explicit exponential map and parallel translation in the secant condition (2.6), alternate forms based on retraction and vector transport are used in practice.

CHAPTER 3

RIEMANNIAN BROYDEN FAMILY OF LIMITED-MEMORY QUASI-NEWTON METHODS

This chapter serves as an introduction to the LRBroyden family of methods and provides a comprehensive development of their convergence analysis. It lays the foundation for understanding and applying these methods in the context of optimization problems.

3.1 RBroyden Family of Methods

On a Riemannian manifold, suppose the stepsize and search direction at iterate $x_k \to x_{k+1}$ are α_k and η_k, then $x_{k+1} = R_{x_k}(\alpha_k \eta_k)$ for a line search optimization algorithm. Let s_k denote $\mathcal{T}_{S_{\alpha_k \eta_k}} \alpha_k \eta_k$, and y_k denote $\text{grad} f(x_{k+1})/\beta_k - \mathcal{T}_{S_{\alpha_k \eta_k}} \text{grad} f(x_k)$ with $\beta_k = \frac{||\alpha_k \eta_k||}{||\mathcal{T}_{R_{\alpha_k \eta_k}} \alpha_k \eta_k||}$. This allows for a convergence analysis under more general assumptions than those of Qi [41], where $\beta_k = 1$ but requiring the retraction to be the exponential mapping and vector transport to be the parallel transport. $\mathcal{T}_S$ is the isometric vector transport with R as associated retraction and $\mathcal{T}_R$ is the differentiated retraction of R. The RBroyden family [22, 27] has the updating formula as follows:

$$\mathcal{B}_{k+1} = \tilde{\mathcal{B}}_k - \frac{\tilde{\mathcal{B}}_k s_k (\tilde{\mathcal{B}}_k^* s_k)^\flat}{(\tilde{\mathcal{B}}_k^* s_k)^\flat s_k} + \frac{y_k y_k^\flat}{y_k^\flat s_k} + \phi_k g(s_k, \tilde{\mathcal{B}}_k s_k) v_k v_k^\flat, \tag{3.1}$$

where $\tilde{\mathcal{B}}_k = \mathcal{T}_{S_{\alpha_k \eta_k}} \circ \mathcal{B}_k \circ \mathcal{T}^{-1}_{S_{\alpha_k \eta_k}}$, $v_k = \frac{y_k}{g(y_k, s_k)} - \frac{\tilde{\mathcal{B}}_k s_k}{g(s_k, \tilde{\mathcal{B}}_k s_k)}$ and $\tilde{\mathcal{B}}_k^*$ denotes the adjoint with respect to the Riemannian metric g.

The non-isometric vector transport can be used but it is not provably convergent in general. An isometric vector transport preserves the distance on the two tangent spaces, i.e., $g_{R(\eta_x)}(\mathcal{T}_{\eta_x} \xi_x, \mathcal{T}_{\eta_x} \zeta_x) = g_x(\xi_x, \zeta_x)$. It guarantees that $\tilde{\mathcal{B}}_k$ is self-adjoint if $\mathcal{B}_k$ is self-adjoint. The isometric vector transport $\mathcal{T}_S$ is not necessarily smooth, but required to be in C^0. And for any $\bar{x} \in \mathcal{M}$, there exists a neighborhood $\mathcal{U}$ of $\bar{x}$ and a constant c_0 such that for all $x, y \in \mathcal{U}$

$$||\mathcal{T}_{S_\eta} - \mathcal{T}_{R_\eta}|| \leq c_0 ||\eta||, \tag{3.2}$$

$$||\mathcal{T}^{-1}_{S_\eta} - \mathcal{T}^{-1}_{R_\eta}|| \leq c_0 ||\eta||, \tag{3.3}$$

where $\eta = R_x^{-1}(y)$ and $\mathcal{T}_R$ denotes the differentiated retraction. Besides the isometry condition shown above, the locking condition [22, 27] is also required:

$$\mathcal{T}_{S_\xi}\xi = \beta \mathcal{T}_{R_\xi}\xi, \quad \beta = \frac{||\xi||}{||\mathcal{T}_{R_\xi}\xi||} \tag{3.4}$$

for all $\xi \in \mathrm{T}_x\mathcal{M}$ and all $x \in \mathcal{M}$. This locking condition and isometry condition are imposed to guarantee the curvature condition $g(s_k, y_k) > 0$ so that the Hessian approximation is positive definite since, unlike in Euclidean space, the second Wolfe condition (3.6) alone cannot guarantee $g(s_k, y_k) > 0$. These two conditions are also necessary for the proposed the LRBroyden family of methods.

This locking condition brings us a derivation of potentially efficient algorithm and facilitates the convergence analysis. Exponential mapping and parallel transport satisfy the locking condition with $\beta = 1$, however, their form is often unknown or computationally expensive. Methods to specify a retraction and an isometric vector transport that satisfy the locking condition have been proposed in [27]: (i) Given a retraction and an isometric vector transport, we can modify the given vector transport so that it satisfies the locking condition. (ii) Given a retraction and bases of tangent space, we can construct directly an isometric vector transport that satisfies the locking condition. (iii) Given a transporter [42], we can construct a retraction and its differentiated retraction that satisfy the locking condition.

The RBroyden family, Algorithm 1, proposed in [27] is presented here to facilitate the convergence proof. The global and q-superlinear convergence of the RBroyden family of methods were shown in [22] when applied to minimize a retraction-convex objective function. To ensure global convergence when applied to nonconvex optimization problems, a cautious update method as outlined in [25] is imposed as follows in (3.7). This update is designed to enhance the algorithm's stability and robustness when dealing with challenging optimization tasks.

$$\mathcal{B}_{k+1} = \begin{cases} \tilde{\mathcal{B}}_k - \dfrac{\tilde{\mathcal{B}}_k s_k (\tilde{\mathcal{B}}_k^* s_k)^\flat}{(\tilde{\mathcal{B}}_k{}^* s_k)^\flat s_k} + \dfrac{y_k y_k^\flat}{y_k^\flat s_k} + \phi_k g(s_k, \tilde{\mathcal{B}}_k s_k) v_k v_k^\flat, & \text{if } \dfrac{y_k^\flat s_k}{\|s_k\|^2} \geq \vartheta(\|\mathrm{grad} f(x_k)\|) \\ \tilde{\mathcal{B}}_k, & \text{otherwise,} \end{cases} \tag{3.7}$$

where ϑ is a monotone increasing function satisfying $\vartheta(0) = 0$ and ϑ is strictly increasing at 0.

3.2 Two-loop Recursive LRBroyden Family of Methods

The RBroyden family relies on the explicit representations of the operators $\mathcal{B}_k$, $\tilde{\mathcal{B}}_k$, $\mathcal{T}_{S_{\alpha_k \eta_k}}$ and $\mathcal{T}^{-1}_{S_{\alpha_k \eta_k}}$. However, for large-scale problems, obtaining these explicit representations can be

Algorithm 1 RBroyden family method proposed in [27]

Require: Riemannian manifold $\mathcal{M}$ with Riemannian metric g; a retraction R; isometric vector transport $\mathcal{T}_S$, with R as associated retraction that satisfies the locking condition; continuously differentiable real-valued function f on $\mathcal{M}$; initial iterate $x_0 \in \mathcal{M}$; initial Hessian approximation $\mathcal{B}_0$ which is a linear transformation of the tangent space $\mathrm{T}_{x_0}\mathcal{M}$ that is symmetric positive definite with respect to the metric g; convergence tolerance ε; Wolfe condition constants $0 < c_1 < \frac{1}{2} < c_2 < 1$;

1: $k = 0$;

2: **while** $\|\mathrm{grad} f(x_k)\| > \varepsilon$ **do**

3: Obtain $\eta_k \in \mathrm{T}_{x_k}\mathcal{M}$ by solving $\mathcal{B}_k\eta_k = -\mathrm{grad} f(x_k)$;

4: Set $x_{k+1} = R_{x_k}(\alpha_k\eta_k)$. where $\alpha_k > 0$ is computed from a line search procedure to satisfy the Wolfe conditions

$$f(x_{k+1}) \leq f(x_k) + c_1\alpha_k g(\mathrm{grad} f(x_k), \eta_k), \tag{3.5}$$

$$\frac{d}{dt} f(R(t\eta_k))|_{t=\alpha_k} \geq c_2 \frac{d}{dt} f(R(t\eta_k))|_{t=0}; \tag{3.6}$$

5: Define $s_k = \mathcal{T}_{S_{\alpha_k\eta_k}}\alpha_k\eta_k$, $y_k = \mathrm{grad} f(x_{k+1})/\beta_k - \mathcal{T}_{S_{\alpha_k\eta_k}}\mathrm{grad} f(x_k)$, where $\beta_k = \frac{||\alpha_k\eta_k||}{||\mathcal{T}_{R_{\alpha_k\eta_k}}\alpha_k\eta_k||}$ and $\mathcal{T}_R$ is the differentiated retraction of R;

6: Define the linear operator $\mathcal{B}_{k+1} : \mathrm{T}_{x_{k+1}}\mathcal{M} \to \mathrm{T}_{x_{k+1}}\mathcal{M}$ by Equation (3.1) with $\phi_k \in (\phi_k^c, \infty)$, $\phi_k^c = 1/(1-\mu_k), \mu_k = (g(y_k, \tilde{\mathcal{B}}_k^{-1}y_k)g(s_k, \tilde{\mathcal{B}}_k s_k))/g(y_k, s_k)^2$;

7: $k = k + 1$;

8: **end while**

challenging or computationally expensive. LRBroyden addresses this problem by storing only some of the most recent s_k and y_k vectors, rather than the entire Hessian approximation. It constructs the Hessian approximation using curvature information, which leads to an acceptable rate of convergence and makes it well-suited for many practical applications. This approach helps alleviate the computational burden associated with explicitly representing these operators in large-scale problems.

Generalizing the limited-memory full Broyden family methods to Riemannian manifolds is one of the main contributions in this dissertaion. In this section, we introduce the two-loop recursive LRBroyden, extending the results obtained in Euclidean space by Reed in [46]. This algorithm incorporates the adaptation of Nocedal's two-loop recursive formula, which was originally developed for LBFGS [37].

Consider $\mathcal{H}_k$, the inverse Hessian approximation update formula:

$$\mathcal{H}_{k+1} = \tilde{\mathcal{H}}_k - \frac{\tilde{\mathcal{H}}_k y_k (\tilde{\mathcal{H}}_k^* y_k)^\flat}{(\tilde{\mathcal{H}}_k^* y_k)^\flat y_k} + \frac{s_k s_k^\flat}{y_k^\flat s_k} + \Phi_k g(y_k, \tilde{\mathcal{H}}_k s_k)\omega_k \omega_k^\flat, \tag{3.8}$$

where $\Phi_k = \frac{(1-\phi_k)g(y_k,s_k)^2}{(1-\phi_k)g(y_k,s_k)^2+\phi_k g(y_k,\tilde{\mathcal{H}}_k y_k)g(s_k,\tilde{\mathcal{B}}_k s_k)}$, $\omega_k = \frac{s_k}{g(y_k,s_k)} - \frac{\tilde{\mathcal{H}}_k y_k}{g(y_k,\tilde{\mathcal{H}}_k y_k)}$ and $\tilde{\mathcal{H}}_k = \mathcal{T}_{S_{\alpha_k\eta_k}} \circ \mathcal{H}_k \circ \mathcal{T}^{-1}_{S_{\alpha_k\eta_k}}$. Note that $\Phi_k = 0$ when $\phi_k = 1$ and $\Phi_k = 1$ when $\phi_k = 0$. We generalize the inverse update formula to Riemannian setting as follows:

$$\mathcal{H}_{k+1} = \mathcal{V}_k^\flat \tilde{\mathcal{H}}_k \mathcal{V}_k + \rho_k s_k s_k^\flat, \tag{3.9}$$

where

$$\rho_k = \frac{1}{g(y_k, s_k)} \text{ and } \mathcal{V}_k = \mathrm{id} - y_k (A s_k + B \tilde{\mathcal{H}}_k y_k)^\flat.$$

$A = \pm\sqrt{\Phi_k}\rho_k$ and $B = (1 \mp \sqrt{\Phi_k})/g(y_k, \tilde{\mathcal{H}}_k y_k)$ are derived by expansion and coefficients comparison with Equation (3.8). Note that BFGS update has $\Phi_k = 1$, $A = \rho_k$ and $B = 0$. If the L most recent s_k and y_k are stored then we have

$$\begin{aligned}
\mathcal{H}_k =& \tilde{\mathcal{V}}_{k-1}^\flat \tilde{\mathcal{V}}_{k-2}^\flat \cdots \tilde{\mathcal{V}}_{k-L}^\flat \tilde{\mathcal{H}}_k^0 \tilde{\mathcal{V}}_{k-L} \cdots \tilde{\mathcal{V}}_{k-2} \tilde{\mathcal{V}}_{k-1} \\
&+ \rho_{k-L} \tilde{\mathcal{V}}_{k-1}^\flat \tilde{\mathcal{V}}_{k-2}^\flat \cdots \tilde{\mathcal{V}}_{k-L+1}^\flat s_{k-L}^{(k)} (s_{k-L}^{(k)})^\flat \tilde{\mathcal{V}}_{k-L+1} \cdots \tilde{\mathcal{V}}_{k-2} \tilde{\mathcal{V}}_{k-1}, \\
&+ \cdots + \rho_{k-1} s_{k-1}^{(k)} (s_{k-1}^{(k)})^\flat,
\end{aligned}$$

where $\tilde{\mathcal{V}}_i = \mathrm{id} - y_i^{(k)} (A_i s_i^{(k)} + B_i \tilde{\mathcal{H}}_i y_i^{(k)})^\flat$ for $i = k-L, \cdots, k-1$ with $A_i = \pm\sqrt{\Phi_i}\rho_i$ and $B_i = (1 \mp \sqrt{\Phi_i})/g(y_i^{(k)}, \tilde{\mathcal{H}}_i y_i^{(k)})$. Note that the vectors $s_i^{(k)}, y_i^{(k)}$ requires vector transports, which yield the

Step 28 of Algorithm 2. Note that the initial inverse Hessian approximation $\tilde{\mathcal{H}}_k^0$ is not necessarily $\tilde{\mathcal{H}}_{k-L}$. It can be any positive definite self-adjoint operator, and is usually set as

$$\mathcal{H}_k^0 = \gamma_k \mathrm{id},$$

where $\gamma_k > 0$ is a constant. The update formulae are valid for $\Phi_k \geq 0$, hence not all family members are included. The algorithm generalized for Riemannian manifolds is shown in Algorithm 2.

In the limited-memory algorithms (both Algorithm 2 and 4) with memory size L, the index $k = 0, 1, \cdots$ represents the iteration step. At step k, denote $l = \max\{k - L, 0\}$, the index $i = l, l + 1, \cdots, k - 1$ represents the local components during the updates of inverse Hessian approximation. $s_i^{(k)}$, $y_i^{(k)}$, $u_i^{(k)}$, $\phi_i^{(k)}$ and $\Phi_i^{(k)}$ are used for updating the inverse Hessian approximations $\mathcal{H}_k^i \to \mathcal{H}_k^{i+1}$, where $\mathcal{H}_k^l$ is denoted as the initial inverse Hessian approximation that $\mathcal{H}_k^0 \equiv \mathcal{H}_k^l \to \mathcal{H}_k^{l+1} \to \cdots \to \mathcal{H}_k^k$. The inverse Hessian approximations $\mathcal{H}_k^i$ for $i = l, l + 1, \cdots, k$ defined on the tangent space of x_k are not explicitly expressed. In Algorithm 2, the notation Φ_k is employed instead of $\Phi_i^{(k)}$ to accommodate for the shifting and reuse of values, as discussed in Section 3.5.

Algorithm 2 exhibits three primary drawbacks. Firstly, the two-loop recursive process is computationally efficient only for LRBFGS. LRBFGS (proposed by Huang et al. [27]) is a specialized case of Algorithm 2 with $A_i = \rho_i,\ B_i = 0$, where $u_i^{(k)}$ is not needed. For other members that can be applied to Algorithm 2, the computation and storage regarding to $u_i^{(k)}$ leads to the extra cost. In addition to the complexity outlined in Table 2.1 for Algorithm 2 in Euclidean space, it is noteworthy to consider the additional cost associated with storing and transporting $u_i^{(k)}$.

Secondly, Algorithm 2 solely supports the LRBroyden family of methods by storing and shifting the values of $\phi_i^{(k)}$ and $\Phi_i^{(k)}$, as indicated in the first approach in Section 3.5. These shifted $\phi_i^{(k)}$ and $\Phi_i^{(k)}$, however, may not satisfy the convergence requirement (3.14). It is not applicable to apply this algorithm if equipping other strategy of choosing $\phi_i^{(k)}$ or $\Phi_i^{(k)}$.

Thirdly, each Φ_k corresponds to two updates in the Broyden family, except when $\Phi_k = 0$ (DFP). Even in the case of $\Phi_k = 1$, other than $A_i = \rho_i,\ B_i = 0$ for LRBFGS, there exists a secondary update with $A_i = -\rho_i,\ B_i = 2/g(y_i^{(k)}, u_i^{(k)})$ which did not perform well, as demonstrated in Reed's findings [46]. Indeed, Reed concluded that the standard LBFGS with $A = \rho_k$ and $B = 0$ stood out as the most efficient and effective method among the other applicable members of Algorithm 2. Due to the drawbacks of the algorithm mentioned above, this conclusion prompts further exploration of the LRBroyden family of methods by allowing for a wider range of ϕ_k values.

Algorithm 2 Two-loop Recursive LRBroyden by generalization of [46]

Require: Riemannian manifold $\mathcal{M}$ with Riemannian metric g; a retraction R; isometric vector transport $\mathcal{T}_S$; Smooth function f on $\mathcal{M}$; initial iterate $x_0 \in \mathcal{M}$; an integer $L > 0$;

$k = 0, \varepsilon > 0, 0 < c_1 < \frac{1}{2} < c_2 < 1, \gamma_0 = 1, l = 0$;

$\mathcal{H}_k^0 = \gamma_k \mathrm{id}$. Obtain $\eta_k \in T_{x_k}\mathcal{M}$ by the following algorithm:

Choose $A_i = \sqrt{\Phi_i}\rho_i$, $B_i = (1 - \sqrt{\Phi_i})/g(y_i^{(k)}, u_i^{(k)})$ or choose $A_i = -\sqrt{\Phi_i}\rho_i$, $B_i = (1 + \sqrt{\Phi_i})/g(y_i^{(k)}, u_i^{(k)})$;

$q \leftarrow y_{k-1}^{(k)}$;

for $i = k-2, k-3, \cdots, l$ **do**

  $\xi_i = g(s_i^{(k)}, q)$;

  $q \leftarrow q - (B_i g(u_i^{(k)}, q) + A_i \xi_i) y_i^{(k)}$;

end for

$r \leftarrow \mathcal{H}_k^0 q$;

for $i = l, l+1, \cdots, k-2$ **do**

  $r \leftarrow r - B_i g(y_i^{(k)}, r) u_i^{(k)} + (\rho_i \xi_i - A_i g(y_i^{(k)}, r)) s_i^{(k)}$;

end for

Set $u_{k-1}^{(k)} = r$;

$q \leftarrow \mathrm{grad} f(x_k)$;

for $i = k-1, k-2, \cdots, l$ **do**

  $\xi_i = g(s_i^{(k)}, q)$;

  $q \leftarrow q - (B_i g(u_i^{(k)}, q) + A_i \xi_i) y_i^{(k)}$;

end for

$r \leftarrow \mathcal{H}_k^0 q$;

for $i = l, l+1, \cdots, k-1$ **do**

  $r \leftarrow r - B_i g(y_i^{(k)}, r) u_i^{(k)} + (\rho_i \xi_i - A_i g(y_i^{(k)}, r)) s_i^{(k)}$;

end for

Set $\eta_k = -r$;

Find α_k that satisfies Wolfe conditions;

Set $x_{k+1} = R_{x_k}(\alpha_k \eta_k)$. If $||\mathrm{grad} f(x_{k+1})|| > \varepsilon$, then break;

Define $s_k^{(k+1)} = \mathcal{T}_{S_{\alpha_k \eta_k}} \alpha_k \eta_k$ and $y_k^{(k+1)} = \mathrm{grad} f(x_{k+1})/\beta_k - \mathcal{T}_{S_{\alpha_k \eta_k}} \mathrm{grad} f(x_k)$, where $\beta_k = \frac{||\alpha_k \eta_k||}{||\mathcal{T}_{R_{\alpha_k \eta_k}} \alpha_k \eta_k||}$. And calculate $\rho_k = 1/g(s_k^{(k+1)}, y_k^{(k+1)})$, $\gamma_{k+1} = g(s_k^{(k+1)}, y_k^{(k+1)})/||y_k^{(k+1)}||^2$;

Set or compute Φ_k;

Let $l = \max\{k - L + 1, 0\}$. Add $s_k^{(k+1)}, y_k^{(k+1)}, \rho_k$ and Φ_k into storage and if $k > L$, then discard vector pair $\{s_{l-1}^{(k)}, y_{l-1}^{(k)}, u_{l-1}^{(k)}\}$ and scalars ρ_{l-1}, Φ_{l-1} from storage; Transport $s_l^{(k)}, s_{l+1}^{(k)}, \cdots, s_{k-1}^{(k)}, y_l^{(k)}, y_{l+1}^{(k)}, \cdots, y_{k-1}^{(k)}$ and $u_l^{(k)}, u_{l+1}^{(k)}, \cdots, u_{k-1}^{(k)}$ from $\mathrm{T}_{x_k}\mathcal{M}$ to $\mathrm{T}_{x_{k+1}}\mathcal{M}$ by $\mathcal{T}_S$, then get $s_l^{(k+1)}, s_{l+1}^{(k+1)}, \cdots, s_{k-1}^{(k+1)}, y_l^{(k+1)}, y_{l+1}^{(k+1)}, \cdots, y_{k-1}^{(k+1)}$ and $u_l^{(k+1)}, u_{l+1}^{(k+1)}, \cdots, u_{k-1}^{(k+1)}$;

Update $u_i^{(k+1)} = \frac{\gamma_{k+1}}{\gamma_k} u_i^{(k+1)}$ for $i = l, l+1, \cdots, k-1$;

$k = k + 1$, go to 2;

3.3 Implementation Techniques

This section introduces an effective technique that can be used for LRBroyden. A d-dimensional Riemannian manifold $\mathcal{M}$ often has elements that can be represented by a vector in $\mathbb{R}^n$. There are some common situations encountered in practice:

1. $\mathcal{M}$ is embedded in $\mathbb{R}^n$ and inherits its metric from a metric on $\mathbb{R}^n$.

2. $\mathcal{M}$ is a subset of $\mathbb{R}^n$ with a metric g_x on $\mathrm{T}_x\mathcal{M}$ that is not necessarily a restriction of a metric on $\mathbb{R}^n$ nor can it necessarily be extended to be a metric on all of $\mathbb{R}^n$.

3. $\mathcal{M}$ is a quotient of a manifold $\bar{\mathcal{M}}$(can be either of the first two types).

4. $\mathcal{M}$ is a product of two or more manifolds(each of which is any of the first three types).

With a choice of basis of $\mathrm{T}_x\mathcal{M}$, one can implement an n-dimensional or a d-dimensional vector to represent a tangent vector in these four cases. The d-dimensional is also called the intrinsic representation [24] of tangent vectors and vector transports on matrix manifold. This intrinsic representation and vector transport by parallelization provide computational benefits for the proposed algorithm.

Given a d-dimensional Riemannian manifold $\mathcal{M}$ with the Riemannian metric $g : (\eta_x, \xi_x) \mapsto g_x(\eta_x, \xi_x) \in \mathbb{R}$, where $\eta_x, \xi_x \in \mathrm{T}_x\mathcal{M}$. Consider two situations: (i) $\mathcal{M}$ is an embedded submanifold [1, section 3.3] of a n-dimensional Euclidean space; (ii) $\mathcal{M}$ is a quotient manifold [1, section 3.4] $\bar{\mathcal{M}}/\mathcal{G} = \{[x]|x \in \bar{\mathcal{M}}\}$, where $\bar{\mathcal{M}}$ is a submanifold of a n-dimensional Euclidean space, $\mathcal{G}$ is a group acting on $\bar{\mathcal{M}}$ and $[x] = \{gx|g \in \mathcal{G}\}$ The tangent space $\mathrm{T}_x[x]$ is called the vertical space $\mathcal{V}_x$ at x. The horizontal space is defined to be the perpendicular space of $\mathcal{V}_x$, i.e., $\mathcal{H}_x \oplus \mathcal{V}_x = \mathrm{T}_x\bar{\mathcal{M}}$. For any tangent vector $\eta_{[x]} \in \mathrm{T}_{[x]}\mathcal{M}$, the unique representation in $\mathcal{H}_x$ is called the horizontal lift of $\eta_{[x]}$ at x, denoted by $\eta_{\uparrow_x}$.

Suppose $\mathcal{L}(x, y)$ is a linear operator from $\mathrm{T}_x\mathcal{M}$ to $\mathrm{T}_y\mathcal{M}$ whose dependence on x and y is jointly smooth and such that $\mathcal{L}(x, x)$ is identity for all x, and we are given a retraction R. It has been shown in [27] that a vector transport $\mathcal{T}$ with associated retraction R can be defined as

$$\mathcal{T}_{\xi_x}\eta_x = \mathcal{L}(x, R_x(\xi_x))\eta_x.$$

If $\mathcal{M}$ is an embedded submanifold in case (i), The transporter by parallelization is defined by

$$\mathcal{L}^{\mathrm{Pl}}(x, y)\eta_x = B_y B_x^{\dagger}\eta_x, \tag{3.10}$$

where $B : \mathcal{V} \to \mathbb{R}^{n\times d} : z \mapsto B_z$ is a smooth tangent basis field defined on an open set $\mathcal{V}$ of $\mathcal{M}$, and $B^\dagger$ denotes the pseudo-inverse of B. $\mathcal{L}(x, y)$ is isometric if B_z forms an orthonormal basis of $\mathrm{T}_z\mathcal{M}$, which implies that the vector transport by parallelization $\mathcal{T}_{\xi_x}\eta_x = B_{R_x(\xi_x)}B_x^\dagger\eta_x$ is isometric.

If $\mathcal{M}$ is a quotient manifold in case (ii), the transporter by parallelization can be defined as

$$\mathcal{L}^{\mathrm{Pl}}(x, y)\eta_x = B_y^h(B_x^h)^\dagger\eta_x, \tag{3.11}$$

where the columns of B_z^h form an orthonormal basis of the horizontal space $\mathcal{H}_z$. This defines the transporters by parallelization for the quotient manifold if (3.11) is independent of the representation chosen in $[x]$ and $[y]$.

Throughout this chapter, the d-dimensional and n-dimensional representations of a tangent vector are called the intrinsic and extrinsic representations respectively. The functions that denote the maps converting from one representation to the other representation are shown as:

$$\mathrm{D2E}_x^{\mathcal{M}} : v_x \mapsto \eta_x = B_x v_x \qquad \text{and} \qquad \mathrm{E2D}_x^{\mathcal{M}} : \eta_x \mapsto v_x = B_x^\dagger\eta_x,$$

where v_x and η_x are intrinsic and extrinsic representation respectively. The intrinsic representation of the transporter by parallelization has the form

$$\mathrm{E2D}_x^{\mathcal{M}} \circ \mathcal{L}^{Pl}(x, y) \circ \mathrm{D2E}_x^{\mathcal{M}} v_x = B_y^\dagger(B_y B_x^\dagger(B_x v_x)) = v_x$$

and is therefore the identity. If the columns of B_x form an orthonormal basis of $\mathrm{T}_x\mathcal{M}$, the transporter is isometric, and the Riemannian metric reduces to the Euclidean metric for the intrinsic representations:

$$g(\eta_x, \xi_x) = g(B_x v_x, B_x u_x) = v_x^T u_x.$$

Many manifolds have been confirmed to have a computationally inexpensive way to transition between v_x and η_x, such as the Stiefel manifold, the Grassmann manifold and the fixed-rank manifold (see [24]), all which are included in the experiments later in this dissertation.

By employing the intrinsic representation, time and spatial complexity can be reduced by manipulating smaller dimensional vectors in LRBroyden. The following section illustrates Algorithm 4 with intrinsic representation. In cases where the intrinsic representation is not available for a given manifold, the algorithm can also be stated with the extrinsic representation, as indicated in Algorithm 2.

3.4 Full LRBroyden Family of Methods

In this section, we introduce the Riemannian Broyden family of limited-memory quasi-Newton algorithm by generalizing the work by Deguchy et al. [12]. The updating formula of the Hessian approximation B_k can be rewritten from Equation (2.3) as:

$$B_{k+1} = B_k + (B_k s_k \;\; y_k) \begin{pmatrix} -\frac{(1-\phi_k)}{s_k^T B_k s_k} & -\frac{\phi_k}{y_k^T s_k} \\ -\frac{\phi_k}{y_k^T s_k} & (1+\phi_k \frac{s_k^T B_k s_k}{y_k^T s_k}) \frac{1}{y_k^T s_k} \end{pmatrix} (B_k s_k \;\; y_k)^T .$$

In the compact representation, the updating formula can be expressed as

$$B_{k+1} = B_0 + \hat{\Psi}_k \hat{M}_k \hat{\Psi}_k^T,$$

where B_0 is the initial guess, $\hat{\Psi}_k \in \mathbb{R}^{n\times p}$ is formulated in terms of the curvature information, $\hat{M}_k \in \mathbb{R}^{p\times p}$ is obtained recursively and their expressions can be found in Algorithm 3. The update of Broyden class can be either rank-one with $p = k+1$ for SR1 update or rank-two with $p = 2(k+1)$ for other members. The inverses of these matrices, denoted H_k, possess a similar expression. All these formula details are incorporated into the following algorithm and the proof by induction can be found in [12].

The generalization LBroyden is shown in Algorithm 3 and Algorithm 4. Algorithm 3 oulines the compact representation of the L (memory size) Broyden updates from $\mathcal{B}_k^0$ and $\mathcal{H}_k^0$ to $\mathcal{B}_k^k$ and $\mathcal{H}_k^k$, where $\mathcal{B}_k^0 \equiv \mathcal{B}_k^l \to \mathcal{B}_k^{l+1} \to \cdots \to \mathcal{B}_k^k$ and $\mathcal{H}_k^0 \equiv \mathcal{H}_k^l \to \mathcal{H}_k^{l+1} \to \cdots \to \mathcal{H}_k^k$ with $l = \max\{k-L, 0\}$. Denote $\mathcal{B}_k^i$ and $\mathcal{H}_k^i$ as $\mathcal{B}_i$ and $\mathcal{H}_i$ for clarity in Algorithm 3. The outputs $\tilde{M}_{k-1}$ and $\tilde{\Psi}_{k-1}$ are used for computing the search direction in Step 5 and Step 6 in Algorithm 4. Algorithm 4 presents the loop for $k = 0, 1, \cdots$, which is based on the intrinsic representation as described in [24]. For manifolds that do not have an acceptable intrinsic representation, it is possible to obtain the extrinsic version by employing similar expressions as in Algorithm 2. The intrinsic representation employed in Algorithm 4 allows for efficient implementation, and the identity implementation of the transporter through parallelization enhances computational efficiency. The vectors s_k, y_k, η_k^d are all d-dimensional vectors and η_k represents the extrinsic representation of η_k^d. The transformation between η_k and η_k^d relies on the orthonormal basis, and more details can be found in [24].

The LRBroyden family of methods, through generalizing the approach in [12], allows for different $\phi_i^{(k)}$ at each step and $\phi_i^{(k)}$ can take on negative values. These $\phi_i^{(k)}$ can be determined in advance or calculated during the updates. During the update process, the matrix times vector terms in Step $3, 6, 7$ are computationally efficient since $\mathcal{B}_k^0$ and $\mathcal{H}_k^0$ are essentially identity matrix multiplied

by scalar values. Additionally, $\hat{M}_l$ and $\tilde{M}_l$ are two symmetric small matrices with a maximum size $2L \times 2L$ and $s_{j+1}^T \mathcal{B}_{j+1} s_{j+1}$ and $y_{j+1}^T \mathcal{H}_{j+1} y_{j+1}$ are scalar values for $j = l, \cdots, k-2$, further contributing to the computational efficiency of the algorithm.

The primary computational cost in Algorithm 3 arises from the calculation of $\hat{\Psi}_j^T s_{j+1}$ at Step 4 and $\tilde{\Psi}_j^T y_{j+1}$ at Step 7. The columns of $\hat{\Psi}_j$ and $\tilde{\Psi}_j$ are essentially the curvature-related vectors s, y multiplied by scalars. Therefore, calculating $\hat{\Psi}_j^T s_{j+1}$ and $\tilde{\Psi}_j^T y_{j+1}$ invloves computing inner products between these curvature-related vectors. By storing the previous inner products, the computation at step k requires calculations of $s_{k-1}^T s_i,\ s_{k-1}^T y_i,\ y_{k-1}^T y_i,\ y_{k-1}^T s_i$ with $i = l, \cdots, k-2$. Suppose the intrinsic representation is d-dimensional, the computational complexity of determining search directions in Algorithm 4 is $8(L-1)d + 8Ld + 2d + \mathcal{O}(L^2)$. Compared to the cost of $8Ld + 2d$ for finding search directions in LRBFGS (as presented in [24]), the additional cost for LRBroyden is $8(L-1)d + \mathcal{O}(L^2)$. This additional cost is acceptable since the memory size L is typically small for solving large-scale problems. The outer structure Algorithm 4 is essentially the same as the existing intrinsic algorithm of LRBFGS [24], hence there is no difference in the computational complexity there.

Algorithm 4 effectively addresses the drawbacks of Algorithm 2 in both the Riemannian and Euclidean versions. Notably, Algorithm 4 does not require the computation, storage and transportation of $u_i^{(k)}$. Moreover, the update mechanism within Algorithm 4 is unique with a chosen $\phi_i^{(k)}$ and the choice of $\phi_i^{(k)}$ can be any real numbers. It is important to note that the experiments conducted by Deguchy et al. [12] did not include a performance comparison across different members of LBroyden. This dissertation makes a significant contribution by exploring and identifying robust and efficient member of the LRBroyden family, as detailed in Chapter 4 and 5.

Algorithm 3 Update $\tilde{M}_j, \tilde{\Psi}_k$ at step k by generalization of [12]

Require: $\mathcal{B}_k^0, \mathcal{H}_k^0$ and $\phi_i^{(k)}\rho_i, s_i, y_i$ for $i = l, \cdots, k-1$, with $l = \max\{k-L, 0\}$. For clarity, we ignore the superscript (k) for $\phi_i^{(k)}$;

1: Define $\hat{M}_l = \begin{cases} -\hat{\beta}_l, & \phi_l = \phi_l^{SR1} \\ \begin{pmatrix} \hat{\alpha}_l & \hat{\beta}_l \\ \hat{\beta}_l & \hat{\delta}_l \end{pmatrix}, & otherwise \end{cases}$ and $\tilde{M}_l = \begin{cases} -\tilde{\beta}_l, & \phi_l = \phi_l^{SR1} \\ \begin{pmatrix} \tilde{\alpha}_l & \tilde{\beta}_l \\ \tilde{\beta}_l & \tilde{\delta}_l \end{pmatrix}, & otherwise \end{cases}$, these scalars are defined in the line $11-13$;

2: **for** $j = l : k-2$ **do**

3: $\hat{\Psi}_j \leftarrow \big[(\mathcal{B}_k^0 s_l \ y_l)E_l \ (\mathcal{B}_k^0 s_{l+1} \ y_{l+1})E_{l+1} \cdots (\mathcal{B}_k^0 s_j \ y_j)E_j\big]$, where $E_j = \begin{cases} (-1 \ 1)^T, & \phi_j = \phi_j^{SR1} \\ I_2, & otherwise; \end{cases}$

4: Calculate $\hat{\Psi}_j^T s_{j+1}$;

5: Calculate $p_{j+1} = \hat{M}_j(\hat{\Psi}_j^T s_{j+1})$;

6: $s_{j+1}^T \mathcal{B}_{j+1} s_{j+1} \leftarrow s_{j+1}^T(\mathcal{B}_k^0 s_{j+1}) + p_{j+1}^T(\hat{\Psi}_j^T s_{j+1})$;

7: Calculate $\tilde{\Psi}_j^T y_{j+1} = \hat{\Psi}_j^T \mathcal{H}_k^0 y_{j+1}$;

8: Calculate $\tilde{p}_{j+1} = \tilde{M}_j(\tilde{\Psi}_j^T y_{j+1})$;

9: $y_{j+1}^T \mathcal{H}_{j+1} y_{j+1} \leftarrow y_{j+1}^T(\mathcal{H}_k^0 y_{j+1}) + \tilde{p}_{j+1}^T(\tilde{\Psi}_j^T y_{j+1})$;

10: $\Phi_{j+1} \leftarrow (1-\phi_{j+1})/((1-\phi_{j+1}) + \rho_{j+1}^2 \phi_{j+1}(y_{j+1}^T \mathcal{H}_{j+1} y_{j+1})(s_{j+1}^T \mathcal{B}_{j+1} s_{j+1}))$;

11: $\hat{\alpha}_{j+1} \leftarrow -(1-\phi_{j+1})/(s_{j+1}^T \mathcal{B}_{j+1} s_{j+1})$, and $\tilde{\alpha}_{j+1} \leftarrow (1 + \Phi_j(y_{j+1}^T \mathcal{H}_{j+1} y_{j+1})\rho_{j+1})\rho_{j+1}$;

12: $\hat{\beta}_{j+1} \leftarrow -\phi_{j+1}\rho_{j+1}$, and $\tilde{\beta}_{j+1} \leftarrow -\Phi_{j+1}\rho_{j+1}$;

13: $\hat{\delta}_{j+1} \leftarrow (1 + \phi_{j+1}(s_{j+1}^T \mathcal{B}_{j+1} s_{j+1})\rho_{j+1})\rho_{j+1}$, and $\tilde{\delta}_{j+1} \leftarrow -(1-\Phi_{j+1})/(y_{j+1}^T \mathcal{H}_{j+1} y_{j+1})$;

14: Form $\hat{M}_{j+1} \leftarrow \begin{cases} \begin{pmatrix} \hat{M}_j - \hat{\beta}_{j+1}\hat{p}_{j+1}\hat{p}_{j+1}^T & -\hat{\beta}_{j+1}\hat{p}_{j+1} \\ -\hat{\beta}_{j+1}\hat{p}_{j+1}^T & -\hat{\beta}_{j+1} \end{pmatrix}, & \phi_{j+1} = \phi_{j+1}^{SR1} \\ \begin{pmatrix} \hat{M}_j + \hat{\alpha}_{j+1}p_{j+1}p_{j+1}^T & \hat{\alpha}_{j+1}p_{j+1} & \hat{\beta}_{j+1}p_{j+1} \\ \hat{\alpha}_{j+1}p_{j+1}^T & \hat{\alpha}_{j+1} & \hat{\beta}_{j+1} \\ \hat{\beta}_{j+1}p_{j+1}^T & \hat{\beta}_{j+1} & \hat{\delta}_{j+1} \end{pmatrix}, & otherwise; \end{cases}$

15: Form $\tilde{M}_{j+1} \leftarrow \begin{cases} \begin{pmatrix} \tilde{M}_j - \tilde{\beta}_{j+1}\tilde{p}_{j+1}\tilde{p}_{j+1}^T & -\tilde{\beta}_{j+1}\tilde{p}_{j+1} \\ -\tilde{\beta}_{j+1}\tilde{p}_{j+1}^T & -\tilde{\beta}_{j+1} \end{pmatrix}, & \phi_{j+1} = \phi_{j+1}^{SR1} \\ \begin{pmatrix} \tilde{M}_j + \tilde{\delta}_{j+1}\tilde{p}_{j+1}\tilde{p}_{j+1}^T & \tilde{\beta}_{j+1}\tilde{p}_{j+1} & \tilde{\delta}_{j+1}\tilde{p}_{j+1} \\ \tilde{\beta}_{j+1}\tilde{p}_{j+1}^T & \tilde{\alpha}_{j+1} & \tilde{\beta}_{j+1} \\ \tilde{\delta}_{j+1}\tilde{p}_{j+1}^T & \tilde{\beta}_{j+1} & \tilde{\delta}_{j+1} \end{pmatrix}, & otherwise; \end{cases}$

16: **end for**

17: **Output:** $\tilde{\Psi}_{k-1} = \mathcal{H}_k^0 \hat{\Psi}_{k-1}$ and $\tilde{M}_{k-1}$.

Algorithm 4 Full LRBroyden by generalization of [12] using intrinsic representation and vector transport by parallelization

Require: Riemannian manifold $\mathcal{M}$ with Riemannian metric g; a retraction R; Smooth function f on $\mathcal{M}$; initial iterate $x_0 \in \mathcal{M}$; an integer $L > 0$. The superscript d means the intrinsic representations;

1: $k = 0, \varepsilon > 0, 0 < c_1 < \frac{1}{2} < c_2 < 1, \gamma_0 = 1, l = 0$;

2: $\mathcal{H}_k^0 = \gamma_k \mathrm{I}$, $\mathcal{B}_k^0 = 1/\gamma_k \mathrm{I}$. Obtain $\eta_k \in T_{x_k}\mathcal{M}$ by the following algorithm:

3: Given s_i, y_i, ρ_i and the method to set $\phi_i^{(k)}$ for $i = k-1, k-2, \cdots, l$;

4: $q \leftarrow \mathrm{grad}^d f(x_k)$;

5: Compute $\tilde{M}_{k-1}, \tilde{\Psi}_{k-1}$ based on Algorithm 3;

6: Set $\eta_k^d = -\mathcal{H}_k^0 q - \tilde{\Psi}_{k-1}\tilde{M}_{k-1}\tilde{\Psi}_{k-1}^T q$;

7: Find α_k that satisfies Wolfe conditions

$$f(x_{k+1}) \leq f(x_k) + c_1\alpha_k(\mathrm{grad}^d f(x_k))^T \eta_k^d,$$
$$(\eta_k^d)^T \mathrm{grad}^d f(x_{k+1}) \geq c_2(\eta_k^d)^T \mathrm{grad}^d f(x_k);$$

8: Compute the extrinsic representation η_k of η_k^d;

9: Set $x_{k+1} = R_{x_k}(\alpha_k \eta_k)$. Compute the intrinsic representation $\mathrm{grad}^d f(x_{k+1})$ of $\mathrm{grad} f(x_{k+1})$. If $\|\mathrm{grad} f(x_{k+1})\| > \varepsilon$, then break;

10: define $s_k = \alpha_k \eta_k^d$ and $y_k = \mathrm{grad}^d f(x_{k+1}) - \mathrm{grad}^d f(x_k)$;

11: **if** $\frac{s_k^T y_k}{\|s_k\|_2^2} \geq \vartheta \|\mathrm{grad}^d f(x_k)\|_2$ **then**

12: Calculate $\rho_k = 1/s_k^T y_k$ and $\gamma_{k+1} = s_k^T y_k / y_k^T y_k$;

13: Let $l = \max\{k - L + 1, 0\}$. Add s_k, y_k and ρ_k into storage and if $k \geq L$, then discard vector pair $\{s_{l-1}, y_{l-1}\}$ and scalar ρ_{l-1} from storage, i.e. store $\{s_l, s_{l+1}, \cdots, s_k\}$, $\{y_l, y_{l+1}, \cdots, y_k\}$ and $\{\rho_l, \rho_{l+1}, \cdots, \rho_k\}$;

14: **else**

15: Set $\gamma_{k+1} \leftarrow \gamma_k$, $\{\rho_l, \cdots, \rho_k\} \leftarrow \{\rho_{l-1}, \cdots, \rho_{k-1}\}$, $\{s_l, \cdots, s_k\} \leftarrow \{s_{l-1}, \cdots, s_{k-1}\}$, $\{y_l, \cdots, y_k\} \leftarrow \{y_{l-1}, \cdots, y_{k-1}\}$;

16: **end if**

17: $k = k + 1$, go to 2;

3.5 Methods of Choosing $\phi_i^{(k)}$

The parameter $\phi_i^{(k)}$ is related to the update formula of $\mathcal{B}_k^i \to \mathcal{B}_k^{i+1}$, while $\Phi_i^{(k)}$ is related to the update formula of $\mathcal{H}_k^i \to \mathcal{H}_k^{i+1}$. The relationship between $\phi_i^{(k)}$ and $\Phi_i^{(k)}$ is as follows and is the same

Table 3.1: Relationship between the values of $\phi_i^{(k)}$ and values of $\Phi_i^{(k)}$.

$\phi_i^{(k)}$	$\phi_i^{(k)c}$	0	1	$+\infty$
$\Phi_i^{(k)}$	$+\infty$	1	0	$\phi_i^{(k)c}$

as Step 10 in Algorithm 3 (in which the superscript is suppressed):

$$\Phi_i^{(k)} = \frac{1-\phi_i^{(k)}}{(1-\phi_i^{(k)}) + (\rho_i^{(k)})^2\phi_i^{(k)} g(y_i^{(k)}, \mathcal{H}_k^i y_i^{(k)}) g(s_i^{(k)}, \mathcal{B}_k^i s_i^{(k)})}. \tag{3.12}$$

To guarantee the convergence of the LRBroyden family of methods, $\phi_i^{(k)}$ is required to be set in an appropriate range. Choosing $\phi_i^{(k)}$ equal to or less than the following negative critical value $\phi_i^{(k)c}$ makes $\hat{\mathcal{B}}_k^{i+1}$ singular or indefinite

$$\phi_i^{(k)c} = \frac{1}{1 - \frac{g(y_i^{(k)}, \mathcal{H}_k^i y_i^{(k)}) g(s_i^{(k)}, \mathcal{B}_k^i s_i^{(k)})}{g(y_i^{(k)}, s_i^{(k)})^2}}. \tag{3.13}$$

The values of $g(y_i^{(k)}, \mathcal{H}_k^i y_i^{(k)})$, $g(s_i^{(k)}, \mathcal{B}_k^i s_i^{(k)})$ and $g(y_i^{(k)}, s_i^{(k)})$ are explicitly calculated during the update indicated in Algorithm 3. All choices of $\phi_i^{(k)}$ should be greater than $\phi_i^{(k)c}$ in order to guarantee the positive definiteness of $\hat{\mathcal{B}}_k^{i+1}$. Based on the formula (3.12), $\Phi_i^{(k)}$ and $\phi_i^{(k)}$ are symmetric with respect to "$y = x$". Consider $\phi_i^{(k)c}$ as a value, the following Table 3.1 illustrates the relationship between the values of $\Phi_i^{(k)}$ and $\phi_i^{(k)}$. In this dissertation, $\phi_i^{(k)}$ is used as the primary parameter in the analysis and experimental setting. The convergence analysis for the LRBroyden family of methods are presented in the next section under the following guard for $\phi_i^{(k)}$:

$$\phi_i^{(k)} \in ((1-\nu)\phi_i^{(k)c}, 1-\delta) \tag{3.14}$$

with $\nu \in (0,1)$ and $\delta \in (0,1]$.

There are two ways to update the Broyden parameter of $\phi_i^{(k)}$ within the LRBroyden family. The first approach is to store, shift and discard the values of $\phi_i^{(k)}$ along with $s_i^{(k)}$, $y_i^{(k)}$ for $i = \max\{k-L, 0\}, \cdots, k-1$ at step k. In this approach, historical values of $\phi_i^{(k)}$ are retained temporarily and updated over time. In contrast, the second approach does not involve storing historical values of $\phi_i^{(k)}$. Instead, it computes or sets the values of $\phi_i^{(k)}$ directly for each update from $\mathcal{B}_k^0 \equiv \mathcal{B}_k^l$ to $\mathcal{B}_k \equiv \mathcal{B}_k^k$ at step k.

Consider LBroyden in Euclidean space with $L = 2$ as an example, suppose B_k^0 is positive definite, Broyden updates $B_k^0 \to B_k^1 \to B_k^2 \equiv B_k$ at step k, and denote U as the Broyden update

matrices. The first three steps following the first approach can be expressed as:

$$B_1 = B_1^0 + U(s_0, y_0, B_1^0, \phi_0),$$

$$B_2 = B_2^0 + U(s_0, y_0, B_2^0, \phi_0) + U(s_1, y_1, B_2^1, \phi_1),$$

$$B_3 = B_3^0 + U(s_1, y_1, B_3^0, \phi_1) + U(s_2, y_2, B_3^1, \phi_2),$$

where B_3 reuses ϕ_1 from last update. However, it may not be acceptable to use the same ϕ_1 if ϕ_1 falls outside the guard range (3.14). Even though $U(s_1, y_1, B_2^1, \phi_1)$ preserves positive definiteness for B_2, the new update $U(s_1, y_1, B_3^0, \phi_1)$ does not guarantee that for B_3 unless it is constrained or replaced with a new computed value.

For the second approach, the $\phi_i^{(k)}$ is computed or set at each step k each update i so that the positive definite property is preserved, as indicated in the following expression of B_3:

$$B_3 = B_3^0 + U(s_1, y_1, B_3^0, \phi_1^{(3)}) + U(s_2, y_2, B_3^1, \phi_2^{(3)}).$$

One way to justify this approach is by considering LBroyden as restarting from a new initial point. Following the safeguard (3.14) for $\phi_i^{(k)}$, the first approach is only applicable when the Broyden parameters are chosen within the restricted Broyden family.

More generally, the first approach is presented in Algorithm 2, where we use Φ_k in place of $\Phi_i^{(k)}$ to account for the shift and reuse of values. The second approach without the need to store historical $\phi_i^{(k)}$ is illustrated in Algorithm 4. It is possible but not necessary to use the shifting $\phi_i^{(k)}$, while the convergence guard (3.14) should always be checked during the update. We primarily consider the following choices of $\phi_i^{(k)}$ in this dissertation:

- BFGS update with $\phi_i^{(k)} = 0$.
- Constant value $c \in \mathbb{R}$ for the update, where $\phi_i^{(k)} = c$ for each i, k. Set $\phi_i^{(k)} = 0$ if the convergence bound (3.14) is not satisfied.
- Davidon update (see Section 4.4) with historical usage, as mentioned in the first approach. Set $\phi_i^{(k)} = 0$ if the convergence bound (3.14) is not satisfied.
- Davidon update without historical usage, as mentioned in the second approach.

3.6 Convergence Analysis

In the Euclidean setting, the global and superlinear convergence of restricted Broyden family of methods was presented in [10], based on the Dennis Moré condition [13, 14]. The result was extended to the full Broyden family in [8] under some strengthened assumptions on the choice of ϕ_k. On a Riemannian manifold, the global and superlinear convergence proof of the RBroyden family with $\phi_k \in [0, 1-\delta]$ was presented in [27], where $\delta \in (0, 1]$. For a limited-memory variant, the R-linear convergence of LBFGS was proved in [33]. In this section, the global convergence proof of the RBroyden family is extended to the wider range $(1-\nu)\phi_k^c \leq \phi_k \leq 1-\delta$, where $\phi_k^c = \frac{1}{1-\frac{(y_k^\flat \tilde{\mathcal{H}}_k y_k)(s_k^\flat \tilde{\mathcal{B}}_k s_k)}{(y_k^\flat s_k)^2}}$. Then the global convergence and R-linear convergence rate for the LRBroyden family are proved.

3.6.1 Basic Assumptions and Preliminary Lemmas

Throughout the convergence analysis, consider the update formula for the RBroyden family shown in Equation (3.1). $\{x_k\}$, $\{\mathcal{B}_k\}$, $\{\tilde{\mathcal{B}}_k\}$, $\{\alpha_k\}$, $\{s_k\}$, $\{y_k\}$, and $\{\eta_k\}$, are infinite sequences generated by Algorithm 1, Ω denotes the sublevel set $\{x : f(x) \leq f(x_0)\}$, and x^* is a local minimizer of f in the level set Ω. The manifold $\mathcal{M}$ is assumed to be compact so Ω is compact, which guarantees the existence of such an x^*.

Assumption 3.1. *The objective function f is twice continuously differentiable.*

Let $\tilde{\Omega}$ be a neighborhood of x^* and ρ be a positive constant such that, for all $y \in \tilde{\Omega}$, $\tilde{\Omega} \subset R_y(\mathbb{B}(0_y, \rho))$ and $R_y(\cdot)$ is a diffeomorphism on $\mathbb{B}(0_y, \rho)$.

Assumption 3.2. *$R_{x_k}(t\eta_k) \in \tilde{\Omega}$ for all $t \in [0, \alpha_k]$.*

The convergence analysis depends on the property of retraction-convexity. For a function $f : \mathcal{M} \to \mathbb{R} : x \mapsto f(x)$ on a Riemannian manifold $\mathcal{M}$ with retraction R, define $m_{x,\eta}(t) = f(R_x(t\eta))$ for $x \in \mathcal{M}$ and $\eta \in \mathrm{T}_x\mathcal{M}$. The function f is retraction-convex with respect to the retraction R in a set $\mathcal{S}$ if for all $x \in \mathcal{S}$, all $\eta \in \mathrm{T}_x\mathcal{M}$ and $\|\eta\| = 1$, $m_{x,\eta}(t)$ is convex for all t which satisfy $R_x(\tau\eta) \in \mathcal{S}$ for all $\tau \in [0, t]$. Moreover, f is strongly retraction-convex in $\mathcal{S}$ if $m_{x,\eta}(t)$ is strongly convex, i.e., there exist two constants $0 < a_0 < a_1$ such that

$$a_0 \leq \frac{d^2 m_{x,\eta}}{dt^2}(t) \leq a_1, \tag{3.15}$$

for all $x \in \mathcal{S}$, all $\|\eta\| = 1$ and all t such that $R_x(\tau\eta) \in \mathcal{S}$ for all $\tau \in [0, t]$.

Assumption 3.3. *f is strongly retraction-convex with respect to the retraction R in $\tilde{\Omega}$.*

The global convergence proof for the RBroyden family is an extension to the existing convergence proof in [27]. Therefore, the following Lemmas 3.1-3.10 are restatements of the existing lemmas in [27]. Lemma 3.11 is novel based on the fact that the bound for ϕ_k is different.

Lemma 3.1. *(see [27, Lemma 3.1]) Suppose Assumption 3.1 holds and $Hessf(x^*)$ is positive definite. Define $\tilde{m}_{x,\eta}(t) = f(R_x(t\eta))$. Then there exists a neighbor $\mathcal{N}$ of x^* and two constants $0 < \tilde{a}_0 < \tilde{a}_1$ such that*

$$\tilde{a}_0 \leq \frac{d^2\tilde{m}_{x,\eta}}{dt^2}(t) \leq \tilde{a}_1,$$

for all $x \in \mathcal{N}$ and t which satisfies $R_x(t\eta) \in \mathcal{N}$.

Lemma 3.2. *(see [27, Lemma 3.2]) Using Lemma 3.1 and the first Wolfe condition (3.5), if Assumption 3.1, 3.2 and 3.3 hold then there exists a constant $a_0 > 0$ such that*

$$\frac{1}{2}a_0\|s_k\|^2 \leq (c_1 - 1)\alpha_k g(\mathrm{grad}f(x_k), \eta_k),$$

where constant a_0 can be chosen as in (3.15).

Lemma 3.3. *(see [27, Lemma 3.3]) Using the locking condition (3.4), if Assumption 3.1, 3.2 and 3.3 hold then there exists constants $0 < a_0 \leq a_1$ such that*

$$a_0 g(s_k, s_k) \leq g(s_k, y_k) \leq a_1 g(s_k, s_k),$$

where constants a_0 and a_1 can be chosen as in (3.15).

Lemma 3.4. *(see [27, Lemma 3.4]) Using Lemma 3.2, if Assumption 3.1, 3.2 and 3.3 hold then there exists constants $0 < a_2 < a_3$ such that*

$$a_2\|\mathrm{grad}f(x_k)\|\cos\theta_k \leq \|s_k\| \leq a_3\|\mathrm{grad}f(x_k)\|\cos\theta_k,$$

for all k, where $\cos\theta_k = \frac{-g(\mathrm{grad}f(x_k),\eta_k)}{\|\mathrm{grad}f(x_k)\|\|\eta_k\|}$.

Lemma 3.5. *(see [27, Lemma 3.5]) Let $\mathcal{M}$ be a Riemannian manifold endowed with two vector transports $\mathcal{T}_1 \in C^0$ and $\mathcal{T}_2 \in C^\infty$ where $\mathcal{T}_1$ satisfies $\|\mathcal{T}_{S_\eta} - \mathcal{T}_{R_\eta}\| \leq c_0\|\eta\|$, $\|\mathcal{T}_{S_\eta}^{-1} - \mathcal{T}_{R_\eta}^{-1}\| \leq c_0\|\eta\|$ and both transports are associated with a same retraction R. Then for any $\bar{x} \in \mathcal{M}$ there exists a constant a_4 and a neighborhood of $\bar{x}$,$\mathcal{U}$, such that for all $x, y \in \mathcal{U}$*

$$\|\mathcal{T}_{1_\eta}\xi - \mathcal{T}_{2_\eta}\xi\| \leq a_4\|\xi\|\|\eta\|,$$

where $\eta = R_x^{-1}y$ and $\xi \in T_x\mathcal{M}$.

Lemma 3.6. *(see [27, Lemma 3.6]) Using Lemma 3.5, let $\mathcal{M}$ be a Riemannian manifold endowed with a retraction R whose differentiated retraction is denoted $\mathcal{T}_R$. Let $\bar{x} \in \mathcal{M}$, then there is a neighborhood $\mathcal{U}$ of $\bar{x}$ and constant $\tilde{a}_4 > 0$ such that for all $x, y \in \mathcal{U}n$ any $\xi \in T_x\mathcal{M}$ with $\|\xi\| = 1$, the effect of the differentiated retraction is bounded with*

$$|\|\mathcal{T}_{R_\eta}\| - 1| \leq \tilde{a}_4\|\eta\|,$$

where $\eta = R_x^{-1}y$.

Lemma 3.7. *(see [27, Lemma 3.7]) Using Lemma 3.6, if Assumption 3.1, 3.2 and 3.3 hold then there exists a constant $a_5 > 0$ such that for all k*

$$f(x_{k+1}) - f(x^*) \leq (1 - a_5 \cos^2\theta_k)(f(x_k) - f(x^*)).$$

Lemma 3.8. *(see [27, Lemma 3.8]) Using Lemma 3.2, if Assumption 3.1, 3.2 and 3.3 hold then there exists constants $0 < a_6 < a_7$ such that for all k*

$$a_6\frac{g(s_k, \tilde{B}_k s_k)}{\|s_k\|^2} \leq \alpha_k \leq a_7\frac{g(s_k, \tilde{B}_k s_k)}{\|s_k\|^2}.$$

Lemma 3.9. *(see [27, Lemma 3.9]) Using Lemma 3.3, Lemma 3.5, Lemma 3.6 and Lemma 3.8, if Assumption 3.1 holds then there exists a constant $a_9 > 0$ such that for all k*

$$g(y_k, y_k) \leq a_9 g(s_k, y_k).$$

Lemma 3.10. *(see [27, Lemma 3.10]) Using the above lemmas, if Assumption 3.1, 3.2 and 3.3 hold then there exists constants $a_{10} > 0, a_{11} > 0, a_{12} > 0$ such that for all k*

$$\frac{g(s_k, \tilde{B}_k s_k)}{g(s_k, y_k)} \leq a_{10}\alpha_k,$$

$$\frac{\|\tilde{B}_k s_k\|^2}{g(s_k, \tilde{B}_k s_k)} \geq a_{11}\frac{\alpha_k}{\cos^2\theta_k},$$

$$\frac{|g(y_k, \tilde{B}_k s_k)|}{g(y_k, s_k)} \leq a_{12}\frac{\alpha_k}{\cos\theta_k}.$$

Lemma 3.11. *(This is a modification of [27, Lemma 3.11]) Using Lemma 3.7, Lemma 3.8, Lemma 3.9 and Lemma 3.10, if Assumption 3.1, 3.2 and 3.3 hold, the parameter $(1-\nu)\phi_k^c \leq \phi_k \leq 1$ where $\nu \in (0, 1)$ and ϕ_k^c defined in (3.13). Then there exists a positive constant a_{13} such that for all $k \geq 1$*

$$\prod_{j=1}^{k} \alpha_j \geq a_{13}^k.$$

Proof. Use hat to denote the coordinates expression of the operator $\mathcal{B}_k$ and $\tilde{\mathcal{B}}_k$, consider trace($\hat{\mathcal{B}}$) and det($\hat{\mathcal{B}}$). trace($\hat{\tilde{\mathcal{B}}}_k$) = trace($\hat{\mathcal{B}}_k$) and det($\hat{\tilde{\mathcal{B}}}_k$) = det($\hat{\mathcal{B}}_k$) since $\mathcal{T}_S$ is an isometric vector transport. The determinant of the update formula is

$$\det(\hat{\mathcal{B}}_{k+1}) = \det(\hat{\mathcal{B}}_k)\frac{g(y_k, s_k)}{g(s_k, \tilde{\mathcal{B}}_k s_k)}[1 + \phi_k(\mu_k - 1)], \tag{3.16}$$

where $\mu_k = (g(y_k, \tilde{\mathcal{B}}_k^{-1} y_k)g(s_k, \tilde{\mathcal{B}}_k s_k))/g(y_k, s_k)^2$. With $\phi_k \geq (1-\nu)\phi_k^c$ and $\phi_k^c = 1/(1-\mu_k)$, it is not hard to have the inequality $\det(\hat{\mathcal{B}}_{k+1}) \geq \det(\hat{\mathcal{B}}_k)\frac{g(y_k,s_k)}{g(s_k,\tilde{\mathcal{B}}_k s_k)}\nu$. With Lemma 3.10, we have

$$\det(\hat{\mathcal{B}}_{k+1}) \geq \det(\hat{\mathcal{B}}_k)\frac{1}{a_{10}\alpha_k}\nu \geq \det(\hat{\mathcal{B}}_1)\prod_{j=1}^{k}\frac{1}{a_{10}\alpha_j}\nu^k.$$

The trace of the update formula is

$$\text{trace}(\hat{\mathcal{B}}_{k+1}) = \text{trace}(\hat{\mathcal{B}}_k) + \frac{\|y_k\|^2}{g(y_k, s_k)} + \phi_k\frac{\|y_k\|^2}{g(y_k, s_k)}\frac{g(s_k, \tilde{\mathcal{B}}_k s_k)}{g(y_k, s_k)} - (1-\phi_k)\frac{\|\tilde{\mathcal{B}}_k s_k\|^2}{g(s_k, \tilde{\mathcal{B}}_k s_k)} - 2\phi_k\frac{g(y_k, \tilde{\mathcal{B}}_k s_k)}{g(y_k, s_k)}. \tag{3.17}$$

Based on the Equation (3.1), the trace is monotone function of ϕ_k so that the inequalities with trace($\hat{\mathcal{B}}_{k+1}$) as Equation (3.20) in [27] involving the trace still holds, that is

$$\text{trace}(\hat{\mathcal{B}}_{k+1}) \leq b_0^k.$$

We know $\det(\hat{\mathcal{B}}_{k+1}) \leq (\frac{\text{trace}(\hat{\mathcal{B}}_{k+1})}{d})^d$ by inequality of arithmetic and geometric means, where d is dimension of the manifold. Then

$$\prod_{j=1}^{k}\frac{1}{a_{10}\alpha_j} \leq \frac{1}{\det(\hat{\mathcal{B}}_1)\nu^k}(\frac{\text{trace}(\hat{\mathcal{B}}_{k+1})}{d})^d \leq \frac{(b_0^d)^k}{\det(\hat{\mathcal{B}}_1)\nu^k d^d}.$$

These derives the inequality $\prod_{j=1}^{k}\alpha_j \geq \frac{\det(\hat{\mathcal{B}}_1)d^d}{(a_{10}b_0^d)^k}\nu^k$, so that there exists a constant $a_{13} > 0$ such that $\prod_{j=1}^{k}\alpha_j \geq a_{13}^k$ for all $k \geq 1$. □

Lemma 3.11 extends the bound of ϕ_k from $[0, 1]$ to $[(1-\nu)\phi_k^c, 1]$. Note that the main difference of proof between Lemma 3.11 and [27, Lemma 3.11] is ν.

3.6.2 Global Convergence Analysis

With the preliminary lemmas in place, the global convergence result for the RBroyden family can be stated and proved in the same way as Theorem 3.1 in [27]:

Theorem 3.1. *Suppose Assumptions 3.1, 3.2 and 3.3 hold and* $(1-\nu)\phi_k^c \leq \phi_k \leq 1-\delta$ *with* $\nu \in (0,1), \delta \in (0,1]$ *and* ϕ_k^c *defined in* (3.13)*. Then the sequence* $\{x_k\}$ *generated by Algorithm 1 converges to a minimizer* x^* *of* f*.*

The next theorem states and proves the main convergence result for Algorithm 4. For clarity, the superscripts and subscripts are reindexed as follows. Suppose $\mathcal{B}_k^0$ is the initial guess and $\mathcal{B}_k^j$ with $j = 1, 2, \cdots, L$ are generated on the tangent space $\mathrm{T}_{x_k}\mathcal{M}$. Denote $\mathcal{B}_k := \mathcal{B}_k^L$. Notice that $\mathcal{B}_k$ can be viewed as updating $\mathcal{B}_k^0$ L times with the RBroyden formula. The curvature pairs $\{s_j^{(k)}, y_j^{(k)}\}$ with $j = 0, 1, \cdots, L-1$ are on the tangent space $\mathrm{T}_{x_k}\mathcal{M}$. One important difference compared to the RBroyden family is that $\{s_j^{(k)}, y_j^{(k)}\}$ are generated at the iterate $\mathcal{B}_{k-L+j}$ but not at the iterate $\mathcal{B}_k^j$, then transported to the current tangent space.

Theorem 3.2. *Using Lemma 3.3, Lemma 3.7 and Lemma 3.9, suppose Assumptions 3.1, 3.2 and 3.3 hold and* $(1-\nu)\phi_i^{(k)c} \leq \phi_i^{(k)} \leq 1-\delta$*. Then the sequence* $\{x_k\}$ *generated by Algorithm 4 converges to a minimizer* x^* *of* f*.*

Proof. Consider $\det(\hat{\mathcal{B}}_k^L)$ and $\mathrm{trace}(\hat{\mathcal{B}}_k^L)$, the following inequalities are based on the formula (3.16) and (3.17):

$$\det(\hat{\mathcal{B}}_k^L) \geq \det(\hat{\mathcal{B}}_k^0) \prod_{j=0}^{L-1} \frac{g(y_j^{(k)}, s_j^{(k)})}{g(s_j^{(k)}, \mathcal{B}_k^j s_j^{(k)})} \nu,$$

$$\mathrm{trace}(\hat{\mathcal{B}}_k^L) \leq \mathrm{trace}(\hat{\mathcal{B}}_k^0) + \sum_{j=0}^{L-1} \left[\frac{\|y_j^{(k)}\|^2}{g(y_j^{(k)}, s_j^{(k)})} + \frac{\|y_j^{(k)}\|^2}{g(y_j^{(k)}, s_j^{(k)})} \frac{g(s_j^{(k)}, \mathcal{B}_k^j s_j^{(k)})}{g(y_j^{(k)}, s_j^{(k)})} + 2\frac{|g(y_j^{(k)}, \mathcal{B}_k^j s_j^{(k)})|}{g(y_j^{(k)}, s_j^{(k)})} \right],$$

where hat denotes the coordinates expressions of corresponding operators. The second inequality is derived from the fact that the trace is a monotone function of $\phi_i^{(k)}$ when $\phi_i^{(k)} < 1$. Notice that $\det(\hat{\mathcal{B}}_k^0)$ and $\mathrm{trace}(\hat{\mathcal{B}}_k^0)$ are bounded since $\hat{\mathcal{B}}_k^0 = 1/\gamma_k \mathrm{id}$. We then prove that $\det(\hat{\mathcal{B}}_k^j)$ is bounded below and $\mathrm{trace}(\hat{\mathcal{B}}_k^j)$ is bounded above for $j = 1, 2, \cdots, L$.

Suppose $\det(\hat{\mathcal{B}}_k^j) \geq M_1$ and $\mathrm{trace}(\hat{\mathcal{B}}_k^j) \leq M_2$, the inequality of the determinant has the form:

$$\det(\hat{\mathcal{B}}_k^{j+1}) \geq \det(\hat{\mathcal{B}}_k^j) \frac{g(y_j^{(k)}, s_j^{(k)})}{g(s_j^{(k)}, \mathcal{B}_k^j s_j^{(k)})} \nu. \tag{3.18}$$

The fraction part on the right hand side can be rewritten as

$$\frac{g(y_j^{(k)}, s_j^{(k)})}{g(s_j^{(k)}, \mathcal{B}_k^j s_j^{(k)})} = \frac{g(y_j^{(k)}, s_j^{(k)})}{\|s_j^{(k)}\|^2} \frac{\|s_j^{(k)}\|^2}{g(s_j^{(k)}, \mathcal{B}_k^j s_j^{(k)})} \geq a_0 \frac{\|s_j^{(k)}\|^2}{g(s_j^{(k)}, \mathcal{B}_k^j s_j^{(k)})},$$

since $\frac{g(y_j^{(k)},s_j^{(k)})}{\|s_j^{(k)}\|^2} \geq a_0$ from Lemma 3.3. Then rewrite inverse of the fraction as

$$\frac{g(s_j^{(k)},\mathcal{B}_k^j s_j^{(k)})}{\|s_j^{(k)}\|^2} = \frac{\hat{s}^T G \hat{\mathcal{B}}_k^j \hat{s}}{\hat{s}^T G \hat{s}} = \frac{\hat{s}^T G^{1/2} G^{1/2} \hat{\mathcal{B}}_k^j G^{-1/2} G^{1/2} \hat{s}}{\hat{s}^T G^{1/2} G^{1/2} \hat{s}} = \frac{s^T M s}{s^T s},$$

where $s = G^{1/2}\hat{s}, M = G^{1/2}\hat{\mathcal{B}}_k^j G^{-1/2}$ and G is the matrix expression of the inner product. Notice that $\text{trace}(M) = \text{trace}(\hat{\mathcal{B}}_k^j) \leq M_2$. Hence $\frac{g(s_j^{(k)},\mathcal{B}_k^j s_j^{(k)})}{\|s_j^{(k)}\|^2} \leq M_2$ with the fact that the Rayleigh quotient $\frac{s^T M s}{s^T s}$ is less than the upper bound of the largest eigenvalue, which is less than the sum of all the eigenvalues. Combining the above we have:

$$\frac{g(y_j^{(k)}, s_j^{(k)})}{g(s_j^{(k)}, \mathcal{B}_k^j s_j^{(k)})} \geq a_0/M_2. \tag{3.19}$$

Plugging all into (3.18) shows us the determinant inequality:

$$\det(\hat{\mathcal{B}}_k^{j+1}) \geq M_1(a_0/M_2)\nu.$$

Then we consider the inequality of $\text{trace}(\hat{\mathcal{B}}_k^{j+1})$:

$$\text{trace}(\hat{\mathcal{B}}_k^{j+1}) \leq \text{trace}(\hat{\mathcal{B}}_k^j) + \frac{\|y_j^{(k)}\|^2}{g(y_j^{(k)}, s_j^{(k)})} + \frac{\|y_j^{(k)}\|^2}{g(y_j^{(k)}, s_j^{(k)})}\frac{g(s_j^{(k)}, \mathcal{B}_k^j s_j^{(k)})}{g(y_j^{(k)}, s_j^{(k)})} + 2\frac{|g(y_j^{(k)}, \mathcal{B}_k^j s_j^{(k)})|}{g(y_j^{(k)}, s_j^{(k)})}. \tag{3.20}$$

Notice that $\frac{\|y_j^{(k)}\|^2}{g(y_j^{(k)},s_j^{(k)})} \leq a_9$ by Lemma 3.9; $\frac{g(s_j^{(k)},\mathcal{B}_k^j s_j^{(k)})}{g(y_j^{(k)},s_j^{(k)})} \leq M_2/a_0$ by (3.19); $\frac{\|s_j^{(k)}\|^2}{g(y_j^{(k)},s_j^{(k)})} \leq 1/a_0$ by Lemma 3.3; $\frac{\|\mathcal{B}_k^j s_j^{(k)}\|}{\|s_j^{(k)}\|} \leq \|\mathcal{B}_k^j\| \leq \text{trace}(\hat{\mathcal{B}}_k^j) \leq M_2$. The second and the third term in (3.20) are bounded above with these inequalities. For the last term $\frac{|g(y_j^{(k)},\mathcal{B}_k^j s_j^{(k)})|}{g(y_j^{(k)},s_j^{(k)})}$, it can be derived that

$$\frac{|g(y_j^{(k)}, \mathcal{B}_k^j s_j^{(k)})|^2}{g(y_j^{(k)}, s_j^{(k)})^2} \leq \frac{\|y_j^{(k)}\|^2}{g(y_j^{(k)}, s_j^{(k)})}\frac{\|\mathcal{B}_k^j s_j^{(k)}\|^2}{g(y_j^{(k)}, s_j^{(k)})} = \frac{\|y_j^{(k)}\|^2}{g(y_j^{(k)}, s_j^{(k)})}\frac{\|\mathcal{B}_k^j s_j^{(k)}\|^2}{\|s_j^{(k)}\|^2}\frac{\|s_j^{(k)}\|^2}{g(y_j^{(k)}, s_j^{(k)})} \leq \frac{a_9}{a_0}M_2^2.$$

Hence the inequality of trace is as follows:

$$\text{trace}(\hat{\mathcal{B}}_k^{j+1}) \leq M_2 + a_9 + a_9 M_2/a_0 + \sqrt{a_9/a_0}M_2.$$

Since $\det(\hat{\mathcal{B}}_k^0)$ and $\text{trace}(\hat{\mathcal{B}}_k^0)$ are bounded, there exists constants M_3, M_4 such that $\det(\hat{\mathcal{B}}_k^L) \geq M_3$ and $\text{trace}(\hat{\mathcal{B}}_k^L) \leq M_4$ by recursion. And $\mathcal{B}_k = \mathcal{B}_k^L$ as the one used for updating x_k, we have

$$\cos\theta_k = \frac{-g(\text{grad}f(x_k), \eta_k)}{\|\text{grad}f(x_k)\|\|\eta_k\|} = \frac{-g(\text{grad}f(x_k), s_k)}{\|\text{grad}f(x_k)\|\|s_k\|} = \frac{g(\mathcal{B}_k s_k, s_k)}{\|\mathcal{B}_k s_k\|\|s_k\|} = \frac{\|s_k\|}{\|\mathcal{B}_k s_k\|}\frac{g(\mathcal{B}_k s_k, s_k)}{\|s_k\|^2}.$$

Notice that $\frac{\|s_k\|}{\|\mathcal{B}_k s_k\|} \geq 1/\text{trace}(\hat{\mathcal{B}}_k) \geq 1/M_4$ since $\frac{\|\mathcal{B}_k s_k\|}{\|s_k\|} \leq \|\hat{\mathcal{B}}_k\| \leq \text{trace}(\hat{\mathcal{B}}_k)$. Also $\frac{g(\mathcal{B}_k s_k, s_k)}{\|s_k\|^2} \geq \lambda_{\min}$, where $\lambda_{\min}$ is the minimum eigenvalue of $\hat{\mathcal{B}}_k$ and is bounded away from 0 because of the explicit bound for $\det(\hat{\mathcal{B}}_k)$ and $\text{trace}(\hat{\mathcal{B}}_k)$. There exists a constant $\delta = \lambda_{\min}/M_4 > 0$ such that

$$\cos\theta_k \geq \delta.$$

Therefore, applying Lemma 3.7 such that for all k,

$$f(x_{k+1}) - f(x^*) \leq (1 - a_5 \cos^2\theta_k)(f(x_k) - f(x^*)) \tag{3.21}$$

completes the proof. □

3.6.3 R-Linear Convergence Analysis of the LRBroyden family

It is possible to show q-superlinear convergence rate for the RBroyden family of methods with the restriction $\sum_{\phi_k \leq 0} \frac{\phi_k}{\phi_k^c} < \infty$ [8]. For the LRBroyden family of methods, R-linear convergence is stated and proved in the following theorem:

Theorem 3.3. *Based on Theorem 3.2, then the sequence $\{x_k\}$ generated by Algorithm 4 satisfies that, there is a constant $0 \leq a_{15} < 1$ such that*

$$f(x_{k+1}) - f(x^*) \leq a_{15}^k (f(x_1) - f(x^*)),$$

holds for all sufficiently large k.

Proof. By applying Inequality (3.21) k times,

$$f(x_{k+1}) - f(x^*) \leq \prod_{i=1}^{k} (1 - a_5 \cos^2\theta_i)^k (f(x_1) - f(x^*)). \tag{3.22}$$

Since $a_5 \cos^2\theta_i \geq a_5\delta^2$ is bounded away from 0, we can find a constant $a_{15} \in [0, 1)$, such that $f(x_{k+1}) - f(x^*) \leq a_{15}^k (f(x_1) - f(x^*))$, which completes the proof. □

Notice that the proof of R-linear convergence rate of the LRBroyden family is much easier than the RBroyden family (see Section 6.2.1 in [22]) due to the explicit bound for each $\cos\theta_i$. Even though imposing the limited-memory constraint may decrease the convergence rate theoretically, however, LRBroyden is still powerful since it reduces the complexity of storage and computational time for each iteration. There is a significant amount of application evidence for the success of LRBFGS in recent years. In the following chapters, the proposed LRBroyden family of methods is implemented and analyzed with different test problems to compare with the state-of-art LRBFGS.

CHAPTER 4

EXPERIMENTS OF LRBROYDEN METHODS

The LRBroyden family of methods is designed for solving large-scale problems and therefore it is preferred to keep the memory size reasonably small. The exploration and investigation of this dissertation explores a more robust and efficient library of LRBroyden for solving large-scale problem. The selection of the parameter $\phi_i^{(k)}$ plays a key role in determining the performance of LRBroyden. The easiest way is to choose a constant ϕ, such as LRBFGS with $\phi = 0$. A dynamic $\phi_i^{(k)}$ is more complicated and is the main interest in this dissertation.

Experiments in this chapter are designed to isolate and address the aspects influencing the performance of the proposed algorithm. When considering various members of the LRBroyden family, their performances for different memory sizes become a notable aspect. It is crucial to bear in mind that the emphasis is on keeping memory size small. Besides the choice of $\phi_i^{(k)}$, the initial stepsize α_0 also plays an important role in the performance of LRBroyden. In Section 4.3, a constant ϕ is tested to illustrate the relationship between ϕ and α_0. LRBroyden with a constant ϕ does not generally provide a better performance than LRBFGS with $\phi = 0$, but some values are competitive. In Section 4.4, a dynamic $\phi_i^{(k)}$ selection by Davidon [11] is introduced. The Davidon's choice of $\phi_i^{(k)}$ incorporating with LRBroyden outperforms LRBFGS in the synthetic problems constructed on Euclidean space and the Stiefel manifold. In the next chapter, a hybrid strategy $\phi_i^{(k)}$ between the choices of Davidon and BFGS is considered.

4.1 Test Problems and Test Data Parameters

We consider quadratic problems in Euclidean space and four well-known optimization problems defined on manifolds: minimization of the Brockett function on the Stiefel manifold, the low-rank matrix completion problem on the 2-factor quotient representation for the fixed rank manifold, finding a low-rank approximation for the solution of the Lyapunov equation on the manifold of symmetric positive semidefinite fixed rank matrices, and the weighted low-rank approximation problem on the fixed-rank manifold. All problems tested here are smooth and unconstrained.

4.1.1 Euclidean Quadratic

A quadratic problem in Euclidean space is defined for symmetric positive definite $A \in \mathbb{R}^{d\times n}$ as

$$\min_{x\in\mathbb{R}^d} 0.5x^T Ax.$$

It is easy to track the true Hessian so that the behavior of the algorithms can be analyzed according to the eigenvalues of Hessian.

In the experiments, the Hessian matrix is set to be $A := QDQ^T$, where D is diagonal. Q is obtained by applying Matlab ORTH command on a matrix whose entries are drawn from the standard normal distribution. The initial point x_0 is generated by the standard normal distribution. The Euclidean quadratic problem is parameterized to easily control the eigenvalues of Hessian matrix and is used to probe characteristics of Hessian that indicate an advantage for particular members of the LRBroyden family [8].

4.1.2 Brockett Cost Function on the Stiefel Manifold

The minimization problem of the Brockett cost function [1] on the Stiefel manifold $\mathrm{St}(p,n) := \{X \in \mathbb{R}^{n\times p} : X^T X = I_p\}$ has the following form:

$$\min_{X\in\mathrm{St}(p,n)} f(X) = \mathrm{trace}(X^T AXN), \tag{4.1}$$

where $N = \mathrm{diag}(\mu_1, \cdots, \mu_p)$ with $0 < \mu_1 < \cdots < \mu_p$, $A \in \mathbb{R}^{n\times n}$ and $A = A^T$. The tangent space of the Stiefel manifold is

$$\mathrm{T}_X\mathrm{St}(p,n) = \{X\Omega + X_\perp K : \Omega^T = -\Omega, K \in \mathbb{R}^{(n-p)\times p}\}.$$

Viewing $\mathrm{St}(p,n)$ as an embedded submanifold of the Euclidean space $\mathbb{R}^{n\times p}$, the metric endowed from the Euclidean space is

$$g(\xi_X, \eta_X) = \mathrm{trace}(\xi_X^T \eta_X),$$

where $\xi_X, \eta_X \in \mathrm{T}_X\mathrm{St}(p,n)$. The corresponding Riemannian gradient is

$$\mathrm{grad} f(X) = P_X(2AXN),$$

where the projection $P_X(\xi_X) = \xi_X - X\mathrm{sym}(X^T\xi_X)$ is onto $\mathrm{T}_X\mathrm{St}(p,n)$ with $\mathrm{sym}(M) := (M + M^T)/2$. The retraction is taken to be

$$R_X(\eta_X) = \mathrm{qf}(X + \eta_X),$$

where qf(A) denotes the Q factor of the QR decomposition of $A \in \mathbb{R}_*^{n\times p}$ with strictly positive elements on the diagonal of R.

In the experiments, $A = M + M^T$ where the elements of M are drawn from the standard normal distribution; $N = \text{diag}(1, \cdots, p-1, p)$ is a diagonal matrix whose diagonal elements are integers from 1 to p. The initial point X_0 is generated by applying Matlab's function ORTH to a matrix whose elements are drawn randomly from the standard normal distribution.

4.1.3 Low-rank Matrix Completion

Let $A \in \mathbb{R}^{m\times n}$ be an $m \times n$ matrix that is only known on a subset Ω of the complete set of entries $\{1, \cdots, m\} \times \{1, \cdots, n\}$. The low-rank matrix completion optimization problem (see [57, 35, 6]) is defined as

$$\min_{X\in\mathcal{M}_r} f(X) := \frac{1}{2}\|P_\Omega(X - A)\|_F^2,$$

where

$$\mathcal{M}_r := \{X \in \mathbb{R}^{m\times n} : \text{rank}(X) = r\}$$

and

$$P_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}, X_{i,j} \mapsto \begin{cases} X_{i,j}, & \text{if } (i,j) \in \Omega \\ 0, & \text{if } (i,j) \notin \Omega. \end{cases}$$

A popular way to parameterize fixed-rank matrices is through full-rank factorization [15]. A rank r matrix $X \in \mathcal{M}_r$ is factorized as

$$X = GH^T,$$

where $G \in \mathbb{R}_*^{m\times r}$ and $H \in \mathbb{R}_*^{n\times r}$ are full column-rank matrices. Such a factorization is not unique as X remains unchanged by

$$(G, H) \mapsto (GM^{-1}, HM^T),$$

for any non-singular matrix $M \in \text{GL}(r)$ where the set GL(r) denotes the general linear group of degree r.

Consider a product space $\bar{\mathcal{M}}_r := \mathbb{R}_*^{m\times r} \times \mathbb{R}_*^{n\times r}$ of matrices with full column rank r, the projection $\pi : \mathbb{R}_*^{m\times r} \times \mathbb{R}_*^{n\times r} \to \mathcal{M}_r : (G, H) \mapsto GH^T$ induces the equivalence relation $\sim$ on $\bar{\mathcal{M}}_r$. The structure of the fibers of π can be characterized by the linear group GL(r). Hence $\mathcal{M}_r$ can be defined as a quotient space

$$\mathcal{M}_r := \bar{\mathcal{M}}_r/\text{GL}(r) \simeq \mathbb{R}_*^{m\times r} \times \mathbb{R}_*^{n\times r}/\text{GL}(r).$$

The product space $\bar{\mathcal{M}}_r$ is referred to as the total space. A point in $\bar{\mathcal{M}}_r$ is denoted by $\bar{x} = (G_{\bar{x}}, H_{\bar{x}})$ or simply (G, H). The tangent space to $\bar{\mathcal{M}}_r$ at $\bar{x}$ is $\mathrm{T}_{\bar{x}}\bar{\mathcal{M}}_r = \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$. Given a matrix $X \in \mathcal{M}_r$ and a tangent vector $\xi \in \mathrm{T}_X\mathcal{M}_r$, the mapping π induces infinitely many representations of ξ : for $\bar{x} \in \pi^{-1}(X)$, any element $\bar{\xi} \in \mathrm{T}_{\bar{x}}\bar{\mathcal{M}}_r$ that satisfies $\mathrm{D}\pi(\bar{x})[\bar{\xi}] = \xi$ can be considered as a representation of ξ. Decompose the tangent space $\mathrm{T}_{\bar{x}}\bar{\mathcal{M}}_r := \mathcal{V}_{\bar{x}} \oplus \mathcal{H}_{\bar{x}}$, where the vertical space $\mathcal{V}_{\bar{x}} := \mathrm{T}_{\bar{x}}(\pi^{-1}(X))$ is the tangent space at $\bar{x}$ to the equivalence class $[\bar{x}]$ and the horizontal space $\mathcal{H}_{\bar{x}}$ is the complementary of $\mathcal{V}_{\bar{x}}$. Consequently, there is a unique representation $\bar{\xi} \in \mathcal{H}_{\bar{x}} \subset \mathrm{T}_{\bar{x}}\bar{\mathcal{M}}_r$ of ξ such that $\mathrm{D}\pi(\bar{x})[\bar{\xi}] = \xi$. The tangent vector $\bar{\xi}$ is called the horizontal lift of ξ.

The optimization of a real-valued function f on $\mathcal{M}_r$ can be seen as the following matrix factorization optimization problem:

$$\min_{(G,H)\in\mathbb{R}_*^{m\times r}\times\mathbb{R}_*^{n\times r}} \bar{f}(G, H) := f \circ \pi(G, H).$$

The Riemannian metric on the total space $\bar{\mathcal{M}}_r$ has the form

$$\bar{g}_{\bar{x}}(\bar{\xi}_{\bar{x}}, \bar{\eta}_{\bar{x}}) = \mathrm{trace}(\bar{\xi}_G^T \bar{\eta}_G (H^T H)) + \mathrm{trace}(\bar{\xi}_H^T \bar{\eta}_H (G^T G)),$$

where $\bar{x} = (G, H) \in \bar{\mathcal{M}}_r$, $\bar{\xi}_{\bar{x}}, \bar{\eta}_{\bar{x}} \in \mathrm{T}_{\bar{x}}\bar{\mathcal{M}}_r$ are denoted as $(\bar{\xi}_G, \bar{\xi}_H), (\bar{\eta}_G, \bar{\eta}_H)$. Given the horizontal lifts $\bar{\xi}_{\bar{x}}, \bar{\eta}_{\bar{x}} \in \mathcal{H}_{\bar{x}}\bar{\mathcal{M}}_r$ as the matrix representation of tangent vectors to the quotient manifold $\mathcal{M}_r$, a metric g in $\mathcal{M}_r$ is induced such that, for any $X \in \mathcal{M}_r$ and $\xi_X, \eta_X \in \mathrm{T}_X\mathcal{M}_r$,

$$g_X(\xi_X, \eta_X) = \bar{g}_{\bar{x}}(\bar{\xi}_{\bar{x}}, \bar{\eta}_{\bar{x}}).$$

The Riemannian gradient has the form

$$\mathrm{grad} f(G, H) = (G - SH(H^T H)^{-1}, H - S^T G(G^T G)^{-1}),$$

where $S = P_\Omega(MN^T - A)$ and the retraction is taken to be

$$R_{(G,H)}(\bar{\eta}_G, \bar{\eta}_H) = (G + \bar{\eta}_G, H + \bar{\eta}_H),$$

where $\bar{\eta}_{\bar{x}} \in \mathcal{H}_{\bar{x}}\bar{\mathcal{M}}_r$.

In the experiments, the matrix is defined as $A := A_L A_R^T$, where $A_L \in \mathbb{R}^{m\times r}, A_R \in \mathbb{R}^{n\times r}$ with i.i.d. standard Gaussian entries; the set of observations Ω is sampled uniformly at random among all sets of $|\Omega|$. As for the initial point X_0, instead of taking a random guess, we use the r leading singular vectors U, V with corresponding diagonal matrix D(entries are leading singular values) of

A to construct $X_0 = UDV^T$. This is an effective way to avoid being far away from the global optimizer. We use two parameters rank(r) and oversampling ratio(OS) to control the difficulties of the problem, where the oversampling ratio is defined as the ratio of the number of samples to the degrees of freedom in a non-symmetric matrix of rank r ($OS = |\Omega|/(r(m+n-r))$).

4.1.4 Computing Low-rank Solutions of Lyapunov Equations

A generalized Lyapunov equation has the form

$$AXM^T + MXA^T = C, \tag{4.2}$$

where A, M are symmetric positive definite and C is symmetric positive semidefinite given matrices. The solution X is symmetric and unique positive semidefinite matrix. It is widely used in signal processing, model reduction, system and control theory. As an alternative to factorization-based methods for solving a Lyapunov equation, Vandereycken and Vandewalle [58] proposed a method based on optimizing an objective function on the Riemannian manifold of symmetric positive semidefinite matrices of fixed rank. The cost function is defined as:

$$F : \mathbb{S}_+(n,n) \to \mathbb{R} : X \mapsto \text{trace}(XAXM) - \text{trace}(XC),$$

where $\mathbb{S}_+(n,n)$ denotes the symmetric positive semidefinite matrices. Adding low-rank constraints, we have

$$\min_{X \in \mathbb{S}_+(r,n)} f(X) = \text{trace}(XAXM) - \text{trace}(XC),$$

where $\mathbb{S}_+(r,n) = \{YY^T : Y \in \mathbb{R}_*^{n\times r}\}$ denotes the symmetric positive semidefinite $n \times n$ matrices with fixed rank r, and $\mathbb{R}_*^{n\times r}$ denotes all full-rank real $n \times r$ matrices.

The tangent space is $\mathrm{T}_{YY^T}S_+(p,r) = \{Y\dot{Y}^T + \dot{Y}Y^T | \dot{Y} \in \mathbb{R}^{n\times r}, \dot{Y} = YS + Y_\perp K, S = S^T \in \mathbb{R}^{r\times r}, K \in \mathbb{R}^{(n-r)\times r}\}$. Viewing $\mathbb{S}_+(r,n)$ as an embedded submanifold of $\mathbb{R}^{n\times n}$, the Riemannian metric endowed from Euclidean inner product is

$$g(\xi_X, \eta_X) = \text{trace}(\xi_X^T \eta_X),$$

where $\eta_X, \xi_X \in \mathrm{T}_X\mathbb{S}_+(r,n)$. The corresponding Riemannian gradient has the form

$$\text{grad} f(X) = P_{\mathrm{T}_X\mathbb{S}_+(r,n)}(AXM + MXA - C),$$

where $P_{\mathrm{T}_X\mathbb{S}_+(r,n)}(Z) = P_Y Z P_Y + P_Y^\perp Z P_Y + P_Y Z P_Y^\perp,\ P_Y^\perp = I - P_Y,\ P_Y = Y(Y^TY)^{-1}Y^T$. The retraction is taken to be

$$R_X(\eta_X) = P_{\mathbb{S}_+(r,n)}(X + \eta_X),$$

where the projection $P_{\mathbb{S}_+(r,n)}(Z) = \sum_{i=1}^{r} \sigma_i v_i v_i^T, Z = V\Sigma V^T, V = [v_1, \cdots, v_n], \Sigma = \text{diag}(\sigma_1, \cdots, \sigma_n)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. Details are in [58].

In the experiments, the matrices A, M, C are generated from the semidiscretization of a steel rail cooling problem [39] with dimension 821 and 3113, and the initial point Y is generated randomly from the standard normal distribution.

4.1.5 Weighted Low-rank Approximation

The weighted low-rank approximation problem [34] determines a matrix approximation X of a given matrix R with respect to a certain weighted norm

$$\min_{X \in \mathcal{M}_r} f(X) = \|R - X\|_W^2,$$

where $R \in \mathbb{R}^{m \times n}$ is given, $\mathcal{M}_r$ denotes the matrices with rank r, $W \in \mathbb{R}^{mn \times mn}$ is a positive definite symmetric weighting matrix, $\|R - X\|_W^2 = \text{vec}\{R - X\}^T W \text{vec}\{R - X\}$ and $\text{vec}\{A\}$ denotes the vectorized form of a matrix A.

Using the singular value decomposition [20], $\mathcal{M}_r = \{UDV^T : U \in \text{St}(m, r), V \in \text{St}(n, r), D = diag(\sigma_1, \cdots, \sigma_r), \sigma_1 \geq \cdots \sigma_r > 0\}$, where $\text{St}(m, k) = \{X \in \mathbb{R}^{m \times k} | X^T X = I_k\}$ is the Stiefel manifold. This representation is not unique but the update is established based on the unique associated tangent vector [62]. For $\forall X = U_r D_r V_r^T \in \mathcal{M}_r$, $\text{T}_X\mathcal{M}_r := \{U_r A V_r^T + U_r B V_{r\perp}^T + U_{r\perp} C V_r^T : A, B, C \text{ are arbitrary matrices}\}$. The Riemannian metric inherited from $\mathbb{R}^{m \times n}$ is

$$g(\xi_X, \eta_X) := \langle \xi_X, \eta_X \rangle_F = \text{vec}\{\xi_X\}^T \text{vec}\{\eta_X\}$$

with $X \in \mathcal{M}_r$ and $\xi_X, \eta_X \in \text{T}_X\mathcal{M}_r$. The orthogonal projection onto the tangent space at $X = U_r D_r V_r^T \in \mathcal{M}_r$ is $P_X : \mathbb{R}^{m \times n} \to \text{T}_X\mathcal{M}_r,\ Z \to P_X Z = U_r U_r^T Z + Z V_r V_r^T - U_r U_r^T Z V_r V_r^T$. The resulting Riemannian gradient on fixed-rank manifold $\mathcal{M}_r$ is

$$\text{grad} f_r := -P_X(2\text{vec}^{-1}(W \text{vec}\{R - X\})).$$

The retraction is taken to be

$$R_X(\eta_X) = P_{\mathcal{M}_r}(X + \eta_X),$$

where $P_{\mathcal{M}_r}(Z) = \sum_{i=1}^{r} \sigma_i u_i v_i^T,\ Z = U\Sigma V^T,\ U = [u_1, \cdots, u_{\min(m,n)}],\ V = [v_1, \cdots, v_{\min(m,n)}],\ \Sigma = \text{diag}(\sigma_1, \cdots, \sigma_{\min(m,n)})$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$.

In the experiments, the real matrix R is generated as $R_1 R_2^T \in \mathbb{R}^{m \times n}$, where $R_1 \in \mathbb{R}^{m \times r}$ and $R_2 \in \mathbb{R}^{n \times r}$ are drawn from the standard normal distribution. The weighted matrix W has the form

$U\Sigma U^T$ with U orthonormal and Σ diagonal. We control W by setting the diagonal of Σ a vector of logarithmically spaced points (logspace$(-1, 1)$ for 10^{-1} and 10) multiplied element-wise by a vector drawn from the uniform distribution on $[0.5, 1.5]$. The initial point $X_0 := U_0 D_0 V_0^T$ is generated with $D_0 \in \mathbb{R}^{r\times r}$ randomly from the standard normal distribution, and $U_0 \in \mathbb{R}^{m\times r}$, $V_0 \in \mathbb{R}^{n\times r}$ by applying Matlab's function ORTH to a matrix whose elements are drawn randomly from the standard normal distribution.

4.2 Notation and Algorithm Parameters

In the experiments, LRBFGS, which employs the two-loop recursion form proposed in Huang et al. [27], serves as the benchmark for evaluating members of the LRBroyden family. Both LRBFGS and LRBroyden are tested with intrinsic representation and vector transport by parallelization. Let L denotes the memory size; iter, nf, ng, nV, nR denote the number of iterations, the number of function evaluations, the number of gradient evaluations, the number of actions of vector transport and the number of actions of a retraction respectively; gf_f denotes the final norm of the gradient and gf_f/gf_0 denotes the ratio of the final norm of the gradient over the initial, t denotes the average run time (seconds). To obtain sufficiently stable timing results, an average time is taken of 10 runs with identical parameters and same initial conditions. The comparisons presented are performed in Matlab 9.8.0 on a Mac platform with 2.7 GHz and 8 GB memory.

Unless otherwise indicated in the description of the experiments, the following linesearch parameters are used for LRBFGS and LRBroyden.

- Line Search type: Wolfe condition with constants $c_1 = 10^{-4}$ and $c_2 = 0.999$;
- Initial stepsize: ONESTEP$[\alpha_0 = 1]$;
- Initial inverse Hessian approximation: $\mathcal{H}_k^0 = \gamma_k \mathrm{id}$ where $\gamma_k = \frac{g(s_{k-1}, y_{k-1})}{g(y_{k-1}, y_{k-1})}$;
- Stopping criteria: $\|gf_f\|/\|gf_0\| < 10^{-6}$(final norm of gradient over the initial).

The initial Hessian approxiamtion $\mathcal{H}_0^0$ is set to be the identity matrix. After one iteration is completed, all methods update with $\mathcal{H}_k^0 = \gamma_k \mathrm{id}$, where $\gamma_k = \frac{g(s_{k-1}, y_{k-1})}{g(y_{k-1}, y_{k-1})}$ is the Barzilai-Borwein (BB) scaling [2] that attempts to estimate the size of the true Hessian matrix (see [38])) along the most recent search direction. Once the search direction is determined after the limited-memory procedure, the stepsize α_k determines how far the iterate moves. $\gamma_k = \frac{g(s_{k-1}, y_{k-1})}{g(y_{k-1}, y_{k-1})}$ with $\alpha_0 = 1$ is an effective way of introducing the scale for LRBFGS (see [33] and [38]), hence are set as the

Table 4.1: Comparison of LRBFGS and LRDavidon without historical usage for the Stiefel Brockett problem. The subscript $-k$ indicates a scale of 10^{-k}.

(n,p)	(1000,5)									
method	LRBFGS					LRDavidon				
L	1	2	4	8	16	1	2	4	8	16
iter	629	589	529	488	469	683	693	609	523	525
nf	681	630	564	514	493	738	740	623	538	541
ng	631	590	530	489	470	684	694	610	524	526
nV	680	629	563	513	493	737	739	622	537	540
nR	1259	1177	1059	976	939	1365	1386	1218	1045	1050
gf_f	5.86_{-4}	6.14_{-4}	6.06_{-4}	6.07_{-4}	6.17_{-4}	6.17_{-4}	6.04_{-4}	6.05_{-4}	6.14_{-4}	6.25_{-4}
gf_f/gf_0	8.80_{-7}	9.23_{-7}	9.09_{-7}	9.11_{-7}	9.27_{-7}	9.27_{-7}	9.07_{-7}	9.09_{-7}	9.22_{-7}	9.39_{-7}
t	9.98_{-1}	9.79_{-1}	9.09_{-1}	9.42_{-1}	1.10	1.08	1.15	1.06	1.02	1.29

default parameters for LRBroyden. The influence of different values of α_0 on the performance of LRBroyden is assessed in the experiments.

To evaluate the performance of two different approaches to update $\phi_i^{(k)}$ for $i = \max\{k - L, 0\}, \cdots, k-1$ discussed in Section 3.5, the Stiefel Brockett problem is tested in this section. The main difference between the two approaches is whether or not to use historical values of $\phi_i^{(k)}$. The results are indicative of the trends observed in both Euclidean and Riemannian problems. In the following experiments, the Stiefel Brockett problem is run 10 times with the same problem setting but varying initial settings for each method with each memory size L. Table 4.1 presents the results for LRBFGS and LRDavidon (see Section 4.4 for details) without historical usage, that is, $\phi_i^{(k)}$ is calculated for each i and k.

Table 4.2 presents the results for LRDavidon with historical usage and LRDavidon with historical usage but adopting the convergence bounds (3.14). At step k, both of them utilize the historical values $\phi_i^{(k)}$ from the previous step $k-1$, where $\phi_i^{(k)} = \phi_i^{(k-1)}$ for $i = \max\{k - L, 0\}, \cdots, k-2$ with only $\phi_{k-1}^{(k)}$ being set or computed at step k. It is worth noting that LRDavidon with historical usage fail to converge when $L = 2, 4$, as indicated in the row "# fail". The average numbers of $\phi_i^{(k)}$ out of the bounds (3.14) are shown as the row "# $\phi_i^{(k)}$ out of bounds/ # total $\phi_i^{(k)}$".

In all the remaining experiments in this dissertation, the $\phi_i^{(k)}$ values in the LRBroyden family are computed or set directly without historical storing. To avoid the failure of convergence, a safeguard for $\phi_i^{(k)}$ is set as follows

$$\phi_i^{(k)} = \begin{cases} 0, & \text{if } \phi_i^{(k)} \leq 0.95\phi_i^{(k)c} \text{ or } \phi_i^{(k)} \geq 0.95 \\ \phi_i^{(k)}, & \text{otherwise.} \end{cases} \tag{4.3}$$

Table 4.2: Comparison of LRDavidon with historical $\phi_i^{(k)}$ vs LRDavidon with historical $\phi_i^{(k)}$ adopting the convergence bounds that $\phi_i^{(k)} = 0$ if $\phi_i^{(k)} \leq 0.95\phi_i^{(k)c}$ or $\phi_i^{(k)} \geq 0.95$ for the Stiefel Brockett problem. The subscript $-k$ indicates a scale of 10^{-k}.

(n,p)	(1000,5)									
method	LRDavidon with historical $\phi_i^{(k)}$					LRDavidon with historical $\phi_i^{(k)}$ adopting (3.14)				
L	1	2	4	8	16	1	2	4	8	16
# fail	-	10	10	7	0	0	0	0	0	0
# $\phi_i^{(k)}$ out of bounds/ # total $\phi_i^{(k)}$	-	-	-	-	-	0/683	246/1325	338/2758	259/4412	248/8312
iter	683	43	100	327	520	683	663	691	555	517
nf	738	99	153	381	532	738	704	712	571	532
ng	684	44	101	328	521	684	664	692	556	518
nR	737	98	152	380	531	737	703	711	570	531
nV	1365	87	199	655	1040	1365	1326	1381	1110	1034
gf_f	6.17_{-4}	2.75_{1}	1.18_{1}	1.66	6.34_{-4}	6.17_{-4}	6.28_{-4}	5.60_{-4}	5.87_{-4}	6.37_{-4}
gf_f/gf_0	9.27_{-7}	4.14_{-2}	1.77_{-2}	2.49_{-3}	9.53_{-7}	9.27_{-7}	9.43_{-7}	8.41_{-7}	8.82_{-7}	9.57_{-7}
t	9.77_{-1}	1.46_{-1}	2.37_{-1}	6.61_{-1}	1.21	10.00_{-1}	9.94_{-1}	1.10	1.02	1.21

The guarded value of $\phi_i^{(k)}$ is set to be the BFGS update since it satisfies the convergence condition and BFGS update is the benchmark in the tests. The utilization of historical $\phi_i^{(k)}$ within LRBroyden does not yield substantial benefits and the convergence guard (4.3) is critical to the robustness of LRBroyden.

4.3 LRBroyden with Constant ϕ

A constant ϕ strategy is the simplest way to select a method in the (L)RBroyden family and potentially achieving effective convergence, e.g., (L)RBFGS method with $\phi = 0$. The first experiment for LRBroyden is to compare the performances when a constant ϕ is selected. Besides the value of constant ϕ, another consideration for the experiments in this section is the initial stepsize α_0. When γ_k is chosen to ensure that the search direction is well scaled, α_0 determines the efficiency of the line search.

For the non-limited-memory Broyden update (2.3) in Euclidean space, Liu and Vander Wiel [32] gave an optimal estimation of the stepsize in the following formula:

$$\hat{\alpha}_0^{(k+1)} = \frac{g_{k+1}^T B_{k+1}^{-1} g_{k+1}}{g_{k+1}^T B_{k+1}^{-1} g_{k+1} - \phi_k (s_k^T B_{k+1} s_k)(g_{k+1}^T B_{k+1}^{-1} v_k)^2}, \tag{4.4}$$

where v_k, B_{k+1} are defined in Equation (2.3) and $g_{k+1} = \text{grad} f(x_{k+1})$. For BFGS ($\phi_k = 0$) update, the optimal estimation is $\hat{\alpha}_0^{(k+1)} = 1$. For Broyden update with a negative parameter ($\phi_k < 0$), the optimal estimation is $\hat{\alpha}_0^{(k+1)} < 1$. For Broyden update with a positive parameter ($\phi_k > 0$), the optimal estimation is $\hat{\alpha}_0^{(k+1)} > 1$. Zhang and Tewarson [61] commented that their algorithms in the

Euclidean Broyden family using negative ϕ_k improve iteration counts but little or no savings are achieved on the number of function evaluations. This is because the initial steps are often too long to provide a sufficient decrease in the function value and a large number of line search iterations result.

The non-limited-memory Broyden does not have the scaling term γ_k in each iterate. However, the findings by Liu and Vander Wiel [32] underscores the importance of α_0 on the performance of different members within Broyden family. Consider the default parameter settings $\gamma_k = \frac{g(s_{k-1},y_{k-1})}{g(y_{k-1},y_{k-1})}$ and $\alpha_0 = 1$ mentioned in Section 4.2, which is a good choice for LRBFGS. If this combination is not suitable for other members in the (L)RBroyden family, many function evaluations may be required to find a suitable value for the stepsize under the Wolfe conditions. The hypothesis here is that without changing the BB scaling $\gamma_k = \frac{g(s_{k-1},y_{k-1})}{g(y_{k-1},y_{k-1})}$, a shortened or lengthened α_0 can improve the performance of (L)RBroyden.

The experiments in this section are designed to illustrate and validate the influence of α_0 on the performance of LRBroyden. For LRBroyden, α_0 is employed after $\mathcal{B}_k^0 \equiv \mathcal{B}_k^l \rightarrow \mathcal{B}_k^{l+1} \rightarrow \cdots \rightarrow \mathcal{B}_k^k$ and $\mathcal{H}_k^0 \equiv \mathcal{H}_k^l \rightarrow \mathcal{H}_k^{l+1} \rightarrow \cdots \rightarrow \mathcal{H}_k^k$ with $l = \max\{k - L, 0\}$, thereby hindering the utilization of the optimal α_0 as proposed by Liu and Vander Wiel [32]. Instead, the value of α_0 is selected as a constant from the set $[-0.5, -0.4, -0.3, \cdots, 0.5]$. Various α_0 are tested to compare performances across different values of a ϕ, where ϕ is selected as a constant from the set $[-0.5, -0.45, -0.4, \cdots, 0.45, 0.5]$. All other parameters are the same as mentioned in Section 4.2. Four cases are tested here: (a) Stiefel Brockett with $n = 1000, p = 1$; (b) Stiefel Brockett with $n = 1000, p = 5$; (c) Low-rank matrix completion with $m = n = 1000, r = 5, OS = 2.5$; (d) Steel Rail Cooling with $n = 3113, r = 5$. Problems are generated by the same procedure as in Section 4.1. Memory size is fixed as $L = 4$.

Figure 4.1 compares the required nf for case (a) Stiefel Brockett with $n = 1000, p = 1$ when various of α_0 are selected with $\phi = -0.25$. It is evident that an α_0 around 0.75 delivers superior performance compared to $\alpha_0 = 1$. Performance deteriorates rapidly when α_0 is either excessively large or exceedingly small. The finding aligns with the results presented by Liu and Vander Wiel [32] and provides the impact of α_0 on the performance of LRBroyden. In the forthcoming experiments, we use the term "best-tuned α_0" to denote the specific value of the constant α_0 that yields the best performance of LRBroyden across the α_0 set with the comparison metric chosen as the number of function evaluations (nf).

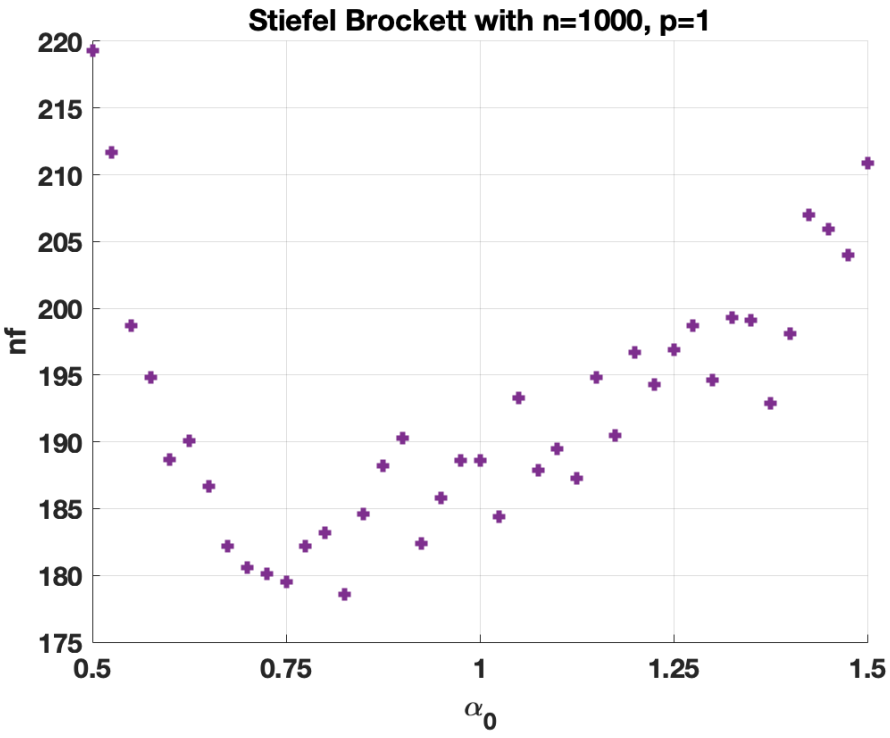


Figure 4.1: $\phi = -0.25$: averaging nf derived from 10 runs versus α_0.

In Figure 4.2, the blue dots represent the best-tuned α_0 when a specific ϕ is selected. These y-axis values are computed by averaging the best-tuned α_0 from 10 runs with identical parameters but different initial conditions. The red plus signs represent the averaging nf values derived from 10 runs with different initial conditions when the specific ϕ and the best-tuned α_0 are selected. In the left blue dots graph, a negative ϕ uses $\alpha_0 < 1$ for better performance while a positive ϕ uses $\alpha_0 > 1$. Notice that α_0 around 1 is proven to be optimal for LRBFGS. This observation aligns with the theoretical conclusion of Liu and Vander Wiel [32] for the Euclidean Broyden update. In the right red plus graph, a closer-to-zero ϕ provides a better performance and LRBFGS keeps its dominance over other constant ϕ.

For a negative constant ϕ, LRBroyden in case (a), (b), (d) demonstrate competitive performance to LRBFGS. Note that ϕ adheres to the convergence safeguard criteria as defined in (4.3) to guarantee convergence. Consequently, some updates use the BFGS update rather than the Broyden family when a negative constant ϕ is employed. Table 4.3 provides the proportions of how many $\phi = 0$ (in the parenthesis) to the total count of ϕ. When ϕ is close to -0.5, all four cases exhibit a substantial occurrence of $\phi = 0$. For cases (a) and (c), the proportion of $\phi = 0$ is diminished

Table 4.3: Number of steps using the BFGS update for cases (a), (b), (c), (d) as depicted in Figure 4.2 when the constant ϕ is negative. The reported numbers represent the total count of ϕ, with the count of instances using the BFGS update enclosed in parentheses due to the convergence safeguard criteria (4.3).

ϕ	-0.5	-0.45	-0.4	-0.35	-0.3	-0.25	-0.2	-0.15	-0.1	-0.05
(a)	586(178)	577(124)	574(119)	578(114)	568(103)	575(88)	572(75)	562(35)	558(15)	532(3)
(b)	2914(1313)	3013(1401)	3034(1189)	2753(1162)	2718(976)	2703(978)	2694(780)	2698(699)	2681(648)	2690(522)
(c)	348(64)	339(59)	343(58)	348(51)	352(40)	331(39)	338(26)	325(17)	318(3)	310(0)
(d)	1454(571)	1434(521)	1386(494)	1374(393)	1326(406)	1337(374)	1218(321)	1318(321)	1242(269)	1249(217)

significantly as ϕ approaches 0. Conversely, for cases (b) and (d), the proportion of $\phi = 0$ is diminished from over 1/3 to less than 1/5 as ϕ ranges from -0.5 to 0.

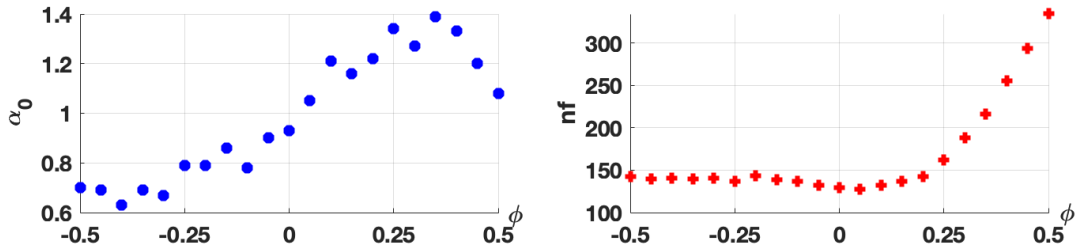
The results in this section reveal that LRBroyden exhibits suboptimal performance when subjected to a positive constant ϕ that deviates from zero. The reasoning behind this is that if ϕ increases further to $\phi = 1$, the algorithm employes the DFP update, known for its lackluster practical performance [38]. On the contrary, adopting a negative constant ϕ in LRBroyden results in frequent utilization of the BFGS updates, owing to the convergence safeguard criteria (4.3) imposed on ϕ. Despite employing a careful selection of α_0, LRBroyden with a constant ϕ struggles to achieve a higher level of robustness compared to LRBFGS. This leads us to consider dynamically choosing of ϕ for a more robust and efficient LRBroyden algorithm, which is disclosed in the next section.

4.4 Davidon's Choice of $\phi_i^{(k)}$

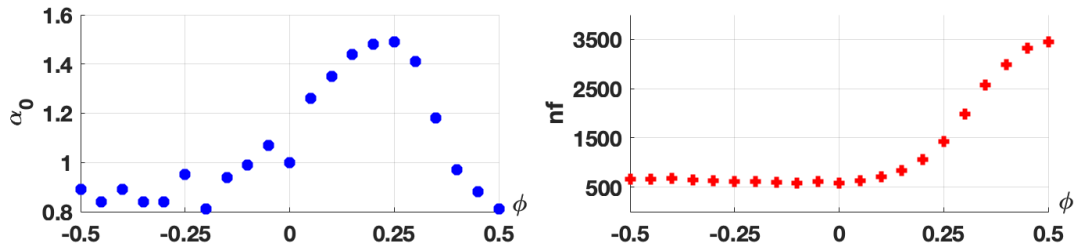
In the preceding section, it was observed that a constant ϕ does not consistently deliver comparable or better performance than LRBFGS. In this section, we explore Davidon's [11] strategy for generating ϕ_k and incorporate it into LRBroyden. Tests are conducted using the synthetic Euclidean quadratic problem the Stiefel Brockett problem, allowing us to compare the performance of LRBFGS and LRBroyden when utilization Davidon's $\phi_i^{(k)}$.

4.4.1 Optimally Conditioned Method

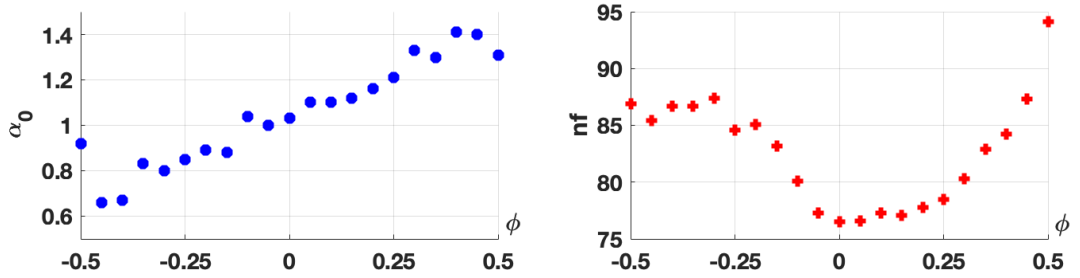
Davidon [11] defines an "optimally conditioned" update for ϕ_k by minimizing the condition number of $\mathcal{B}_k^{-1}\mathcal{B}_{k+1}$. The condition number of a nonsingular matrix A is defined as $\kappa(A) := \|A\|_2\|A^{-1}\|_2$. If A is symmetric and positive definite, $\kappa(A)$ is the ratio of the largest to the smallest eigenvalue of A. Following the update (2.3) in Euclidean form, it has been proved that $\kappa(B_{k+1}) \leq \kappa(B_k) \cdot \kappa(B_k^{-1/2}B_{k+1}B_k^{-1/2})$. Since $B_k^{-1/2}B_{k+1}B_k^{-1/2}$ is a rank two update of the identity, it has



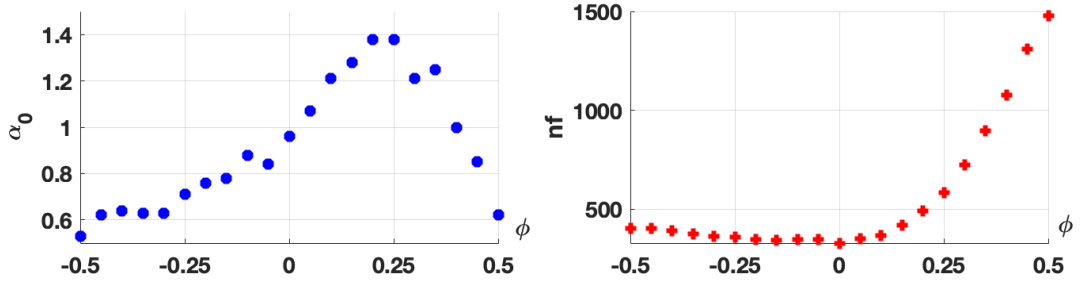
(a) Stiefel Brockett: $n = 1000, p = 1, L = 4$



(b) Stiefel Brockett: $n = 1000, p = 5, L = 4$



(c) Low-rank matrix completion: $m = 1000, n = 1000, r = 5, OS = 2.5, L = 4$



(d) Steel Rail Cooling Problem: $n = 3113, r = 5, L = 4$

Figure 4.2: Performance of LRBroyden with different fixed ϕ values. Left: the best-tuned α_0 versus ϕ; Right: averaging nf versus ϕ.

Table 4.4: Range of ϕ_k^D with different values of a, b, c.

	$\frac{b(a-b)}{ac-b^2}$	$\frac{b}{b-c}$	ϕ_k^D
$b \leq a, c$	$(0,1)$	$< \phi_k^c$	$(0,1)$
$a > b > c$	> 1	> 1	$(1, \phi_k^{SR1}]$
$a < b < c$	$(\phi_k^c, 0)$	$(\phi_k^c, 0)$	$[\phi_k^{SR1}, 0)$

two eigenvalues denoted as $\lambda_\pm(\phi_k)$ and remaining are unit eigenvalues. Minimizing the condition number of $B_k^{-1}B_{k+1}$ is then equivalent to

$$\min_{\phi_k} \frac{\max(\lambda_+(\phi_k), 1)}{\min(\lambda_-(\phi_k), 1)}, \quad \lambda_-(\phi_k) > 0.$$

Given the RBroyden update formulas (3.1) and (3.8), the Riemannian version of the strategy can be generalized as:

$$\phi_k^D = \begin{cases} \frac{g(y_k,s_k)(g(y_k,\tilde{\mathcal{H}}_k y_k)-g(y_k,s_k))}{g(s_k,\tilde{\mathcal{B}}_k s_k)g(y_k,\tilde{\mathcal{H}}_k y_k)-g(y_k,s_k)^2}, & \text{if } g(y_k,s_k) \leq \frac{2g(s_k,\tilde{\mathcal{B}}_k s_k)g(y_k,\tilde{\mathcal{H}}_k y_k)}{g(s_k,\tilde{\mathcal{B}}_k s_k)+g(y_k,\tilde{\mathcal{H}}_k y_k)} \\ \frac{g(y_k,s_k)}{g(y_k,s_k)-g(s_k,\tilde{\mathcal{B}}_k s_k)}, & \text{otherwise.} \end{cases}$$

The first value $\frac{g(y_k,s_k)(g(y_k,\tilde{\mathcal{H}}_k y_k)-g(y_k,s_k))}{g(s_k,\tilde{\mathcal{B}}_k s_k)g(y_k,\tilde{\mathcal{H}}_k y_k)-g(y_k,s_k)^2}$ is equal to $\phi_k^c(1 - g(y_k,\tilde{\mathcal{H}}_k y_k)/g(y_k,s_k))$ and is therefore greater than ϕ_k^c (3.13). The second $\frac{g(y_k,s_k)}{g(y_k,s_k)-g(s_k,\tilde{\mathcal{B}}_k s_k)}$ coincides with the parameter for SR1 update (2.7) and can also be proved to be greater than ϕ_k^c. If $a = g(y_k,\tilde{\mathcal{H}}_k y_k)$, $b = g(y_k,s_k)$, $c = g(s_k,\tilde{\mathcal{B}}_k s_k)$, the two values in the expression ϕ_k^D are $\frac{b(a-b)}{ac-b^2}$ and $\frac{b}{b-c}$ respectively. Table 4.4 shows values of ϕ_k^D in three different situations, where ϕ_k^D is guaranteed to be greater than ϕ_k^c. The only situation that falls outside the bound $(1-\nu)\phi_k^c \leq \phi_k \leq 1-\delta$ in Theorem 3.2 is when $a > b > c$, that $\phi_k^D \in (1, \phi_k^{SR1}]$. However, with the condition $b > c$, the denominator term of the SR1 update (2.7) is guaranteed to be positive. The trace is therefore bounded above which ensures the convergence.

Denote $a = g(y_k,\tilde{\mathcal{H}}_k y_k)$, $b = g(y_k,s_k)$, $c = g(s_k,\tilde{\mathcal{B}}_k s_k)$. The values of a and c can be obtained directly in Algorithm 3 while b is calculated and stored during the updates in Algorithm 4. This optimally conditioned idea of choosing ϕ_k does not require extra computations since it uses parameters that are already computed.

Byrd et al. [8] and Huang et al. [27] show that (R)Broyden with Davidon's ϕ performs better than (R)BFGS in the specific setting where the initial Hessian approximation is set to be a diagonal matrix with one extremely large element. The better performance can be explained by the self-correcting property [10, 8] of the Broyden family. From the determinant of the Hessian

approximation Equation (3.16):

$$\det(\hat{\mathcal{B}}_{k+1}) = \det(\hat{\mathcal{B}}_k)\frac{g(y_k, s_k)}{g(s_k, \tilde{\mathcal{B}}_k s_k)}[1 + \phi_k(\mu_k - 1)],$$

when $g(s_k, \tilde{\mathcal{B}}_k s_k)$ is small compared to $g(y_k, s_k)$, the determinant increases hence some of the eigenvalues of $\hat{\mathcal{B}}_k$ increases. With the fact that $\mu_k \geq 1$ and $1 + \phi_k(\mu_k - 1) > 0$, the ability to correct small eigenvalues is strong when $\phi_k \geq 0$. From the trace of the Hessian approximation Equation (3.17):

$$\text{trace}(\hat{\mathcal{B}}_{k+1}) = \text{trace}(\hat{\mathcal{B}}_k) + \frac{\|y_k\|^2}{g(y_k, s_k)} + \phi_k \frac{\|y_k\|^2}{g(y_k, s_k)} \frac{g(s_k, \tilde{\mathcal{B}}_k s_k)}{g(y_k, s_k)} - (1 - \phi_k) \frac{\|\tilde{\mathcal{B}}_k s_k\|^2}{g(s_k, \tilde{\mathcal{B}}_k s_k)} - 2\phi_k \frac{g(y_k, \tilde{\mathcal{B}}_k s_k)}{g(y_k, s_k)},$$

the fourth term on the right-hand side is the only one that is guaranteed to be negative. Thus, we must rely on this term to reduce the trace of $\hat{\mathcal{B}}_k$. When $\phi_k < 0$, the fourth term remains negative and increases in magnitude and the third term becomes negative, hence the update is more able to correct large eigenvalues. This is consistent with the ϕ_k^D formula that negative ϕ_k^D is selected when $g(y_k, \tilde{\mathcal{H}}_k y_k) < g(y_k, s_k) < g(s_k, \tilde{\mathcal{B}}_k s_k)$ (Table 4.4). When there are large eigenvalues in the Hessian approximation, $g(s_k, \tilde{\mathcal{B}}_k s_k)$ tends to be large and a negative ϕ_k^D is chosen to correct the large eigenvalues.

4.4.2 Experiments in Euclidean Space

In the context of the limited-memory Broyden algorithm in Euclidean space, previous research has not presented conclusive evidence supporting the selection of $\phi_i^{(k)}$ that outperforms the BFGS update, as detailed in Section 2.3. This section is dedicated to designing experiments that aims to showcase the scenarios in which LRBroyden with Davidon's $\phi_i^{(k)}$ (LRDavidon) demonstrates superior perfomrance compared to LRBFGS. Our experimental approach is motivated by the findings of Byrd, Liu and Nocedal [8] and Huang et al. [27] in the context of (R)Broyden.

The first experiment involves solving the quadratic problem in Euclidean space, i.e., $\min_{x \in \mathbb{R}^n} 0.5 x^T A x$. The behavior of the algorithms can be analyzed based on the eigenvalues of the true Hessian A. The Hessian matrix is constructed as $A := QDQ^T$, where Q is obtained by applying Matlab ORTH command to a matrix whose entries are drawn from the standard normal distribution. D is diagonal with elements drawn from the uniform distribution on the interval $(0, 1]$. In the first case, one element in D is set to 1000; in the second case, two elements in D are set to 1000, 999. Both cases have invariable positive definite non-singular Hessian matrices. The initial point x_0 is generated from a standard normal distribution, and α_0 is set to 1 for both LRBFGS and LRDavidon. The

chosen Euclidean geometry for LRDavidon and LRBFGS aligns with the Euclidean LBroyden [12] and the Euclidean LBFGS [33].

The results are presented in Table 4.5 and 4.6. Both cases exhibit a condition number exceeding 10^6. The primary difference in the eigenvalues of A between the two cases lies in the value of the second largest eigenvalue, which is 0.9973 and 999 respectively. In the first case, the largest eigenvalue 1000 is significantly larger than the other eigenvalues. This choice of eigenvalue distribution is inspired by the experiments in [8] and [27] where (R)Broyden with Davidon's ϕ_k demonstrated strong performance. Table 4.5 reveals that LRDavidon outperforms LRBFGS in terms of iterations, function evaluations and time for each value of $L = 4, 8, 16, 32$. In fact, the performance is better by a factor of at least 2 for each L. The best performance in terms of time for LRDavidon ($L = 32$) surpasses that of LRBFGS ($L = 64$).

In Table 4.6, LRDavidon consistently outperforms LRBFGS from $L = 4$ to $L = 32$, but the degree of the superiority is smaller compared to the first case. This aligns with expectation, as the property of an "extremely largest eigenvalue" is diminished due to the eigenvalue 999. The least computational time for LRDavidon ($L = 32$) is greater than for LRBFGS ($L = 128$). For $L = 128$ in the first case and $L = 64, 128$ in the second case, iterations, function evaluations and time are either less than or very close to those of LRBFGS, but the computational time for LRDavidon is greater than for LRBFGS. This discrepancy is attributed to the additional cost of LRDavidon becoming more noticeable when the memory size is large. However, it is important to note that the LRBroyden family of methods are typically considered for large-scale problems, and as such, L should be kept low. In practice, it is important and natural to run with a fixed L as small as possible for large-scale problems. The results in this section support the argument that LRDavidon with a small L has the potential to be competitive or even superior for problems with local Hessians characterized by skewed spectra.

Table 4.5: Comparison of LRBFGS and LRDavidon in Euclidean quadratic with $n = 1000$. The 3 largest and smallest eigenvalues of A are 1000, 0.9973, 0.9969 and 2.6×10^{-3}, 1.8×10^{-3}, 2.6×10^{-4}. The subscript $-k$ indicates a scale of 10^{-k}.

method	LRBFGS							LRDavidon						
L	1	4	8	16	32	64	128	1	4	8	16	32	64	128
iter	1306	855	678	513	326	174	114	1251	479	272	212	136	130	117
nf	1749	1049	813	612	379	194	121	1680	515	289	222	142	136	120
ng	1365	857	679	515	327	176	116	1317	480	273	213	137	132	118
nV	1748	1048	812	611	378	193	120	1679	514	288	221	141	135	119
nR	2669	1711	1356	1027	652	349	229	2567	958	543	424	272	261	234
gf_f	5.53_{-4}	6.30_{-4}	6.09_{-4}	5.92_{-4}	5.64_{-4}	5.97_{-4}	5.91_{-4}	6.21_{-4}	5.71_{-4}	5.89_{-4}	5.57_{-4}	6.00_{-4}	6.19_{-4}	6.05_{-4}
gf_f/gf_0	8.52_{-7}	9.19_{-7}	9.23_{-7}	8.95_{-7}	8.59_{-7}	8.61_{-7}	9.12_{-7}	9.19_{-7}	8.56_{-7}	8.63_{-7}	8.50_{-7}	8.81_{-7}	9.05_{-7}	8.67_{-7}
t	9.95_{-1}	6.16_{-1}	5.06_{-1}	4.08_{-1}	2.96_{-1}	1.84_{-1}	1.91_{-1}	9.43_{-1}	3.13_{-1}	1.90_{-1}	1.66_{-1}	1.35_{-1}	2.22_{-1}	3.96_{-1}

Table 4.6: Comparison of LRBFGS and LRDavidon in Euclidean quadratic with $n = 1000$. The 3 largest and smallest eigenvalues of A are 1000, 999, 0.9971 and 2.4×10^{-3}, 1.7×10^{-3}, 2.4×10^{-4}. The subscript $-k$ indicates a scale of 10^{-k}.

method	LRBFGS							LRDavidon						
L	1	4	8	16	32	64	128	1	4	8	16	32	64	128
iter	819	1297	1080	858	606	300	135	867	1093	726	537	322	222	123
nf	998	1454	1206	963	683	333	142	1055	1173	760	554	333	228	128
ng	827	1299	1082	860	607	302	136	876	1095	728	538	324	223	124
nV	997	1453	1205	962	682	332	141	1054	1172	759	553	332	227	127
nR	1644	2595	2161	1717	1212	601	270	1742	2187	1453	1074	645	444	246
gf_f	1.01_{-3}	9.54_{-4}	9.63_{-4}	9.64_{-4}	9.03_{-4}	8.48_{-4}	9.49_{-4}	1.03_{-3}	9.29_{-4}	9.79_{-4}	9.71_{-4}	8.89_{-4}	9.67_{-4}	9.77_{-4}
gf_f/gf_0	9.33_{-7}	8.92_{-7}	8.95_{-7}	8.97_{-7}	8.68_{-7}	7.99_{-7}	8.72_{-7}	9.57_{-7}	8.68_{-7}	9.19_{-7}	8.67_{-7}	8.36_{-7}	8.95_{-7}	8.98_{-7}
t	6.42_{-1}	9.92_{-1}	8.00_{-1}	6.96_{-1}	5.74_{-1}	3.56_{-1}	1.96_{-1}	6.51_{-1}	7.81_{-1}	5.60_{-1}	4.40_{-1}	3.50_{-1}	4.54_{-1}	4.51_{-1}

4.4.3 Experiments on Riemannian Manifold

The experiments on Riemannain manifold are motivated by those in Euclidean space. On a Riemannian manifold, the problem of Brockett cost function on the Stiefel Manifold (4.1) is constructed with $A := QDQ^T$. Q is obtained by applying Matlab ORTH command to a matrix whose entries are drawn from the standard normal distribution. In the cases presented in Table 4.7 and 4.9, D is diagonal with one element equal to 100 and others from the uniform distribution on the interval $(0, 1]$. In the cases presented in Table 4.8 and 4.10, D is diagonal with all elements drawn from the uniform distribution on the interval $(0, 1]$. $N = \text{diag}(1, \cdots, p-1, p)$ is a diagonal matrix whose diagonal elements being integers from 1 to p.

It can be observed that LRDavidon outperforms LRBFGS in Table 4.7 and 4.9, where the largest eigenvalue significantly surpasses the others. For each value of $L = 4, 8, 16, 32$ in Table 4.7 and $L = 32, 64$ in Table 4.9, LRDavidon's performance is better by a factor of at least 2. Furthermore,

Table 4.7: Comparison of LRBFGS and LRDavidon for the Stiefel Brockett problem with $n = 1000,\ p = 1,\ D$ is diagonal with one element equal to 100 and others from the uniform distribution on the interval $(0, 1]$. The 3 largest and smallest eigenvalues of Hessian at the solution are $200,\ 1.997,\ 1.996$ and $4.7 \times 10^{-3},\ 1.2 \times 10^{-3},\ 3.5 \times 10^{-4}$. The subscript $-k$ indicates a scale of 10^{-k}.

method	LRBFGS							LRDavidon						
L	1	4	8	16	32	64	128	1	4	8	16	32	64	128
iter	2200	1282	802	580	399	272	216	1942	633	339	256	222	217	204
nf	2785	1503	931	670	456	298	231	2451	695	364	269	230	223	209
ng	2212	1284	803	582	400	273	218	1953	634	341	258	223	218	206
nV	2784	1502	930	669	455	297	230	2450	694	363	268	229	222	208
nR	4411	2565	1604	1161	798	544	433	3894	1266	679	513	444	434	409
gf_f	5.79_{-6}	5.88_{-6}	5.69_{-6}	5.95_{-6}	5.55_{-6}	5.02_{-6}	5.73_{-6}	5.91_{-6}	6.13_{-6}	5.47_{-6}	5.28_{-6}	5.80_{-6}	5.73_{-6}	5.52_{-6}
gf_f/gf_0	8.72_{-7}	8.83_{-7}	8.89_{-7}	9.01_{-7}	8.42_{-7}	7.95_{-7}	8.65_{-7}	9.00_{-7}	9.46_{-7}	8.31_{-7}	8.14_{-7}	8.91_{-7}	8.65_{-7}	8.50_{-7}
t	1.74	9.75_{-1}	6.33_{-1}	4.92_{-1}	3.92_{-1}	3.11_{-1}	3.22_{-1}	1.54	4.73_{-1}	2.73_{-1}	2.33_{-1}	2.41_{-1}	4.25_{-1}	1.14

Table 4.8: Comparison of LRBFGS and LRDavidon for the Stiefel Brockett problem with $n = 1000,\ p = 1,\ D$ is diagonal with all elements from the uniform distribution on the interval $(0, 1]$. The 3 largest and smallest eigenvalues of Hessian at the solution are $1.993,\ 1.991,\ 1.990$ and $5.6 \times 10^{-3},\ 3.5 \times 10^{-3},\ 1.6 \times 10^{-3}$. The subscript $-k$ indicates a scale of 10^{-k}.

method	LRBFGS							LRDavidon						
L	1	4	8	16	32	64	128	1	4	8	16	32	64	128
iter	271	220	200	197	180	163	160	275	256	226	203	190	170	166
nf	286	230	207	202	185	168	164	293	258	227	204	191	171	167
ng	272	221	201	198	181	164	161	276	257	227	204	191	171	167
nV	285	229	206	201	184	167	163	292	257	226	203	190	170	166
nR	542	440	400	394	360	325	319	550	513	453	406	379	339	331
gf_f	4.87_{-7}	4.61_{-7}	5.03_{-7}	4.97_{-7}	5.06_{-7}	4.61_{-7}	4.74_{-7}	4.73_{-7}	5.15_{-7}	4.93_{-7}	5.21_{-7}	5.31_{-7}	5.13_{-7}	5.06_{-7}
gf_f/gf_0	8.57_{-7}	8.12_{-7}	8.85_{-7}	8.75_{-7}	8.91_{-7}	8.11_{-7}	8.33_{-7}	8.32_{-7}	9.07_{-7}	8.68_{-7}	9.17_{-7}	9.34_{-7}	9.04_{-7}	8.90_{-7}
t	2.31_{-1}	2.24_{-1}	2.20_{-1}	2.25_{-1}	2.23_{-1}	2.19_{-1}	2.24_{-1}	2.33_{-1}	2.36_{-1}	2.29_{-1}	2.34_{-1}	2.36_{-1}	2.79_{-1}	3.55_{-1}

in Table 4.9, LRDavidon remains competitive with LRBFGS for $L = 4, 8$. These observations align with the experimental findings in Euclidean space when employing a low value of L.

In Table 4.8 and 4.10, LRDavidon is competitive to LRBFGS when L is maintained at a low value. It is worth noting that the computational time for LRDavidon increases at a faster rate than LRBFGS when $L = 64, 128$, consistent with observations made for the Euclidean quadratic problems. The limited-memory characteristics of these methods are recommended for enhancing robustness in dealing with large-scale problems. Choosing L as small as possible for LRBroyden should always be kept in mind. The choices of $L = 32, 64, 128$ tested in this section primarily serve to observe the behavior of each algorithm. However, it is worth noting that such large memory size selections are seldom employed or taken into consideration in practical applications.

Table 4.9: Comparison of LRBFGS and LRDavidon for the Stiefel Brockett problem with $n = 1000,\ p = 5,\ D$ is diagonal with one element equal to 100 and others from the uniform distribution on the interval $(0, 1]$. The 3 largest and smallest eigenvalues of Hessian at the solution are $1000,\ 800,\ 600$ and $8.9 \times 10^{-4},\ 7.9 \times 10^{-4},\ 1.4 \times 10^{-5}$. The subscript $-k$ indicates a scale of 10^{-k}.

method	LRBFGS							LRDavidon						
L	1	4	8	16	32	64	128	1	4	8	16	32	64	128
iter	10896	9121	7777	6345	4225	2827	1787	9523	8985	8138	4904	1903	1213	1006
nf	11949	9740	8260	6784	4573	3084	1937	10442	9216	8334	5012	2059	1269	1026
ng	10897	9122	7778	6346	4226	2828	1788	9524	8986	8139	4905	1904	1214	1007
nV	11948	9739	8259	6783	4572	3083	1936	10441	9215	8333	5011	2058	1268	1025
nR	21792	18243	15553	12689	8451	5655	3574	19045	16970	16277	9807	3807	2426	2013
gf_f	1.61_{-4}	4.31_{-5}	4.19_{-5}	4.41_{-5}	4.25_{-5}	4.23_{-5}	4.32_{-5}	4.87_{-5}	4.76_{-5}	4.14_{-5}	4.36_{-5}	4.23_{-5}	4.35_{-5}	4.37_{-5}
gf_f/gf_0	7.45_{-6}	9.77_{-7}	9.08_{-7}	9.60_{-7}	9.16_{-7}	9.16_{-7}	9.22_{-7}	1.13_{-6}	9.91_{-7}	9.01_{-7}	9.29_{-7}	9.18_{-7}	9.38_{-7}	9.48_{-7}
t	1.59_{1}	1.42_{1}	1.34_{1}	1.36_{1}	1.22_{1}	1.24_{1}	1.32_{1}	1.41_{1}	1.45_{1}	1.43_{1}	1.08_{1}	6.24	7.15	1.47_{1}

Table 4.10: Comparison of LRBFGS and LRDavidon for the Stiefel Brockett problem with $n = 1000,\ p = 5,\ D$ is diagonal with all elements from the uniform distribution on the interval $(0, 1]$. The 3 largest and smallest eigenvalues of Hessian at the solution are $9.965,\ 9.953,\ 9.951$ and $9.2 \times 10^{-4},\ 8.3 \times 10{-}4,\ 4.5 \times 10^{-4}$. The subscript $-k$ indicates a scale of 10^{-k}.

method	LRBFGS							LRDavidon						
L	1	4	8	16	32	64	128	1	4	8	16	32	64	128
iter	1596	1304	1155	1200	1080	1044	934	1631	1446	1190	1230	1193	1143	974
nf	1714	1360	1180	1225	1104	1064	951	1748	1455	1192	1231	1194	1145	976
ng	1598	1305	1156	1201	1081	1045	935	1632	1447	1191	1231	1194	1144	975
nV	1713	1359	1179	1224	1103	1063	950	1747	1454	1191	1230	1193	1144	975
nR	3193	2609	2310	2400	2161	2087	1869	3262	2893	2380	2459	2385	2286	1948
gf_f	3.64_{-6}	3.87_{-6}	3.96_{-6}	4.00_{-6}	3.76_{-6}	4.03_{-6}	3.98_{-6}	3.90_{-6}	2.71_{-3}	3.98_{-6}	4.05_{-6}	4.06_{-6}	4.03_{-6}	4.09_{-6}
gf_f/gf_0	8.61_{-7}	9.15_{-7}	9.35_{-7}	9.45_{-7}	8.89_{-7}	9.53_{-7}	9.41_{-7}	9.21_{-7}	6.20_{-4}	9.42_{-7}	9.57_{-7}	9.60_{-7}	9.53_{-7}	9.67_{-7}
t	4.30_{-1}	4.24_{-1}	4.32_{-1}	4.27_{-1}	4.32_{-1}	4.45_{-1}	4.50_{-1}	4.39_{-1}	4.35_{-1}	4.36_{-1}	4.54_{-1}	4.59_{-1}	4.85_{-1}	7.65_{-1}

4.4.4 Experiments on $\phi_i^{(k)}$ Distribution

This section leverages the self-correcting property to offer an interpretation of the preceding observations. Distributions of Davidon's $\phi_i^{(k)}$, denoted as $\phi_i^{(k)D}$, are illustrated in Figures 4.3, 4.4 and 4.5 in one typical run for each problem. The first two figures depict scenarios where LRDavidon demonstrates significant superiority, while the third figure showcases the scenario where LRDavidon exhibits similar performance to LRBFGS. While the mean of the distribution of $\phi_i^{(k)D}$ is close to 0, the actual distribution deviates from 0. The existence of a range of $\phi_i^{(k)D}$ values, some significantly different from 0, and the observed performance in the experiments indicate that the LRDavidon update does indeed differ from the LRBFGS update.

For the Euclidean quadratic problem depicted in Figure 4.3, where LRDavidon demonstrates significant superiority, the distribution of $\phi_i^{(k)D}$ is centered at 0 with a negativity bias. The mean value of γ_k is approximately 0.6191, signifying the scaling of the initial inverse Hessian approximation, given that $\mathcal{H}_k^0 = \gamma_k \mathrm{id}$. In other words, $\mathcal{B}_k^0$ is scaled by approximately $1/\gamma_k \approx 1.6152$, a value that greater than all eigenvalues except the largest one (see the caption in Table 4.5). Figure 4.4 demonstrates similar outcomes. In another Riemmanian case that LRDavidon exhibits similar performance to LRBFGS, illustrated in Figure 4.5, the distribution of $\phi_i^{(k)D}$ noticeably shifts towards the right, with a greater concentration of positive values compared to negative ones.

Roughly speaking, when an extremely large eigenvalue exists in the Hessian matrix approaching the solution, $1/\gamma_k$ that scales the initial Hessian approximation of LRBroyden tends to be larger compared to the remaining eigenvalues. By introducing negative $\phi_i^{(k)D}$, LRDaviden enhances its capacity to effectively address the impact of these dominant eigenvalues more than LRBFGS does. As a result, this enhancement contributes to the overall improvement in the performance of LRBroyden. The degree of superiority weakens as the skewness diminishes, but LRDavidon remains competitive when the memory size L is maintained at a relatively small value. This is particularly relevant for problems featuring a high-dimensional feasible set, where maintaining a small memory size is necessary.

The last experiment in this chapter aims to illustrate the outstanding quality of $\phi_i^{(k)}$ values for the BFGS and the Davidon updates. A perturbation is added to the value of $\phi_i^{(k)}$ for LRBFGS and LRDavidon methods, adding or subtracting a random number in the range of $(0, 0.01]$. The methods with the randomness in the $\phi_i^{(k)}$ are denoted as "Perturbation LRBFGS" and "Perturbation LRDavidon" in Figure 4.6. The settings of the test problem are the same as the synthetic Stiefel

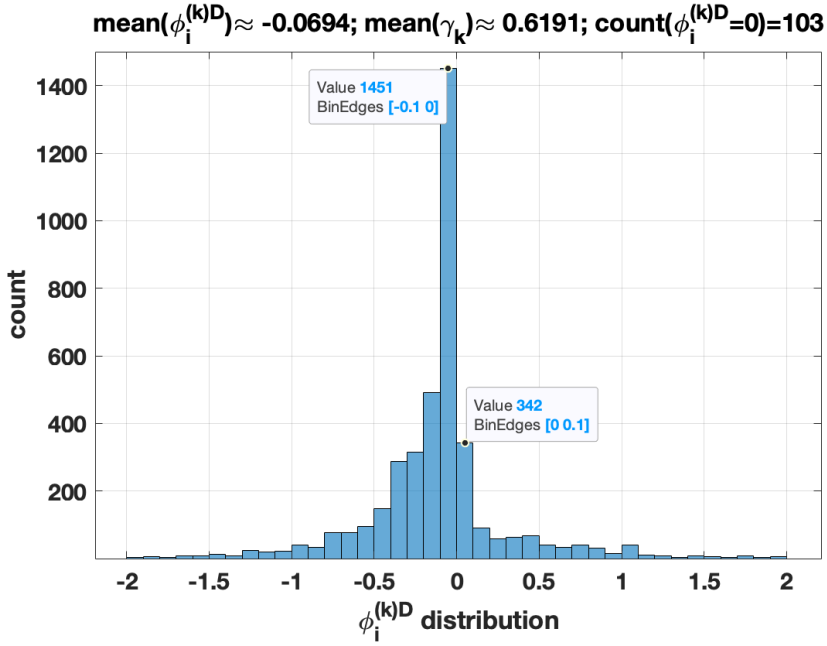


Figure 4.3: $\phi_i^{(k)D}$ distribution for the Euclidean quadratic problem (Table 4.5) with $L = 32$.

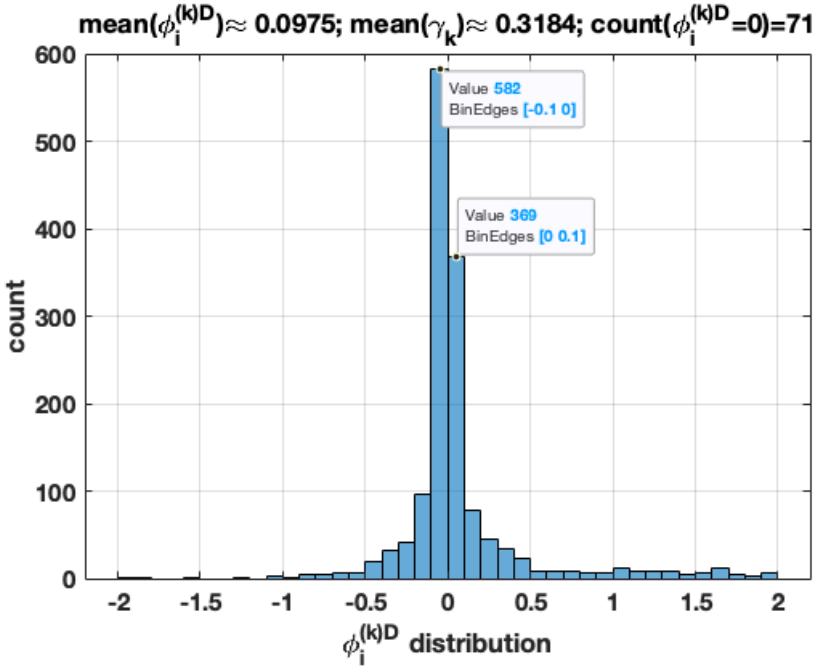


Figure 4.4: $\phi_i^{(k)D}$ distribution for the Stiefel Brockett problem (Table 4.7) with $L = 4$.

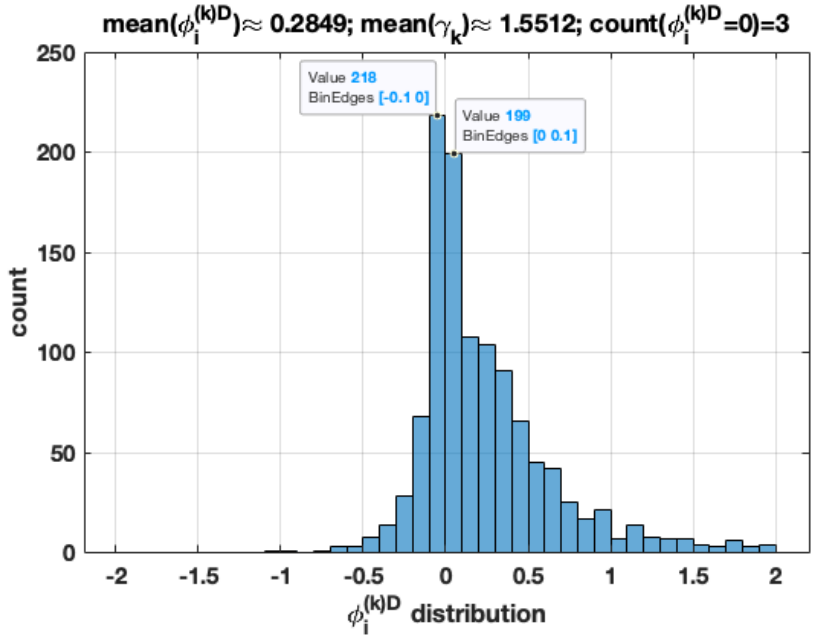


Figure 4.5: $\phi_i^{(k)D}$ distribution for the Stiefel Brockett problem (Table 4.8) with $L = 4$.

Brockett problem in Table 4.9. The choices of BFGS and Davidon for $\phi_i^{(k)}$ in the LRBroyden family of methods have clearly shown their good performance. However, their performance degrades noticeably when $\phi_i^{(k)}$ is perturbed. The use of a perturbation in these experiments indicates that this degradation occurs even when $\phi_i^{(k)}$ is in a neighborhood of 0 or $\phi_i^{(k)D}$. Therefore, we conclude that a variable $\phi_i^{(k)}$ is crucial to the robust performance of the limited-memory RBroyden family, especially when L is kept relatively small for high-dimensional problems.

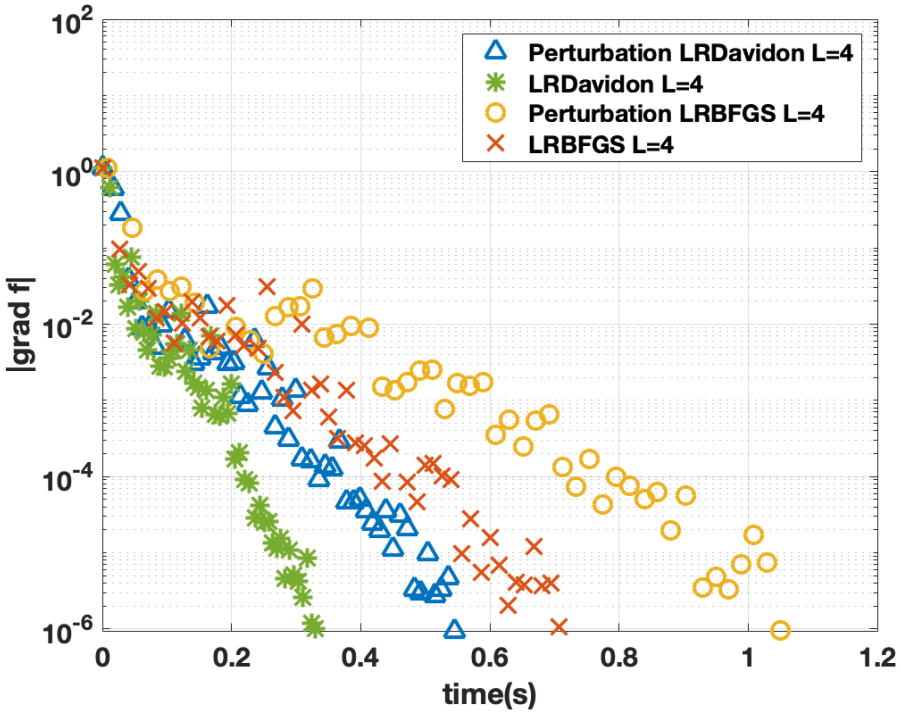


Figure 4.6: Comparison of LRBFGS and LRDavidon with perturbation in $\phi_i^{(k)}$ for the Stiefel Brockett problem case in Table 4.7.

CHAPTER 5

RIEMANNIAN HYBRID LRDAVIDON-BFGS METHOD

This chapter delves further into the study of the LRBroyden family of methods. Motivated by the robustness exhibited by LRBFGS and LRDavidon within the entire family, we explore a hybrid strategy for choosing $\phi_i^{(k)}$ between the options of Davidon and BFGS in Section 5.1. The synthetic Euclidean quadratic problem tested in Table 4.5 is used to illustrate the advantage of the hybrid method. Then in Section 5.2, more general matrix problems are tested for comparison of LRDavidon and LRBFGS. The results demonstrate that LRDavidon is generally competitive for a wide range of problems. Section 5.3 focuses on determining the parameter that controls the balance between the choices in the hybrid strategy for each problem. Simultaneously, empirical investigation is used to determine the initial stepsize for the optimal performance of the hybrid method. In Section 5.4, the hybrid strategy with heuristically selected parameters is shown empirically to achieve more robust performance than LRBFGS and LRDavidon for general matrix problems.

5.1 Hybrid LRDavidon-BFGS Strategy for $\phi_i^{(k)}$

LRBFGS with $\phi = 0$ outperforms other members of LRBroyden family that use a constant ϕ. Previous findings have demonstrated that the dynamic choice, LRDavidon with $\phi_i^{(k)D}$, surpasses LRBFGS in synthetic problems. For enhanced robustness, it makes sense to introduce a hybrid method that can inherit the advantages from both choices. This section introduces a strategy by adapting $\phi_i^{(k)}$ between the choices of Davidon and BFGS.

As shown in Section 4.4, Davidon's ϕ_k^D is determined by minimizing the condition number of $\mathcal{B}_k^{-1}\mathcal{B}_{k+1}$. However, the condition number of Davidon's update may be very close to that of BFGS. Therefore, it is natural to propose an algorithm that selects the Davidon's update only on the iterations that the optimal condition number is significantly smaller than the BFGS, and selects the BFGS otherwise. Based on the discussion in [53], the formula for the hybrid strategy in LRBroyden, denoted as ϕ_k^H, has the following expression :

$$\phi_k^H = \begin{cases} \phi_k^D, & \text{if } \kappa(\text{BFGS}) > \delta\ \kappa(\text{Davidon}) \\ 0, & \text{otherwise,} \end{cases}$$

where κ represents the condition number of $\mathcal{B}_k^{-1}\mathcal{B}_{k+1}$ at each update and $\delta \geq 1$ is a constant. Denote $a :\equiv g(y_k, \tilde{\mathcal{H}}_k y_k)$, $b :\equiv g(y_k, s_k)$ and $c :\equiv g(s_k, \tilde{\mathcal{B}}_k s_k)$, the formula for κ(BFGS) and κ(Davidon) are generalized to Rimannian manifold (see [53] for the expression in Euclidean space):

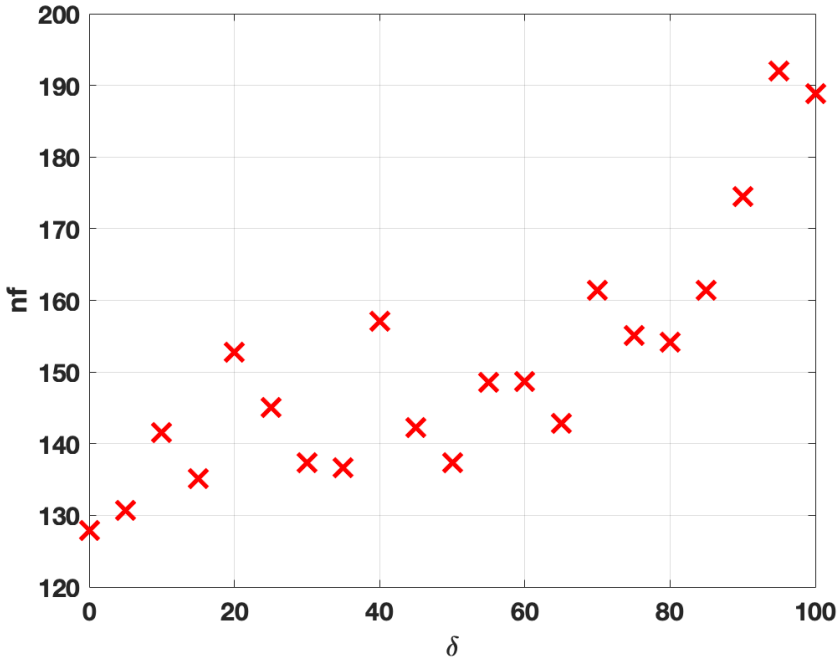
$$\kappa(\text{BFGS}) = [(a+b)^2 c - 2b^3 + (a+b)\sqrt{(a+b)^2c^2 - 4b^3 c}]/2b^3,$$

$$\kappa(\text{Davidon}) = \begin{cases} [2ac - b^2 + 2\sqrt{a^2c^2 - ab^2c}]/b^2, & \text{if } 2ac > b(a+c) \\ (a-b)/(b-c), & \text{if } 2ac \leq b(a+c), a > b \\ (c-b)/(b-a), & \text{if } 2ac \leq b(a+c), b > a, \end{cases}$$

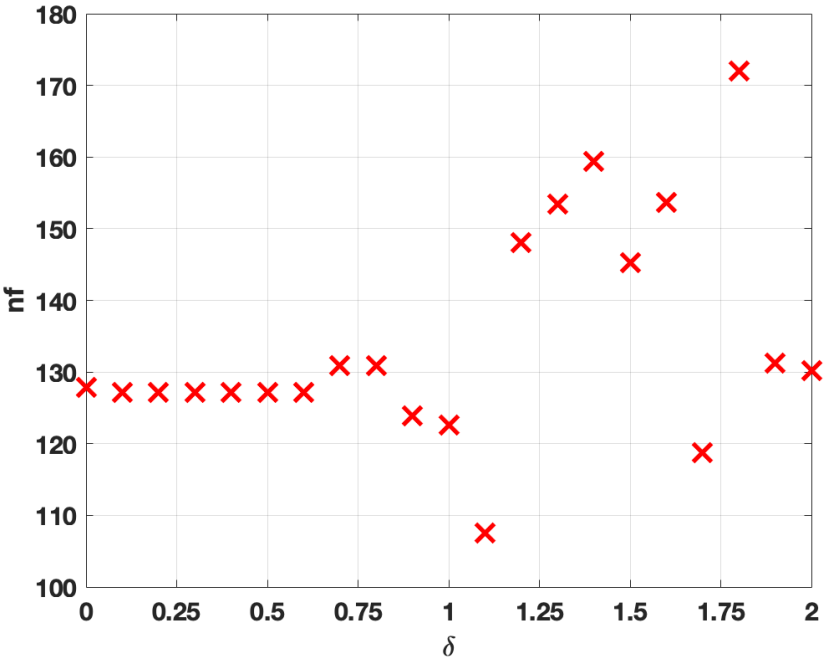
where a, b, c can be obtained directly during the updates in Algorithm 3. The hybrid method uses the BFGS update when κ(Davidon) is not obviously smaller than κ(BFGS). With a relatively large value of the constant δ, the hybrid method uses the BFGS update most of the time and the Davidon's update the rest.

By adopting the the hybrid strategy into the LRBroyden family, denoted as Hybrid LRDavidon-BFGS, we conduct an initial experiment on the synthetic Euclidean quadratic problem (refer to Table 4.5), where LRDavidon is observed to significantly outperform LRBFGS. The experiment is conducted to illustrate the effectiveness of the hybrid method in comparison to using only the BFGS or Davidon update individually.

Results for different values of δ are presented in Figure 5.1, with the memory size L fixed at 32. The red crosses represent the average number of function evaluations (nf) obtained from 10 runs using specific values of δ and different initial conditions. Notably, the number of function evaluations of LRBFGS is 379 and the number of function evaluations of LRDavidon is 142 as specified in Table 4.5. In the larger broader of δ depicted in graph (a), there is a clear trend that larger δ values lead to higher values of nf. This observation is reasonable since the hybrid method selects more BFGS update (use notation $\phi_i^{(k)\text{BFGS}}$ for $\phi_i^{(k)} = 0$) as δ increases. Despite the superior performance of $\phi_i^{(k)D}$ in the certain case, the results shown in graph (b) confirm that occasionally choosing $\phi_i^{(k)\text{BFGS}}$ is indeed beneficial and plays a helpful role in the hybrid method. The use of the hybrid strategy with δ selected from a range between 1 and 2 results in fewer function evaluations compared to using $\delta = 0$, i.e., LRDavidon. This conclusion suggests that, at the very least, the hybrid strategy does not degrade performance. With a well-suited choice of δ, the hybrid strategy can actually enhance the performance.



(a) Euclidean Quadratic: δ in range $(0, 100)$



(b) Euclidean Quadratic: δ in range $(0, 2)$

Figure 5.1: Performance of Hybrid LRDavidon-BFGS: averaging nf versus δ.

5.2 LRDavidon on General Problems

Before delving further into the exploration of the hybrid strategy, we concentrate on general problems on a Riemannian manifold, generated using the procedure outlined in Section 4.1. The memory size is selected from the set $\{1, 2, 4, 8, 16\}$ and the initial stepsize is set to $\alpha_0 = 1$. Subsequent sections will consider different initial stepsizes for LRDavidon.

The results are presented in Table 5.1-5.4. For both LRBFGS and LRDavidon, it appears that the best-performing memory size is either $L = 4$ or 8 for the first three problems, and $L = 1$ for the weighted low-rank approximation problem. The observations, which demonstrate better performance with a relatively small memory size, are consistent with the discussions and insights provided in Section 4.4.

LRBFGS requires fewer iterations and less computational time in some cases, in line with the conclusions drawn in [8] and [27], which suggest that Davidon's update does not consistently outperform BFGS update in general. Nevertheless, LRDavidon remains competitive and does not significantly hinder the overall performance of the method. This motivates us to apply the hybrid strategy to these general problems.

Another noteworthy observation is that LRDavidon requires fewer function evaluations even when it demands more iterations than LRBFGS, as observed in cases such as $L = 8$ in Table 5.2 and $L = 8$ in Table 5.3. The reason behind this is that LRDavidon directly accepts more unit stepsizes, while these unit stepsizes are often too large for LRBFGS, resulting in additional function evaluations during the line search process. The results and discussions in Section 4.3 suggest that a larger stepsize may be more suitable for LRDavidon, which explains its better performance in terms of function evaluations.

Table 5.1: Comparison of LRBFGS and LRDavidon for the Stiefel Brockett problem. The subscript $-k$ indicates a scale of 10^{-k}.

(n,p)	(1000,5)									
method	LRBFGS					LRDavidon				
L	1	2	4	8	16	1	2	4	8	16
iter	629	589	529	488	469	683	693	609	523	525
nf	681	630	564	514	493	738	740	623	538	541
ng	631	590	530	489	470	684	694	610	524	526
nV	680	629	563	513	493	737	739	622	537	540
nR	1259	1177	1059	976	939	1365	1386	1218	1045	1050
gf_f	5.86_{-4}	6.14_{-4}	6.06_{-4}	6.07_{-4}	6.17_{-4}	6.17_{-4}	6.04_{-4}	6.05_{-4}	6.14_{-4}	6.25_{-4}
gf_f/gf_0	8.80_{-7}	9.23_{-7}	9.09_{-7}	9.11_{-7}	9.27_{-7}	9.27_{-7}	9.07_{-7}	9.09_{-7}	9.22_{-7}	9.39_{-7}
t	9.98_{-1}	9.79_{-1}	9.09_{-1}	9.42_{-1}	1.10	1.08	1.15	1.06	1.02	1.29

Table 5.2: Comparison of LRBFGS and LRDavidon for the low-rank matrix completion problem. The subscript $-k$ indicates a scale of 10^{-k}.

(m,n,r,OS)	(1000,1000,5,2.5)									
method	LRBFGS					LRDavidon				
L	1	2	4	8	16	1	2	4	8	16
iter	84	81	75	71	72	84	85	81	72	72
nf	93	91	83	80	80	93	97	87	78	78
ng	89	86	80	76	77	89	90	86	77	77
nV	92	90	82	79	79	92	96	86	77	77
nR	172	166	153	147	148	172	174	166	147	149
gf_f	5.63_{-5}	4.96_{-5}	4.89_{-5}	5.62_{-5}	4.51_{-5}	5.63_{-5}	5.57_{-5}	5.05_{-5}	5.66_{-5}	5.58_{-5}
gf_f/gf_0	8.89_{-7}	7.85_{-7}	7.73_{-7}	8.88_{-7}	7.12_{-7}	8.89_{-7}	8.80_{-7}	7.96_{-7}	8.95_{-7}	8.82_{-7}
t	3.97_{-1}	3.93_{-1}	3.55_{-1}	3.86_{-1}	4.15_{-1}	3.98_{-1}	4.02_{-1}	3.92_{-1}	3.87_{-1}	4.23_{-1}

Table 5.3: Comparison of LRBFGS and LRDavidon for the Steel Rail cooling problem. The subscript $-k$ indicates a scale of 10^{-k}.

(n,r)	(3113,5)									
method	LRBFGS					LRDavidon				
L	1	2	4	8	16	1	2	4	8	16
iter	373	344	325	358	381	374	385	367	363	361
nf	397	373	345	389	421	397	413	374	373	372
ng	376	347	328	361	385	377	388	369	365	363
nV	396	372	344	388	420	396	412	373	372	371
nR	748	690	652	718	766	750	772	735	727	723
gf_f	1.38_{-6}	1.38_{-6}	1.35_{-6}	1.39_{-6}	1.38_{-6}	1.37_{-6}	1.41_{-6}	1.39_{-6}	1.43_{-6}	1.29_{-6}
gf_f/gf_0	9.15_{-7}	9.19_{-7}	8.97_{-7}	9.20_{-7}	9.17_{-7}	9.11_{-7}	9.36_{-7}	9.21_{-7}	9.50_{-7}	8.58_{-7}
t	1.45	1.31	1.14	1.43	1.64	1.48	1.58	1.32	1.60	1.71

Table 5.4: Comparison of LRBFGS and LRDavidon for the weighted low-rank approximation problem. The subscript $-k$ indicates a scale of 10^{-k}.

(m,n,r)	(100,20,5)									
method	LRBFGS					LRDavidon				
L	1	2	4	8	16	1	2	4	8	16
iter	45	46	45	44	44	45	48	42	42	41
nf	47	50	50	49	49	47	52	44	45	44
ng	46	48	48	48	47	46	50	44	44	44
nV	46	49	49	48	48	46	51	43	44	43
nR	90	94	92	91	90	90	97	85	85	84
gf_f	2.67_{-4}	2.45_{-4}	2.00_{-4}	2.41_{-4}	2.32_{-4}	2.67_{-4}	2.83_{-4}	2.79_{-4}	2.98_{-4}	2.74_{-4}
gf_f/gf_0	7.24_{-7}	6.59_{-7}	5.39_{-7}	6.50_{-7}	6.25_{-7}	7.24_{-7}	7.72_{-7}	7.58_{-7}	8.11_{-7}	7.41_{-7}
t	1.90_{-1}	2.01_{-1}	2.31_{-1}	2.89_{-1}	3.68_{-1}	1.90_{-1}	2.09_{-1}	2.07_{-1}	2.70_{-1}	3.69_{-1}

5.3 Parameter Selection of Hybrid LRDavidon-BFGS

This section is dedicated to the exploration and determination of the parameters for Hybrid LRDavidon-BFGS. Two key parameters that require pre-determination are α_0, other than the unit length, and δ, which governs the balance of the hybrid strategy. Subsequently, the performance of the hybrid method, considering various choices of these parameters, is compared to LRBFGS and LRDavidon for further investigation.

The parameters δ is selected from the range $\{1, 1.05, 1.1, \cdots, 1.95, 2\}$ and α_0 is chosen from $\{0.5, 0.6, \cdots, 1.4, 1.5\}$. Other problem settings remain consistent with those in the tests conducted in Section 5.2. In line with the findings in Tables 5.1-5.4, the memory size is fixed at 4. The weighted low-rank approximation problem is excluded from this testing, as the hybrid strategy does not yield differences when $L = 1$ (the best-tuned memory size in Table 5.4).

The results comparing the performances of LRBFGS, LRDavidon and Hybrid LRDavidon-BFGS are presented In Figure 5.2. In these graphs, the blue dots represent the best-tuned α_0 for different values of δ within the hybrid strategy. The values of α_0 are calculated by averaging results from 10 runs with identical parameters but different initial conditions. The red Plus signs represent the average number of function evaluations from 10 runs with different initial conditions when the specific δ and the best-tuned α_0 are selected. Additionally, the magenta, cyan, and green vertical dashed lines indicate the number of function evaluations for LRBFGS, LRDavidon with $\alpha_0 = 1$, and LRDavidon with the appropriate α_0 values (the selected α_0 values are determined in tests similar to those in Section 4.3), respectively.

From the left-hand side α_0 v.s. δ graphs, it is evident that α_0 decreases and approaches 1 as δ increases. The trend is reasonable since a larger δ means the method is closer to LRBFGS, making the best-tuned initial stepsize closer to unit length. From the right-hand side nf v.s. δ graphs, it is clear that using an adaptive α_0 can enhance the performance of LRDavidon and make it more competitive with LRBFGS. The hybrid strategy is observed to require fewer function evaluations than LRBFGS for δ within the range $[1, 2]$. This demonstrates that the hybrid strategy is a suitable choice within the LRBroyden family of methods.

Denote $\phi_i^{(k)H}$ as $\phi_i^{(k)}$ chosen in hybrid strategy, we present the histogram of $\phi_i^{(k)D}$ and $\phi_i^{(k)H}$ distribution in Figure 5.3, 5.4, 5.5 for the three problems. These figures display the distribution as well as the mean values of all used $\phi_i^{(k)D}$ and $\phi_i^{(k)H}$ in one typical run for each problem. There is a noticeable trend that $\phi_i^{(k)H}$ shifts to the left compared to $\phi_i^{(k)D}$, and the mean value of $\phi_i^{(k)H}$ is smaller than the mean value of $\phi_i^{(k)D}$. Furthermore, the proportions of how many $\phi_i^{(k)D}$ and $\phi_i^{(k)H}$ values are equal to zero clearly indicate that there are significantly more $\phi_i^{(k)H}$ values equal to zero compared to $\phi_i^{(k)D}$, which aligns with the desired balance in the hybrid strategy.

By combining the results in Figure 5.2 with the observations in Figure 5.3, 5.4 and 5.5, it is evident that the hybrid strategy enhances performance, particularly for the Stiefel Brockett problem. Notably, more than 1/3 of $\phi_i^{(k)H}$ values are equal to zero as shown in Figure 5.3 and the distribution of $\phi_i^{(k)H}$ is more concentrated around 0.

In the case of the low-rank matrix completion problem, the superiority of the hybrid strategy appears to be less pronounced compared to the other two problems. The observation is supported by Figure 5.4, which shows that the proportion of $\phi_i^{(k)H} = 0$ is not significantly different from $\phi_i^{(k)D} = 0$. Additionally, the superiority of $\phi_i^{(k)H}$ diminishes as δ increases, with the larger δ making the algorithm closer to LRBFGS. These observations suggest that $\phi_i^{(k)\text{BFGS}}$ does not provide much assistance in the hybrid strategy for the low-rank matrix completion problem.

For the Steel Rail problem, over half of $\phi_i^{(k)H}$ values are equal to zero as indicated in Figure 5.5. There is a slight trend indicating that the superiority of $\phi_i^{(k)H}$ becomes more pronounced as δ increases from 1 to 2. This is in line with the observation that LRDavidon, even with an appropriate α_0, cannot reach the same level of performance as LRBFGS as shown in Table 5.3. The mean value of $\phi_i^{(k)H}$ is close to zero given a large number of $\phi_i^{(k)H} = 0$. These findings collectively suggest that the hybrid strategy plays a beneficial role for the Steel Rail problem compared to LRDavidon or LRBFGS alone.

It is worth noting that the appropriate values of α_0 greater than 1 in Figure 5.3 and 5.4 correspond to cases where the mean value of $\phi_i^{(k)H}$ is greater than 0. Conversely, the α_0 values less than 1 in Figure 5.5 are associated with the mean value of $\phi_i^{(k)H}$ less than 0. This observation aligns with the conclusion drawn in Section 4.3, which suggests that negative $\phi_i^{(k)}$ values tend to perform better with $\alpha_0 < 1$ while positive $\phi_i^{(k)}$ values prefer $\alpha_0 > 1$.

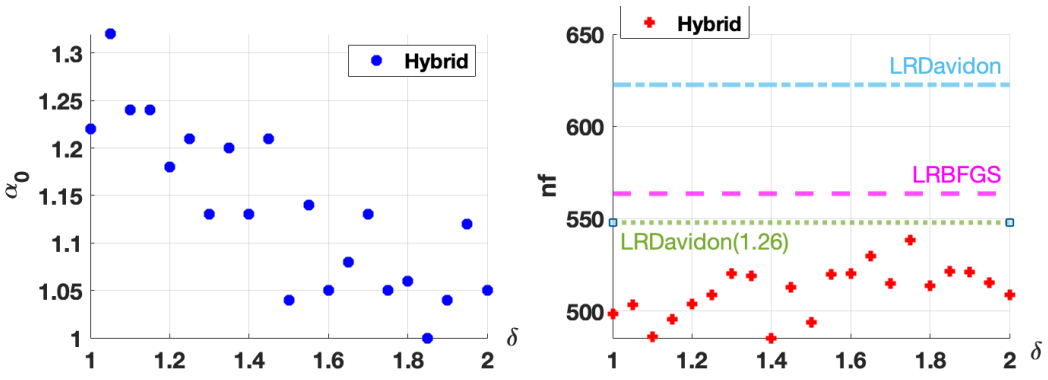
In the context of Hybrid LRDavidon-BFGS, $\phi_i^{(k)D}$ provides flexibility, while incorporating $\phi_i^{(k)\text{BFGS}}$ can be viewed as a safeguard. The best-performing δ values and the empirically determined α_0 values are employed in the experiments conducted in Section 5.4, which offer a more detailed assessment of performance.

5.4 Comparison of LRBFGS, LRDavidon and Hybrid LRDavidon-BFGS

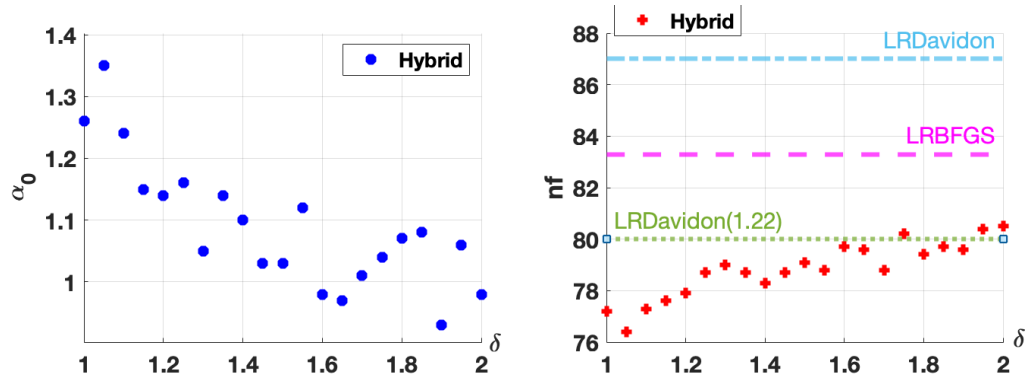
The following experiments compare the performances between LRBFGS, LRDavidon and Hybrid LRDavidon-BFGS. For Hybrid LRDavidon-BFGS, $\delta = 1.4$ with $\alpha_0 = 1.13$, $\delta = 1.05$ with $\alpha_0 = 1.35$, $\delta = 1.6$ with $\alpha_0 = 0.94$ are chosen respectively for the Stiefel Brockett problem, the low-rank matrix completion problem and the Steel Rail Cooling problem. The memory size is fixed as $L = 4$. Figure 5.6 provides performance comparison between time and $|\text{grad} f|$. Hybrid LRDavidon-BFGS, denoted as "Hybrid" with green circle, demonstrates faster convergence compared to the other methods. This highlights the robustness and efficiency of Hybrid LRDavidon-BFGS.

In summary, Hybrid LRDavidon-BFGS has demonstrated its robustness compared to LRBFGS and LRDavidon. It offers a more resilient choice within the LRBroyden family of methods. The numerical results emphasize the importance of both the $\phi_i^{(k)}$ strategy and the α_0 selection in enhancing the performance of LRBroyden. This suggests that there is still substantial potential for exploring the capabilities of the Broyden family of quasi-Newton methods by optimizing various parameter settings.

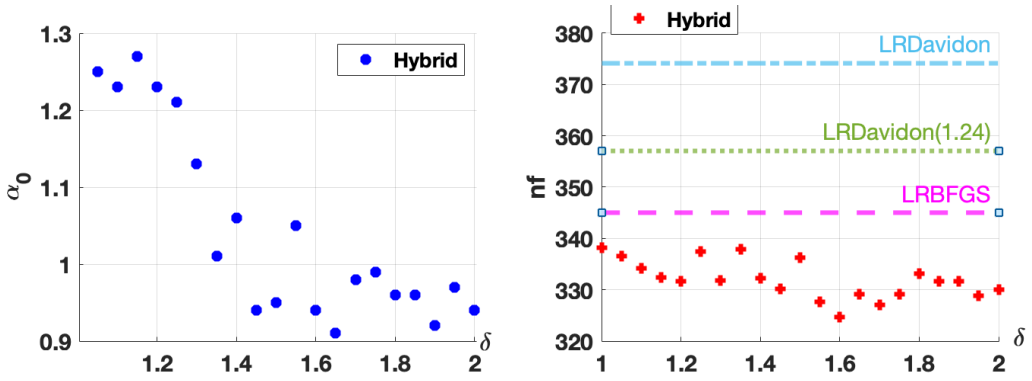
In the subsequent chapters, the LRBroyden family of methods are integrated with stochastic approximation techniques. The resulting quasi-Newton algorithm, which leverages the update of the Hybrid LRDavidon-BFGS, has achieved significant success in various applications.



(a) Stiefel Brockett: $n = 1000, p = 5, L = 4$

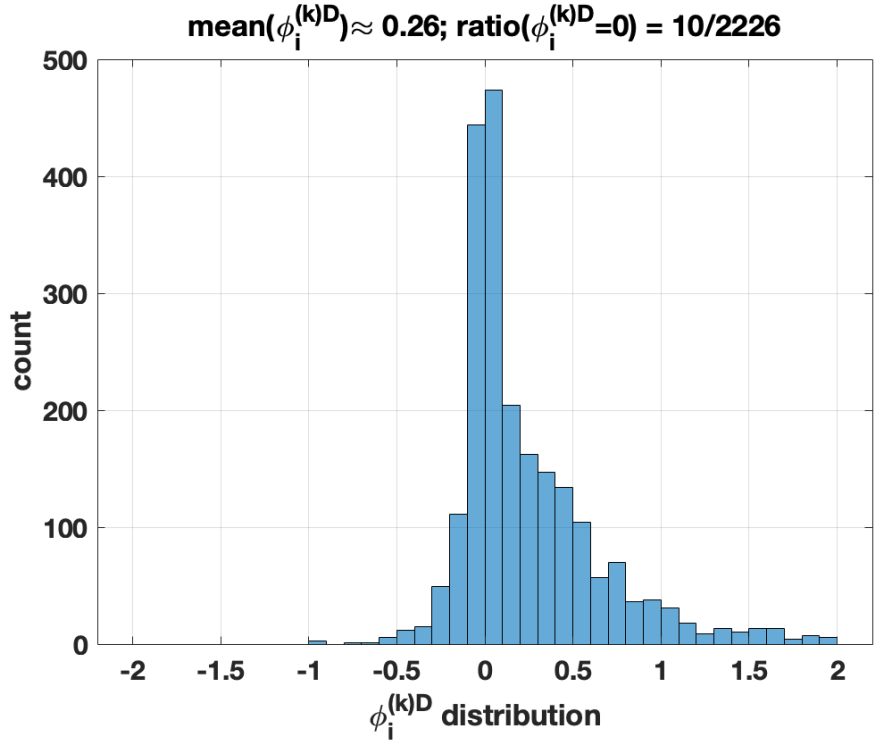


(b) Low-rank matrix completion: $m = 1000, n = 1000, r = 5, OS = 2.5, L = 4$

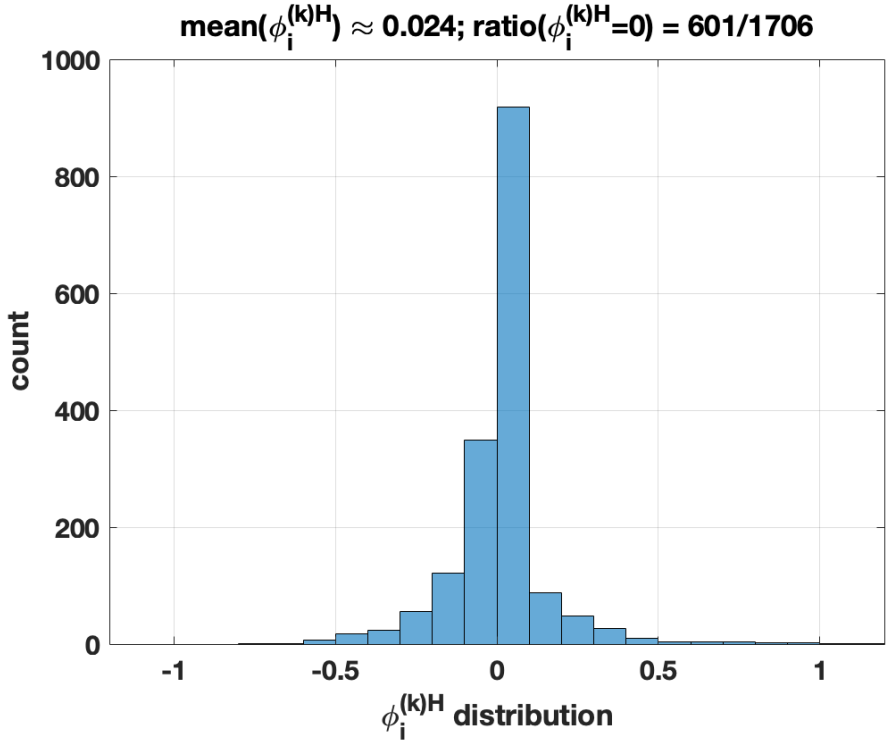


(c) Steel Rail Cooling problem: $n = 3113, r = 5, L = 4$

Figure 5.2: Performance of Hybrid LRDavidon-BFGS with different δ: Left: the best-tuned α_0 versus δ; Right: averaging nf versus δ, the average values of nf for LRBFGS, LRDavidon, LRDavidon(adaptive intial stepsize) are highlighted as the vertical line.

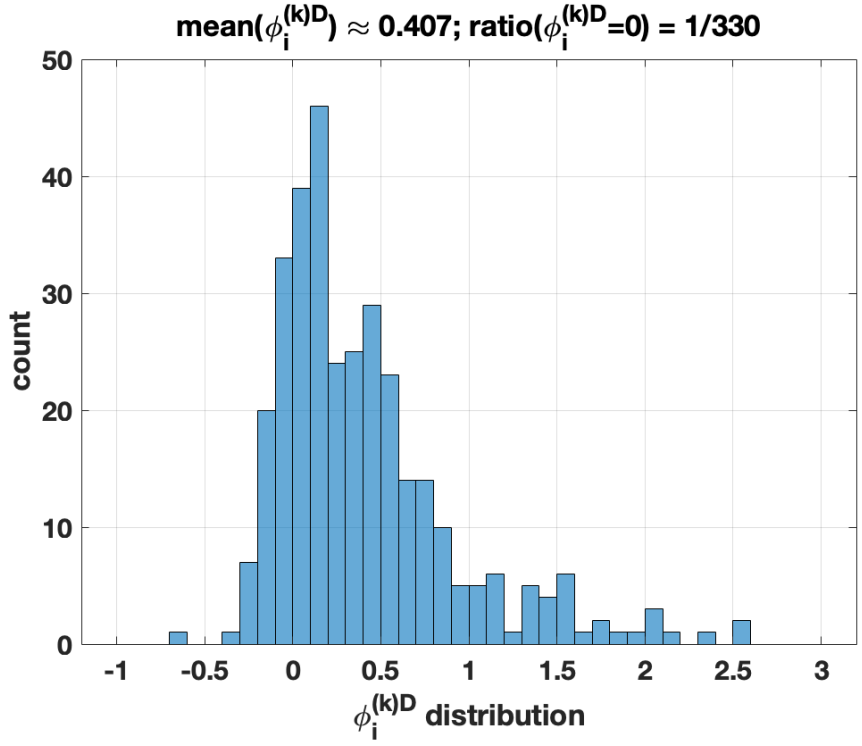


(a) LRDavidon $\alpha_0 = 1, L = 4$

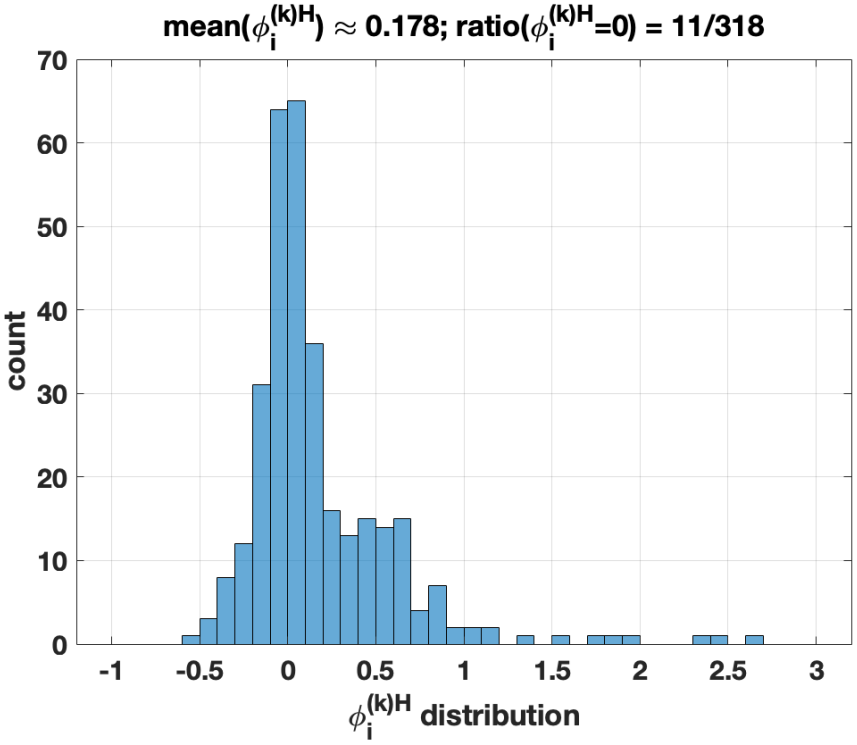


(b) Hybrid LRDavidon-BFGS $\delta = 1.4, \alpha_0 = 1.13, L = 4$

Figure 5.3: Comparison of $\phi_i^{(k)D}$ and $\phi_i^{(k)H}$ distribution for the Stiefel Brockett problem.

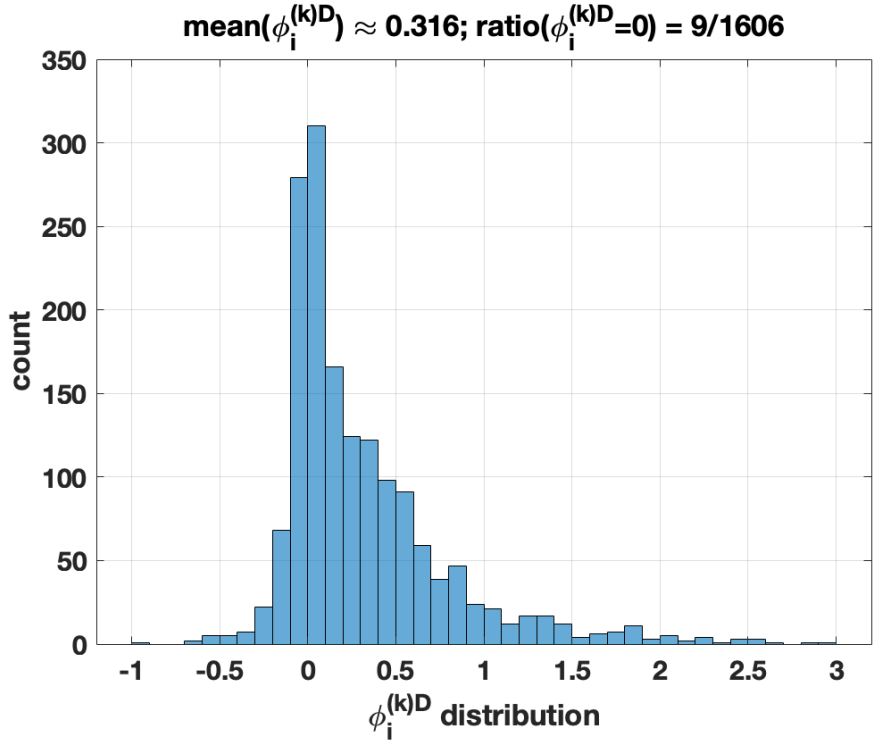


(a) LRDavidon $\alpha_0 = 1, L = 4$

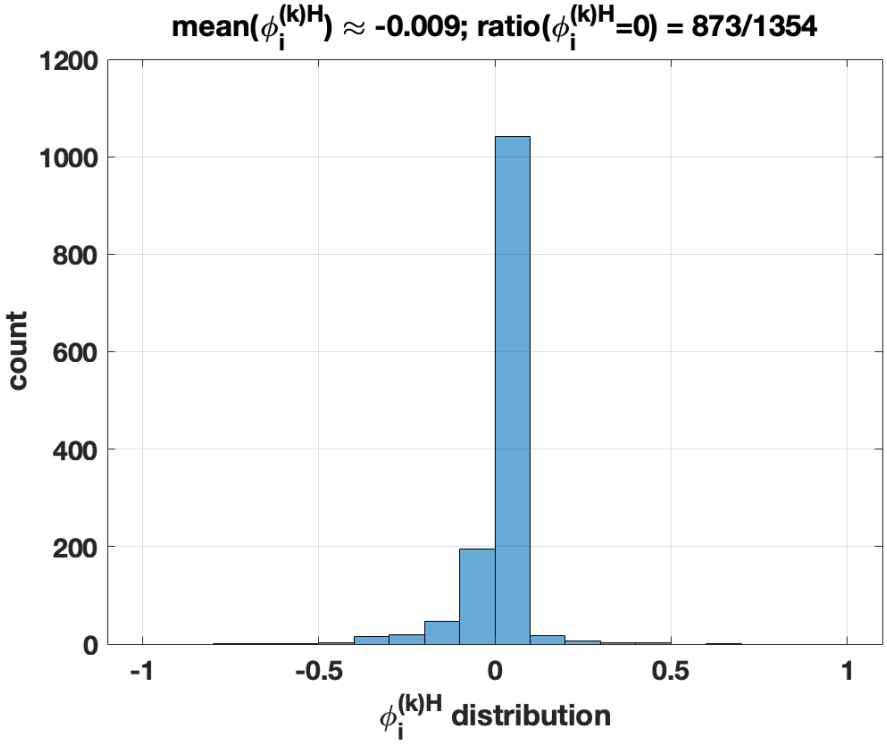


(b) Hybrid LRDavidon-BFGS $\delta = 1.05, \alpha_0 = 1.35, L = 4$

Figure 5.4: Comparison of $\phi_i^{(k)D}$ and $\phi_i^{(k)H}$ distribution for the low-rank matrix completion problem.

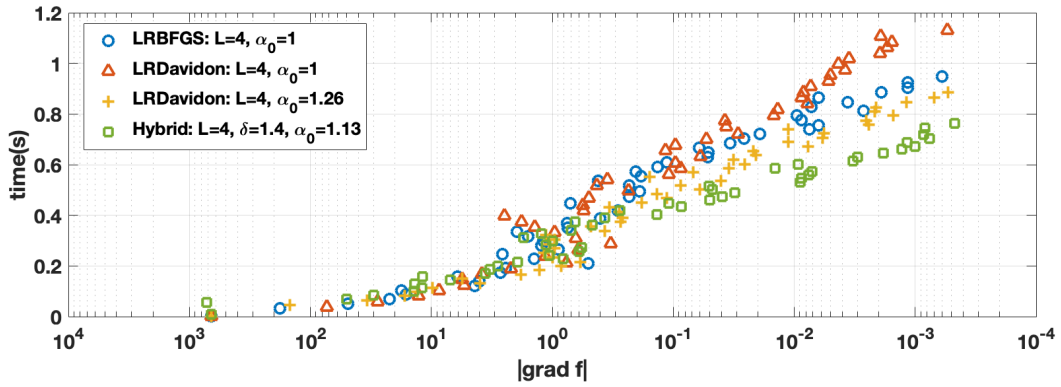


(a) LRDavidon $\alpha_0 = 1, L = 4$

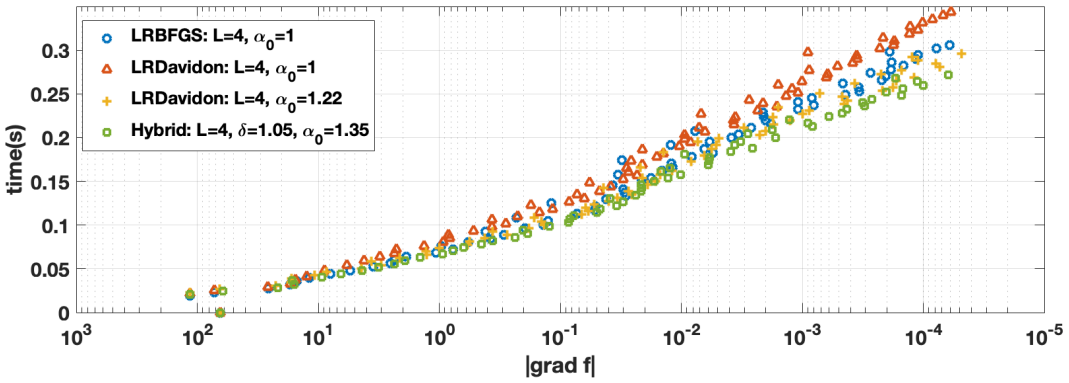


(b) Hybrid LRDavidon-BFGS $\delta = 1.6, \alpha_0 = 0.94, L = 4$

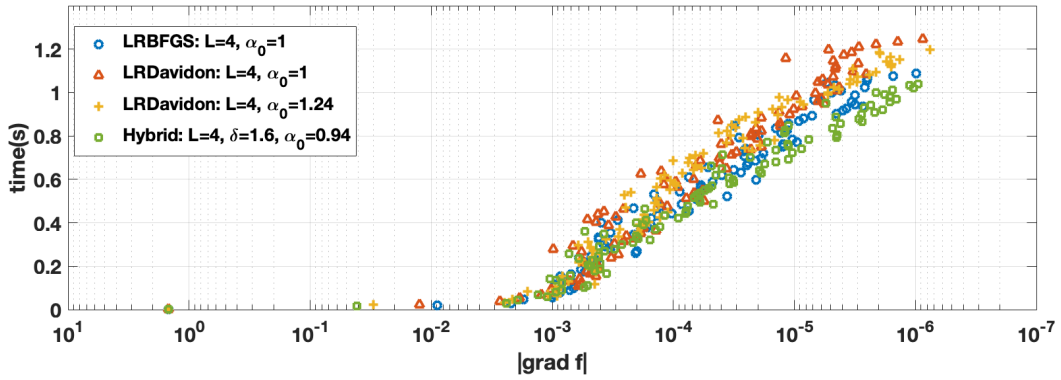
Figure 5.5: Comparison of $\phi_i^{(k)D}$ and $\phi_i^{(k)H}$ distribution for the Steel Rail cooling problem.



(a) Stiefel Brockett: $n = 1000, p = 5, L = 4$



(b) Low-rank matrix completion: $m = 1000, n = 1000, r = 5, OS = 2.5, L = 4$



(c) Steel Rail Cooling problem: $n = 3113, r = 5, L = 4$

Figure 5.6: Comparison of LRBFGS, LRDavidon and Hybrid LRDavidon-BFGS: time(s) versus |grad|.

CHAPTER 6

RIEMANNIAN STOCHASTIC BROYDEN FAMILY OF QUASI-NEWTON METHODS

We continue our study on the Riemannian Broyden family of quasi-Newton methods but shift our focus to its application in conjunction with stochastic approximation methods. The limited-memory LRBroyden family of methods, including LRBFGS, has demonstrated excellent performance in large-scale optimization with respect to the dimension of $\mathcal{M}$. In the context of the recent advancements in statistical estimation and machine learning, the dimensions and complexities of objective functions in target optimization problems have surged significantly. Many intriguing large-scale problems in this domain can be formulated as minimizing the average value of a large but finite number of loss functions, expressed as follows:

$$\min_{\omega \in \mathcal{M}} \{ f(\omega) := \frac{1}{n} \sum_{i=1}^{n} f_i(\omega) \}. \tag{6.1}$$

Here, f is a smooth real-valued function defined on a Riemannian manifold $\mathcal{M}$. This type of problem finds applications in various fields such as machine learning, statistical inference, and image processing. When n is large, computing the gradient of the full objective at every iteration faces slowdowns due to the necessity of processing every data point before updating. Therefore, it is imperative to employ stochastic approximation algorithms that update based on a small subset of gradient terms.

Intuitively, when both n and the dimension of $\mathcal{M}$ are large, a quasi-Newton method that operates in the stochastic approximation regime is promising. While applying the stochastic method to quasi-Newton methods is not a novel concept, the majority of approaches have primarily focused on adapting LRBFGS. In this chapter, we propose, implement and analyze the limited-memory variant of Riemannian stochastic quasi-Newton algorithms within the full Broyden family.

In Section 6.1, we provide a review of commonly used stochastic algorithms designed to address the large-scale finite-sum problems (6.1). Subsequently, in Section 6.2, we introduce the limited-memory variant of Riemannian stochastic Broyden with variance reduction (LR-SBroyden-VR). The convergence analysis for the proposed LR-SBroyden-VR method is presented in Section 6.3.

6.1 Stochastic Methods for Large-scale Optimization

6.1.1 Euclidean Stochastic Methods

Stochastic gradient descent (SGD) is a simple iterative approach for optimizing an object function that is a linear combination of differentiable functions (see (6.1)). SGD replaces the actual full gradient by an estimate from a randomly selected subset of the functions. Define the sub-sampled function $f_{\mathcal{S}}$ as

$$f_{\mathcal{S}}(\omega) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} f_i(\omega).$$

The batch size is defined as the size of sub-sample, denoted as $b := |\mathcal{S}|$. The SGD uses the update:

$$\omega_{k+1} = \omega_k - \alpha_k \nabla f_S(\omega_k),$$

where α_k is a positive stepsize. Both stochastic and full-gradient approaches offer different trade-offs in terms of per-iteration costs and expected per-iteration improvement in minimizing the cost function. In a full-gradient approach, a line search condition is employed to select a stepsize that ensures a sufficient decrease in the objective function. In SGD, the direction of the update is based on an estimation of the gradient computed from a subset. Due to this nature, the estimated gradient may not accurately represent the true gradient hence the direction is not guaranteed to be a descent direction. Instead of relying on a line search, SGD commonly uses a fixed or diminishing stepsize.

Some adaptive stochastic gradient algorithms have been proposed to accelerate SGD. These algorithms mainly adjust stepsizes per coordinate and use a momentum term [43] that includes previous gradients in calculating the current update. Let g_k denote the stochastic gradient vector, and the upper index j represents the j-th coordinate element.

AdaGrad [17] uses the update:

$$\omega_{k+1}^j = \omega_k^j - \alpha_k g_k^j / \sqrt{v_k^j},$$

where $v_k^j = \sum_{t=1}^k (g_t^j)^2$ is the sum of the j-th squared stochastic gradient coordinates over the past iterates. The key motivation of AdaGrad is to have different stepsizes for different coordinates.

Adam [31] additionally employs a momentum term to modify the search direction with the following update:

$$\omega_{k+1}^j = \omega_k^j - \alpha_k m_k^j / \sqrt{v_k^j},$$

where $m_k^j = \beta_1 m_{k-1}^j + (1-\beta_1) g_k^j$ with $m_0^j = 0$ and $v_k^j = \beta_2 v_{k-1}^j + (1-\beta_2)(g_k^j)^2$ with $v_0^j = 0$. Adam method updates the exponential moving averages of the gradient (m_k) and the squared gradient (v_k) where $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates of these moving averages.

AMSGrad [45] guarantees convergence while preserving the practical benefits of Adam. AMSGrad has the update:

$$\omega_{k+1}^j = \omega_k^j - \alpha_k m_k^j / \sqrt{\hat{v}_k^j},$$

where $\hat{v}_k^j = \max\{\hat{v}_{k-1}^j, v_k^j\}$ maintains a non-increasing stepsize. m_k and v_k^j have the same definition as in Adam.

In the realm of stochastic approximation, a quasi-Newton method can be applied. However, the direct application of classical quasi-Newton updating techniques leads to noisy curvature estimates, which can degrade the robustness of the updates. To address this challenge, Byrd, Hansen, Nocedal, and Singer [7] propose a transformation of the classical BFGS update, in its limited-memory form, into a stochastic method by averaging curvature estimates using sub-sampled Hessian-vector products. Building upon this foundation, Moritz et al. [36] enhance the approach by incorporating a variance-reduced strategy. More details of the variance reduction approach are elucidated in Section 6.1.3.

6.1.2 Riemannian Stochastic Methods

The first Riemannian SGD (RSGD) algorithm was proposed in 2013 by Bonnabel [4]. This algorithm extends the conventional Euclidean SGD approach to the domain of Riemannian manifolds, which has the update form:

$$\omega_{k+1} = \mathrm{Exp}_{\omega_k}(-\alpha_k \ \mathrm{grad} f_S(\omega_k)),$$

where $\mathrm{grad} f_S(\omega_k) \in \mathrm{T}_{\omega_k}\mathcal{M}$ denotes the Riemannian stochastic gradient of $f_S(\omega)$ at ω_k. The Riemannian stochastic gradient lies in the tangent space $\mathrm{T}_{\omega_k}\mathcal{M}$ of the Riemannian manifold $\mathcal{M}$. In practical scenarios, when the exponential map is either unknown in closed-form or computationally expensive, a retraction $R_{\omega_k}(-\alpha_k \ \mathrm{grad} f_S(\omega_k))$ is often employed as an alternative. The retraction allows for a more feasible computation of the updated parameter, making the algorithm applicable in situations where the direct use of the exponential map is challenging.

Adaptive stochastic gradient algorithms, such as RAdaGrad, RAdam, RAMSGrad, have been investigated and assessed in the context of Riemannain manifolds, as documented in the works of Becigneul et al. [3] and Kasai et al. [29]. While these algorithms often demonstrate superior

empirical performance compared to RSGD, they do face certain challenges due to their reliance solely on first-order information. The challenges include several issues such as relatively slow convergence, difficulty in escaping saddle points and poor performance for problems with mild to severe ill-conditioning. Depending on the specific characteristics of the optimization problem, a careful choice of algorithms, possibly combining first and second-order information, may be necessary for achieving efficient and robust optimization outcomes.

Beyond relying solely on Riemannian gradient-based methods, researchers delved into leveraging second-order information to mitigate the challenges posed by high nonlinearity and ill-conditioning in objective functions. A notable advancement in this direction is the introduction of Riemannian quasi-Newton methods. Among these, the variance-reduced stochastic LRBFGS method emerged as particularly successful, demonstrating scalability with the number of variables and effectiveness across a diverse range of applications, as detailed in works [49] and [30]. It was found that the method does more than rescaling the gradient, i.e., the improved performance is the result of incorporating curvature information.

In essence, the variance-reduced stochastic LRBFGS method showcases the advantages of considering not only the gradient but also the curvature of the objective function on a Riemannian manifold. By doing so, it addresses challenges associated with nonlinearity and ill-conditioning, making it a valuable tool for optimization tasks in various application domains.

6.1.3 Accelerating Stochastic Methods with Variance Reduction

Stochastic gradient-based optimization methods provide computational advantages through the utilization the stochastic estimates of gradient. However, both Riemannian and Euclidean methods, including (R)SGD, (R)AdaGrad, (R)Adam and (R)AMSGrad, suffer from the adverse effect of the noisy gradient estimates. Suppose $\nabla f(\omega_k)$ is the full gradient of $f(\omega)$ at ω_k. While the expectation (with respect to S) of the stochastic gradient satisfies $\mathbb{E}[\nabla f_S(\omega_k)|\omega_k] = \nabla f(\omega_k)$, the inherent randomness introduces variance. The variance, in turn, results in a slow convergence rate, imposing constraints on the stepsize [5]. Typically, a diminishing stepsize α_k must be chosen to ensure convergence and to avoid oscillation.

To address this limitation, methods that aim to reduce variance have been developed in Euclidean space. Examples include the stochastic average gradient (SAG) method [52], and the stochastic dual coordinate ascent (SDCA) method [54]. A more popular approach, stochastic variance reduction gradient (SVRG) [28], was proposed. SVRG is known for its broader applicability

across a range of problems and its reduced storage requirements, making it a widely adopted method in the optimization community.

SVRG is outlined in Algorithm 5. The key point of SVRG is to use the full gradient $\nabla f(\omega_k)$ that is occasionally computed (during each iteration of the outer k-loop in Algorithm 5) to correct the current stochastic gradient $\nabla f_S(x_t)$ (computed and used during each iteration of the inner t-loop in Algorithm 5). It is important to note that both ω_k and x_t converge to the same minimizer ω^*, implying that $\nabla f(\omega_k) \to 0$ and $\nabla f_S(\omega_k) \to \nabla f_S(\omega^*)$. As a result,

$$\nabla f_S(x_t) - \nabla f_S(\omega_k) + \nabla f(\omega_k) \to \nabla f_S(x_t) - \nabla f_S(\omega^*) \to 0.$$

This reduction in the variance of the update, as rigorously analyzed in [28], contributes to improved convergence rate. In practical implementations, it is common to opt for option I-A due to its lower storage requirements. However, option I-B, despite necessitating more storage, theoretically offers a superior convergence rate, as indicated in [28, 59].

Algorithm 5 SVRG in Euclidean space

Require: Choose an initial iterate ω_0, Update frequency m_k and stepsize α_t^k;

1: **for** $k = 0, 1, \cdots, K-1$ **do**
2: Compute the full gradient $\nabla f(\omega_k)$;
3: Set $x_0 = \omega_k$;
4: **for** $t = 0, 1, \cdots, m_k - 1$ **do**
5: Choose the sub-sample S uniformly at random from $\{1, 2, \cdots, n\}$;
6: Compute the modified stochastic gradient $g_t^k = \nabla f_S(x_t) - \nabla f_S(\omega_k) + \nabla f(\omega_k)$;
7: $x_{t+1} = x_t - \alpha_t^k g_t^k$;
8: **end for**
9: Option I-A: $\omega_{k+1} = x_{m_k}$;
10: Option I-B: $\omega_{k+1} = x_t$ for randomly chosen $t \in \{0, 1, \cdots, m_k - 1\}$;
11: **end for**

The first Riemannian variance-reduced SGD (R-SVRG) is documented in [60]. This Riemannian variant deviates in two crucial aspects when compared to the Euclidean SVRG: the first is the computation of the modified stochastic gradient (Step 6 in Algorithm 5) uses parallel transport to combine gradients from different tangent spaces:

$$g_t^k = \mathrm{grad} f_S(x_k) - P_{\gamma_k}^{1 \leftarrow 0}(\mathrm{grad} f_S(\omega_k) - \mathrm{grad} f(\omega_k)),$$

where P is the parallel transport and γ_k is the unique minimizing geodesic satisfying $\gamma_k(0) = \omega_k, \gamma_k(1) = x_t$. The second is the exponential map is used for the update (Step 7 in Algorithm 5):

$$x_{t+1} = \mathrm{Exp}_{x_t}(-\alpha_t^k g_t^k).$$

R-SVRG inherits the advantages of the Euclidean SVRG method, while introducing factors dependent on the curvature of the manifold, influencing its convergence properties [60]. Computationally efficient retraction and vector transport were considered instead of the exponential map and parallel transport in [51]. Empirical evidence from (R)-SVRG papers suggests that (R)-SVRG is considerably faster than (R)SGD for solving certain finite-sum optimization problems (6.1), primarily because the stepsize does not need to diminish. This method is also extendable to stochastic quasi-Newton algorithms.

Moreover, the stochastic variant of the entire LRBroyden family is proposed with the variance reduction method in the next section. This suggests that the benefits of variance reduction are not limited to a specific optimization method but can be integrated into a broader family of optimization algorithms. The combination of stochastic optimization, Riemannian geometry, and variance reduction method enhances the efficiency and effectiveness of the optimization process, especially for problems defined on Riemannian manifolds.

6.2 LR-SBroyden-VR Methods

In this section, we introduce the LR-SBroyden-VR family of methods, building upon the concepts used to develop the variance-reduced stochastic LRBFGS outlined in [30]. This new algorithm is a natural extension, involving the update of the modified stochastic gradient through the premultiplication of a linear inverse Hessian approximation operator. LR-SBroyden-VR provides users with the flexibility to choose from the complete LRBroyden family and integrates the stochastic variance reduction method into the optimization process. This extension allows for a versatile optimization approach, leveraging both the benefits of Riemannian geometry and the efficiency gains offered by variance reduction in stochastic optimization scenarios.

The primary challenge in developing a Riemannian stochastic quasi-Newton method lies in the generalization of the Hessian approximation. This challenge arises from the recognition that obtaining a precise curvature information y from randomly selected stochastic gradient estimations is not feasible. In Euclidean space, a common solution to this challenge is to compute average

curvature estimates using sub-sampled Hessian-vector products, as discussed in [7, 36]. However, extending this average estimates to Riemannian manifolds introduces additional complexities due to the non-Euclidean geometry, requiring careful consideration of the manifold structure in the development of effective stochastic quasi-Newton methods.

To overcome this challenge, Kasai et. al [30] introduced a novel stochastic LRBFGS incorporating variance reduction (VR) method. Algorithm 6 proposed in this section extends the idea to the entire LRBroyden family. The key point is to generate the curvature pair $\{s, y\}$ using the full Riemannian gradient which is calculated only occasionally when employing the VR method (Step 15 in Algorithm 6). There are m_k inner iterations (t-loop in Algorithm 6) between each full Riemannian gradient and curvature computation in a VR method. This tactic collects the precise curvature information and keeps using the generated inverse Hessian approximation during the m_k iterations of the t-loops.

In the t-loop, the first step is to uniformly select the sub-sample i_t (Step 4 in Algorithm 6), which has the size of b. The modified stochastic gradient is computed as $\zeta_t = (\mathcal{T}_{\omega_k}^{x_t})^{-1}\mathrm{grad}f_{i_t}(x_t) - (\mathrm{grad}f_{i_t}(\omega_k) - \mathrm{grad}f(\omega_k))$ (Step 6 in Algorithm 6), where the stochastic gradient $\mathrm{grad}f_{i_t}(x_t)$ at x_t is transported to $\mathrm{T}_{\omega_k}\mathcal{M}$ by the inverse vector transport $(\mathcal{T}_{\omega_k}^{x_t})^{-1}$. By doing so, the main calculations can be completed on $\mathrm{T}_{\omega_k}\mathcal{M}$. x_t is updated with a search direction generated by the inverse Hessian approximation operator $\mathcal{H}_k$. More specifically, the application of the inverse Hessian approximation $\mathcal{H}_k\zeta_t$ is calculated at ω_k (Step 7 in Algorithm 6). The search direction is transported back to x_t by $\mathcal{T}_{\omega_k}^{x_t}\mathcal{H}_k\zeta_t$ (Step 8 in Algorithm 6), then x_t is updated by the retraction (Step 9 in Algorithm 6).

After the m_k iterations from x_0 to x_t, ω_k is updated in the k-loop (Step 11,12 in Algorithm 6). Option I-A is natural of practical use because no additional computation and storage is needed, whereas option I-B provides a better theoretical convergence rate (see [30]). In this dissertation, only I-A is considered in the analysis and the experiments. The full gradient is calculated only once per iteration of the k-loop (Step 13 in Algorithm 6). The tangent vector pointing from ω_k to ω_{k+1} is represented as $\eta_k = R_{\omega_k}^{-1}(\omega_{k+1})$, which is calculated by the inverse of the retraction (Step 14 in Algorithm 6). η_k is used in updating the curvature pairs (Step 15 in Algorithm 6). The stored curvature pairs need to be transported from ω_k to ω_{k+1} (Step 16 in Algorithm 6). In the Step 17, we can compute and store the required information for the Hessian approximation $\mathcal{H}_{k+1}$. The key part $\tilde{M}_k$ can be determined and stored as shown in Algorithm 3. Then it can be used repeatedly when the application of the inverse Hessian approximation $\mathcal{H}_k\zeta_t$ is calculated at Step 7.

This strategy provides precise and comprehensive curvature information for the quasi-Newton updates. This avoids the need for the sub-sample of the Hessian [7, 36]. There is no need to transport the inverse Hessian approximation operator since the inverse Hessian approximation preparation and calculations of curvature information are performed only on the tangent space of the outer k-loop. This is very important since the extra cost per step for LRBroyden plays a pivotal role in achieving superior or competitive performance compared to LRBFGS. In LR-SBroyden-VR, this extra cost is diminished with the double loop structure. The drawback is that $\{s_k, y_k\}$ may be out-of-date since the update is not timely if m_k is large. How to choose an appropriate m_k is explored in the experiments.

Algorithm 6 LR-SBroyden-VR

Require: Riemannian manifold $\mathcal{M}$ with Riemannian metric g; a retraction R; isometric vector transport $\mathcal{T}_S$, with R as associated retraction that satisfies the locking condition; continuously differentiable real-valued finite-sum function f on $\mathcal{M}$; initial iterate $\omega_0 \in \mathcal{M}$; batch size b; update frequency m_k; stepsize α_t^k; memory size L; number of epochs K; convergence tolerance ε;

1: **for** $k = 0, 1, \cdots, K-1$ **do**
2: Set $x_0 = \omega_k$;
3: **for** $t = 0, 1, \cdots, m_k - 1$ **do**
4: Choose the sub-sample $i_t \subset \{1, 2, \cdots, n\}$ uniformly at random, where $|i_t| = b$;
5: Transport the k-th step sto-gradient back to $\mathrm{T}_{\omega_k}\mathcal{M}$ by $(\mathcal{T}_{\omega_k}^{x_t})^{-1}\mathrm{grad}f_{i_t}(x_t)$;
6: Calculate $\zeta_t = (\mathcal{T}_{\omega_k}^{x_t})^{-1}\mathrm{grad}f_{i_t}(x_t) - (\mathrm{grad}f_{i_t}(\omega_k) - \mathrm{grad}f(\omega_k))$ at ω_k;
7: Calculate the search direction $\mathcal{H}_k\zeta_t$ by Step 6 in Algorithm 4;
8: Transport the search direction to $\mathrm{T}_{x_t}\mathcal{M}$ by $\mathcal{T}_{\omega_k}^{x_t}\mathcal{H}_k\zeta_t$;
9: Update $x_{t+1} = R_{x_t}(-\alpha_t^k\mathcal{T}_{\omega_k}^{x_t}\mathcal{H}_k\zeta_t)$;
10: **end for**
11: Option I-A: $\omega_{k+1} = x_{m_k}$;
12: Option I-B: $\omega_{k+1} = g_{m_k}(x_1, \cdots, x_{m_k})$ (g_{m_k} is the Riemannian centroid on the manifold) or $\omega_{k+1} = x_t$ for randomly chosen $t \in \{0, 1, \cdots, m_k - 1\}$;
13: Calculate the full gradient $\mathrm{grad}f(\omega_{k+1})$;
14: Calculate the tangent vector $\eta_k = R_{\omega_k}^{-1}(\omega_{k+1})$;
15: Calculate $s_k^{(k+1)} = \mathcal{T}_{\omega_k}^{\omega_{k+1}}\eta_k$ and $y_k^{(k+1)} = \mathrm{grad}f(\omega_{k+1})/\beta_k - \mathcal{T}_{\omega_k}^{\omega_{k+1}}\mathrm{grad}f(\omega_k)$, where $\beta_k = \|\eta_k\|/\|\mathcal{T}_{R_{\eta_k}}\eta_k\|$;
16: Update curvature pairs as Step 28 in Algorithm 2;
17: Prepare for the Hessian approximation as Step 5 in Algorithm 4;
18: **end for**
19: Output ω_k if satisfies stopping criteria.

6.3 Convergence Analysis

In the context of the LR-SBroyden-VR method proposed in this chapter, the global convergence analysis and the convergence rate analysis on a strongly retraction-convex function are presented in the following context. The presentation follows a known procedure, and modifications to certain lemmas in the convergence proof system are elucidated. The modifications to the existing proof system indicate the adjustments made to accommodate the specific characteristics and features of the LR-SBroyden-VR compared to the LR-SBFGS-VR method.

6.3.1 Assumptions and Preliminary Lemmas

For the sake of clarity in the proofs, several notations are introduced to facilitate the analysis of the LR-SBroyden-VR method: Use ω_t^k to represent x_t in the k-th epoch in Algorithm 6. Use subscript i_t^k to represent sub-sample chosen for stochastic gradient at ω_t^k (Step 4 in Algorithm 6). Use η_t^k to represent the search direction from ω_t^k to ω_{t+1}^k, i.e., $R_{\omega_t^k}^{-1}(\omega_{t+1}^k)$. Ignoring the superscripts of Step 15-16 in Algorithm 6, use $s_{k-1}, s_{k-2}, \cdots, s_{k-L}$ and $y_{k-1}, y_{k-2}, \cdots, y_{k-L}$ to represent the stored curvature pairs for updating $\omega_k \to \omega_{k+1}$ using $\mathcal{H}_k$. Use $\mathbb{E}_{i_t^k}[\cdot]$ to denote the expectation taken with respect to the distribution of the random variable i_t^k. Use $\mathbb{E}[\cdot]$ to denote the total expectation with respect to the joint distribution of all random variables, e.g., $\mathbb{E}[f(\omega_t^k)] = \mathbb{E}_{i_1^k}\mathbb{E}_{i_2^k}\cdots\mathbb{E}_{i_t^k}[f(\omega_t^k)]$. The following assumptions are made:

Assumption 6.1. *The objective function f and its components are twice continuously differentiable.*

Assumption 6.2. *For a sequence $\{\omega_t^k\}$ generated by Algorithm 6, there exists a compact and connected set $\Omega \in \mathcal{M}$ such that $\omega_t^k \in \Omega$ for all $k, t > 0$. For each $k \geq 1, t \geq 0$, there exists a ϱ-totally retractive neighborhood $\tilde{\Omega}_k$ of ω_k such that ω_t^k stays in $\tilde{\Omega}_k$, where a ϱ-totally retractive neighborhood $\tilde{\Omega}_k$ is a set that for all $y \in \tilde{\Omega}_k$, $\tilde{\Omega}_k \subset R_y(\mathbb{B}(0_y, \varrho))$ and $R_y(\cdot)$ is a diffeomorphism on $\mathbb{B}(0_y, \varrho)$. Furthermore, the iterates ω_t^k stay continuously in a ϱ-totally retractive neighborhood $\tilde{\Omega}$ of ω^*.*

Assumption 6.3. *f and its components are strongly retraction-convex with respect to the retraction R in $\tilde{\Omega}$.*

Assumption 6.4. *The vector transport is isometric, the conditions in* (3.2),(3.3) *and the locking condition* (3.4) *are satisfied.*

Assumption 6.5. *f is bounded below, and a decaying stepsize sequence $\{\alpha_t^k\}$ satisfies $\sum \alpha_t^k = \infty$ and $\sum(\alpha_t^k)^2 < \infty$. If the Assumption 6.3 is satisfied, α_t^k can be relaxed to a fixed stepsize.There exists $A > 0$ such that for all $\omega \in \tilde{\Omega}$ and $i \in \mathbb{N}$, we have $\|\mathrm{grad} f(\omega)\| \leq A$ and $\|\mathrm{grad} f_i(\omega)\| \leq A$.*

Assumption 6.6. *Riemannian stochastic gradient is bounded as $\mathbb{E}_{i_t^k}[\|\mathrm{grad} f_{i_t^k}(\omega_t^k)\|^2] < C^2$.*

With these assumptions, the following lemmas are stated for the final theorem proof:

Lemma 6.1. *(see [27, Lemma 3.1]) Define $m_{\omega,\eta}(\tau) = f(R_\omega(\tau\eta))$. If Assumptions 6.1, 6.2 and 6.3 hold then there exists constants $a_1 > a_0 > 0$ such that*

$$a_0 \leq \frac{d^2 m_{\omega,\eta}}{d\tau^2} \leq a_1,$$

for all $\omega \in \tilde{\Omega}$, $\eta \in \mathrm{T}_\omega\mathcal{M}$, $\|\eta\| = 1$ and all τ such that $R_\omega(t\eta) \in \tilde{\Omega}$ for all $t \in [0, \tau]$.

Lemma 6.2. *If Assumptions 6.1, 6.2 and 6.3 hold, then there exists constants $a_1 > a_0 > 0$ such that*

$$g(\mathrm{grad} f(\omega_t^k), \alpha_t^k \eta_t^k) + \frac{1}{2} a_0 (\alpha_t^k \|\eta_t^k\|)^2 \leq f(\omega_{t+1}^k) - f(\omega_t^k) \leq g(\mathrm{grad} f(\omega_t^k), \alpha_t^k \eta_t^k) + \frac{1}{2} a_1 (\alpha_t^k \|\eta_t^k\|)^2,$$

where constants a_0, a_1 can be chosen as in Lemma 6.1.

Proof. Define $m_t^k(\tau) = f(R_{\omega_t^k}(\tau \eta_t^k / \|\eta_t^k\|))$, By Taylor's Theorem,

$$\begin{aligned} f(\omega_{t+1}^k) - f(\omega_t^k) &= f(R_{\omega_t^k}(\alpha_t^k \eta_t^k)) - f(R_{\omega_t^k}(0)) \\ &= m_t^k(\alpha_t^k \|\eta_t^k\|) - m_t^k(0) \\ &= \frac{dm_t^k(0)}{d\tau} \alpha_t^k \|\eta_t^k\| + \frac{1}{2} \frac{d^2 m_t^k(p)}{d\tau^2} (\alpha_t^k \|\eta_t^k\|)^2 \\ &= g(\mathrm{grad} f(\omega_t^k), \alpha_t^k \eta_t^k) + \frac{1}{2} \frac{d^2 m_t^k(p)}{d\tau^2} (\alpha_t^k \|\eta_t^k\|)^2, \end{aligned} \tag{6.2}$$

where $0 \leq p \leq \alpha_t^k \|\eta_t^k\|$. Then the inequality is easily derived from Lemma 6.1. □

Lemma 6.3. *If Assumptions 6.1-6.4 hold, then there exists constants $a_1 > a_0 > 0$ such that*

$$a_0 g(s_k, s_k) \leq g(s_k, y_k) \leq a_1 g(s_k, s_k),$$

where constants a_0 and a_1 can be chosen as in Lemma 6.1.

Proof. s_k, y_k are generated between ω_0^k (dnoted as ω_k in Algorithm 6) and $\omega_{m_k}^k$ (dnoted as ω_{k+1} in Algorithm 6), where m_k is the frequency value. $\eta_k = R_{\omega_k}^{-1}(\omega_{k+1})$ is calculated. The lemma can be proved under the locking condition and Taylor's Theorem (see LEMMA 3.3 in [27]). □

Lemma 6.4. *(see [27, Lemma 3.5]) Let $\mathcal{M}$ be a Riemannian manifold endowed with two vector transports $\mathcal{T}_1 \in C^0$ and $\mathcal{T}_2 \in C^\infty$, where both transports are associated with the same retraction R. Under Assumption 6.4, for any $\bar{\omega} \in \mathcal{M}$, there exists a neighborhood $\mathcal{U}$ of $\bar{\omega}$ and a constant such that for all $\omega, y \in \mathcal{U}$,*

$$\|\mathcal{T}_{1_\eta}\xi - \mathcal{T}_{2_\eta}\xi\| \leq c_1\|\xi\|\|\eta\|,$$

where $\xi, \eta \in \mathrm{T}_\omega\mathcal{M}$ and $\eta = R_\omega^{-1}(y)$.

Lemma 6.5. *If Assumptions 6.1-6.4 hold, then there exists a constant $a_2 > 0$ such that*

$$g(y_k, y_k) \leq a_2 g(s_k, y_k).$$

Proof. Define $y_k^P = \mathrm{grad}f(\omega_{k+1}) - P_{\gamma_k}^{1\leftarrow 0}\mathrm{grad}f(\omega_k)$, where $\gamma_k(\tau) = R_{\omega_k}(\tau\eta_k)$, i.e., the retraction line from ω_k to ω_{k+1} and P is the parallel transport along $\gamma_k(\tau)$. Then $\|P_{\gamma_k}^{0\leftarrow 1}y_k^P = \bar{H}_k\eta_k\| \leq b_0\|\eta_k\|^2 = b_0\|s_k\|^2$, where $\bar{H}_k = \int_0^1 P_{\gamma_k}^{0\leftarrow\tau}\mathrm{Hess}f(\gamma_k(\tau))P_{\gamma_k}^{\tau\leftarrow 0}\,d\tau$ with $b_0 > 0$. Then

$$\begin{aligned}
\|y_k\| &\leq \|y_k - y_k^P\| + \|y_k^P\| = \|y_k - y_k^P\| + \|P_{\gamma_k}^{0\leftarrow 1}y_k^P\| \\
&\leq \|y_k - y_k^P\| + \|P_{\gamma_k}^{0\leftarrow 1}y_k^P - \bar{H}_k\eta_k\| + \|\bar{H}_k\eta_k\| \\
&\leq \|\mathrm{grad}f(\omega_{k+1})/\beta_k - \mathcal{T}_{\eta_k}\mathrm{grad}f(\omega_k) - \mathrm{grad}f(\omega_{k+1}) + P_{\gamma_k}^{1\leftarrow 0}\mathrm{grad}f(\omega_k)\| + b_0\|s_k\|^2 + \|\bar{H}_k\eta_k\| \\
&\leq \|\mathrm{grad}f(\omega_{k+1})/\beta_k - \mathrm{grad}f(\omega_{k+1})\| + \|P_{\gamma_k}^{1\leftarrow 0}\mathrm{grad}f(\omega_k) - \mathcal{T}_{\eta_k}\mathrm{grad}f(\omega_k)\| + b_0\|s_k\|^2 + \|\bar{H}_k\eta_k\| \\
&\leq b_1\|\mathrm{grad}f(\omega_{k+1})\| + b_2\|s_k\|\|\mathrm{grad}f(\omega_k)\| + b_0\|s_k\|^2 + b_3\|s_k\| \\
&\leq b_4\|s_k\|, \qquad (6.3)
\end{aligned}$$

where b_1, b_2, b_3 and $b_4 > 0$. Therefore following Lemma 6.3,

$$\frac{g(y_k, y_k)}{g(s_k, y_k)} \leq \frac{g(y_k, y_k)}{a_0 g(s_k, s_k)} \leq \frac{b_4^2}{a_0}$$

as desired. □

The next lemma is to bound the trace and determinant of $\mathcal{B}_k = (\mathcal{H}_k)^{-1}$ at ω_k. Denote the updated Hessian and inverse Hessian approximation as $\mathcal{B}_k^j$ and $\mathcal{H}_k^j$ with $j = 0, 1, \cdots, L$, where L is the memory size. $\mathcal{B}_k^0 \to \mathcal{B}_k^1 \to \cdots \to \mathcal{B}_k^L \equiv \mathcal{B}_k$ and $\mathcal{H}_k^0 \to \mathcal{H}_k^1 \to \cdots \to \mathcal{H}_k^L \equiv \mathcal{H}_k$. The hat notation $\hat{\mathcal{B}}_k$ denotes the coordinate expression of the operator.

Lemma 6.6. *If Assumptions 6.1-6.4 hold, then there exists constants M_3, M_4 such that*

$$\mathrm{trace}(\hat{\mathcal{B}}_k) \leq M_3,$$

$$\det(\hat{\mathcal{B}}_k) \geq M_4,$$

for all $k \geq 1$*, under the convergence bounds* $(1-\nu)\phi_i^{(k)c} \leq \phi_i^{(k)} \leq 1-\delta$ (3.14) *for Broyden parameter* $\phi_i^{(k)}$.

Proof. The proof is similar to the proof part in Theorem 3.2, where Lemma 6.3(corresponding to Lemma 3.3) and Lemma 6.5(corresponding to Lemma 3.9) are needed. There is no requirement for the stepsize α_t^k during the proof, which makes it no difference after equipping the stochastic method. Suppose $\det(\hat{\mathcal{B}}_k^j) \geq M_1$ and $\text{trace}(\hat{\mathcal{B}}_k^j) \leq M_2$, we have

$$\det(\hat{\mathcal{B}}_k^{j+1}) \geq a_0 M_1/\nu M_2,$$

$$\text{trace}(\hat{\mathcal{B}}_k^{j+1}) \leq (1 + a_2/a_0 + \sqrt{a_2/a_0})M_2 + a_2,$$

where ν is the constant defined in (3.14). With the fact that the initial guess $\hat{\mathcal{B}}_k^0 = \frac{g(y_{k-1},y_{k-1})}{g(s_{k-1},y_{k-1})}\text{id}$, $\det(\hat{\mathcal{B}}_k^0)$ and $\text{trace}(\hat{\mathcal{B}}_k^0)$ are bounded and memory size L is a constant. Then constants M_3, M_4 can be found to prove the lemma. □

In Algorithm 6, ω_k is the point calculating the whole gradient. Search direction by Hessian approximation is computed at this point then transport back to the current iterate, hence do not need to transport the Hessian operator. For proof convenience, consider the version that transports Hessian operator to the current iterate ω_t^k: $\mathcal{H}_t^k := \mathcal{T}_{\omega_k}^{\omega_t^k} \circ \mathcal{H}_k \circ (\mathcal{T}_{\omega_k}^{\omega_t^k})^{-1}$. Meanwhile consider $\zeta_t^k := \mathcal{T}_{\omega_k}^{\omega_t^k}\zeta_t$ so that the search direction at ω_t^k(x_t in the algorithm) is $-\mathcal{H}_t^k\zeta_t^k$. Given $A, B \in \mathbb{R}^{n\times n}$, $A \preceq B$ means that $B-A$ is positive semi-definite. Lemma 6.7 bounds the eigenvalues of the inverse Hessian approximation operator.

Lemma 6.7. *If Assumption 6.1-6.4 hold, then there exists constants* $0 < a_3 < a_4 < \infty$ *such that*

$$a_3\text{id} \preceq \hat{\mathcal{H}}_k \preceq a_4\text{id},$$

$$a_3\text{id} \preceq \hat{\mathcal{H}}_t^k \preceq a_4\text{id},$$

for all $k \geq 1$.

Proof. The first inequality regarding to $\hat{\mathcal{H}}_k = \hat{\mathcal{B}}_k^{-1}$ is easily proved from Lemma 6.6, with the help that the determinant of a matrix is the product of its eigenvalues and the trace corresponds to the sum. Since the vector transport is isometric, the trace and determinant are identical after the transport $\mathcal{H}_t^k := \mathcal{T}_{\omega_k}^{\omega_t^k} \circ \mathcal{H}_k \circ (\mathcal{T}_{\omega_k}^{\omega_t^k})^{-1}$. This completes the proof of the second inequality. □

Lemma 6.8. *The iterates of Algorithm 6 satisfy the following inequality for all k:*

(i) $\mathbb{E}_{i_t^k}[f(\omega_{t+1}^k)] - f(\omega_t^k) \leq -\alpha_t^k g(\mathrm{grad} f(\omega_t^k), \mathbb{E}_{i_t^k}[\mathcal{H}_t^k \zeta_t^k]) + \frac{1}{2}(\alpha_t^k)^2 a_1 \mathbb{E}_{i_t^k}[\|\mathcal{H}_t^k \zeta_t^k\|^2]$.

(ii) Let $\omega \in \mathcal{M}$ and ω' be in a ϱ-totally retractive neighborhood of ω. It holds that

$$2a_0(f(\omega) - f(\omega')) \leq \|\mathrm{grad} f(\omega)\|^2.$$

Proof. (i)Substitute η_t^k by $-\mathcal{H}_t^k \zeta_t^k$ in Lemma 6.2 on the right inequality part:

$$f(\omega_{t+1}^k) - f(\omega_t^k) \leq g(\mathrm{grad} f(\omega_t^k), -\alpha_t^k \mathcal{H}_t^k \zeta_t^k) + \frac{1}{2} a_1 (\alpha_t^k \| - \mathcal{H}_t^k \zeta_t^k \|)^2.$$

Then taking the expectations with respect to i_t^k brings us the desired inequality, since only ω_t^k doe not depend on the i_t^k.

(ii)The second inequality can be proved by the other side in Lemma 6.2 by substitute ω_t^k into ω, ω_{t+1}^k into ω' and $\alpha_t^k \eta_t^k$ into ζ:

$$f(\omega') - f(\omega) \geq g(\mathrm{grad} f(\omega), \zeta) + \frac{1}{2} a_0 \|\zeta\|^2 \geq \min_{\xi \in \mathrm{T}_\omega \mathcal{M}} (g(\mathrm{grad} f(\omega), \xi) + \frac{1}{2} a_0 \|\xi\|^2),$$

where the minimum equals $-\frac{1}{2a_0}\|\mathrm{grad} f(\omega)\|^2$ when $\xi = -\mathrm{grad} f(\omega)/a_0$. □

Lemma 6.9. *If Assumption 6.1-6.6 hold, then*

(i) $\mathbb{E}[f(\omega_{t+1}^k)] \leq f(\omega_t^k) - \alpha_t^k a_3 \|\mathrm{grad} f(\omega_t^k)\|^2 + \frac{9}{2} a_1 (\alpha_t^k a_4 A)^2$.

(ii) $\mathbb{E}\left[\sum_{k=1}^K \sum_{t=1}^{m_k} \alpha_t^k \|\mathrm{grad} f(\omega_t^k)\|^2\right] < \infty$.

(iii) $\liminf_{k\to\infty} \mathbb{E}[\|\mathrm{grad} f(\omega_t^k)\|^2] = 0$.

Proof. (i)Note that $\mathbb{E}_{i_t^k}[\zeta_t^k] = \mathrm{grad} f(\omega_t^k)$ since ζ_t^k is an unbiased estimate of $\mathrm{grad} f(\omega_t^k)$ [28]. Taking expectation with respect to ω_t^k in Lemma 6.8 (i) yields:

$$\begin{aligned} \mathbb{E}[f(\omega_{t+1}^k)] - f(\omega_t^k) &\leq -\alpha_t^k g(\mathrm{grad} f(\omega_t^k), \mathcal{H}_t^k \mathbb{E}_{i_t^k}[\zeta_t^k]) + \frac{1}{2}(\alpha_t^k)^2 a_1 \mathbb{E}_{i_t^k}[\|\mathcal{H}_t^k \zeta_t^k\|^2] \\ &\leq -\alpha_t^k a_3 \|\mathrm{grad} f(\omega_t^k)\|^2 + \frac{1}{2}(\alpha_t^k)^2 a_1 \mathbb{E}_{i_t^k}[a_4^2 \|\zeta_t^k\|^2], \end{aligned} \tag{6.4}$$

where the second inequality comes from Lemma 6.7. Finally, the proof is completed with the Assumption 6.6: $\|\zeta_t^k\| = \|\mathrm{grad} f_{i_t^k}(\omega_t^k) - \mathcal{T}_{\omega_k}^{\omega_t^k}[\mathrm{grad} f_{i_t^k}(\omega_k) - \mathrm{grad} f(\omega_k)]\| \leq 3A$ by the reduced variance strategy.

(ii)Taking the total expectation of (i):

$$\mathbb{E}[f(\omega_{t+1}^k)] - \mathbb{E}[f(\omega_t^k)] \leq -\alpha_t^k a_3 \mathbb{E}[\|\mathrm{grad} f(\omega_t^k)\|^2] + \frac{9}{2} a_1 (\alpha_t^k a_4 A)^2.$$

Then sum for all the iterates from ω_0^0 to $\omega_{m_k}^K$:

$$\mathbb{E}[f(\omega_{m_k}^K)] - \mathbb{E}[f(\omega_0^0)] \leq -a_3 \sum_{k=1}^{K} \sum_{t=1}^{m_k} \alpha_t^k \mathbb{E}[\|\mathrm{grad} f(\omega_t^k)\|^2] + \frac{9}{2} a_1 a_4^2 A^2 \sum_{k=1}^{K} \sum_{t=1}^{m_k} (\alpha_t^k)^2.$$

Assumption 6.5 gives the lower bound of f and the limit sum of $(\alpha_t^k)^2$, which completes proof after rearranging the inequality.

(iii)This is easily proved by contradiction based on (ii). □

6.3.2 Global Convergence Theorem

Theorem 6.1. *Let $\omega^* \in \mathcal{M}$ be a non-degenerate local minimizer of f. Suppose Assumptions 6.1-6.6 hold for Algorithm 6 and the Broyden parameter satisfies $(1-\nu)\phi_i^{(k)c} \leq \phi_i^{(k)} \leq 1 - \delta$ (3.14). Then*

$$\lim_{k \to \infty} \mathbb{E}[\|\mathrm{grad} f(\omega_t^k)\|^2] = 0.$$

Proof. The proof starts with defining $h(\omega) := \|\mathrm{grad} f(\omega)\|^2$ and from the Taylor's Theorem,

$$h(\omega_{t+1}^k) - h(\omega_t^k) \leq -2\alpha_t^k g(\mathrm{grad} h(\omega_t^k), \mathrm{Hess} f(\omega_t^k)[\eta_t^k]) + \frac{1}{2}(\alpha_t^k)^2 a_5 \|\eta_t^k\|^2.$$

The derivative is similar with Lemma 6.2, while the first term on the right side is different because of the derivative of $h(\omega)$. Then taking the expectation with respect to i_t^k like Lemma 6.8 (i) yields:

$$\begin{aligned}
\mathbb{E}_{i_t^k}[h(\omega_{t+1}^k)] - h(\omega_t^k) &\leq -2\alpha_t^k g(\mathrm{grad} f(\omega_t^k), \mathrm{Hess} f(\omega_t^k)[\mathcal{H}_t^k \mathbb{E}_{i_t^k}[\zeta_t^k]]) + \frac{1}{2}(\alpha_t^k)^2 a_5 \mathbb{E}_{i_t^k}[\|\mathcal{H}_t^k \zeta_t^k\|^2] \\
&\leq 2\alpha_t^k \|\mathrm{grad} f(\omega_t^k)\| \, \|\mathrm{Hess} f(\omega_t^k)[\mathcal{H}_t^k \mathrm{grad} f(\omega_t^k)]\| + \frac{1}{2}(\alpha_t^k)^2 a_5 a_4^2 \mathbb{E}_{i_t^k}[\|\zeta_t^k\|^2] \\
&\leq 2\alpha_t^k a_1 a_4 \|\mathrm{grad} f(\omega_t^k)\|^2 + \frac{9}{2}(\alpha_t^k)^2 a_5 a_4^2 A^2. \qquad (6.5)
\end{aligned}$$

Taking the total expectation, we have

$$\mathbb{E}[h(\omega_{t+1}^k)] - \mathbb{E}[h(\omega_t^k)] \leq 2\alpha_t^k a_1 a_4 \mathbb{E}[\|\mathrm{grad} f(\omega_t^k)\|^2] + \frac{9}{2}(\alpha_t^k)^2 a_5 a_4^2 A^2.$$

The first term on the right side has a convergent sum from Lemma 6.9 (ii) so that $\mathbb{E}[h(\omega_{t+1}^k)] - \mathbb{E}[h(\omega_t^k)]$ is the term of a convergent sum. We can define $Z_K^+ := \sum_{k=1}^K \sum_{t=1}^{m_k} \max(0, \mathbb{E}[h(\omega_{t+1}^k)] - \mathbb{E}[h(\omega_t^k)])$ and $Z_K^- := \sum_{k=1}^K \sum_{t=1}^{m_k} \max(0, \mathbb{E}[h(\omega_t^k)] - \mathbb{E}[h(\omega_{t+1}^k)])$. S_K^+ is non-decreasing and upper bounded, $\mathbb{E}[h(\omega_{m_k}^K)] = h(\omega_0^0) + S_K^+ - S_K^- \geq 0$ indicates that non-decreasing S_K^- is also upper bounded. Both two converges therefore $\mathbb{E}[h(\omega_{m_k}^K)]$ converges. Then the limit of $\mathbb{E}[\|\mathrm{grad} f(\omega_t^k)\|]^2$ must go to zero from Lemma 6.9 (iii). □

Lemma 6.10. *(Lemma 3.4 in [23]). For any $\bar{\omega} \in \mathcal{M}$, there exists $\tau_1 > 0, \tau_2 > 0$ and δ_{τ_1,τ_2} such that for all ω in a sufficiently small neighborhood of $\bar{\omega}$ and all $\xi \in \mathrm{T}_\omega \mathcal{M}$ with $\|\xi\| \leq \delta_{\tau_1,\tau_2}$, it holds that*

$$\tau_1 \mathrm{dist}(\omega, R_\omega(\xi)) \leq \|\xi\| \leq \tau_2 \mathrm{dist}(\omega, R_\omega(\xi)),$$

where $\mathrm{dist}(\cdot,\cdot)$ *is the shortest distance between two points on $\mathcal{M}$.*

Lemma 6.11. *Suppose Assumptions 6.1, 6.2, 6.4 and 6.6 hold. Let $c_2 > 0$ be a constant such that*

$$\|P_\gamma^{\omega \leftarrow y}(\mathrm{grad} f_i(y)) - \mathrm{grad} f_i(\omega)\| \leq c_2 \mathrm{dist}(\omega, y),$$

where $\omega, y \in \tilde{\Omega}$, $i = 1, 2, \cdots, n$, $\gamma(t) := R_y(t\eta)$ is a curve defined by a retraction and $P_\gamma^{\omega \leftarrow y}$ is a parallel translation operator along γ from y to ω. Then, the upper bound of the variance of $\mathbb{E}_{i_t^k}[\|\zeta_t^k\|^2]$ is given by

$$\mathbb{E}_{i_t^k}[\|\zeta_t^k\|^2] \leq 4(c_2^2 + \tau_2^2 C^2 c_1^2)(7(\mathrm{dist}(\omega_t^k, \omega^*))^2 + 4(\mathrm{dist}(\omega_k, \omega^*))^2).$$

Proof. The first inequality is guaranteed by the fact that $\|P_\gamma^{\omega \leftarrow y}(\mathrm{grad} f(y)) - \mathrm{grad} f(\omega)\|$ is bounded above by a constant times $\|s_k\|$ in the proof of Lemma 6.5.

For the second part, consider $\zeta_t^k = \mathrm{grad} f_{i_t^k}(\omega_t^k) - \mathcal{T}_{\omega_k}^{\omega_t^k}[\mathrm{grad} f_{i_t^k}(\omega_k) - \mathrm{grad} f(\omega_k)]$:

$$
\begin{aligned}
& \mathbb{E}_{i_t^k}[\|\zeta_t^k\|^2] \\
= \; & \mathbb{E}_{i_t^k}[\| \left(\mathrm{grad} f_{i_t^k}(\omega_t^k) - \mathcal{T}_{\omega^*}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega^*)\right) \\
& + \left(\mathcal{T}_{\omega^*}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega^*) - \mathcal{T}_{\omega_k}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega_k)\right) + \mathcal{T}_{\omega_k}^{\omega_t^k} \mathrm{grad} f(\omega_k)\|^2] \\
\leq \; & 2\mathbb{E}_{i_t^k}\left[\|\mathrm{grad} f_{i_t^k}(\omega_t^k) - \mathcal{T}_{\omega^*}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega^*)\|^2\right] \\
& + 2\mathbb{E}_{i_t^k}\left[\|\mathcal{T}_{\omega_k}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega_k) - \mathcal{T}_{\omega^*}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega^*) - \mathcal{T}_{\omega_k}^{\omega_t^k} \mathrm{grad} f(\omega_k)\|^2\right] \\
= \; & 2\mathbb{E}_{i_t^k}\left[\|\mathrm{grad} f_{i_t^k}(\omega_t^k) - \mathcal{T}_{\omega^*}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega^*)\|^2\right] \\
& + 2\mathbb{E}_{i_t^k}\left[\|\mathcal{T}_{\omega_k}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega_k) - \mathcal{T}_{\omega^*}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega^*)\|^2\right] + 2\|\mathrm{grad} f(\omega_k)\|^2 \\
& - 4g(\mathcal{T}_{\omega_k}^{\omega_t^k} \mathrm{grad} f(\omega_k), \mathcal{T}_{\omega_k}^{\omega_t^k} \mathrm{grad} f(\omega_k) - \mathcal{T}_{\omega^*}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega^*)) \\
= \; & 2\mathbb{E}_{i_t^k}\left[\|\mathrm{grad} f_{i_t^k}(\omega_t^k) - P^{\omega_t^k \leftarrow \omega^*} \mathrm{grad} f_{i_t^k}(\omega^*) + P^{\omega_t^k \leftarrow \omega^*} \mathrm{grad} f_{i_t^k}(\omega^*) - \mathcal{T}_{\omega^*}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega^*)\|^2\right] \\
& + 2\mathbb{E}_{i_t^k}\left[\|\mathcal{T}_{\omega_k}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega_k) - \mathrm{grad} f_{i_t^k}(\omega_t^k) + \mathrm{grad} f_{i_t^k}(\omega_t^k) - \mathcal{T}_{\omega^*}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega^*)\|^2\right] \\
& - 2\|\mathrm{grad} f(\omega_k)\|^2 \\
\leq \; & 4\mathbb{E}_{i_t^k}\left[\|\mathrm{grad} f_{i_t^k}(\omega_t^k) - P^{\omega_t^k \leftarrow \omega^*} \mathrm{grad} f_{i_t^k}(\omega^*)\|^2\right] \\
& + 4\mathbb{E}_{i_t^k}\left[\|P^{\omega_t^k \leftarrow \omega^*} \mathrm{grad} f_{i_t^k}(\omega^*) - \mathcal{T}_{\omega^*}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega^*)\|^2\right] \\
& + 4\mathbb{E}_{i_t^k}\left[\|\mathcal{T}_{\omega_k}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega_k) - \mathrm{grad} f_{i_t^k}(\omega_t^k)\|^2\right] \\
& + 4\mathbb{E}_{i_t^k}\left[\|\mathrm{grad} f_{i_t^k}(\omega_t^k) - \mathcal{T}_{\omega^*}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega^*)\|^2\right] \\
\leq \; & 4c_2^2(\mathrm{dist}(\omega_t^k, \omega^*)) + 4\tau_2^2 C^2 c_1^2(\mathrm{dist}(\omega_k, \omega^*)) \\
& + 4\mathbb{E}_{i_t^k}\left[\|\mathcal{T}_{\omega_k}^{\omega_t^k} \mathrm{grad} f_{i_t^k}(\omega_k) - P^{\omega_t^k \leftarrow \omega_k} \mathrm{grad} f_{i_t^k}(\omega_k) + P^{\omega_t^k \leftarrow \omega_k} \mathrm{grad} f_{i_t^k}(\omega_k) - \mathrm{grad} f_{i_t^k}(\omega_t^k)\|^2\right] \\
& + 2[4c_2^2(\mathrm{dist}(\omega_t^k, \omega^*)) + 4\tau_2^2 C^2 c_1^2(\mathrm{dist}(\omega_k, \omega^*))] (\text{evaluated from the second equality}) \\
\leq \; & 12[c_2^2(\mathrm{dist}(\omega_t^k, \omega^*)) + \tau_2^2 C^2 c_1^2(\mathrm{dist}(\omega_k, \omega^*))] \\
& + 2[4\tau_2^2 C^2 c_1^2(\mathrm{dist}(\omega_k, \omega_t^k)) + 4c_2^2(\mathrm{dist}(\omega_k, \omega_t^k)] \\
\leq \; & 4(c_2^2 + \tau_2^2 C^2 c_1^2)(3(\mathrm{dist}(\omega_t^k, \omega^*))^2 + 2(\mathrm{dist}(\omega_k, \omega_t^k))^2) \\
\leq \; & 4(c_2^2 + \tau_2^2 C^2 c_1^2)(7(\mathrm{dist}(\omega_t^k, \omega^*))^2 + 4(\mathrm{dist}(\omega_k, \omega^*))^2), \qquad (6.6)
\end{aligned}
$$

where the last inequality comes from

$$(\mathrm{dist}(\omega_k, \omega_t^k))^2 \leq (\mathrm{dist}(\omega_k, \omega^*) + \mathrm{dist}(\omega_t^k, \omega^*))^2 \leq 2(\mathrm{dist}(\omega_k, \omega^*))^2 + 2(\mathrm{dist}(\omega_t^k, \omega^*))^2.$$

□

6.3.3 Local Convergence Rate Theorem

Theorem 6.2. *Let $\omega^* \in \mathcal{M}$ be a non-degenerate local minimizer of f. Suppose Assumptions 6.1, 6.2, 6.3, 6.4 and 6.6 hold for Algorithm 6. Let α be a positive number satisfying $a_3 a_0^2 \tau_1^2 > 14\alpha a_1 a_4^2 (c_2^2 + \tau_2^2 C^2 c_1^2)$. For any sequence $\{\omega_k\}$ generated by the algorithm converging to ω^* with Option I-A under a fixed step-size $\alpha_t^k = \alpha$, there exists $0 < K^* < K$ such that for all $k > K^*$,*

$$\mathbb{E}[f(\omega_{k+1}) - f(\omega^*)] \leq [1 - 2\alpha a_0 a_3 + \frac{4\alpha^2 a_1 a_4^2}{a_0 \tau_1^2}(4m_k + 7)(c_2^2 + \tau_2^2 C^2 c_1^2)]\mathbb{E}[f(\omega_k) - f(\omega^*)],$$

where a_0, a_1 from Lemma 6.1, a_3, a_4 from Lemma 6.6, τ_1, τ_2 from Lemma 6.10, c_1, c_2, C from Lemma 6.11.

Proof. From Lemma 6.8,

$$\mathbb{E}_{i_t^k}[f(\omega_{t+1}^k)] - f(\omega_t^k) \leq -2\alpha a_3 a_0 (f(\omega_t^k) - f(\omega^*)) + \frac{1}{2}\alpha^2 a_1 a_4^2 \mathbb{E}_{i_t^k}[\|\zeta_t^k\|^2].$$

Noting that $\|\mathrm{grad} f(\omega^*)\| = 0$, from the left side of Lemma 6.2 and Lemma 6.10,

$$f(\omega_t^k) - f(\omega^*) \geq \frac{a_0}{2}\|R_{\omega^*}^{-1}(\omega_t^k)\|^2 \geq \frac{a_0 \tau_1^2}{2}(\mathrm{dist}(\omega_t^k, \omega^*))^2.$$

With these two inequalities and Lemma 6.11, we have

$$\begin{aligned} \mathbb{E}_{i_t^k}[f(\omega_{t+1}^k)] - f(\omega_t^k) \quad \leq \quad & \left[-\alpha a_3 a_0^2 \tau_1^2 + 14\alpha^2 a_1 a_4^2 (c_2^2 + \tau_2^2 C^2 c_1^2)\right] (\mathrm{dist}(\omega_t^k, \omega^*))^2 \\ & + 8\alpha^2 a_1 a_4^2 (c_2^2 + \tau_2^2 C^2 c_1^2)(\mathrm{dist}(\omega_k, \omega^*))^2. \end{aligned} \tag{6.7}$$

Sum the inequalities over $t = 0, \cdots, m_k - 1$ and take the total expectation:

$$\begin{aligned} \mathbb{E}[f(\omega_{m_k}^k) - f(\omega_0^k)] \quad \leq \quad & \left[-\alpha a_3 a_0^2 \tau_1^2 + 14\alpha^2 a_1 a_4^2 (c_2^2 + \tau_2^2 C^2 c_1^2)\right] \sum_{t=0}^{m_k - 1} \mathbb{E}[(\mathrm{dist}(\omega_t^k, \omega^*))^2] \\ & + 8 m_k \alpha^2 a_1 a_4^2 (c_2^2 + \tau_2^2 C^2 c_1^2) \mathbb{E}[(\mathrm{dist}(\omega_k, \omega^*))^2] \\ \leq \quad & \left[-\alpha a_3 a_0^2 \tau_1^2 + 14\alpha^2 a_1 a_4^2 (c_2^2 + \tau_2^2 C^2 c_1^2)\right] \sum_{t=0}^{m_k - 1} \frac{2}{a_0 \tau_1^2} \mathbb{E}[f(\omega_t^k) - f(\omega^*)] \\ & + 8 m_k \alpha^2 a_1 a_4^2 (c_2^2 + \tau_2^2 C^2 c_1^2) \frac{2}{a_0 \tau_1^2} \mathbb{E}[f(\omega_k) - f(\omega^*)]. \end{aligned} \tag{6.8}$$

Note that $\omega_{m_k}^k = \omega_{k+1}$ and $\omega_0^k = \omega_k$, we have

$$\mathbb{E}[f(\omega_{m_k}^k) - f(\omega_0^k)] = \mathbb{E}[f(\omega_{k+1}) - f(\omega^*) + f(\omega^*) - f(\omega_k)] = \mathbb{E}[f(\omega_{k+1}) - f(\omega^*)] - \mathbb{E}[f(\omega_k) - f(\omega^*)].$$

Also note that $-a_3 a_0^2 \tau_1^2 + 14\alpha a_1 a_4^2 (c_2^2 + \tau_2^2 C^2 c_1^2) < 0$, the desired inequality follows by rearranging the above inequality. □

6.3.4 Remark

The convergence rate and the required α are related to the inverse Hessian approximation bounds a_3 and a_4 that are in turn related to the bounds on $\det(\hat{\mathcal{B}}_k)$ and $\text{trace}(\hat{\mathcal{B}}_k)$. Given a strategy to choose $\phi_i^{(k)}$ in the full Broyden family, theoretically it would be possible to change the value of these bounds so that the α or the rate can be changed for better performance.

Consider the bound for stepsize α:

$$\alpha < \frac{a_3 a_0^2 \tau_1^2}{14 a_1 a_4^2 (c_2^2 + \tau_2^2 C^2 c_1^2)} := \Gamma \frac{a_3}{a_4^2},$$

where Γ is a constant, a_3, a_4 bound the Hessian approximation operator $\mathcal{H}_k$ and $\mathcal{H}_t^k$ as shown in Lemma 6.7. Given the bounds M_3, M_4 in Lemma 6.6 that satisfy $\text{trace}(\hat{\mathcal{B}}_k) \leq M_3$, $\det(\hat{\mathcal{B}}_k) \geq M_4$, it follows that:

$$a_3 = 1/M_3,$$

$$a_4 = (M_3)^{L-1}/M_4,$$

where m is the dimension of $\mathcal{M}$. This results are based on the sum and the product of the eigenvalues equal the trace and the determinant of a matrix respectively. We can have a wider range of stepsize if $\det(\hat{\mathcal{B}}_k)$ has a larger lower bound and $\text{trace}(\hat{\mathcal{B}}_k)$ has a smaller upper bound.

Observing the update expression for $\det(\hat{\mathcal{B}}_k)$ that $\mathcal{B}_k \equiv \mathcal{B}_k^L$:

$$\det(\hat{\mathcal{B}}_k^L) \geq \det(\hat{\mathcal{B}}_k^{L-1}) \frac{g(y_{k-1}, s_{k-1})}{g(s_{k-1}, \mathcal{B}_k^{L-1} s_{k-1})} \nu \geq \cdots,$$

where BFGS has $\nu = 1$ while Broyden has $\nu \in (0, 1]$ so that BFGS gives the largest lower bound for the determinant. Meanwhile, the trace expression is a monotone function of $\phi_i^{(k)}$ based on (3.1), which means the upper bound for the trace becomes smaller with a negative $\phi_i^{(k)}$ than the bound for BFGS. Note that the trace upper bound M_3 has power term in the expression, its impact may be more weighted in the bound of α. Although the proof is not rigorous, it suggests that LR-SBroyden-VR may accept a wider range of stepsize in the performance evaluation experiments.

CHAPTER 7

EXPERIMENTS OF RIEMANNIAN STOCHASTIC BROYDEN FAMILY OF QUASI-NEWTON METHODS

In this chapter, we conduct a comprehensive comparative analysis between LR-SBroyden-VR and the existing limited-memory Riemannian stochastic BFGS in the work of Kasai et al. [30], denoted as LR-SBFGS-VR. The Broyden parameter selection adheres to the principles established by Davidon, where $\phi_i^{(k)}$ represents the selected parameter value. Additionally, a hybrid selection strategy, as expounded in Chapter 5, is incorporated into LR-SBroyden-VR. All quasi-Newton methods are employed in mini-batched stochastic iterations, sharing a unified approach to the generation of curvature pairs. This generation process occurs exclusively at the outer k-loop, as outlined in Algorithm 6. The comparative analysis aims to provide insights into the performance and efficiency of LR-SBroyden-VR in comparison to LR-SBFGS-VR under various optimization scenarios.

An important observation highlighted in Chapter 5 of LRBroyden is its capacity to accommodate larger stepsizes than LRBFGS during the line search procedure. This characteristic proves to be particularly significant in the context of stochastic variance-reduced algorithms, where a fixed stepsize is employed. The ability to accept larger steps while maintaining stability is crucial, potentially leading to faster convergence rates in scenarios where a fixed stepsize might be excessively conservative. In the stochastic variance-reduced algorithm, where a fixed stepsize is essential, LR-SBroyden-VR's adaptability to larger steps emerges as a pivotal factor, enhancing its overall performance in stochastic optimization problems.

In stochastic variance-reduced algorithms, the determination of certain parameters often involves empirical experimentation. In addition to the stepsize and memory size, the selection of batch size and frequency value (referred to as stochastic parameters) introduce various trade-offs between per-iteration costs and per-iteration improvement. The numerical results presented in this chapter provide compelling evidence that the choice of these stochastic parameters significantly impacts the performance of stochastic quasi-Newton methods. We provide heuristic suggestions for

selecting these parameters for the proposed algorithm, helping guide the effective implementation of the algorithm in practical stochastic optimization scenarios.

In Section 7.2, a comprehensive metric is introduced to assess the computational costs associated with Riemannian stochastic methods. Section 7.3 provides details on a series of meticulously designed experiments. These experiments are carefully crafted to isolate and address various parameters influencing the performance of Riemannian stochastic quasi-Newton algorithms. Finally, Section 7.4 presents a collection of experiments of three distinct algorithms: R-SVRG, LR-SBFGS-VR, and LR-SBroyden-VR. These experiments aim to provide a thorough understanding of the algorithms' behavior under different scenarios and illustrate their comparative strengths and weaknesses.

7.1 Test Problems

We consider three problems in the experiments: principal component analysis (PCA) on the Grassmann manifold, the joint diagonalization problem on the Stiefel manifold and the low-rank matrix completion problem on the Grassmann manifold. These problems are rewritten into the form of a finite-sum cost function, as defined in Equation 6.1.

7.1.1 PCA Problem

The PCA problem is to minimize the sum of squared residual error between projected data points and the original data as follows:

$$\min_{U \in \mathrm{St}(r,m)} \frac{1}{n} \sum_{i=1}^{n} \|z_i - UU^T z_i\|_2^2, \tag{7.1}$$

where z_i is a data vector of size $m \times 1$ and U is an orthonormal matrix projector. The problem is equivalent to

$$\min_{U \in \mathrm{St}(r,m)} -\frac{1}{n} \sum_{i=1}^{n} z_i^T UU^T z_i. \tag{7.2}$$

Meanwhile, $U \mapsto UO$ for all orthogonal matrices $O \in \mathcal{O}(r)$ have the same cost function value. This makes the problem an optimization problem on the Grassmann manifold $\mathrm{Gr}(r, m)$ (see [51]).

The synthetic matrix $Z \in \mathbb{R}^{m \times n} (m \ll n)$ storing z_i is generated by the following procedure. First, we generate a $m \times n$ matrix A with elements independently drawn from the standard normal distribution. The data is then centered by subtracting the mean of each row from the matrix A. Then a Singular Value Decomposition (SVD) is performed on the centralized matrix, resulting

in two orthogonal matrices $U_* \in \mathbb{R}^{m\times m}$ containing left singular vectors, $V_* \in \mathbb{R}^{n\times n}$ containing right singular vectors and a rectangular diagonal matrix $S \in \mathbb{R}^{m\times n}$. We construct the rectangular diagonal matrix S_{syn} and generate the synthetic matrix by $Z := U_* S_{syn} V_*^T$. The initial point is set as $\mathrm{ORTH}(U_*(:, 1:r) + randn(m, r))$ in Matlab, where $U_*(:, 1:r)$ stores the r leading eigenvectors of Z, representing the true solution.

7.1.2 ICA Problem

The second test problem is the joint diagonalization on the Stiefel manifold. This is a preprocessing step widely used in Independent Component Analysis (ICA) or the blind source separation problem. The optimization problem defined on the Stiefel manifold is

$$\min_{U\in \mathrm{St}(r,m)} -\frac{1}{n}\sum_{i=1}^{n} \|\mathrm{diag}(U^T C_i U)\|_F^2, \tag{7.3}$$

where $\|\mathrm{diag}(A)\|_F^2$ returns the sum of the squared diagonal elements of A, and C_i are known symmetric matrices (see [56, 50]).

The matrices C_i are selected as $C_i = \mathrm{diag}(m, m-1, \cdots, 1) + \epsilon_C(R_i + R_i^T)$, where the elements of $R_i \in \mathbb{R}^{m\times m}$ are independently drawn from the standard normal distribution. The initial point is randomly generated as an $m \times r$ matrix with orthonormal column.

7.1.3 Low-rank Matrix Completion Problem

The low-rank matrix completion problem is considered on the Grassmann manifold. Partitioning $A = [a_1, \cdots, a_n]$ so that the problem is well-defined for using a stochastic method. The problem is defined as follows:

$$\begin{aligned} &\min_{X\in\mathbb{R}_r^{m\times n}} \|P_\Omega X - P_\Omega A\|_F^2 \\ &= \min_{U\in\mathbb{R}^{m\times r}, W\in\mathbb{R}^{r\times n}} \|P_\Omega(UW) - P_\Omega A\|_F^2 \\ &= \min_{U\in\mathbb{R}^{m\times r}, w_i\in\mathbb{R}^r} \frac{1}{n}\sum_{i=1}^{n} \|P_{\Omega_i}(Uw_i) - P_{\Omega_i}(a_i)\|_2^2, \end{aligned} \tag{7.4}$$

where P_{Ω_i} is the sampling operator for the i-th column. Given U, w_i admits a closed-form solution [6]. As a consequence, the problem depends only on the column space of U, and is on the Grassmann manifold (see [29]).

We generate $A_L \in \mathbb{R}^{m\times r}$ and $A_R \in \mathbb{R}^{r\times n}$ with i.i.d. standard normal entries and then compute the thin SVD of the product $A_L A_R = U_* S V_*^T$ where $U_* \in \mathbb{R}^{m\times r}$, $S \in \mathbb{R}^{r\times r}$ and $V_* \in \mathbb{R}^{r\times n}$. The elements in the diagonal matrix S are then replaced with values that satisfy the condition number,

denoted as CN–the ratio of the maximal to the minimal singular values of the matrix. The known index set Ω is uniformly sampled at random among all sets of $|\Omega|$, which yields the oversampling ratio OS. The initial point is randomly generated as an $m \times r$ matrix with orthonormal columns.

7.2 Algorithm Parameters and Evaluation Criteria

In the experiments, three different algorithms are employed: R-SVRG, LR-SBFGS-VR and LR-SBroyden-VR (Broyden parameters chosen as Davidon's choice and the hybrid strategy). All these stochastic optimization methods incorporate the variance reduction method proposed in [60]. The experimentation involves a fixed stepsize α, with the batch size b determining the number of gradient components calculated at each iteration. Additionally, the frequency value m_k is employed to specify how often the entire gradient and curvature information are calculated and updated. The choice of these two stochastic parameters, b and m_k, is determined empirically, following a methodology similar to existing stochastic optimization algorithms. The performance evaluation is conducted as a function of α and L, with these values chosen from the predefined sets as outlined below.

- Batch size: b is chosen from $\{1, 10, 20\}$;
- Frequency value: m_k is chosen from $\{n/20, n/10, n/5, n, 5n\}$;
- Stepsize: α is tuned from $\{10^{-1}, 5 \times 10^{-2}, 10^{-2}, 5 \times 10^{-3}, 10^{-3}\}$ as a fixed number;
- Memory size: L is chosen from $\{1, 2, 4, 8\}$.

In a single iteration, which comprises one k-loop iteration and m_k t-loop iterations, both LR-SBroyden-VR and LR-SBFGS-VR involve the following computational steps: 1 evaluation of a full gradient, m_k evaluations of a stochastic Euclidean gradient with batch size b, $m_k + 1$ projections of these gradients onto the manifold, m_k evaluations of a retraction, 1 evaluation of an inverse retraction, $m_k + 2L$ evaluations of a vector transport, m_k evaluations of an inverse vector transport, m_k applications of the inverse Hessian approximation $\mathcal{H}_k \zeta_t$, 1 computation of $\mathcal{H}_k$ preparation. Notice that the total sum index n is usually much larger than the manifold dimension, and the parameter m_k is related to the value of n.

Suppose the algorithm operates on $\mathrm{St}(r, m)$ and $\mathrm{Gr}(r, m)$ with qf retraction, parallelization transport and the intrinsic representation. Let J denote the dominant computation for each component's Euclidean gradient, as indicated in Table 7.1. Consequently, a full Euclidean gradient

Table 7.1: Dominant computations of Euclidean gradient for each test problem.

Problem	low-rank matrix completion	PCA	joint diagonalization
J	$OS \cdot nr + 2m^2r$	$4mr$	$2m^2r$

Table 7.2: Dominant computations of the updates in LR-SBroyden-VR.

	grad$f$	gradf_{i_t}	R	$\mathcal{T}\&\mathcal{T}^{-1}$	$\mathcal{H}_k\zeta_t$
operation	$nJ + 8mr^2 - 2r^3$	$bJ + 8mr^2 - 2r^3$	$10mr^2 - 14r^3/3$	-	$8Ld$
multiplicity	1	m_k	m_k	$2m_k + 2L$	m_k

and a stochastic Euclidean gradient require nJ and bJ computations, respectively. Table 7.2 presents the dominant computations and their multiplicities in LR-SBroyden-VR, where all the terms shown are related to n. Notice that d represents the intrinsic dimension of the manifold, where $d = mr - r(r+1)/2$ for Stiefel manifold and $d = r(m-r)$ for Grassmann manifold.

In the experiments, the computational cost is measured by aggregating the major operations performed during the iterates and then dividing this sum by nJ. This measurement is denoted as the number of equivalent Euclidean gradient computations, represented as $\#nJ$ in the following numerical results. It serves as an indicator of the amount of work required by each method in comparison to the Euclidean non-stochastic gradient-based optimization algorithms. In Euclidean space, $\#\text{grad}/n$ is a useful metric as it signifies the number of passes needed to traverse the entire dataset. However, $\#nJ$ provides more accurate reflection of the total work performed on a Riemannian manifold. Finally, the stopping criteria are set as $\|gf_f\|/\|gf_0\| < 10^{-6}$ or the maximum value of $\#nJ$ is reached. The maximum $\#nJ$ threshold, is dependent on the choices of problem and parameter settings.

7.3 Parameter Selection

The experiments in this section are designed to determine the appropriate stochastic parameters batch size b and frequency value m_k for each algorithm and problem. The problems are set with a fixed size of $n = 1000$. The hypothesis is batch size $b = 10$ would yield the most stable and robust performance when compared to alternatives like $b = 1, 20$. Frequency value m_k plays a crucial role in the quality of the curvature information within stochastic quasi-Newton methods. The hypothesis is that choosing a smaller m_k than the typical choices for R-SVRG ($2n$ for convex and $5n$ for nonconvex as indicated in [28]) may lead to improved performance in stochastic quasi-

Newton methods. For the LR-SBroyden-VR algorithm, we opt for the Broyden parameter choice of Davidon's $\phi_i^{(k)}$, denoted as LR-SDavidon-VR. This choice of Broyden parameter is observed to be robust and effective in the non-stochastic LRBroyden family of methods. To test our hypotheses, we proceed to compare the performance of R-SVRG, LR-SBFGS-VR and LR-SDavidon-VR across different problems.

7.3.1 Knowledge of Stochastic Parameters

The practical success of stochastic gradient-based methods has been demonstrated in many domains in recent years. However, theoretical foundations supporting the idea that different choices of stepsize, batch size, and frequency size can result in significantly different behaviors are not extensively established. This consideration has received substantial attention in the context of the Stochastic Gradient Descent (SGD) method, as evident in work [19]. These papers primarily focused on the phenomenon of heavy-tail behavior caused by the noise in SGD gradients, which leads to slow convergence as the iterates approach a minimizer. The ratio of stepsize to batch size controls the magnitude of scaling the stochastic noise, specifically, a higher ratio leads to a higher probability to escape from the basin near the minimum.

As discussed in Section 6.1, the gradient variance of SGD imposes a limitation on the stepsize, which must be a decaying sequence. This constraint leads to a reduction in the convergence rate. It is noteworthy that this restriction also applies to some widely used adaptive stochastic gradient methods, such as AdaGrad, Adam, AMSGrad both in Euclidean space and on Riemannian manifold. In the pursuit of faster convergence, the SVRG method introduced in Section 6.1.3 has emerged as an empirically successful approach, allowing for a constant stepsize. This method effectively accelerates the convergence rate compared to the methods with decaying stepsize and also is suitable for serving as the basis for proposing stochastic quasi-Newton methods. The fixed value of stepsize, however, is typically chosen empirically.

In the first SVRG paper by Johnson and Zhang [28], it was recommended to set the frequency value m_k to $2n$ for convex problems and $5n$ for nonconvex problems. However, the authors did not provide explicit reasons for these choices. Subsequent papers, including LR-SBFGS-VR [30], simply adopted these recommendations without conducting further exploration. One possible reason for this could be the desire to set it to a safe value, given that there exists a theoretical lower bound for m_k in the nonconvex problems. This bound was established by Reddi et. al [44] in the relationships

between the stepsize, batch size and frequency size as follows:

$$b < n^{2/3}, \alpha = \mu b/(L^* n^{2/3}),\ m_k = \lfloor n/(3b\mu) \rfloor, \tag{7.5}$$

where α, b, m_k represent the stepsize, batch size and frequency size, respectively. $\mu \in (0, 1)$ is a constant, and L^* is Lipschitz constant for continuous gradients. It is important to note that the stepsize and frequency size of SVRG depend on n. In the experiments conducted in [44], the choices were $b = 10, m_k = n/b$ and a best-tuned α.

Despite the progress in understanding the relationships between batch size b, frequency value m_k and stepsize α for stochastic quasi-Newton methods both in Euclidean space and on Riemannian manifold, the optimal choices for these parameters remain open questions. Most findings in the stochastic quasi-Newton literature are based on heuristic evidence derived from empirical experimentation. In the following experiments, we present the empirical evidence that help us assess our hypotheses regarding the selection of these parameters. The preferred parameter settings are used in the subsequent comparison of LR-SBroyden-VR, LR-SBFGS-VR and R-SVRG.

7.3.2 Experiments to Determine b and m_k

In this section, we present the results of our investigation into the appropriate batch size b and frequency value m_k for LR-SDavidon-VR, LR-SBFGS-VR and R-SVRG in the three test problems. The stochastic quasi-Newton algorithms are tested with memory sizes $L = 2$ and $L = 4$, as these two choices demonstrated good performances. It is important to note that the algorithm cannot not always find sufficiently accurate solution within the prescribed amount of work. The results are obtained from 10 runs, each with 5 different random initial points, resulting in a total of 50 runs.

The values presented in the following tables represent the average $\#nJ$ calculated from the successful runs that reached $|gf_f|/|gf_0| < 10^{-6}$ within the prescribed amount of work, where $\#nJ$ is defined in Section 7.2. The values in the parentheses indicate the count of successful runs out of total 50. For example, "- (0)" means all 50 runs do not find the sufficiently accurate solution. The entries highlighted in blue in each table represent the minimum $\#nJ$ values that achieved a 100% success rate and are considered the best performances. Note that the metric $\#nJ$, which measures operation counts and computational costs, is machine independent. Therefore, it provides a more reliable measure than computational time, as it is not affected by variations in machine performance. This makes $\#nJ$ a valuable metric for assessing algorithm efficiency and comparing performance in this dissertation.

PCA

The synthetic PCA problem is generated with $m = 50, n = 1000, r = 5$ following instructions in Section 7.1, and S_{syn} has the diagonal elements $\{100, 80, 60, 40, 5, 1 \cdots 1\}$. Tables 7.3, 7.4 and 7.5 show the results for stochastic quasi-Newton algorithms with $L = 2$, $L = 4$ and R-SVRG respectively. These tables provide insights into the performance of these algorithms on the synthetic PCA problem under different memory and parameter settings.

The best batch size b for both LR-SBFGS-VR and LR-SDavidon-VR is found to be $b = 10$, while a batch size of $b = 1$ performs poorly due to the lack of information from each stochastic gradient. A larger batch size $b = 20$ does not lead to a decrease in the $\#nJ$ value as each evaluation of the stochastic gradient becomes more expensive. For the frequency value, a smaller m_k generally leads to higher success rates overall, with LR-SDavidon-VR achieving its best performance at $b = 10$, $m_k = n/10$. LR-SBFGS-VR performs best at $b = 10$, $m_k = n/10$ with $L = 2$ and at $b = 10$, $m_k = n/20$ with $L = 4$. Interestingly, increasing the memory size from $L = 2$ to $L = 4$ has a negative impact on LR-SBFGS-VR's performance.

For R-SVRG, the best performances were observed at $b = 1$ and $m_k = n$ in the PCA problem. Larger step sizes, such as $\alpha = 0.5$, were also tested but cannot find sufficiently accurate solution in most cases. Generally, smaller frequency values appear to be more appropriate for the two stochastic quasi-Newton algorithms compared to R-SVRG. This is mainly due to the requirement of updating curvature pairs in the stochastic quasi-Newton algorithms.

Fixing the dimension of the manifold as $m = 50$ and $r = 5$, the second experiment presented in Table 7.6 is conducted with $n = 10000$. The batch size $b = 10$ remains a suitable choice, but increasing it to $b = 20$ tends to produce better performance. This is reasonable because with a larger problem size, a larger batch size provides more information for each stochastic gradient. Even though the batch size increases, the proportion of stochastic components to the total decreases from 10/1000 to 20/10000. The best performing frequency value m_k increases from $1000/10 = 100$ to $10000/50 = 200$, compensating for the fact that each stochastic gradient accounts for fewer components in total. These findings indicate that the choice of parameters batch size and frequency value should be adapted to the specific problem size to achieve optimal performance.

ICA

The synthetic ICA problem is generated with $m = 10, n = 1000, r = 9$ following instructions in Section 7.1, and $\epsilon_C = 0.1$. Tables 7.7, 7.8 and 7.9 show the results for the stochastic quasi-newton

Table 7.3: Comparison of LR-SBFGS-VR and LR-SDavidon-VR for the PCA problem with $n = 1000$ and different b, m_k. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

$L=2$		$m_k = n$			$m_k = n/5$		
	stepsize	5_{-2}	1_{-2}	5_{-3}	5_{-2}	1_{-2}	5_{-3}
$b=1$	LR-SBFGS-VR	- (0)	859 (2)	743 (12)	- (0)	502 (44)	490 (49)
	LR-SDavidon-VR	- (0)	754 (6)	737 (24)	- (0)	464 (50)	559 (50)
$b=10$	LR-SBFGS-VR	- (0)	630 (16)	681 (31)	- (0)	387 (50)	740 (50)
	LR-SDavidon-VR	- (0)	598 (31)	670 (34)	647 (22)	320 (50)	442 (50)
$b=20$	LR-SBFGS-VR	- (0)	665 (16)	764 (33)	671 (4)	427 (50)	897 (50)
	LR-SDavidon-VR	- (0)	658 (23)	789 (35)	620 (9)	350 (50)	478 (50)
$L=2$		$m_k = n/10$			$m_k = n/20$		
	stepsize	5_{-2}	1_{-2}	5_{-3}	5_{-2}	1_{-2}	5_{-3}
$b=1$	LR-SBFGS-VR	301 (1)	399 (50)	563 (49)	619 (8)	443 (50)	599 (50)
	LR-SDavidon-VR	482 (6)	421 (50)	561 (50)	434 (39)	426 (49)	633 (50)
$b=10$	LR-SBFGS-VR	512 (15)	335 (50)	559 (50)	426 (35)	352 (50)	578 (50)
	LR-SDavidon-VR	471 (39)	277 (50)	460 (50)	265 (50)	299 (50)	670 (50)
$b=20$	LR-SBFGS-VR	551 (10)	540 (50)	613 (50)	486 (38)	411 (50)	637 (50)
	LR-SDavidon-VR	470 (36)	312 (50)	519 (50)	357 (50)	340 (50)	763 (50)

Table 7.4: Comparison of LR-SBFGS-VR and LR-SDavidon-VR for the PCA problem with $n = 1000$ and different b, m_k. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

$L=4$		$m_k = n$			$m_k = n/5$		
	stepsize	5_{-2}	1_{-2}	5_{-3}	5_{-2}	1_{-2}	5_{-3}
$b=1$	LR-SBFGS-VR	- (0)	679 (1)	791 (11)	- (0)	388 (9)	520 (27)
	LR-SDavidon-VR	- (0)	772 (2)	827 (9)	- (0)	507 (48)	547 (50)
$b=10$	LR-SBFGS-VR	- (0)	618 (10)	783 (35)	- (0)	340 (32)	774 (49)
	LR-SDavidon-VR	- (0)	602 (31)	906 (50)	463 (8)	410 (50)	503 (50)
$b=20$	LR-SBFGS-VR	- (0)	652 (15)	773 (30)	- (0)	319 (42)	891 (49)
	LR-SDavidon-VR	- (0)	689 (21)	781 (26)	718 (10)	393 (50)	581 (50)
$L=4$		$m_k = n/10$			$m_k = n/20$		
	stepsize	5_{-2}	1_{-2}	5_{-3}	5_{-2}	1_{-2}	5_{-3}
$b=1$	LR-SBFGS-VR	- (0)	322 (14)	706 (50)	- (0)	419 (37)	603 (50)
	LR-SDavidon-VR	239 (1)	385 (50)	607 (50)	534 (11)	429 (50)	758 (50)
$b=10$	LR-SBFGS-VR	- (0)	386 (39)	633 (50)	108 (5)	388 (46)	541 (50)
	LR-SDavidon-VR	452 (18)	307 (50)	741 (50)	371 (50)	418 (50)	801 (50)
$b=20$	LR-SBFGS-VR	389 (1)	406 (33)	750 (50)	523 (1)	454 (45)	585 (50)
	LR-SDavidon-VR	480 (23)	332 (50)	861 (50)	393 (46)	522 (50)	906 (50)

Table 7.5: Comparison of R-SVRG for the PCA problem with $n = 1000$ and different b, m_k. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

R-SVRG	$m_k = 5n$			$m_k = n$		
stepsize	1_{-1}	5_{-2}	1_{-2}	1_{-1}	5_{-2}	1_{-2}
b=1	2590 (40)	2063 (49)	2414 (26)	1024 (50)	1624 (43)	2353 (16)
b=10	1724 (50)	1711 (46)	1977 (22)	1141 (47)	1251 (45)	2007 (29)
b=20	2089 (46)	1966 (28)	2148 (23)	1317 (43)	1475 (39)	2062 (25)
R-SVRG	$m_k = n/5$			$m_k = n/10$		
stepsize	1_{-1}	5_{-2}	1_{-2}	1_{-1}	5_{-2}	1_{-2}
b=1	1424 (38)	1848 (11)	2145 (3)	1853 (17)	2279 (2)	2512 (2)
b=10	1823 (17)	1722 (13)	1796 (37)	- (0)	2825 (1)	2189 (27)
b=20	2056 (3)	1685 (2)	2325 (25)	- (0)	- (0)	2161 (24)

Table 7.6: Comparison of LR-SBFGS-VR and LR-SDavidon-VR for the PCA problem with $n = 10000$ and different b, m_k. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

$L = 2$		$m_k = n/10$		$m_k = n/50$		$m_k = n/100$	
	stepsize	1_{-2}	5_{-3}	1_{-2}	5_{-3}	1_{-2}	5_{-3}
$b = 1$	LR-SBFGS-VR	389 (22)	197 (50)	156 (50)	121 (50)	134 (50)	208 (50)
	LR-SDavidon-VR	371 (47)	178 (50)	110 (50)	143 (50)	159 (50)	196 (50)
$b = 10$	LR-SBFGS-VR	231 (50)	123 (50)	76 (50)	173 (50)	134 (50)	174 (50)
	LR-SBroyden-VR	145 (50)	110 (50)	69 (50)	106 (50)	95 (50)	144 (50)
$b = 20$	LR-SBFGS-VR	237 (50)	141 (50)	88 (50)	194 (50)	137 (50)	175 (50)
	LR-SDavidon-VR	176 (50)	137 (50)	61 (50)	106 (50)	91 (50)	147 (50)
$b = 50$	LR-SBFGS-VR	296 (47)	180 (50)	91 (50)	264 (50)	186 (50)	195 (50)
	LR-SDavidon-VR	222 (50)	160 (50)	83 (50)	126 (50)	102 (50)	161 (50)
$L = 4$		$m_k = n/10$		$m_k = n/50$		$m_k = n/100$	
	stepsize	1_{-2}	5_{-3}	1_{-2}	5_{-3}	1_{-2}	5_{-3}
$b = 1$	LR-SBFGS-VR	133 (9)	201 (25)	296 (16)	270 (40)	327 (34)	259 (50)
	LR-SDavidon-VR	469 (34)	207 (50)	119 (50)	141 (50)	117 (50)	212 (50)
$b = 10$	LR-SBFGS-VR	117 (24)	145 (44)	213 (45)	175 (50)	254 (50)	203 (50)
	LR-SBroyden-VR	162 (50)	119 (50)	81 (50)	107 (50)	95 (50)	222 (50)
$b = 20$	LR-SBFGS-VR	172 (32)	188 (48)	242 (49)	195 (50)	175 (49)	203 (50)
	LR-SDavidon-VR	168 (50)	128 (50)	72 (50)	118 (50)	91 (50)	238 (50)
$b = 50$	LR-SBFGS-VR	220 (31)	194 (46)	123 (48)	209 (50)	186 (49)	225 (50)
	LR-SDavidon-VR	202 (50)	170 (50)	84 (50)	138 (50)	101 (50)	283 (50)

Table 7.7: Comparison of LR-SBFGS-VR and LR-SDavidon-VR for the ICA problem with different b, m_k. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

$L=2$		$m_k = n$			$m_k = n/5$		
	stepsize	5_{-2}	1_{-2}	5_{-3}	5_{-2}	1_{-2}	5_{-3}
$b=1$	LR-SBFGS-VR	- (0)	187 (1)	774 (24)	- (0)	187 (1)	774 (24)
	LR-SDavidon-VR	- (0)	644 (5)	553 (47)	- (0)	305 (48)	217 (50)
$b=10$	LR-SBFGS-VR	- (0)	310 (23)	567 (48)	499 (1)	286 (47)	284 (50)
	LR-SDavidon-VR	- (0)	407 (45)	399 (50)	209 (3)	192 (49)	222 (50)
$b=20$	LR-SBFGS-VR	- (0)	387 (30)	599 (46)	102 (2)	249 (48)	347 (50)
	LR-SDavidon-VR	- (0)	447 (39)	487 (50)	325 (9)	235 (49)	291 (50)
$L=2$		$m_k = n/10$			$m_k = n/20$		
	stepsize	5_{-2}	1_{-2}	5_{-3}	5_{-2}	1_{-2}	5_{-3}
$b=1$	LR-SBFGS-VR	- (0)	395 (39)	299 (50)	- (0)	319 (46)	326 (50)
	LR-SDavidon-VR	- (0)	215 (50)	268 (50)	- (0)	223 (49)	305 (50)
$b=10$	LR-SBFGS-VR	70 (1)	192 (50)	244 (50)	309 (10)	214 (50)	350 (50)
	LR-SDavidon-VR	522 (24)	159 (50)	254 (50)	330 (49)	206 (50)	372 (50)
$b=20$	LR-SBFGS-VR	87 (3)	248 (49)	352 (49)	187 (14)	249 (48)	410 (50)
	LR-SDavidon-VR	435 (34)	198 (50)	353 (50)	239 (49)	229 (49)	438 (50)

algorithms with $L = 2$, $L = 4$ and R-SVRG respectively. It can be observed that the stochastic quasi-Newton algorithms perform the best with $b = 10$, $m_k = n/10$. A smaller batch size lowers the success rate while a larger batch size increases the value of $\#nJ$, i.e., the total computational cost. The frequency value is chosen as the best-tuned value to utilize the curvature information effectively. These findings are consistent with the observations in the PCA problem, where a batch size of 10 and a frequency value of $n/10$ are the best choices for the stochastic quasi-Newton algorithms. It appears that these parameter settings are effective across different problem types, at least for the synthetic problems tested in Section 7.4.

Low-rank Matrix Completion

The synthetic low-rank matrix completion problem is generated with $m = 50, n = 1000, r = 5, OS = 5$ and $CN = 10$ following instructions in Section 7.1. To avoid redundancy, only $b = 10$ are listed in Table 7.10 and only $m_k = n, n/10$ are listed in Table 7.11. The frequency value $m_k = n/10$ consistently performs well for the stochastic quasi-Newton methods. An obvious decrease in the success rate is observed for LR-SBFGS-VR from $L = 2$ to $L = 4$, while LR-SDavidon-VR remains more robust to a larger memory size. The rationale behind this can be explained as the self-correcting property provided by the more flexible Broyden parameter. For R-SVRG, $b = 1, m_k = n$ is the best stochastic parameter pair as highlighted in blue.

Table 7.8: Comparison of LR-SBFGS-VR and LR-SDavidon-VR for the ICA problem with different b, m_k. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

$L=4$		$m_k = n$			$m_k = n/5$		
	stepsize	5_{-2}	1_{-2}	5_{-3}	5_{-2}	1_{-2}	5_{-3}
$b=1$	LR-SBFGS-VR	- (0)	197 (1)	187 (7)	- (0)	111 (6)	122 (14)
	LR-SDavidon-VR	- (0)	279 (2)	690 (19)	- (0)	210 (17)	314 (50)
$b=10$	LR-SBFGS-VR	- (0)	254 (17)	244 (21)	- (0)	93 (24)	208 (37)
	LR-SDavidon-VR	241 (1)	602 (38)	407 (47)	- (0)	226 (49)	210 (50)
$b=20$	LR-SBFGS-VR	- (0)	307 (22)	352 (30)	106 (3)	126 (24)	280 (36)
	LR-SDavidon-VR	- (0)	497 (35)	499 (50)	129 (5)	239 (48)	315 (50)
$L=4$		$m_k = n/10$			$m_k = n/20$		
	stepsize	5_{-2}	1_{-2}	5_{-3}	5_{-2}	1_{-2}	5_{-3}
$b=1$	LR-SBFGS-VR	- (0)	97 (6)	224 (19)	- (0)	120 (8)	217 (30)
	LR-SDavidon-VR	- (0)	324 (34)	248 (50)	- (0)	263 (50)	307 (50)
$b=10$	LR-SBFGS-VR	- (0)	100 (28)	399 (38)	60 (2)	167 (32)	377 (49)
	LR-SDavidon-VR	169 (2)	163 (50)	246 (50)	187 (8)	178 (48)	325 (49)
$b=20$	LR-SBFGS-VR	77 (3)	131 (29)	346 (48)	52 (4)	222 (36)	457 (50)
	LR-SDavidon-VR	78 (4)	203 (50)	363 (50)	156 (16)	207 (50)	400 (49)

Table 7.9: Comparison of R-SVRG for the ICA problem with different b, m_k. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

R-SVRG	$m_k = 5n$			$m_k = n$		
stepsize	1_{-1}	5_{-2}	1_{-2}	1_{-1}	5_{-2}	1_{-2}
b=1	1029 (45)	728 (49)	598 (48)	274 (48)	241 (47)	491 (47)
b=10	879 (49)	780 (48)	973 (48)	273 (47)	280 (46)	399 (48)
b=20	1261 (46)	1228 (47)	1389 (47)	397 (48)	435 (45)	516 (48)
R-SVRG	$m_k = n/10$			$m_k = n/20$		
stepsize	1_{-1}	5_{-2}	1_{-2}	1_{-1}	5_{-2}	1_{-2}
b=1	203 (47)	337 (45)	891 (44)	535 (46)	1298 (36)	1591 (12)
b=10	199 (47)	274 (48)	643 (46)	402 (48)	782 (34)	1042 (19)
b=20	422 (44)	422 (46)	621 (45)	581 (40)	777 (38)	1170 (21)

Table 7.10: Comparison of LR-SBFGS-VR and LR-SDavidon-VR for the low-rank matrix completion problem with different b, m_k. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

$b=10$		$m_k=n$			$m_k=n/5$		
	stepsize	5_{-2}	1_{-2}	5_{-3}	5_{-2}	1_{-2}	5_{-3}
$L=2$	LR-SBFGS-VR	334 (2)	277 (25)	302 (40)	325 (7)	156 (50)	165 (50)
	LR-SDavidon-VR	306 (5)	274 (47)	270 (49)	217 (37)	116 (50)	142 (50)
$L=4$	LR-SBFGS-VR	- (0)	184 (8)	234 (17)	- (0)	193 (16)	171 (39)
	LR-SDavidon-VR	162 (1)	289 (34)	293 (48)	276 (10)	135 (50)	147 (50)
$b=10$		$m_k=n/10$			$m_k=n/20$		
	stepsize	5_{-2}	1_{-2}	5_{-3}	5_{-2}	1_{-2}	5_{-3}
$L=2$	LR-SBFGS-VR	238 (26)	114 (50)	171 (50)	193 (44)	135 (50)	207 (50)
	LR-SDavidon-VR	146 (48)	110 (50)	154 (50)	98 (50)	137 (50)	196 (50)
$L=4$	LR-SBFGS-VR	212 (1)	136 (31)	175 (45)	61 (2)	171 (36)	221 (47)
	LR-SDavidon-VR	229 (25)	109 (50)	177 (50)	172 (43)	142 (50)	207 (50)

Table 7.11: Comparison of R-SVRG for the low-rank matrix completion problem with different b, m_k. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

R-SVRG	$m_k=n$			$m_k=n/10$		
stepsize	1_{-1}	5_{-2}	1_{-2}	1_{-1}	5_{-2}	1_{-2}
b=1	112 (35)	79 (50)	155 (50)	154 (38)	217 (45)	514 (50)
b=10	176 (50)	199 (50)	256 (50)	296 (37)	201 (48)	312 (50)
b=20	313 (50)	347 (50)	502 (50)	473 (36)	422 (50)	460 (50)

Conclusion. Based on the experiments conducted on in this section, here are some heuristic suggestions for choosing stochastic parameters b and m_k. A batch size of around 10 appears to be a good starting point for problems with a size of $n = 1000$. Setting a batch size that is too small may result in a failed run due to insufficient gradient information, while setting a batch size that is too large can increase computational costs without significant improvements in convergence. For the frequency value, the performance of LR-SBFGS-VR and LR-SDavidon-VR improves then deteriorates as m_k decreases. A frequency value of around $n/10$ is effective in the tested cases for problems with a size of $n = 1000$. As the problem size n increases, both batch size and frequency value should be scaled up to meet the increased demands for gradient information per iteration.

These heuristic suggestions can serve as a starting point for selecting stochastic parameters in stochastic quasi-Newton algorithms. However, it is essential to keep in mind that the optimal parameter choices may vary depending on the specific problem and algorithm, so experimentation and tuning are often necessary for practical applications. In the next section, all problems with $n = 1000$ are tested. Following the above consequences and considerations, $b = 10$ and $m_k = n/10$ are selected for the stochastic quasi-Newton algorithms.

7.4 Comparison of Performances Between Stochastic Algorithms

In the previous section, we derived empirical stochastic parameter pairs for R-SVRG, LR-SBFGS-VR and LR-SDavidon-VR methods across three test problems. In this section, we assess the performance of these methods concerning stepsize α and memory size L. Drawing insights from Chapter 5, we posit that the Broyden family algorithm with Davidon's $\phi_i^{(k)}$ offer advantages in the stepsize selection. Specifically, we hypothesize that a larger stepsize is more suitable when using $\phi_i^{(k)D}$ than with $\phi = 0$ during the updates, potentially influencing the convergence speed of the stochastic quasi-Newton methods.

This section provides a comprehensive comparison of the proposed LR-SBroyden-VR with LR-SBFGS-VR ([30]) and R-SVRG([60]) with BB scaling. The $\phi_i^{(k)}$ of LR-SBroyden-VR is chosen as Davidon's choice (4.4) and the hybrid strategy (5.1) in the experiments, denoted as LR-SDavidon-VR and Hybrid respectively. The superority of LR-SDavidon-VR and Hybrid is mainly obeserved in ill-conditioned problems. An ill-conditioned problem occurs when the curvature along some directions is much higher than in others. In such cases, bouncing around is likely to occur during gradient descent, as we may overstep along certain directions. The stochatic gradient descent methods are adversely affected by ill-conditioning (see, e.g., [7]), while stochastic quasi-Newton methods

are beneficial in collecting curvature information through Hessian-vector products. Compared to using only the BFGS update, we demonstrate that an appropriate choice of Hessian approximation in the full Broyden family can provide more significant improvements over the stochastic gradient descent method.

7.4.1 Principal Components Analysis(PCA)

It is well-known that PCA boils down to computing the r eigenvectors of ZZ^T that have the largest eigenvalues, or r dominant singular vectors of Z^T. Z is defined as $Z := U_* S_{syn} V_*^T$. The difficulty of the PCA problem can be modulated by altering the r largest elements in S_{syn}. Presented below are three scenarios that are investigated for PCA problem, differing only in the values of the fifth-leading diagonal elements.

Scenario 1: $m = 50, n = 1000, r = 5, \text{diag}(S_{syn}) = \{100, 80, 60, 40, 20, 1, \cdots, 1\}$.

Scenario 2: $m = 50, n = 1000, r = 5, \text{diag}(S_{syn}) = \{100, 80, 60, 40, 5, 1, \cdots, 1\}$.

Scenario 3: $m = 50, n = 1000, r = 5, \text{diag}(S_{syn}) = \{100, 80, 60, 40, 3, 1, \cdots, 1\}$.

Set $b = 10$, $m_k = n/10$ for LR-SBFGS-VR and LR-SDavidon-VR based on the observations from the previous section. The numerical results, obtained by averaging over 50 runs, are presented in Table 7.12, 7.13, 7.14. The row labeled "BFGS/both/Davi" represents the number of successful runs out of 50 runs for LR-SBFGS-VR succeeding/ both succeeding/ only LR-SDavidon-VR succeeding. The additional method in the comparison noted as "Hybrid" represents LR-SBroyden-VR with the hybrid strategy of $\phi_i^{(k)}$ between Davidon and BFGS, as proposed in Chapter 5.

Scenario 1 involves distinctly larger leading singular values compared to the remaining values, making it ideal for R-SVRG. In this scenario, both quasi-Newton algorithms perform similarly, with two noteworthy observations: the first one is LR-SBFGS-VR tends to be faster than LR-SDavidon-VR when $\alpha = 0.001$; the second is LR-SDavidon-VR demonstrates relatively more stable performance when α and L are large.

Table 7.12: Results for the PCA problem Scenario 1. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

Scenario 1: diag(S_{syn}) $= \{100, 80, 60, 40, 20, 1, \cdots, 1\}$						
	stepsize	1_{-1}	5_{-2}	1_{-2}	5_{-3}	1_{-3}
$b = 10, m_k = n/10$	R-SVRG	58 (50)	58 (50)	259 (48)	390 (45)	817 (4)
$L = 1$	LR-SBFGS-VR	151 (50)	89 (50)	113 (50)	243 (50)	521 (50)
	LR-SDavidon-VR	133 (50)	87 (50)	113 (50)	243 (50)	521 (50)
$L = 2$	LR-SBFGS-VR	1009 (29)	321 (49)	102 (50)	175 (50)	563 (50)
	LR-SDavidon-VR	275 (50)	97 (50)	100 (50)	165 (50)	610 (50)
$L = 4$	LR-SBFGS-VR	57 (4)	331 (14)	160 (49)	228 (50)	612 (50)
	LR-SDavidon-VR	749 (29)	231 (50)	114 (50)	197 (50)	694 (50)
	BFGS/both/Davi	0/4/25				
$L = 8$	LR-SBFGS-VR	67 (4)	68 (12)	117 (46)	201 (46)	735 (50)
	LR-SDavidon-VR	67 (5)	616 (43)	142 (50)	250 (50)	854 (50)
	BFGS/both/Davi	0/4/1	0/12/31			

The other two Scenarios involve cases that are inherently difficult to solve, and it is evident that stochastic quasi-Newton algorithms are more effective in handling them than R-SVRG. The robustness and effectiveness of LR-SDavidon-VR and Hybrid are clearly demonstrated in these two scenarios. Results from 7.13 indicate that the stochastic quasi-Newton algorithms perform similarly when $L = 1$, but LR-SDavidon-VR exhibits its advantage as L increases. The success rate of LR-SDavidon-VR significantly improves with a relatively large α, especially when $L = 4, 8$. LR-SBFGS-VR shows a slight advantage when α is set to be the smallest value 0.001. However, LR-SDavidon-VR performs better in other situations, including its best performance at $L = 2, \alpha = 0.01$.

An even more difficult Scenario 3 in 7.14 further highlights this phenomenon. R-SVRG does not find sufficiently accurate solution with all different stepsizes. In case of "BFGS/both/Davi", especially with $\alpha = 5_{-2}$, it is observed that there is no situation where LR-SDavidon-VR fails but LR-SBFGS-VR succeeds when $L > 2$.

For PCA problem, Hybrid slightly amplifies the advantage and disadvantage of LR-SDavidon-VR. The parameter δ is determined by testing only at the best-tuned memory and stepsize. Table 7.15 shows us the results for different δ values in Hybrid, where $\delta = 0$ corresponds to LR-SDavidon-VR and an increasing δ indicates that the algorithm is approaching LR-SBFGS-VR. In this case, $\delta = 1$ is selected for the Hybrid. One typical example from Scenario 2 is shown in Figure 7.1, which compares $\#nJ$ versus the norm of gradient and the optimality gap.

Table 7.13: Results for the PCA problem Scenario 2. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

Scenario 2: $\text{diag}(S_{syn}) = \{100, 80, 60, 40, 5, 1, \cdots, 1\}$						
	stepsize	1_{-1}	5_{-2}	1_{-2}	5_{-3}	1_{-3}
$b = 1, m_k = n$	R-SVRG	1025 (50)	1772 (50)	4452 (45)	5036 (44)	8950 (13)
$L = 1$	LR-SBFGS-VR	730 (3)	746 (45)	448 (50)	2349 (50)	1165 (50)
	LR-SDavidon-VR	920 (4)	688 (44)	439 (50)	2374 (50)	1198 (50)
	BFGS/both/Davi	2/1/3	4/40/5			
	Hybrid $\delta = 1$	920 (4)	688 (44)	439 (50)	2374 (50)	1156 (50)
$L = 2$	LR-SBFGS-VR	- (0)	896 (25)	411 (50)	562 (50)	1085 (50)
	LR-SDavidon-VR	738 (7)	668 (49)	295 (50)	468 (50)	1293 (50)
	BFGS/both/Davi		0/25/24			
	Hybrid $\delta = 1$	1066 (7)	653 (48)	289 (50)	468 (50)	1395 (50)
$L = 4$	LR-SBFGS-VR	- (0)	502 (3)	769 (50)	693 (50)	1309 (50)
	LR-SDavidon-VR	- (0)	892 (46)	305 (50)	723 (50)	1456 (50)
	BFGS/both/Davi		0/3/43			
	Hybrid $\delta = 1$	- (0)	873 (46)	304 (50)	714 (50)	1574 (50)
$L = 8$	LR-SBFGS-VR	- (0)	557 (2)	287 (22)	606 (41)	1565 (50)
	LR-SDavidon-VR	- (0)	911 (14)	352 (50)	683 (50)	1782 (50)
	BFGS/both/Davi		0/2/12			
	Hybrid $\delta = 1$	- (0)	1214 (16)	356 (50)	674 (50)	1931 (50)

Table 7.14: Results for the PCA problem Scenario 3. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

Scenario 3: $\text{diag}(S_{syn}) = \{100, 80, 60, 40, 3, 1, \cdots, 1\}$						
	stepsize	1_{-1}	5_{-2}	1_{-2}	5_{-3}	1_{-3}
$b = 10, m_k = n/10$	R-SVRG	- (0)	- (0)	- (0)	- (0)	- (0)
$L = 1$	LR-SBFGS-VR	- (0)	1042 (17)	824 (50)	5544 (50)	1641 (50)
	LR-SDavidon-VR	400 (1)	1327 (18)	824 (50)	5523 (50)	1672 (50)
	BFGS/both/Davi		13/4/14			
	Hybrid $\delta = 1$	400 (1)	1327 (18)	824 (50)	5523 (50)	1672 (50)
$L = 2$	LR-SBFGS-VR	- (0)	1369 (13)	803 (50)	906 (50)	1547 (50)
	LR-SDavidon-VR	1054 (5)	967 (37)	460 (50)	779 (50)	2008 (50)
	BFGS/both/Davi		2/11/26			
	Hybrid $\delta = 1$	1349 (3)	1114 (38)	466 (50)	789 (50)	2200 (50)
$L = 4$	LR-SBFGS-VR	- (0)	193 (2)	1003 (42)	1289 (48)	1871 (50)
	LR-SDavidon-VR	- (0)	1030 (15)	474 (50)	1255 (50)	2335 (50)
	BFGS/both/Davi		0/2/13			
	Hybrid $\delta = 1$	- (0)	893 (18)	446 (50)	1265 (50)	2762 (50)
$L = 8$	LR-SBFGS-VR	- (0)	0 (0)	476 (18)	731 (33)	2031 (50)
	LR-SDavidon-VR	- (0)	1828 (4)	487 (50)	910 (50)	3024 (50)
	Hybrid $\delta = 1$	- (0)	905 (2)	476 (50)	903 (50)	3535 (50)

Table 7.15: Comparison of different δ in Hybrid for the PCA problem. The reported numbers are $\#nJ$ that measure the computational cost.

Scenario 2					
δ	0	1	2	3	BFGS
$L = 2, \alpha = 0.01$	295	289	303	393	411
$L = 4, \alpha = 0.01$	305	304	346	359	769
Scenario 3					
δ	0	1	2	3	LR-SBFGS-VR
$L = 2, \alpha = 0.01$	460	466	510	681	803
$L = 4, \alpha = 0.01$	474	446	613	558	1003

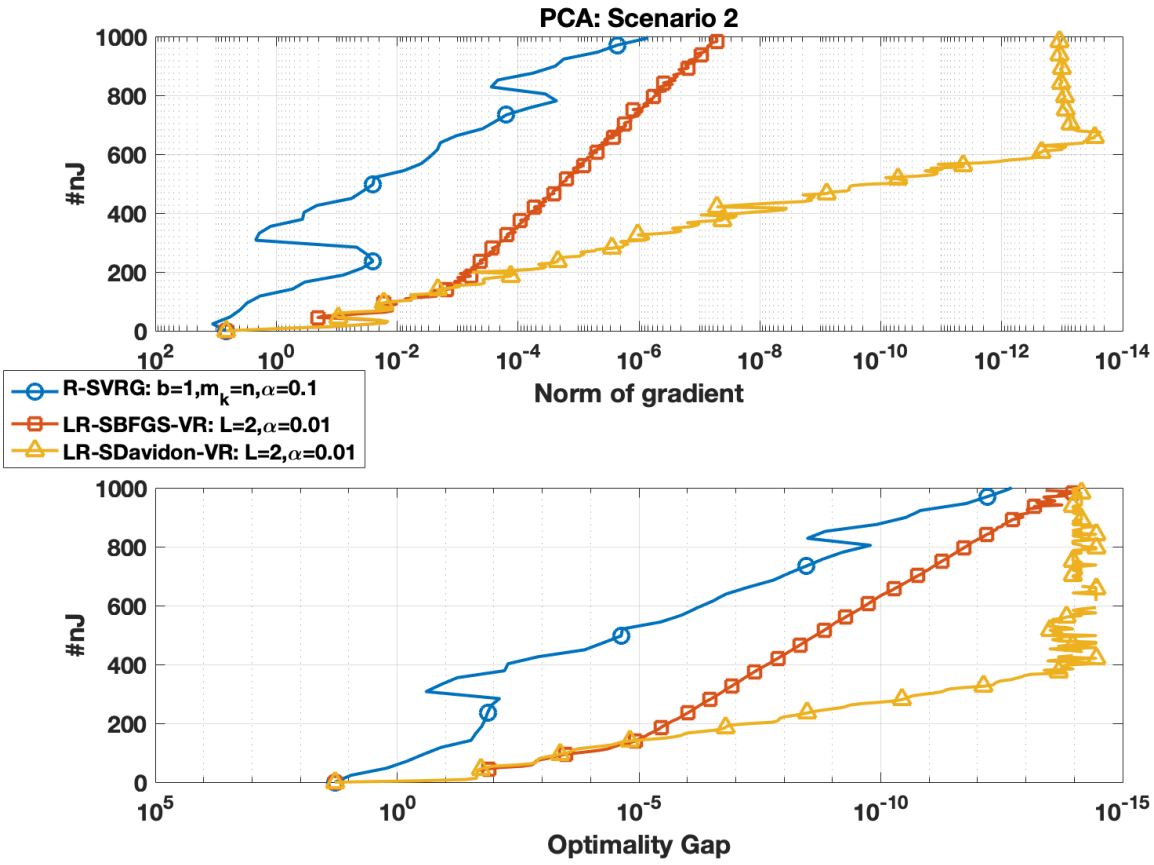


Figure 7.1: Comparison of stochastic algorithms for the PCA problem: Top: $\#nJ$ versus $\|g_f\|$; Bottom: $\#nJ$ versus $|f - f*|$.

7.4.2 Joint Diagonalization in ICA

Two scenarios with different rank are tested for the ICA problem on the Stiefel manifold, where higher rank makes the problem harder to solve.

Scenario 1: $m = 10, n = 1000, r = 5$.

Scenario 2: $m = 10, n = 1000, r = 9$.

Results are shown in Table 7.16: In scenario 1, R-SVRG performs the best with a large stepsize, while quasi-Newton algorithms have the best performance at $L = 1$. In scenario 2, however, LR-SDavidon and Hybrid outperform LR-SBFGS-VR and R-SVRG. In both scenarios, LR-SDavidon-VR and Hybrid work much better with a relatively large memory size since they do not suffer as much from the obsolescence of the curvature information. Additionally and significantly, they remain competitive to LR-SBFGS-VR in most other situations.

The selected δ for Hybrid is 2 based on the results in Table 7.17. Unlike PCA problem where $\delta = 1$ yields a similar performance compared to LR-SDavidon-VR, Hybrid shows a more pronounced improvement in ICA problem . This is reasonable since the superiority of LR-SDavidon-VR compared to LR-SBFGS-VR in ICA problem is not as significant as in PCA problem. A larger δ means the algorithm selects $\phi_i^{(k)} = 0$ more frequently during the updates.

Figure 7.4.3 reports the results tested in Scenario 2. It can be observed that R-SVRG has an obvious lower decaying rate. The diversity between the quasi-Newton algorithms take place after the iterate is approaching to the solution. Hybrid with $\delta = 2$ keeps its superiority until reaching the high precision.

7.4.3 Low-rank Matrix Completion Problem

The following three scenarios are synthetic low-rank matrix completion problems with different condition number.

Scenario 1: $m = 50, n = 1000, r = 5, OS = 5, CN = 10$.

Scenario 2: $m = 50, n = 1000, r = 5, OS = 5, CN = 20$.

Scenario 3: $m = 50, n = 1000, r = 5, OS = 5, CN = 50$.

Table 7.18 shows the results for all three scenarios, notice that only $\alpha = 0.05, 0.01$ are included to avoid redundancy. We know from previous experiments that a too large stepsize may result in a failed run and a much too small one may slower the speed. With the value of CN increasing, the

Table 7.16: Results for the ICA problem. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

Scenario 1: $r = 5$						
	stepsize	1_{-1}	5_{-2}	1_{-2}	5_{-3}	1_{-3}
$b = 10, m_k = n/10$	R-SVRG	24 (50)	25 (50)	41 (50)	72 (50)	355 (28)
$L = 1$	LR-SBFGS-VR	277 (49)	64 (50)	39 (50)	57 (50)	254 (50)
	LR-SDavidon-VR	291 (46)	68 (50)	38 (50)	57 (50)	254 (50)
$L = 2$	LR-SBFGS-VR	- (0)	494 (7)	54 (50)	63 (50)	301 (50)
	LR-SBroyden-VR	375 (5)	253 (47)	43 (50)	64 (50)	313 (50)
$L = 4$	LR-SBFGS-VR	- (0)	30 (1)	37 (34)	117 (47)	306 (50)
	LR-SDavidon-VR	- (0)	381 (7)	53 (50)	67 (50)	327 (50)
$L = 8$	LR-SBFGS-VR	- (0)	31 (1)	39 (34)	63 (42)	325 (50)
	LR-SDavidon-VR	- (0)	36 (4)	74 (50)	72 (50)	354 (50)
Scenario 2: $r = 9$						
	stepsize	1_{-1}	5_{-2}	1_{-2}	5_{-3}	1_{-3}
$b = 10, m_k = n/10$	R-SVRG	208 (39)	307 (50)	476 (43)	698 (39)	1217 (1)
$L = 1$	LR-SBFGS-VR	482 (7)	251 (50)	193 (50)	550 (50)	569 (50)
	LR-SDavidon-VR	539 (6)	270 (50)	188 (50)	546 (50)	593 (50)
	Hybrid $\delta = 2$	497 (11)	234 (50)	190 (50)	540 (50)	583 (50)
$L = 2$	LR-SBFGS-VR	- (0)	- (0)	196 (50)	225 (50)	598 (50)
	LR-SDavidon-VR	- (0)	454 (25)	153 (50)	227 (50)	672 (50)
	Hybrid $\delta = 2$	- (0)	575 (29)	133 (50)	224 (50)	805 (50)
$L = 4$	LR-SBFGS-VR	- (0)	- (0)	166 (27)	392 (45)	625 (50)
	LR-SDavidon-VR	- (0)	99 (3)	154 (50)	213 (50)	768 (50)
	Hybrid $\delta = 2$	- (0)	225 (3)	137 (50)	183 (50)	678 (50)
$L = 8$	LR-SBFGS-VR	- (0)	- (0)	80 (24)	153 (39)	537 (44)
	LR-SDavidon-VR	- (0)	81 (1)	172 (50)	248 (50)	787 (50)
	Hybrid $\delta = 2$	- (0)	74 (2)	172 (50)	183 (50)	616 (50)

Table 7.17: Comparison of different δ in Hybrid for the ICA problem. The reported numbers are $\#nJ$ that measure the computational cost.

Scenario 2					
δ	0	1	2	3	LR-SBFGS-VR
$L = 2, \alpha = 0.01$	153	150	133	142	196
$L = 4, \alpha = 0.01$	154	140	137	135	166

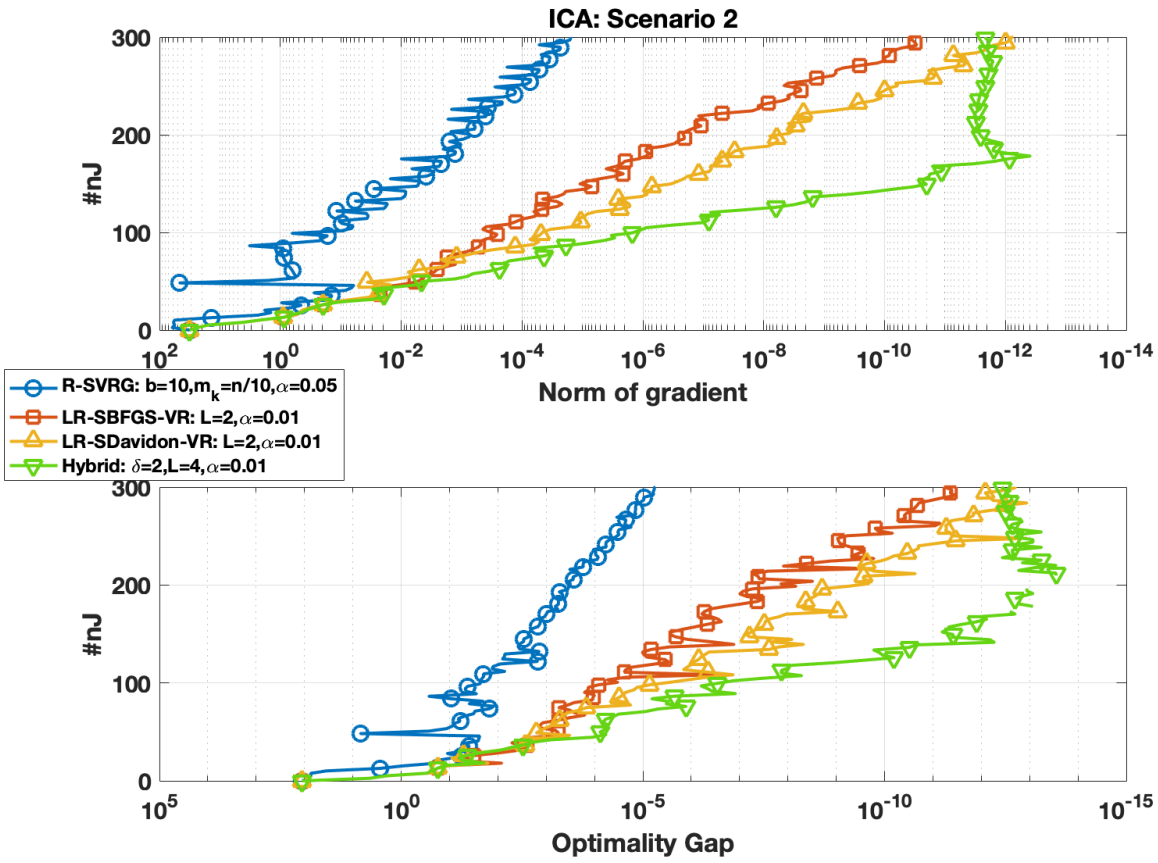


Figure 7.2: Comparison of stochastic algorithms for the ICA problem: Top: $\#nJ$ versus $\|g_f\|$; Bottom: $\#nJ$ versus $|f - f^*|$.

Table 7.18: Results for the low-rank matrix completion problem. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

		Scenario 1: $CN = 10$		Scenario 2: $CN = 20$		Scenario 3: $CN = 50$	
	stepsize	5_{-2}	1_{-2}	5_{-2}	1_{-2}	5_{-2}	1_{-2}
$L = 1$	LR-SBFGS-VR	148 (44)	160 (50)	224 (24)	303 (50)	456 (12)	944 (50)
	LR-SDavidon-VR	135 (44)	162 (50)	235 (25)	294 (49)	597 (15)	980 (49)
$L = 2$	LR-SBFGS-VR	186 (4)	136 (50)	306 (5)	284 (45)	- (0)	803 (31)
	LR-SBroyden-VR	166 (34)	114 (50)	220 (24)	198 (50)	474 (13)	545 (50)
$L = 4$	LR-SBFGS-VR	- (0)	128 (17)	- (0)	244 (22)	- (0)	621 (8)
	LR-SDavidon-VR	156 (3)	117 (50)	178 (7)	183 (50)	142 (2)	441 (50)
$L = 8$	LR-SBFGS-VR	- (0)	91 (12)	- (0)	157 (12)	- (0)	285 (6)
	LR-SDavidon-VR	68 (1)	126 (50)	77 (1)	179 (50)	- (0)	295 (50)

problem becomes harder to solve and both methods require more computations to converge. The accepted memory sizes for LR-SBFGS-VR are still restricted and this restriction is more severe with higher CN values. On the other hand, the memory size can provide more useful information for LR-SDavidon-VR where the best performance occurs at $L = 8$ when $CN = 20, 50$.

There are some modifications to the Low-rank matrix completion problem. Firstly, the initial point is changed from a randomly generated orthonormal-column $m \times r$ matrix to r dominant left singular vector U_0, where $[U_0, S_0, V_0] = svds(P_\Omega A, r)$ in Matlab. This is similar what was done in the non-stochastic experiments (see Section 4.1.3), where the initial guess is prevented from being far away from the minimizer. Doing so can be used to show the robustness of the proposed algorithm on the choice of initial condition. Secondly, the stopping criterion is set as $\|f\| < 10^{-4}$ since the optimal function value of the low-rank matrix completion problem is 0. This stopping criteria is equivalent to the optimality gap between the current objective value and the optimal objective value, which is more accurate to measure the performance of a non-convex optimization problem. The following results are run with 50 different seeds in generating the sparse matrix A and the known index Ω.

Table 7.19 shows the $\#nJ$ values and success rates for algorithms reaching to the new criterion $\|f\| < 10^{-4}$. The results are consistent with the observations in Table 7.18 and an obvious advantage of LR-SDavidon-VR can be observed, especially when $L = 4, 8$. Table 7.20 shows the δ selection in Hybrid, where $\delta = 1.5$ provides a better performance at $L = 8$. Hybrid also makes up for the deficiency of LR-SDavidon-VR when α is small, making it competitive with LR-SBFGS-VR. One typical example for Scenario 3 is shown in Figure 7.4.3, note that the Function Value in the second graph is equivalent to the optimality gap in the previous problem results. The blue curve is the best

Table 7.19: Results for the low-rank matrix completion problem Scenario 3 with modified stopping criteria and initial condition. The reported numbers are $\#nJ$ that measure the computational cost (with the number of successful runs out of 50 in the parentheses).

		Scenario 3: $CN = 50$		
	stepsize	5_{-2}	1_{-2}	5_{-3}
$b = 1, m_k = n$	R-SVRG	- (0)	- (0)	- (0)
$L = 1$	LR-SBFGS-VR	219 (18)	477 (50)	2966 (49)
	LR-SDavidon-VR	237 (15)	501 (50)	2991 (50)
	Hybrid $\delta = 1.5$	234 (20)	500 (50)	2960 (50)
$L = 2$	LR-SBFGS-VR	217 (1)	283 (49)	409 (50)
	LR-SDavidon-VR	181 (19)	267 (50)	548 (50)
	Hybrid $\delta = 1.5$	183 (18)	274 (50)	501 (50)
$L = 4$	LR-SBFGS-VR	- (0)	356 (41)	388 (50)
	LR-SDavidon-VR	163 (1)	198 (50)	456 (50)
	Hybrid $\delta = 1.5$	148 (1)	195 (50)	295 (50)
$L = 8$	LR-SBFGS-VR	- (0)	159 (20)	293 (37)
	LR-SDavidon-VR	- (0)	189 (50)	326 (50)
	Hybrid $\delta = 1.5$	66 (1)	169 (50)	310 (50)

Table 7.20: Comparison of different δ in Hybrid for the low-rank matrix completion problem. The reported numbers are $\#nJ$ that measure the computational cost.

Scenario 3					
δ	0	1	1.5	2	3
$L = 4, \alpha = 0.01$	198	189	195	229	246
$L = 8, \alpha = 0.01$	189	189	169	174	189

we can obtain with R-SVRG for this difficult case, where the function value is slightly descending but the norm of gradient is slightly rising. Hybrid is the winner from very beginning of the iterates.

Conclusion. This section yields several noteworthy insights. In comparison to the existing LR-SBFGS-VR algorithm, LR-SDavidon-VR, and Hybrid exhibit similar convergence speeds in well-conditioned cases. However, their superiority becomes particularly pronounced when tackling ill-conditioned problems, where these algorithms outperform the first-order R-SVRG commonly used in such scenarios. Another noteworthy observation is the enhanced robustness of LR-SDavidon-VR and Hybrid when employed with larger memory sizes and stepsizes. The increased memory size lends support to the hypothesis that certain members within the full Broyden family offer superior estimates in approximating curvature.

While the method of generating curvature pairs imposes limitations on the behavior of LR-SBFGS-VR when $L > 2$, these constraints are less conspicuous in the case of LR-SDavidon-VR and Hybrid. Furthermore, the compatibility of LR-SDavidon-VR and Hybrid with larger stepsizes aligns

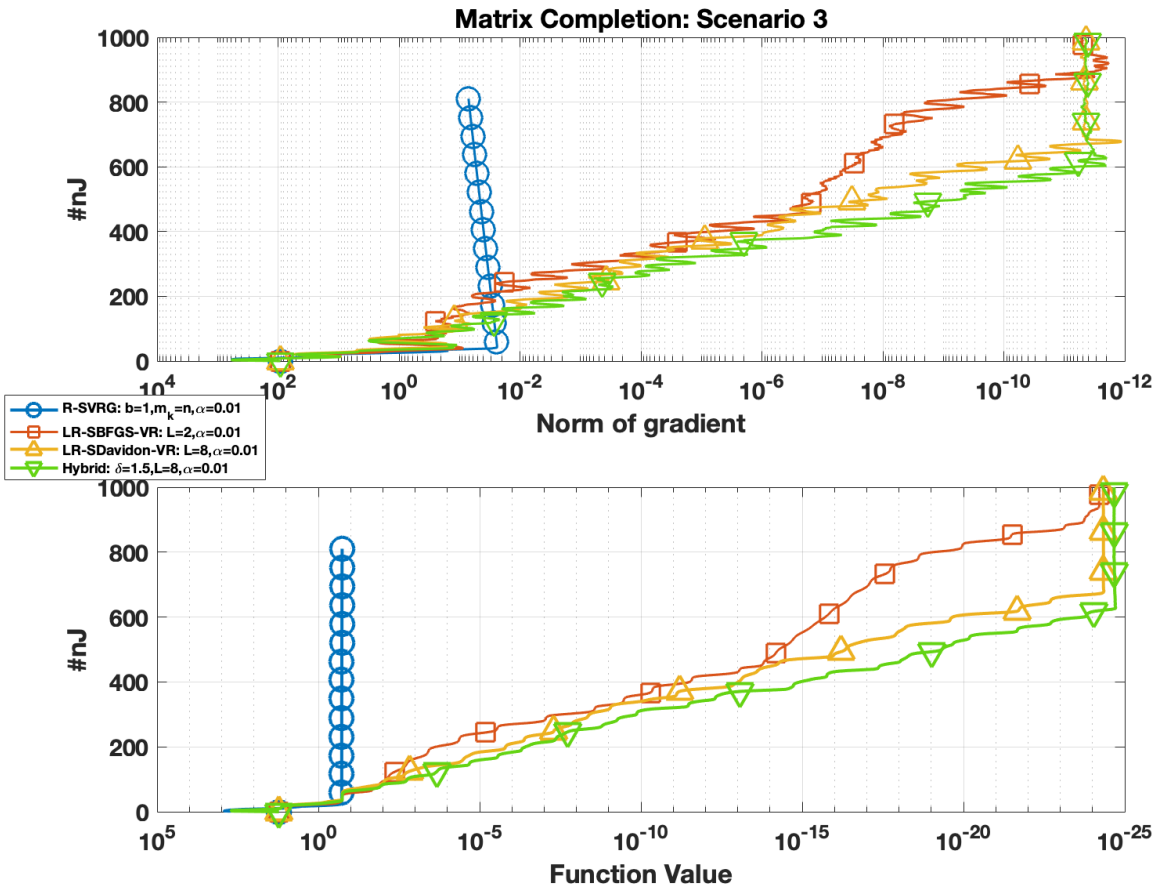


Figure 7.3: Comparison of stochastic algorithms for the low-rank matrix completion problem: Top: $\#nJ$ versus $\|g_f\|$; Bottom: $\#nJ$ versus $|f|$.

with the findings from Section 4.3. This advantage not only underscores the practical convenience of LR-SDavidon-VR and Hybrid, but also addresses the challenge of stepsize selection. Overall, these findings underscore the effectiveness of LR-SDavidon-VR and Hybrid, especially in challenging and ill-conditioned optimization scenarios, making them valuable tools for practical applications.

To illustrate implicit common factors in the experiments, a comparison is established between Euclidean quadratic problem, Brockett problem on the Stiefel manifold and the three problems tested in this chapter. Table 7.21 provides insights into different scenarios, the second and third columns are cases where LR-SDavidon-VR(LRDavidon) and LR-SBFGS-VR(LRSBFGS) exhibit competitive performance. In contrast, the fourth and fifth columns highlight instances where LR-SDavidon-VR(LRDavidon) significantly outperforms LR-SBFGS-VR(LRSBFGS). The "max,min eig" denotes the maximum and minimum eigenvalues of the Hessian from the initial point to the solution point. A larger difference between the maximum and minimum eigenvalues at the solution point signifies that the "Davidon better cases" correspond to ill-conditioned problems as desired. It was established in Section 4.4 that mere ill-conditioning is insufficient for LRDavidon to outperform LRBFGS. However, for LR-SDavidon-VR, ill-conditioning alone proves sufficient for outperforming LR-SBFGS-VR. This sheds light on the nuanced interplay of problem characteristics and algorithm performance, showcasing the unique strengths of LR-SDavidon-VR in addressing ill-conditioned problems.

The comparison involves two key variables: the mean value of γ_k and the mean value of $\phi_i^{(k)}$, where γ represents the initial scaling of inverse Hessian approximation, such that $\mathcal{H}_k^0 = \gamma_k \cdot \mathrm{id}$. Observing the mean value of γ, LR-SDavidon-VR(LRDavidon) exhibits a significantly larger mean value of γ_k than LR-SBFGS-VR(LRSBFGS) in the "Davidon better cases". This indicates that the initial scaling of $\mathcal{H}_k^0$ is larger for LR-SDavidon-VR(LRDavidon), resulting in a smaller scaling of $\mathcal{B}_k^0$.

Notably, the mean value of $\phi_i^{(k)}$ tends to be positive for most LR-SDavidon-VR(LRDavidon) cases. The positive values of $\phi_i^{(k)}$ enhance the self-correcting property of the Broyden family, particularly in correcting small eigenvalues. Examing the mean value of $\phi_i^{(k)}$ further reveals that in the "Davidon better cases", the mean value of $\phi_i^{(k)}$ for LR-SDavidon-VR(LRDavidon) is smaller compared to the "Competitive cases". This suggests that the updates prioritize correcting the large eigenvalues, which is reasonable since the minimum eigenvalue at the solution point is smaller in the "Davidon better cases". These findings provide compelling evidence that the self-correcting property of the Broyden family plays a pivotal role in algorithm performance. LR-SDavidon-

Table 7.21: Comparison of mean(γ_k) and mean($\phi_i^{(k)}$) between LR-SBFGS-VR and LR-SBroyden-VR. Row "max,min eig" represents the maximum and minimum eigenvalues of the Hessian from initial point to solution.

	Davidon Competitive to BFGS		Davidon better than BFGS	
max,min eig	$1000, 0.00026$		$1000, 0.00026$	
EucQuadratic	mean(γ_k)	mean($\phi_i^{(k)}$)	mean(γ_k)	mean($\phi_i^{(k)}$)
LRBFGS	0.0029	0	0.2289	0
LRDavidon	0.0028	2.2838	0.6191	-0.0694
max,min eig	$846, -933 \rightarrow 1781, 0.2935$		$1000, -6 \rightarrow 1000, 0.00014$	
StieBrockett	mean(γ_k)	mean($\phi_i^{(k)}$)	mean(γ_k)	mean($\phi_i^{(k)}$)
LRBFGS	0.0062	0	0.0143	0
LRDavidon	0.0042	0.2941	0.0833	0.1048
max,min eig	$2.9, -17 \rightarrow 20, 0.8$		$2.9, -17 \rightarrow 20, 0.048$	
PCA	mean(γ_k)	mean($\phi_i^{(k)}$)	mean(γ_k)	mean($\phi_i^{(k)}$)
LR-SBFGS-VR	0.2886	0	0.3025	0
LR-SDavidon-VR	0.4232	0.6096	1.0454	0.3077
max,min eig	$152, -46 \rightarrow 288, 103$		$140, -37 \rightarrow 320, 26$	
ICA	mean(γ_k)	mean($\phi_i^{(k)}$)	mean(γ_k)	mean($\phi_i^{(k)}$)
LR-SBFGS-VR	0.0032	0	0.0045	0
LR-SDavidon-VR	0.0036	0.7107	0.0072	0.4039
max,min eig	$122, -19 \rightarrow 92, 9$		$129, -30 \rightarrow 97, 0.5$	
matrix completion	mean(γ_k)	mean($\phi_i^{(k)}$)	mean(γ_k)	mean($\phi_i^{(k)}$)
LR-SBFGS-VR	0.0057	0	0.0056	0
LR-SDavidon-VR	0.0065	0.4919	0.0103	0.3953

VR(LRDavidon) demonstrates a more balanced approach in addressing both large and small eigenvalues, contributing to its effectiveness in optimization scenarios with varying eigenvalue magnitudes.

CHAPTER 8

CONCLUSION AND FURTHER RESEARCH

The dissertation focuses on the development, investigation, and application of efficient and robust limited-memory Broyden quasi-Newton methods specifically tailored for optimization problems on a Riemannian manifold. The employment of limited-memory variants and the incorporation of a stochastic method within LR-SBroyden-VR underscore important considerations pertaining to computational efficiency and convergence properties within the domain of Riemannian optimization.

The major contributions of this dissertation are:

1. Developing limited-memory family, LRBroyden, derived from the full Broyden family and completing its convergence theory. The efficiency of the state-of-the-art algorithm is achieved through the utilization of a compact representation and an intrinsic representation, enabling users to set the Broyden parameters at all steps.

2. Empirically evaluating the LRBroyden family through systematic experiments in both Euclidean and Riemannian settings. Previous research on limited-memory extensions of the Broyden family primarily concentrated on Euclidean space, lacking persuasive experimental results to demonstrate the robustness advantages of limited-memory extensions across the entire Broyden family. The primary emphasis of the empirical results presented in this dissertation lies in the comparison between LRBFGS, LRDavidon and a hybrid strategy by:

   (a) Exploring the empirical relationship between a fixed ϕ and the initial stepsize. The heuristic consequence is shown to be consistent with the theoretical analysis.

   (b) Providing a particular case where LRDavidon outperforms the state-of-the-art LRBFGS in both Euclidean and Riemannian settings. The good performance is explained by the self-correcting property with the distribution of the $\phi_i^{(k)}$ in sequence of iterates.

   (c) Exploiting a hybrid strategy of generating $\phi_i^{(k)}$ between the parameter choices of Davidon and BFGS. The algorithm with an appropriate initial stepsize is observed to be more efficient and robust than LRBFGS in a general large-scale problem.

3. Developing LR-SBroyden-VR and completing its convergence theory, expanding the scope of stochastic quasi-Newton methods to full Broyden family.

4. Empirically evaluating the LR-SBroyden-VR in comparison with LR-SBFGS-VR and R-SVRG on three problems: principal component analysis on the Grassmann manifold, joint

diagonalization problem on the Stiefel manifold and low-rank matrix completion problem on the Grassmann manifold. The evaluation concentrates on:

(a) Providing an appropriate measurement on the computational costs for the stochastic algorithms.

(b) Providing heuristics to choose between various stochastic parameters in different cases. Previous research did not pay much attention to these parameters. However, they significantly influence the behavior of the stochastic optimization methods on large-scale problems.

(c) Comparing and evaluating the performances of various stochastic algorithms by systematic numerical experiments. Stochastic quasi-Newton algorithms are observed to be more efficient than R-SVRG on the ill-conditioned problems as desired. Meanwhile, LR-SBroyden-VR with Davidon and Hybrid Broyden parameters outperform LR-SBFGS-VR on these problems. Davidon's $\phi_i^{(k)D}$ provides us a wider acceptance of algorithm parameters and the hybrid strategy strengthens the superiority of the robustness of LR-SBroyden-VR.

5. Contributing a C++ toolbox for the above mentioned algorithms and stochastic problems to ROPTLIB. Providing comprehensive toolbox for LRBroyden and its stochastic form.

There are several avenues for future research. First of all, a strategy to select $\phi_i^{(k)}$ in the limited-memory variant of Broyden family is still an open question in both Euclidean and Riemannian settings. Results in this dissertation are limited to the family members between Davidon's $\phi_i^{(k)D}$ and BFGS's $\phi_i^{(k)\mathrm{BFGS}}$. For a specific problem, it will always be interesting and worthwhile to consider a particular way of choosing $\phi_i^{(k)}$ in LRBroyden.

Researchers paid much attention on SGD and its related methods. However, there are many opportunities in the area of Riemannian stochastic quasi-Newton methods. The batch size, frequency value and stepsize play significant roles in affecting the performance of LR-SBroyden-VR and are determined by experiments in our results. Further study is needed in the consideration of these stochastic parameter settings and in determining them in a manner infromed by a more complete theoretical understanding. This effort must be supported by additional systematic comparisons with existing methods on real-world applications.

Finally, this dissertation does not include a comparison with adaptive stochastic gradient methods since their convergecne speed is limited due to the decaying stepsize. Very recently, researchers started to combine the variance reduction with some adaptive gradient-based methods, e.g., with AdaGrad [16] in Euclidean space. Further study is needed on generalization to the Riemannian settings and compare them with LR-SBroyden-VR.

REFERENCES

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, Princeton, NJ, 2008.

[2] Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[3] Gary Becigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*, 2019.

[4] Silvere Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

[5] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[6] Nicolas Boumal and P-A Absil. Low-rank matrix completion via preconditioned optimization on the Grassmann manifold. *Linear Algebra and its Applications*, 475:200–239, 2015.

[7] Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[8] Richard H Byrd, Dong C Liu, and Jorge Nocedal. On the behavior of Broyden's class of quasi-Newton methods. *SIAM Journal on Optimization*, 2(4):533–557, 1992.

[9] Richard H Byrd, Jorge Nocedal, and Robert B Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1):129–156, 1994.

[10] Richard H Byrd, Jorge Nocedal, and Ya-Xiang Yuan. Global convergence of a cass of quasi-Newton methods on convex problems. *SIAM Journal on Numerical Analysis*, 24(5):1171–1190, 1987.

[11] William C Davidon. Optimally conditioned optimization algorithms without line searches. *Mathematical Programming*, 9(1):1–30, 1975.

[12] Omar DeGuchy, Jennifer B Erway, and Roummel F Marcia. Compact representation of the full Broyden class of quasi-Newton updates. *Numerical Linear Algebra with Applications*, 25(5):e2186, 2018.

[13] John E Dennis and Jorge J Moré. A characterization of superlinear convergence and its application to quasi-Newton methods. *Mathematics of Computation*, 28(126):549–560, 1974.

[14] John E Dennis, Jr and Jorge J Moré. Quasi-Newton methods, motivation and theory. *SIAM Review*, 19(1):46–89, 1977.

[15] Shuyu Dong, Bin Gao, Wen Huang, and Kyle A Gallivan. On the analysis of optimization with fixed-rank matrices: a quotient geometric view. *arXiv preprint arXiv:2203.06765*, 2022.

[16] Benjamin Dubois-Taine, Sharan Vaswani, Reza Babanezhad, Mark Schmidt, and Simon Lacoste-Julien. SVRG meets AdaGrad: painless variance reduction. *Machine Learning*, pages 1–51, 2022.

[17] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

[18] Jennifer B Erway and Roummel F Marcia. On efficiently computing the eigenvalues of limited-memory quasi-Newton matrices. *SIAM Journal on Matrix Analysis and Applications*, 36(3):1338–1359, 2015.

[19] Mert Gurbuzbalaban, Umut Simsekli, and Lingjiong Zhu. The heavy-tail phenomenon in SGD. In *International Conference on Machine Learning*, pages 3964–3975. PMLR, 2021.

[20] Roger A Horn and Charles R Johnson. Norms for vectors and matrices. *Matrix Analysis*, pages 313–386, 1990.

[21] S Hoshino. A formulation of variable metric methods. *IMA Journal of Applied Mathematics*, 10(3):394–403, 1972.

[22] Wen Huang. *Optimization algorithms on Riemannian manifolds with applications*. PhD thesis, The Florida State University, 2013.

[23] Wen Huang, P-A Absil, and Kyle A Gallivan. A Riemannian symmetric rank-one trust-region method. *Mathematical Programming*, 150(2):179–216, 2015.

[24] Wen Huang, P-A Absil, and Kyle A Gallivan. Intrinsic representation of tangent vectors and vector transports on matrix manifolds. *Numerische Mathematik*, 136(2):523–543, 2017.

[25] Wen Huang, P-A Absil, and Kyle A Gallivan. A Riemannian BFGS method without differentiated retraction for nonconvex optimization problems. *SIAM Journal on Optimization*, 28(1):470–495, 2018.

[26] Wen Huang, P-A Absil, Kyle A Gallivan, and Paul Hand. ROPTLIB: an object-oriented C++ library for optimization on Riemannian manifolds. *ACM Transactions on Mathematical Software (TOMS)*, 44(4):1–21, 2018.

[27] Wen Huang, Kyle A Gallivan, and P-A Absil. A Broyden class of quasi-Newton methods for Riemannian optimization. *SIAM Journal on Optimization*, 25(3):1660–1685, 2015.

[28] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in Neural Information Processing Systems*, 26:315–323, 2013.

[29] Hiroyuki Kasai, Pratik Jawanpuria, and Bamdev Mishra. Riemannian adaptive stochastic gradient algorithms on matrix manifolds. In *International Conference on Machine Learning*, 2019.

[30] Hiroyuki Kasai, Hiroyuki Sato, and Bamdev Mishra. Riemannian stochastic quasi-Newton algorithm with variance reduction and its convergence analysis. In *International Conference on Artificial Intelligence and Statistics*, pages 269–278. PMLR, 2018.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computing Research Repository*, abs/1412.6980, 2014.

[32] Chuanhai Liu and Scott A Vander Wiel. Statistical quasi-Newton: A new look at least change. *SIAM Journal on Optimization*, 18(4):1266–1285, 2008.

[33] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989.

[34] Jonathan H Manton, Robert Mahony, and Yingbo Hua. The geometry of weighted low-rank approximations. *IEEE Transactions on Signal Processing*, 51(2):500–514, 2003.

[35] Bamdev Mishra, Gilles Meyer, Silvere Bonnabel, and Rodolphe Sepulchre. Fixed-rank matrix factorizations and Riemannian low-rank optimization. *Computational Statistics*, 29:591–621, 2014.

[36] Philipp Moritz, Robert Nishihara, and Michael Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *Artificial Intelligence and Statistics*, pages 249–258. PMLR, 2016.

[37] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

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

[39] Thilo Penzl. A cyclic low-rank smith method for large sparse Lyapunov equations. *SIAM Journal on Scientific Computing*, 21(4):1401–1418, 1999.

[40] Chunhong Qi. *Numerical optimization methods on Riemannian manifolds*. PhD thesis, The Florida State University, 2011.

[41] Chunhong Qi, Kyle A Gallivan, and P-A Absil. Riemannian BFGS algorithm with applications. In *Recent Advances in Optimization and its Applications in Engineering: The 14th Belgian-French-German Conference on Optimization*, pages 183–192. Springer, 2010.

[42] Chunhong Qi, Kyle A Gallivan, and Pierre-Antoine Absil. An efficient BFGS algorithm for riemannian optimization. In *Proceedings of the 19th International Symposium on Mathematical Theory of Network and Systems (MTNS 2010)*, volume 1, pages 2221–2227. Citeseer, 2010.

[43] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[44] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International Conference on Machine Learning*, pages 314–323. PMLR, 2016.

[45] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations*, 2018.

[46] Martin B Reed. L-Broyden methods: a generalization of the L-BFGS method to the limited-memory Broyden family. *International Journal of Computer Mathematics*, 86(4):606–615, 2009.

[47] Wolfgang Ring and Benedikt Wirth. Optimization methods on Riemannian manifolds and their application to shape space. *SIAM Journal on Optimization*, 22(2):596–627, 2012.

[48] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[49] Anirban Roychowdhury and Srinivasan Parthasarathy. Accelerated stochastic quasi-Newton optimization on Riemann manifolds. *arXiv*, abs/1704.01700, 2017.

[50] Hiroyuki Sato. Riemannian Newton-type methods for joint diagonalization on the Stiefel manifold with application to independent component analysis. *Optimization*, 66(12):2211–2231, 2017.

[51] Hiroyuki Sato, Hiroyuki Kasai, and Bamdev Mishra. Riemannian stochastic variance reduced gradient algorithm with retraction and vector transport. *SIAM Journal on Optimization*, 29(2):1444–1472, 2019.

[52] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162:83–112, 2017.

[53] Robert B Schnabel. Analyzing and improving quasi-Newton methods for unconstrained optimization. Technical report, Cornell University, 1977.

[54] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(1), 2013.

[55] David F Shanno and Kang Hoh Phua. Matrix conditioning and nonlinear optimization. *Mathematical Programming*, 14:149–160, 1978.

[56] Fabian J Theis, Thomas P Cason, and P A Absil. Soft dimension reduction for ICA by joint diagonalization on the stiefel manifold. In *Independent Component Analysis and Signal Separation: 8th International Conference, ICA 2009, Paraty, Brazil, March 15-18, 2009. Proceedings 8*, pages 354–361. Springer, 2009.

[57] Bart Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

[58] Bart Vandereycken and Stefan Vandewalle. A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2553–2579, 2010.

[59] Jinshan Zeng, Yixuan Zha, Ke Ma, and Yuan Yao. On stochastic variance reduced gradient method for semidefinite optimization. *arXiv preprint arXiv:2101.00236*, 2021.

[60] Hongyi Zhang, Sashank J. Reddi, and Suvrit Sra. Riemannian SVRG: Fast stochastic optimization on Riemannian manifolds. In *Neural Information Processing Systems*, 2016.

[61] Yin Zhang and RP Tewarson. Quasi-Newton algorithms with updates from the preconvex part of Broyden's family. *IMA Journal of Numerical Analysis*, 8(4):487–509, 1988.

[62] Guifang Zhou. *Rank-constrained optimization: A Riemannian manifold approach.* PhD thesis, The Florida State University, 2015.

BIOGRAPHICAL SKETCH

Shuguang Zhang, son of Ming Zhang and Haixia Shu, was born on September 8th, 1993 in Hefei, Anhui province of P.R. China. He completed his bachelor degree in Mathematics in 2014 at Nankai University in China. He enrolled in the Master degree of Mathematics in 2015 and started his Doctoral program in 2017 at Florida State University, worked with Prof. Kyle A. Gallivan and Prof. Wen Huang.

Shuguang's research interests include Riemannian Broyden family of limited-memory quasi-Newton methods.