FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

RIEMANNIAN MANIFOLD TRUST-REGION METHODS WITH APPLICATIONS TO EIGENPROBLEMS

By

CHRISTOPHER G. BAKER

A Treatise submitted to the
School of Computational Science
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Degree Awarded:
Summer Semester, 2008

The members of the Committee approve the Treatise of Christopher G. Baker defended on May 22, 2008.

Kyle Gallivan
Professor Co-Directing Treatise

Pierre-Antoine Absil
Professor Co-Directing Treatise

Anjaneyulu Krothapalli
Outside Committee Member

Gordon Erlebacher
Committee Member

Anuj Srivastava
Committee Member

Yousuff Hussaini
Committee Member

The Office of Graduate Studies has verified and approved the above named committee members.

This work is dedicated to my parents, for their constant encouragement...

And to Kelly, my partner and greatest supporter. I love you.

ACKNOWLEDGEMENTS

I must first acknowledge my dissertation committee. It has been my pleasure to interact with them over my decade at Florida State, both in and outside of the classroom.

The largest influence on my scholarship comes from Kyle Gallivan. It has been over eight years since I sat for my first lecture with Kyle. Five courses and three diplomas later, I know that if only a fraction of his insight and intuition have rubbed off on me, then I will never want for an interesting problem to occupy my whiteboard. His precision seemed nearly infinite, and it was only his willingness to be distracted to one of a multitude of other conversations that saved me from madness—albeit a madness exhibiting perfect prose, exhaustive analysis and *just one more* set of experiments. I look forward to both our future collaborations and conversations.

And here it can't be helped but to rely on cliché: this dissertation would never have happened without the guidance of Pierre-Antoine Absil. He was both my advisor and my friend through all of the nonlinearities that define Riemannian geometry, publication, dissertation writing... graduate school. He has been a constant source of reassurance and encouragement during this process; a role model; my Belgian big brother.

I must acknowledge both the Department of Computer Science and the School of Computational Science, the places that I called home over the years, and the people who kept everything running there: Eleanor McNealy, Michele Locke, Anne Johnson, Lynn Lacombe, Cecelia Farmer, and Debra Crews. Mimi Burbank devoted sweat and tears to assemble this LaTeX template, so that the rest of us wouldn't have to work so hard. Many thanks go to the system administrators who kept the computers and networks running: Chris Cprek, Tom Green, Dana Lutton, Jeff McDonald, Daniel Whelan and Ryan Carlyle at SCS; Bill Goldman and Roger Retallack at Sandia.

The majority of my past three years was spent at the Computer Science Research Institute of Sandia National Laboratories. I am grateful for the opportunity to have worked in such

an encouraging environment, with such wonderful and talented people. Special thanks go to my managers: Scott Collis, Ken Alvin and David Womble. But very special thanks go to Rich Lehoucq, the best mentor that a student could ask for. His advice on writing, on career, on finding and fixing the ugliest bugs; his support and understanding as I juggled work, school and life; and his willingness to share his own experience. The training I've received under Rich's supervision will undoubtedly benefit me for the rest of my career.

To my home away from home in the Department of Religious Studies and the support and encouragement of many people there, especially Dr. John Kelsay and my advisor-in-law, Dr. John Corrigan. Thanks for letting me pretend to be a religion major every once in a while.

I would be remiss not to mention all of the friends who helped me maintain my sanity over the years, in the humidity and the high desert: Mike and Kristin Pasquier, Michael and Shaynna Gueno, Howell Williams and Steve Bryant, Brian Adams and Judy Hill. To Kelly Dickson, always stand by your NaN. *Many thanks* to Dr. Clayton Webster, for all of the advice, assistance and coffee; and for having my back, buddy. Most of all, thanks to Heidi Thornquist and Denis Ridzal: without them, my code would not compile, my proofs would never have been proven and I likely would have starved to death years ago.

To my family, in particular my parents, Lynn and Frank and Steve and Dottie; and my grandparents, Dorothy and Lois and Earl. They instilled in me an appreciation for the importance of education, as well as the curiosity and persistence without which any research would ever be accomplished.

And to **Kelly**: for encouragement, understanding, support and peace. For being my unconditional partner through the whole thing. For being a reminder that there are more important things than research and for making me want to be a better *person*.

— CGB, 2008

This research benefited from funding from NSF Grants OCI0324944 and CCR9912415.

A portion of this work was conducted while employed at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy; contract/grant number: DE-AC04-94AL85000.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

ABSTRACT

This dissertation proposes a new class of methods for optimizing smooth functions over Riemannian manifolds. A novel optimization paradigm, retraction-based Riemannian optimization, is proposed. This paradigm uses mappings called retractions to lift the objective function from the Riemannian manifold to the tangent space, an abstract Euclidean space, where methods from Euclidean optimization can be easily applied. The retraction is then used to move back to the Riemannian manifold, where the process repeats. We justify this approach through the derivation of second-order optimality conditions for a solution computed in this manner.

This framework allows for the description of two novel Riemannian optimization methods. The Riemannian Trust-Region (RTR) method adapts the mechanisms of Euclidean trust-region methods to a Riemannian setting. Analysis shows that the RTR method retains the global and local convergence properties of its Euclidean counterparts. The combination of robust global convergence and fast local convergence provides superior performance not available with previously described Riemannian optimization methods. The Implicit Riemannian Trust-Region (IRTR) method improves on the classical trust-region mechanism by eliminating its inherent inefficiencies: an over-constraining trust-region radius or a wasteful rejection mechanism.

These solvers are applied to the problem of computing extreme eigenspaces of a symmetric matrix pencil. This problem can be characterized as a Riemannian optimization problem, the optimization of the generalized Rayleigh quotient over the Grassmann manifold. Standard solvers for the eigenvalue problem are analyzed in the context of Riemannian optimization. A performance analysis of the RTR and IRTR methods applied to the eigenvalue problem demonstrates that they are competitive with these standard solvers.

INTRODUCTION

Numerical optimization is at the root of many problems in computer science, engineering, physics, and other computational sciences. Consider a function $f : \mathcal{D} \to \mathbb{R}$, mapping elements from a set $\mathcal{D}$ to the real numbers. The task of optimization is to find elements in the set $\mathcal{D}$ whose image under f is extreme compared to surrounding elements. When the set $\mathcal{D}$ is the vector space $\mathbb{R}^n$, there is an extensive collection of theory and methodology available to solve the problem. In this case, the problem is referred to as an *unconstrained Euclidean optimization* problem. If there are constraints on the search space for minimizers (i.e., $\mathcal{D} \subset \mathbb{R}^n$), the problem is a *constrained Euclidean optimization* problem.

Many problems of significant importance can be cast as optimization problems where the objective function is defined over a Riemannian manifold. We refer here to these as problems of *Riemannian optimization*. In the case that the Riemannian manifold is an embedded submanifold of $\mathbb{R}^n$, the problem may be rewritten as a constrained optimization problem, so that Euclidean solvers can be applied. However, there may be multiple reasons why this is not optimal. In some cases, the Riemannian manifold may have no tractable embedding in Euclidean space. In the case that there is an efficient embedding, a Euclidean characterization may increase the dimensionality so as to make a Riemannian approach more attractive. Furthermore, the theory and practice of constrained optimization requires more effort than unconstrained Euclidean optimization. The Riemannian optimization methods discussed in this dissertation resemble unconstrained Euclidean methods; they should be thought of as unconstrained search in a constrained search space.

In response to this situation, a number of efforts [Shu86, Mah96, EAS98, OW00, Man02, ADM+02, DPM03, HT04] have focused on the development of general techniques for conducting optimization over Riemannian manifolds. Most of the literature is concerned with transferring traditional Euclidean optimization methods to Riemannian manifolds. This dissertation is similar in that regard: it proposes a class of trust-region methods

on Riemannian manifolds. However, the main interest here is in producing efficient algorithms for solving large-scale optimization problems. For that reason and to assist in the development of the Riemannian trust-region methods to follow, we begin by proposing a new paradigm for Riemannian optimization: the retraction-based Riemannian optimization approach.

Following a review of Euclidean optimization, Chapter 1 presents the material from Riemannian geometry necessary for the development of optimization methods on Riemannian manifolds. The concept of retractions is introduced and formalized, allowing the description of retraction-based Riemannian optimization. Chapter 2 describes the Riemannian Trust-Region method and analyzes its convergence. Chapter 3 discusses the Implicit Riemannian Trust-Region method and proves that it retains the convergence of the RTR method. Finally, Chapter 4 illustrates the potential of these methods by applying them to the computation of selected eigenvalues and eigenvectors of a symmetric/definite matrix pencil.

CHAPTER 1

RIEMANNIAN OPTIMIZATION

Much of the current work in Riemannian optimization derives from work in Euclidean optimization. By constructing analogous concepts in a Riemannian setting for the familiar concepts of Euclidean space, a large amount of material can be transferred from the latter setting to the former. In particular, our goals include a body of optimization theory on Riemannian manifolds along with a group of algorithms for solving Riemannian optimization problems.

This chapter begins with a brief review of the necessary material from unconstrained Euclidean optimization. The prerequisite theory, as well as the selected algorithms in Section 1.2, will illustrate the machinery needed for Riemannian optimization. Section 1.3 will introduce the topic of Riemannian optimization and review Riemannian geometry and the necessary concepts. Section 1.4 describes the retraction-based Riemannian optmization paradigm, the setting for the Riemannian trust-region methods developed in this dissertation. Section 1.5 describes the manifolds of interest in this dissertation.

1.1 Euclidean Optimization

Numerical optimization consists of a collection of theory and algorithms for finding extreme points of real-valued functions. Take a function $f : \mathbb{R}^n \to \mathbb{R}$. The function f is referred to as the *objective function*. The goal of the optimization is to find points in $\mathbb{R}^n$ which are extreme under f. These extreme points are minimizers and maximizers. A local minimizer is a point which is minimal under f within a certain neighborhood. A formal definition follows.

Definition 1 (Local Minimizer)**.** *Take a function $f : \mathbb{R}^n \to \mathbb{R}$ and a point $x \in \mathbb{R}^n$. If there exists $\epsilon > 0$ such that*

$$f(x) \leq f(y) \tag{1.1}$$

for all $y \neq x$ *satisfying* $\|y - x\| < \epsilon$, *then* x *is a* local minimizer of f.

A *strict local minimizer* is defined by making strict the inequality (1.1). A *(strict) local maximizer* is defined as in Definition 1, by making the appropriate changes to (1.1). The concepts of *(strict) global maximizers and minimizers* come from replacing the neighborhood of the extreme point in Definition 1 by the entire domain of f. It is easily shown that the local maximizers (minimizers) of f are exactly the local minimizers (maximizers) of the objective function $-f$. Therefore, most discussion of optimization is limited to the minimization of the objective function. This dissertation follows that tradition.

The search for minimizers of a function is aided by the necessary and critical conditions on minimality. Assume that x_* is a local minimizer. Then x_* necessarily satisfies the following:

$$\nabla f(x_*) = 0 \tag{1.2}$$

$$\nabla^2 f(x_*) \text{ is positive semidefinite .} \tag{1.3}$$

Equation (1.2) is the *first-order necessary condition for optimality*. Points satisfying this condition are called *critical points* or *stationary points*. Equation (1.3) is the *second-order necessary condition for optimality*. In this dissertation, the terms $\nabla f(x)$ and $\nabla^2 f(x)$ are referred to as the *gradient* and the *Hessian* of f at x. Unless stated otherwise, all functions are assumed to be twice continuously differentiable.

To prove that a point x_* is a local minimizer requires a slightly stronger result, the *sufficient conditions for optimality*:

$$\nabla f(x_*) = 0 \quad \text{and} \quad \nabla^2 f(x_*) \text{ is positive definite .} \tag{1.4}$$

The requirement of positive definiteness for $\nabla^2 f(x_*)$ is the source of much difficulty for large-scale numerical optimization. This requirement is usually too expensive to be verified in practice. However, the requirement that x_* is a critical point is alone *not* enough to guarantee optimality of x_*; x_* could be a local minimizer, a local maximizer, or neither. Nevertheless, many numerical algorithms for optimization are concerned with the discovery of critical points. The next section outlines some popular methods for Euclidean optimization.

1.2 Methods for Euclidean Optimization

Numerical optimization methods typically employ an iteration like

$$x_{k+1} = x_k + \alpha_k p_k \ ,$$

where the *step length* α_k and the *search direction* p_k are chosen by various methods. The goal is to produce a sequence $\{x_k\}$ which converges to a local minimizer. Finding a global minimizer of a function with multiple local minimizers is significantly more difficult; this problem is the focus of *global optimization*. We will not address this topic in this thesis.

It is typical to assume the search direction is a *descent direction*:

$$p_k^T \nabla f(x_k) < 0 \ .$$

This guarantees that, for a small enough step size α_k, the inequality $f(x_{k+1}) < f(x_k)$ can be satisfied. One intuitive and commonly used choice is the *direction of steepest descent*, given by

$$p_k = -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|} \ .$$

The direction of steepest descent is the search direction which produces the greatest decrease under f in the immediate vicinity of x_k. This is the search direction used in the Steepest Descent optimization method.

Other popular choices for search directions often take the form

$$p_k = -B_k^{-1} \nabla f(x_k) \ .$$

If B_k is positive definite, it can be shown that p_k is descent direction:

$$p_k^T \nabla f(x_k) = -\nabla f(x_k)^T B_k^{-1} \nabla f(x_k) < 0 \ .$$

Choosing B_k as the identity clearly recovers the method of Steepest Descent. Newton's Method uses the choice $B_k = \nabla^2 f(x_k)$, whereas quasi-Newton methods will choose B_k as some (positive-definite) approximation to the Hessian that is computed using lower-order information about the objective function.

After choosing a search direction p_k, it still remains to choose a step size α_k. If the search direction is a descent direction, then it is possible to choose a value for α_k such that $f(x_{k+1}) < f(x_k)$. In this way, it can be guaranteed that the sequence $\{x_k\}$ decreases under f, i.e., the iteration moves "downhill".

However, it is not enough to require that our objective function decreases. When attempting to ensure convergence, most methods attempt to give some *sufficient decrease* in the objective function. One condition on decrease is the so-called *Armijo condition*:

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f(x_k)^T p_k \ , \tag{1.5}$$

for some $c_1 \in (0, 1)$. This condition alone is not enough to ensure convergence. A second condition, known as the *curvature condition*, requires α_k to satisfy

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \ , \tag{1.6}$$

where $c_2 \in (c_1, 1)$. These conditions, the Armijo condition and the curvature condition, are collectively known as the *Wolfe conditions*. It can be proven that if f is smooth and bounded below, there exist step lengths which satisfy the Wolfe conditions.

The search directions prescribed above (Steepest Descent and Newton's Method) are often coupled with algorithms that guarantee some sufficient decrease condition. Trust-region implementations usually include a similar sufficient decrease mechanism in order to establish a global convergence theory.

Steepest Descent and Newton's Method are two of the conceptually simplest optimization algorithms. For this reason they have typically been the initial targets when translating Euclidean optimization methods to a Riemannian paradigm. Descriptions of these methods follow. As this dissertation is concerned with the description of and application of trust-region methods on Riemannian manifolds, a review of'ion methods will follow.

1.2.1 Steepest Descent Method

As noted above, choosing a descent direction p_k at iterate x_k allows us to guarantee a descent in the objective function, in some neighborhood of x_k. For small enough distances, the amount of the decrease is proportional to the cosine of the angle between the search direction and the gradient

$$\cos\theta = \frac{p_k^T \nabla f_{(x_k)}}{\|p_k\| \|\nabla f(x_k)\|} \ .$$

Therefore, nearby x_k, the direction maximizing the decrease (the "direction of steepest descent") is $p_k = -\nabla f(x_k)$.

This choice of search direction defines the method of *steepest descent* or *gradient descent*. Approaches requiring the Hessian may not be practical if the Hessian is expensive or unavailable. Steepest descent is attractive for requiring only first-order information about the objective function, namely the gradient. Under assumptions of sufficient decrease (such as those discussed above), steepest descent has provable global convergence to a critical point. Furthermore, only local minimizers are stable attractors of such an implementation,

so steepest descent will converge to a local minimizer in all situations except the most pathological.

In some cases, sufficient knowledge about the objective function may even make it possible to choose an optimal step size, i.e., some α_k minimizing the one-dimensional function

$$\phi_k(\alpha) \doteq f(x_k + \alpha p_k) \ .$$

(In particular, the eigenvalue problem considered in Chapters 4 permits this optimization.) In general, this may not be possible. Inexpensive functions may permit a thorough search along p_k; more expensive functions may require more sophisticated search (e.g., interpolation of ϕ_k). Choosing a step size to satisfy sufficient decrease is commonly done via back-tracking methods [NW99].

Unfortunately, the convergence of steepest descent can be an obstacle to employment of the method. The asymptotic rate of convergence is linear, with a coefficient depending on the eccentricity of the Hessian of the objective function.

The characterization of the search for α_k suggests an illustrative interpretation of steepest descent: first-order information about the objective function is exploited to formulate a one-dimensional problem at each iteration. In order to improve the convergence, it is necessary to include higher-order information about the objective function.

1.2.2 Newton's Method

Newton's method was originally described as a technique for root finding. Given a function $F : \mathbb{R}^n \to \mathbb{R}^n$, Newton's method uses the approximation

$$F(x_k + p) \approx F(x_k) + D_{x_k} F(x_k)[p] \tag{1.7}$$

and chooses a search direction p_k which sets the right hand side to zero.

The first-order necessary optimality condition (Equation (1.2)) requires that a local minimizer x_* of f satisfies $\nabla f(x_*) = 0$, i.e., it is a root of the function $F(x) = \nabla f(x)$. It is possible therefore to apply Newton's method to this $F(x)$ in an attempt to find a critical point of the objective function; this is done in the hope that it is also a local minimizer. Inserting $F(x_k) = \nabla f(x_k)$ into Equation (1.7), setting the right hand side to zero and solving for p yields

$$p_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k) \ .$$

It should first be noted that the method involves the solution of a linear system of order n. Even in the scenario where $\nabla^2 f(x_k)$ is invertible, the cost of solving this system exactly may be prohibitive.

When the matrix is invertible, it may still be indefinite. It was stated earlier that $p_k = -B_k^{-1}\nabla f(x_k)$ is guaranteed to be a descent direction only if B_k is positive definite. Newton's method attempts to find a critical point of the second-order Taylor expansion of f around x_k. An indefinite Hessian means that this quadratic expansion has no local minimizer; ergo, finding a critical point does not necessarily find a local minimizer.

However, in the case where x_k is "close enough" to a local minimizer and the Hessian $\nabla^2 f(x_k)$ is positive definite, Newton's method will converge to the local minimizer. In the special case that the objective function is quadratic and convex, Newton's method will find the local minimizer after one iteration. Furthermore, when Newton's method converges, it does so with a quadratic rate of convergence.

In considering the quadratic expansion of the objective function around the current iterate, note that Newton's method performs best in the case that the Hessian is positive definite. In this case, finding a critical point is equivalent to minimizing the quadratic expansion. One common strategy for employing Newton's method requires first using another optimization method (such as steepest descent) to bring the iteration close enough to a local minimizer that Newton's method will converge.

1.2.3 Trust-region Methods

Newton's method is most successful when the Hessian is positive definite, in which case it is (implicitly) minimizing a quadratic expansion of the function. The second-order information permits a higher rate of convergence than that achieved by steepest descent. However, the global convergence properties of steepest descent are desirable as well, in the cases where the Hessian is not positive definite or invertible. Trust-region methods use ideas from both of these, and they have the ability to capture the beneficial convergence properties of each.

Trust-region methods operate by forming a (usually) quadratic expansion of the objective function, referred to as a *model*:

$$f(x_k + s) \approx m_{x_k}(s) \doteq f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T H_k s \ , \tag{1.8}$$

where H_k is some symmetric operator called the *model Hessian*, that may or may not be

equivalent to the Hessian of the objective function.

The update $s_k = x_{k+1} - x_k$ to the current iterate is chosen by conducting a minimization of m_{x_k}. Taylor's theorem instructs us that such an expansion is only valid nearby the expansion point (i.e., the current iterate x_k). Trust-region methods encode this advice by limiting the maximum step size allowed for s_k. This is done in an adaptive way which limits user interaction while still providing robust convergence. The model m_{x_k} is intended to provide a well understood adjunct for the objective function, and the trust-region mechanism works by comparing the accuracy of this approximation against the objective function.

After the model minimization is complete, the performance of the model at the computed update s_k is analyzed by comparing the model decrease to the decrease in the objective function:

$$\rho_{x_k}(s_k) = \frac{f(x_k) - f(x_k + s_k)}{m_{x_k}(0) - m_{x_k}(s_k)} \ . \tag{1.9}$$

The value of $\rho_{x_k}(s_k)$ is used to adjust the trust-region radius; a small value indicates that the quadratic model performed poorly and should be considered over a smaller area, while a larger value (e.g., closer to 1) indicates that the model minimization is serving well the objective of minimizing f. The value of $\rho_{x_k}(s_k)$ is also used an as acceptance criterion for the proposal $x_k + s_k$. A simple version of a trust-region method is presented in Algorithm 1. More sophisticated techniques exist for updating the trust-region radius which take into account the fidelity of the previous model [NW99, CGT00].

One significant difference between trust-region methods and Newton's method is characterized by the kernel step: Newton's method chooses the update

$$x_{k+1} - x_k = s_k^{\text{NEWTON}} = -\nabla^2 f(x_k)^{-1} \nabla f(x_k) \ ,$$

while the trust-region method chooses s_k as an approximate minimizer of m_{x_k} (subject to the trust-region constraint). If the Hessian $\nabla^2 f(x_k)$ is positive definite, the model minimization is performed exactly, and $\|\nabla^2 f(x_k)^{-1} \nabla f(x_k)\| \leq \Delta_k$, then the two algorithms produce the same update. However, many trust-region implementations will not solve the model minimization exactly; this results in a less expensive update than the exact linear system solve dictated by Newton's method. Furthermore, the singularity and definiteness of the Hessian dictate whether the Newton update exists and whether it is a descent direction. In contrast, the model minimization step of the trust-region method is always well-posed. In

Algorithm 1 Euclidean Trust-Region Algorithm

Require: smooth function $f : \mathbb{R}^n \to \mathbb{R}$

Require: $\bar{\Delta} > 0$, $\Delta_0 \in (0, \bar{\Delta})$, and $\rho' \in [0, \frac{1}{4})$, initial iterate $x_0 \in \mathbb{R}^n$.

for $k = 0, 1, 2, \ldots$ **do**
  Obtain s_k by (approximately) minimizing $m_{x_k}(s)$ subject to $\|s\| \leq \Delta_k$
  Evaluate $\rho_k = \rho_{x_k}(s_k)$ as in (1.9)
  if $\rho_k < \frac{1}{4}$ **then**
    Set $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  else if $\rho_k > \frac{3}{4}$ and $\|s_k\| = \Delta_k$ **then**
    Set $\Delta_{k+1} = \min(2\Delta_k, \bar{\Delta})$
  else
    Set $\Delta_{k+1} = \Delta_k$
  end if
  if $\rho_k > \rho'$ **then**
    Set $x_{k+1} = x_k + s_k$
  else
    Preserve $x_{k+1} = x_k$
  end if
end for

Output: Sequences of iterates $\{x_k\}$.

fact, indefiniteness in the Hessian implies existence of at least one direction which can reduce the model infinitely. In this case, the trust-region radius Δ_k acts as a guide for how far this direction should be followed.

The acceptance mechanism in the trust-region method enforces its operation as a descent method, which enables a much stronger global convergence analysis than is possible for Newton's method. Under very mild conditions, trust-region methods enjoy global convergence to a critical point. Similar to steepest descent, there is stable convergence only to local minimizers. Furthermore, methods exist for solving the model minimization which enable superlinear convergence. This combination of strong global convergence along with fast local convergence makes trust-region methods very appealing for a large class of problems.

1.3 Riemannian Optimization

The previous sections discussed unconstrained Euclidean optimization, where given a function $f : \mathbb{R}^n \to \mathbb{R}$, the goal was to solve

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

It is often the case that the objective function is not defined over all of $\mathbb{R}^n$ or that our interest only covers some subset of $\mathbb{R}^n$. Constrained optimization considers this problem:

$$\min_{x \in R^n,\ c(x)=0} f(x) \ ,$$

where $c : \mathbb{R}^n \to \mathbb{R}^d$ is the constraint function. The goal now is to minimize the function for all x which also satisfy $c(x) = 0$. The nature of extreme points differs from that of unconstrained optimization; the theory and algorithms must be adapted [NW99, NS95].

This dissertation is concerned with the optimization of functions defined on Riemannian manifolds. In some cases, this can be described as a constrained optimization problem. For example, if the manifold in question is the unit sphere,

$$S^{n-1} = \left\{ x \in \mathbb{R}^n : x^T x = 1 \right\} \ ,$$

then the minimization of the function $f : S^{n-1} \subset \mathbb{R}^n \to \mathbb{R}$ can be posed as a constrained minimization problem:

$$\min_{x \in R^n,\ c(x)=0} f(x) \ , \quad \text{with } c(x) = x^T x - 1 \ .$$

Many of the Riemannian manifolds discussed in this dissertation and in the literature can, like the unit sphere, be described as constrained sets in some Euclidean space, allowing the rich methodology of constrained optimization to be applied. However, one purpose of this dissertation and literature is to describe a unifying theory and set of algorithms for optimizing functions defined on general Riemannian manifolds, taking advantage of the unique geometric features available and providing well-motivated, black-box numerical methods for problems not easily described under the constrained optimization paradigm. In this regard, the Riemannian optimization methods discussed in this dissertation seek to perform an *unconstrained optimization* of a function on a Riemannian manifold; this

should be contrasted with using a constrained Euclidean optimization to solve a Riemannian optimization problem.

To illustrate, consider again the example of the unit sphere $S^{n-1} \subset \mathbb{R}^n$. Recall the generic update described for iterative optimization methods:

$$x_{k+1} = x_k + \alpha_k s_k \ .$$

Note that for some $x_k \in S^{n-1}$, x_{k+1} is a member of $\mathbb{R}^n$ for any choice of $\alpha_k s_k$; $\mathbb{R}^n$ is a vector space, so that $x_k + \alpha_k s_k$ is well-defined. However, there are relatively few choices of α_k and s_k which place x_{k+1} on the sphere S^{n-1}. Constrained optimization approaches when applied to this problem take one of two approaches. *Feasible point methods* constrain the selection of α_k and s_k so that all of the iterates $\{x_k\}$ are members of S^{n-1}—specifically, so that they satisfy $c(x_k) = 0$. *Infeasible point methods* allow the iteration sequence $\{x_k\}$ to leave the sphere, but enact measures to guarantee that limit points x_* of the sequence satisfy $c(x_*) = 0$. Methods of the latter type typically need the ability to evaluate the fitness[1] of infeasible points (points not satisfying $c(x) = 0$) and therefore assume that the objective function is defined off the sphere; the former does not.

There is clearly a link between techniques of optimization on manifolds and standard constrained optimization approaches. However, there are manifolds that are not defined as constrained sets[2] in $\mathbb{R}^n$; an important example is the Grassmann manifold (see Section 1.5.2). Also, there are constrained sets that do not admit a regular manifold structure; a simple example is $\{x \in \mathbb{R}^n : \|x\|_\infty = 1\}$. The application areas thus overlap, but are not identical. On the problems that can be tackled by both approaches, an interesting feature of Riemannian optimization schemes is that they are feasible point methods. Feasibility is advantageous in several cases. For example, the cost function is sometimes undefined outside the feasible set; or, the value of the cost function may have little if any relevance outside the feasible set; moreover, if the algorithm runs out of time or computing resources, it should be able to terminate and return a feasible point.

A Riemannian optimization technique considers a function described on a Riemannian manifold and produces a sequence of iterates on the manifold which targets a local minimizer.

[1] with respect to f

[2] Clearly, by Nash's embedding theorem [Nas56], every Riemannian manifold can be smoothly isometrically embedded in a Euclidean space; but this is only an existence theorem, and such an embedding may be elusive or computationally intractable.

Describing these methods requires reviewing some theory from Riemannian manifolds. This is a brief overview; a more thorough treatment of this theory can be found in [dC92, Boo75]. A *Riemannian manifold* is a real differentiable manifold endowed with a Riemannian metric. This is one of many types of manifold, but it provides us the tools needed to perform optimization: differentiability to perform calculus and a Riemannian metric to perform geometry.

A manifold is a set of points which is locally Euclidean. Consider a set $\mathcal{M}$. This set is a *d-dimensional manifold* if every point $x \in \mathcal{M}$ has a neighborhood which resembles Euclidean space, i.e., there exists a subset $U \subset \mathcal{M}$, $x \in U$, and a homeomorphism $\phi : U \to \mathbb{R}^d$. The subset U is a coordinate neighborhood of x, and the pair (U, ϕ) is a chart. A collection of such pairs $\mathcal{A} = \{(U_\alpha, \phi_\alpha)\}$ which cover $\mathcal{M}$ is an atlas on $\mathcal{M}$. The homeomorphic nature of the charts allows us to do topology on the manifold.

Additional requirements on the atlas allow us to perform calculus. Consider two charts, $\{(U_\alpha, \phi_\alpha)\}$ and $\{(U_\beta, \phi_\beta)\}$. The composition $\phi_{\alpha\beta} = \phi_\alpha \circ \phi_\beta^{-1}$ is called a *transition map*:

$$\phi_{\alpha\beta} : \phi_\beta(U_\alpha \cap U_\beta) \subset \mathbb{R}^d \to \phi_\alpha(U_\alpha \cap U_\beta) \subset \mathbb{R}^d \ .$$

If all of the transition maps of the manifold $\mathcal{M}$ are differentiable, then the atlas $\mathcal{A}$ provides a *differentiable structure* for $\mathcal{M}$. Therefore, we refer to $(\mathcal{M}, \mathcal{A})$ as a *differentiable manifold*. When the atlas is clear from context, we may simply refer to $\mathcal{M}$ as a differentiable manifold.

1.3.1 Tangent Spaces

The sphere S^{n-1} example illustrated that, even for an embedded submanifold of $\mathbb{R}^n$, the usual notion of movement between two points must be reconsidered. The foundation of our generic iterative method is the update equation

$$x_{k+1} = x_k + \alpha_k s_k \ .$$

To construct a meaningful manifold analogue it is necessary to find representations of the constituents of this equation. The points x_{k+1} and x_k are members of manifold $\mathcal{M}$; the α_k is a real scalar; and the search direction s_k is a member of the tangent space $T_{x_k}\mathcal{M}$.

For any differentiable manifold $\mathcal{M}$, one can attach to every point $x \in \mathcal{M}$ a tangent space $T_x\mathcal{M}$. This space serves to describe first-order variations of the manifold at x; i.e., the directions moving through a point on the manifold.

Consider a smooth mapping, $\gamma : \mathbb{R} \to \mathcal{M}$, satisfying $\gamma(\tau) = x$. This mapping is a curve on $\mathcal{M}$ that passes through x. A first attempt at describing a "direction at x" might consider the derivative $\gamma'(\tau)$:

$$\gamma'(\tau) = \lim_{h \to 0} \frac{\gamma(\tau + h) - \gamma(\tau)}{h} \ .$$

However, this definition includes the term $\gamma(\tau+h)-\gamma(\tau)$, the "subtraction" of two manifold points. This operation requires a vector space and is not defined for a general manifold.

A solution to this problem is to consider a smooth function $f : \mathcal{M} \to \mathbb{R}$ and to note that $f \circ \gamma$ is a smooth function from $\mathbb{R}$ to $\mathbb{R}$. This function therefore has a well-defined derivative:

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

This approach, combining curves and smooth functions on differentiable manifolds, allows us to define the concept of a tangent vector. Let $\mathcal{F}_x(\mathcal{M})$ be the set of smooth functions defined on a neighborhood of $x \in \mathcal{M}$, and let γ be a curve on $\mathcal{M}$ satisfying $\gamma(0) = x$. Define $\dot{\gamma}(0)$ as the mapping from $\mathcal{F}_x(\mathcal{M})$ to $\mathbb{R}$ satisfying

$$\dot{\gamma}(0)f \doteq \frac{\mathrm{d}f(\gamma(t))}{\mathrm{d}t}\Big|_{t=0}, \quad f \in \mathcal{F}_x(\mathcal{M}) \ .$$

This mapping is a *tangent vector to the curve γ at $t = 0$*. The formal definition of tangent vectors follows.

Definition 2 (tangent vector)**.** *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*

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

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* foot *of the tangent vector ξ_x.*

This definition, while initially seeming overly complex, has many useful properties. To begin with, it is available for a general differentiable manifold, relying on no particular knowledge of the manifold (e.g., embedding in $\mathbb{R}^n$). In this way, the tangent vector is shown to be intrinsic to differential manifolds, independent of our Euclidean intuition. Furthermore, the definition rests on the concept of directional derivatives of a smooth function. In addition to describing directions on the manifold, these derivatives are one of the motivating factors

for developing tangent vectors, and they will be crucial to the description of the Riemannian gradient vector.

Note that this definition of a tangent vector $\xi_x = \dot{\gamma}$ relies on the curve γ. In fact, there are infinitely many curves which realize a particular tangent vector, as defined above. Two curves γ_1 and γ_2, $\gamma_1(0) = x = \gamma_2(0)$, are said to be *tangent at* x if, given a chart (U, ϕ) at x, they satisfy

$$\frac{\mathrm{d}\phi(\gamma_1(t))}{\mathrm{d}t}\Big|_{t=0} = \frac{\mathrm{d}\phi(\gamma_2(t))}{\mathrm{d}t}\Big|_{t=0} .$$

This implies that the mappings $\dot{\gamma}_1(0)$ and $\dot{\gamma}_2(0)$ are equivalent. This definition can be used to define equivalence classes on all curves through x; these equivalence classes are the tangent vectors at x, and the collection $T_x\mathcal{M}$ of all tangent vectors to $x \in \mathcal{M}$ is the *tangent space to* $\mathcal{M}$ *at* x.

For a manifold $\mathcal{M}$ of dimension d, the tangent space $T_x\mathcal{M}$ is a vector space of dimension d. In addition to providing a home for search directions (and in particular, the gradient of a function), the tangent space provides a friendly setting for us to conduct optimization, as discussed later with the introduction of retractions.

1.3.2 Riemannian Metric

The tangent space provides us with a vector space that approximates the manifold. Tangent vectors allow us to take directional derivatives of a function on $\mathcal{M}$. A Riemannian metric will allow us to compute angles and vector lengths on the tangent plane; these are the underpinnings of geometry.

A *Riemannian metric* g is a correspondence between each point $x \in \mathcal{M}$ and a symmetric bilinear form $g_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$, with some smoothness requirements (see [dC92, Boo75] for more info). This inner product, defined at each point on the manifold, turns each tangent space into an abstract Euclidean space capable of supporting a wide variety of algorithms. A *Riemannian manifold* is the combination $(\mathcal{M}, g)$ of a differential manifold and a Riemannian metric; the metric g is said to provide a Riemannian structure for the differentiable manifold $\mathcal{M}$. When the metric g is unclear or unimportant, we may refer to $\mathcal{M}$ as a Riemannian manifold.

Due to the presence of the word "metric", the Riemannian metric is sometimes mistaken as a tool for measuring the lengths of curves on a Riemannian manifold. Whereas the

Riemannian metric directly provides only an inner product on the tangent spaces, the norm induced by this inner product can be used to define a distance metric on $\mathcal{M}$ as follows:

$$d(x,y) = \inf_{\gamma} \left\{ \int_0^1 \|\dot{\gamma}(t)\|_{g_{\gamma(t)}} \mathrm{d}t \right\} , \tag{1.10}$$

where γ is a curve on $\mathcal{M}$ with $\gamma(0) = x$ and $\gamma(1) = y$. This definition of distance, as well as the concept of a distance minimizing curve, is significant to the discussion of geodesics.

This definition of distance on the manifold allows another definition of neighborhoods on the manifold. We denote by $\mathcal{B}_\delta(x)$ the open ball of radius δ around x:

$$\mathcal{B}_\delta(x) = \{y \in \mathcal{M} : d(x,y) < \delta\} .$$

This definition of neighborhoods is used to define local minimizers for a function defined on a manifold. Given a function $f : \mathcal{M} \to \mathbb{R}$, a point x_* is a *strict local minimizer* if there exists some $\delta > 0$ such that

$$f(x) < f(y) \quad \text{for all } y \in \mathcal{B}_\delta(x) .$$

In order to clarify different types of equations, various notations will be used for the Riemannian metric:

$$g_x(\eta,\xi) = g(\eta,\xi) = \langle \eta, \xi \rangle_x = \langle \eta, \xi \rangle .$$

1.3.3 Affine Connections, Geodesics and the Exponential Map

Straight lines in Euclidean space can be defined in a number of ways: a straight line defines the shortest path between points; or a straight line is a curve with zero acceleration ("a straight curve"). The intuitive role of straight lines is played out on Riemannian manifolds by geodesics. A *geodesic* γ on a manifold $\mathcal{M}$ is a curve with zero acceleration, i.e., satisfying

$$\frac{\mathrm{D}^2}{\mathrm{d}t^2}\gamma(t) = 0 \tag{1.11}$$

for all t in the domain of the geodesic. An explanation of this equation follows.

The term $\frac{\mathrm{D}^2}{\mathrm{d}t^2}\gamma$ refers to the acceleration vector of the curve γ. Given a curve γ, there is a well-defined tangent vector $\dot{\gamma}(t)$ at each point $\gamma(t)$ along the curve. The curve defines a vector field $\dot{\gamma}$, a smooth mapping between the interval domain of the curve and the tangent plane at each point along the curve. The acceleration of the curve indicates the

the instantaneous change of $\dot{\gamma}$. However, each tangent vector $\dot{\gamma}(t)$ resides in the tangent space $T_{\gamma(t)}\mathcal{M}$; differentiating $\dot{\gamma}$ requires working with tangent vectors from different tangent spaces. This is enabled by a differetial tool called the affine connection.

An affine connection is so named because it "connects" local tangent spaces, allowing the differentiation of tangent vectors to be performed. Let $\mathcal{X}(\mathcal{M})$ be the set of all smooth vector fields on $\mathcal{M}$. An *affine connection* ∇ is a mapping from $\mathcal{X}(\mathcal{M}) \times \mathcal{X}(\mathcal{M})$ to $\mathcal{X}(\mathcal{M})$. This is a differential operator, and is required to satisfy the following for all $x \in \mathcal{M}$, $f \in \mathcal{F}_x(\mathcal{M})$, $a, b \in \mathbb{R}$, and $\eta, \xi, \zeta \in \mathcal{X}_x(\mathcal{M})$:

1. $\nabla_{f\eta}\xi = f\nabla_\eta\xi$: $\mathcal{F}(\mathcal{M})$-linearity in the first argument;
2. $\nabla_\eta(a\xi + b\zeta) = a\nabla_\eta\xi + b\nabla_\eta\zeta$: $\mathbb{R}$-linearity in the second argument; and
3. $\nabla_\eta(f\xi) = (\eta f)\xi + f\nabla_\eta\xi$: Product rule/Leibniz's law.

At a point x on $\mathcal{M}$, the connection maps tangent vectors $(\eta, \xi) \in T_x\mathcal{M} \times T_x\mathcal{M}$ to a tangent vector $\nabla_\eta\xi \in T_x\mathcal{M}$. The result $\nabla_\eta\xi$ is the *covariant derivative of ξ with respect to η*.

For a general differential manifold $\mathcal{M}$, there exist an infinite number of affine connections. However, for a Riemannian manifold $(\mathcal{M}, g)$, there exists a unique connection, called the *Riemannian connection* or the *Levi-Civita connection*, which exhibits compatibility with the Riemannian metric [dC92, Boo75]. This dissertation assumes the use of the Riemannian connection.

Returning to the discussion of the geodesic, it is now possible to state the form of the acceleration vector of a curve γ at t:

$$\frac{\mathrm{D}^2}{\mathrm{d}t^2}\gamma(t) = \frac{\mathrm{D}}{\mathrm{d}t}\dot{\gamma}(t) = \nabla_{\dot{\gamma}(t)}\dot{\gamma}(t) \ .$$

For the geodesic constraint (1.11) to be satisfied, the change of the tangent vector along the curve must be zero, with respect to the affine connection.

It should be noted that geodesics are also length minimizing curves, so that they satisfy both intuitive definitions of straight lines. Similar to straight lines in Euclidean space, a geodesic is uniquely determined by its starting point and direction. Given a point $x \in \mathcal{M}$ and a tangent vector ξ at x, there is a unique geodesic $\gamma(t; x, \xi)$ satisfying $\gamma(0) = x$ and $\dot{\gamma}(0) = \xi$. Furthermore, geodesics satisfy a homogeneity property; given scalar a, the geodesic γ satisfies

$$\gamma(t; x, a\xi) = \gamma(at; x, \xi) \ .$$

The tangent vector has already been introduced as a notion of direction on the manifold. The properties, that a geodesic is the shortest path between two points and uniquely determined by tangent vector, suggests these curves may be useful for movement along the manifold. The mapping

$$\mathrm{Exp}_x : T_x\mathcal{M} \to \mathcal{M} : \eta \mapsto \gamma(1; x, \eta) \ , \tag{1.12}$$

called the *exponential map*, is a one-to-one mapping between a neighborhood of x and the tangent space $T_x\mathcal{M}$.

The exponential map is useful in that it underpins the concept of normal coordinates. Given a point x in $\mathcal{M}$, there is a ball $B_\epsilon(0_x)$ in $T_x\mathcal{M}$ of radius ϵ around the origin 0_x of $T_x\mathcal{M}$ such that Exp_x is a diffeomorphism of $B_\epsilon(0_x)$ onto an open subset of $\mathcal{M}$. Then $\mathrm{Exp}_x(B_\epsilon(0_x)) = U$ is called a *normal neighborhood* of x, and Exp_x defines a diffeomorphism between the Euclidean space $T_x\mathcal{M}$ and U. The supremum of these ϵ's is the *injectivity radius* $i_x(\mathcal{M})$ at x, and $i(\mathcal{M}) := \inf_{x\in\mathcal{M}} i_x$ is the *injectivity radius* of $\mathcal{M}$. Normal coordinates are defined in a normal neighborhood U by considering an orthonormal basis $\{e_i\}$ of $T_x\mathcal{M}$ and taking $(u_1, \ldots, u_d)$ as the coordinates of $y = \mathrm{Exp}_x\left(\sum_{i=1}^n u_i e_i\right)$. These coordinates provide one way of uniquely representing points in a neighborhood of x.

1.3.4 Gradients and Hessians

The concepts presented in the previous sections allow us to begin to adapt the theory from Euclidean optimization to Riemannian optimization. The first and second-order necessary conditions for optimality, Equations 1.2 and 1.3, depend on the concepts of the gradient and the Hessian of an objective function. The equivalent terms for a function on a Riemannian manifold are defined here.

The gradient for a Euclidean objective function is defined as the direction of steepest ascent. Because the gradient encodes information about the first-order derivatives of a function, it also provides the information necessary for computing the directional derivatives of a function. The definition of tangent vectors was based upon computing directional derivatives of Riemannian functions. Therefore, it should not be surprising that the Riemannian gradient is a tangent vector.

For a function f defined on a Riemannian manifold $(\mathcal{M}, g)$, the *Riemannian gradient*

grad $f(x)$ *of* f *at* x is the unique tangent vector satisfying

$$\langle \operatorname{grad} f(x), \eta \rangle_x = \mathrm{D}f(x)[\eta], \quad \forall \eta \in T_x\mathcal{M} \ .$$

Recall that the definition of a tangent vectors identifies $\mathrm{D}f(x)[\eta] = \eta f$. Note that the notation grad $f(x)$ is used to differentiate between the Euclidean gradient $\nabla f(x)$ and that the identity of the gradient is tied to the Riemannian metric. The differentiability of the manifold allows us to conduct calculus on the manifold, computing directional derivatives. However, identifying the gradient requires the ability to perform geometry, which requires the Riemannian metric.

Implementing methods utilizing second-order information about f, such as Newton's method or a trust-region method, requires the Riemannian Hessian. The Euclidean Hessian contains the second derivative information of the objective function, i.e., it gives an idea about the changes in the gradient in a particular direction. The Riemannian Hessian does the same: given a direction η, it indicates the instantaneous change to the Riemannian gradient. The affine connection provides the ability to conduct differentiation of tangent vectors. In this way, the identity of the Riemannian Hessian is tied to the chosen affine connection.

The *Riemannian Hessian of* f *at* x is the linear mapping from $T_x\mathcal{M}$ to $T_x\mathcal{M}$ defined by

$$\operatorname{Hess} f(x)[\eta] = \nabla_\eta \operatorname{grad} f(x) \ ,$$

for all $\eta \in T_x\mathcal{M}$, where ∇ is the Riemannian connection chosen for $\mathcal{M}$. The requirement that ∇ is the Riemannian connection (instead of an arbitrary affine connection) ensures that the Riemannian Hessian is symmetric with respect to the Riemannian metric:

$$\langle \operatorname{Hess} f(x)[\eta], \xi \rangle_x = \langle \operatorname{Hess} f(x)[\xi], \eta \rangle_x$$

for all $\eta, \xi \in T_x\mathcal{M}$.

1.3.5 Some Methods for Riemannian Optimization

Recall the generic iteration from earlier:

$$x_{k+1} = x_k + \alpha_k s_k \ .$$

The previous section introduced the tools necessary to describe this iteration on a Riemannian manifold. The iterates x_k and x_{k+1} a manifold points, the update vector s_k is a tangent vector, and the addition operation can be implemented via the exponential map. This results in a new, generic iteration of the form

$$x_{k+1} = R_{x_k}(\alpha_k s_k) \ .$$

In this fashion, we can describe the steepest descent and Newton methods from Section 1.1 on the Riemannian manifold. Algorithm 2 outlines the Riemannian Newton method. The history of Newton's method on manifolds can be traced back to Luenberger [Lue72], if not earlier. Gabay [Gab82] proposed a Newton method on embedded submanifolds of $\mathbb{R}^n$. Smith [Smi93, Smi94] and Udrişte [Udr94] formulated and analyzed the method on general Riemannian manifolds. Related work includes [Shu86, EAS98, OW00, Man02, MM02, ADM+02, DPM03, HT04].

Algorithm 2 Riemannian Newton Algorithm

Require: Complete Riemannian manifold $(\mathcal{M}, g)$; scalar field f on $\mathcal{M}$
Input: Initial iterate $x_0 \in \mathcal{M}$.

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: {Compute Newton step}
3: Obtain η_k by solving

$$\operatorname{Hess} f(x_k)[\eta] = -\operatorname{grad} f(x_k)$$

4: {Compute next iterate}
5: Compute step-size α_k
6: Set $x_{k+1} = \operatorname{Exp}_{x_k}(\alpha_k \eta_k)$
7: **end for**

Output: Sequences of iterates $\{x_k\}$.

Now let $(\mathcal{M}, g)$ be a Riemannian manifold of dimension d. Recall the trust-region method of Section 1.2, and consider defining a trust-region method for a cost function f on $\mathcal{M}$. Given a current iterate x, it is tempting to choose a coordinate neighborhood U_α containing x, translate the problem to $\mathbb{R}^d$ through the chart ϕ_α, build a quadratic model m, solve the trust-region problem in $\mathbb{R}^d$ and bring back the solution to $\mathcal{M}$ through ϕ_α^{-1}. One difficulty is that there are in general infinitely many α such that $x \in \Omega_\alpha$. Each choice will yield a different model function $m \circ \phi_\alpha$ and a different trust region $\{y \in \mathcal{M} : \|\phi_\alpha(y)\| \leq \Delta\}$, hence a different next iterate x_+. This kind of situation is pervasive in numerics on manifolds; it is usually addressed by exploiting the exponential map to work in normal coordinates.

1.4 Retraction-based Riemannian Optimization

The exponential map is useful because it can be used to map between points in a neighborhood of $x \in \mathcal{M}$ and the tangent space $T_x\mathcal{M}$. Unfortunately, the strengths of the exponential map, coming from its strict definition, can also be a weakness from a computational point of view. In practice, the zero-acceleration condition (1.11) defining the geodesic can be very expensive to ensure. As pointed out in [Man02], the systematic use of the exponential mapping may not be desirable in all cases: other local mappings to $T_x\mathcal{M}$ may reduce the computational cost while preserving the useful convergence properties of the considered method.

Much as in the work of Shub [Shu86, ADM+02] and some recent work [AMS08], this dissertation forgoes the stricture of the exponential map in favor of an alternative mechanism called a *retraction*. Retractions relax the exponential map to provide a slightly less powerful, yet significantly more flexible tool. A formal definition follows.

Definition 3 (retraction)**.** *A* retraction *on a manifold $\mathcal{M}$ is a mapping R on the tangent bundle TM into $\mathcal{M}$ with the following properties. Let R_x denote the restriction of R to $T_x\mathcal{M}$.*

1. *R is continuously differentiable.*
2. *$R_x(0_x) = x$, where 0_x denotes the zero element of $T_x\mathcal{M}$.*
3. *$DR_x(0_x) = \mathrm{id}_{T_x\mathcal{M}}$, the identity mapping on $T_x\mathcal{M}$, with the canonical identification $T_{0_x}T_x\mathcal{M} \simeq T_x\mathcal{M}$.*

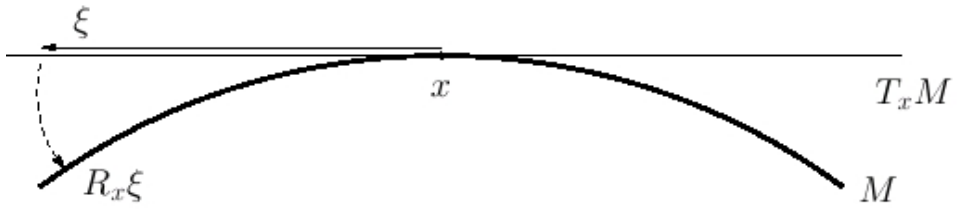

Figure 1.1: Illustration of retractions.

Similar to the exponential map, a retraction maps vectors in the tangent plane $T_x\mathcal{M}$ to points on the manifold $\mathcal{M}$ near x, as illustrated in Figure 1.1. It follows from the inverse

function theorem (see [dC92, Ch. 0, Th. 2.10]) that R_x is a local diffeomorphism at 0_x, namely, R_x is not only C^1 but also bijective with differentiable inverse on a neighborhood V of 0_x in $T_x\mathcal{M}$. In particular, the exponential mapping is a retraction (see Proposition 2.9 in [dC92, Ch. 3] and the proof thereof), the *Riemannian exponential retraction*. The Riemannian exponential retraction exists for every Riemannian manifold, so that every Riemannian manifold is guaranteed to have at least one retraction.

Any other retraction can be thought of as a first-order approximation of the exponential mapping. However, the constraints on the retraction are less stringent than those on the manifold. Both the exponential map Exp_x and a general retraction R_x agree that

$$\mathrm{Exp}_x(0_x) = x = R_x(0_x) \quad \text{and} \quad \mathrm{D}\,\mathrm{Exp}_x(0_x) = \mathrm{id}_{T_x\mathcal{M}} = \mathrm{D}\,R_x(0_x).$$

However, no assumption is made on the second and higher derivatives of the retractions; in particular, $\mathrm{D}^2(\mathrm{Exp}_x^{-1} \circ R_x)(0_x)$ need not vanish. Where the exponential retraction satisfies the zero-acceleration constraint (1.11), a general retraction is required only to satisfy the *local rigidity condition* $\mathrm{D}\,R_x(0_x) = \mathrm{id}_{T_x\mathcal{M}}$. This condition ensures that the curve $\gamma_\eta = R_x(t\eta)$ satisfies $\dot{\gamma}(0) = \eta$. That is, the retraction, which may have nonzero acceleration, defines a curve on the manifold which initially "moves" in the direction specified by the tangent vector. This is contrasted by the exponential map, which moves in a straight line.

To illustrate the distinction between the exponential map and a general retraction, consider analogous mechanisms in Euclidean space. The exponential map is equivalent to moving along a certain direction in a straight line, resulting from the connection between exponential maps and geodesics. However, a retraction, which has a local rigidity condition but no second-order requirements, is equivalent to moving along a curve that initially moved in the specified direction. The benefit of the retraction is that for many manifolds of interest, there exist retractions which are significantly cheaper to compute than the exponential map, as will be discussed in Section 1.5.

A retraction can be used as a mechanism for mapping tangent vectors to nearby manifold points. In this regard, it can be used for movement on the manifold, just as the exponential map was used in Algorithm 2. In addition, this dissertation follows [ABG07] and considers another use of the retraction. By creating a correspondence between the manifold and the tangent plane, the retraction can be used to "lift" a function f defined on the manifold to

the tangent plane as follows:

$$\hat{f}_x : T_x\mathcal{M} \to \mathbb{R} : \eta \mapsto f(R_x(\eta)) \ .$$

The function $\hat{f}_x$ is the *pullback of* f *through* R_x.

The significance of this is that the pullback $\hat{f}_x$ is defined on the tangent space, an abstract Euclidean space. As a result, a large number of algorithms from Euclidean optimization can be easily moved to the manifold. The use of retractions to move the optimization problem from the manifold to the tangent bundle constitutes *retraction-based Riemannian optimization*. Algorithm 3 illustrates a generic Riemannian optimization algorithm using retractions.

Algorithm 3 Generic Riemannian Optimization Algorithm

Require: smooth function $f : \mathcal{M} \to \mathbb{R}$, retraction R
Require: initial iterate $x_0 \in \mathcal{M}$.
1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Compute pullback $\hat{f}_{x_k} : T_{x_k}\mathcal{M} \to \mathbb{R}$
3: Conduct optimization of $\hat{f}_{x_k}$ on $T_{x_k}\mathcal{M}$ to produce s_k
4: Compute next iterate $x_{k+1} = R_{x_k}(s_k)$
5: **end for**
Output: Sequences of iterates $\{x_k\}$.

The "lift-solve-retract" technique can be applied to generalize a wide variety of Euclidean optimization methods to optimization on manifolds. This approach, which finds its roots in the work of Shub [Shu86], seems to have received little attention in the literature until recently [ADM+02, ABG04]. Previous efforts focused on the exponential map as a mechanism for mapping tangent vectors back to the manifold.

Note that it is theoretically possible to choose once and for all the retraction as the Riemannian exponential mapping. This corresponds to the strategy used by numerous Riemannian optimization methods when they compute the exponential of a tangent update vector in order to obtain a new iterate on the manifold; see [Smi94, Udr94, EAS98, Yan07].

However, the use of a retraction may be more computationally feasible. Furthermore, because it has the tangent plane as its domain, the optimization of the pullback $\hat{f}$ is in general significantly easier than the optimization of f on the manifold. In this way, retraction-based approaches explicitly optimize $\hat{f}$ instead of f. The following results justify this approach.

As with most gradient-based Euclidean optimization methods, our optimization strategy is to search for points which satisfy the second-order optimality conditions, i.e., those critical points with a positive-definite Hessian. By choosing to optimize $\hat{f}$ in place of f, we need reassurance that this approach makes sense. This reassurance comes by showing that satisfaction of the optimality conditions for $\hat{f}$ imply satisfaction of those conditions for f. This requires understanding the correspondence between the gradients and Hessians of $\hat{f}$ and f.

The correspondence between the grad $\hat{f}$ and grad f is simple. For a general retraction R and pullback $\hat{f} \doteq f \circ R$, it holds that

$$\operatorname{grad} f(x) = \operatorname{grad} \hat{f}_x(0_x) \ . \tag{1.13}$$

The result of Equation (1.13) is that the search for a critical point of $\hat{f}$ results in a critical point of f.

The gradient vector contains the first order information about a function. The equivalence of the gradients exists because the retraction approximates the exponential map to the first-order. However, for a general retraction R and pullback $\hat{f} = f \circ R$, there is little that can be said to relate the mappings Hess $f(x)$ and Hess $(f \circ R_x)(0_x)$. Two situations exist which alleviate this problem. They are described in the following lemmas. Proofs can be found in [ABG07, AMS08].

Lemma 4. *Suppose that*

$$\frac{D}{\mathrm{d}t}\left(\frac{\mathrm{d}}{\mathrm{d}t}R(t\xi)\right)|_{t=0} = 0, \quad \textit{for all } \xi \in T\mathcal{M}, \tag{1.14}$$

where $\frac{D}{\mathrm{d}t}$ *denotes the covariant derivative along the curve* $t \mapsto R(t\xi)$ *(see [dC92, Ch. 2, Prop. 2.2]). Then* Hess $f(x) =$ Hess $\hat{f}_x(0_x)$.

Lemma 4 places an additional constraint on a retraction. Equation (1.14) is the *zero initial acceleration* condition, and a retraction satisfying this equation is a *second-order retraction*. Note that this is trivially satisfied by the exponential retraction, which by definition has zero acceleration everywhere.

Lemma 5. *Let* R *be a* C^2 *retraction, let* f *be a* C^2 *cost function, and let* v *be a stationary point of* f *(i.e.,* grad $f(v) = 0$*). Then* Hess $\hat{f}_v(0_v) =$ Hess $f(v)$.

In the case that the chosen retraction does not satisfy Equation (1.14), Lemma 5 shows that $\operatorname{Hess} f(x) = \operatorname{Hess}(f \circ R_x)(0_x)$ at critical points. Because we are interested in the use of general retractions for the purpose of optimization, this is the more significant result. It implies that if a point x satisfies the sufficient conditions for optimality for $\hat{f} = f \circ R$, i.e.,

$$\operatorname{grad} \hat{f}_x(0_x) = 0_x \qquad\qquad \operatorname{Hess} \hat{f}_x(0_x) > 0 \ ,$$

then x satisfies the sufficient conditions for optimality of f, i.e.,

$$\operatorname{grad} f(x) = 0_x \qquad\qquad \operatorname{Hess} f(x) > 0 \ .$$

This development provides the theoretical basis for retraction-based Riemannian optimization.

1.5 Riemannian Manifolds of Interest

The theory presented thus far relates to general Riemannian manifolds, and the algorithms presented in this dissertation are applicable to general Riemannian manifolds. However, the application studied in Chapter 4 will consider the Grassmann manifold and the related Stiefel manifold. The orthogonal Stiefel manifold is the set

$$\operatorname{St}(p, n) = \left\{ X \in \mathbb{R}^{n \times p} : X^T X = I_p \right\} \ .$$

This is the set of rank-p orthonormal bases for $\mathbb{R}^n$. The manifold is of particular interest here as its elements form the foundation for matrix factorizations, such as the SVD.

The Grassmann manifold $\operatorname{Grass}(p, n)$ is the set of p-dimensional subspaces of $\mathbb{R}^n$. Whereas the Stiefel matrix has a direct matrix representation, the Grassmann manifold admits many different representations, some being more amenable to computation than others. For example, an element of $\operatorname{Grass}(p, n)$ can be uniquely represented by its projector [MS85]. However, this requires n^2 parameters to represent a point on the Grassmann manifold, a manifold of dimension $np - p^2$. Another possibility is to rely on the definition of $\operatorname{Grass}(p, n)$ as a quotient of Lie groups; see [EAS98] and references therein. Yet another possibility is to use the coordinate charts; see, e.g., [HM94, Section C4]. The latter has the drawback of relying on arbitrarily fixed reference points.

As in [AMS04, ABG07, BAG08], the approach considered in this dissertation is to treat the Grassmann manifold $\operatorname{Grass}(p, n)$ as the quotient

$$\operatorname{Grass}(p, n) = \mathbb{R}_*^{n \times p} / \operatorname{GL}_p$$

of the set $\mathbb{R}_*^{n\times p}$ of full-rank $n\times p$ matrices by the set of of transformations GL_p which preserve the column space. The result is that an element $\mathcal{X} \in \mathrm{Grass}(p,n)$ (i.e., a p-dimensional subspace of $\mathbb{R}^n$) can be represented by any basis $X \in \mathbb{R}_*^{n\times p}$ whose columns span $\mathcal{X}$.

The following subsections will present the necessities for the Stiefel and Grassmann manifolds: tangent spaces, Riemannian metrics, retractions, gradients, and Hessians.

1.5.1 Geometry of the Stiefel Manifold

The orthogonal Stiefel manifold is an embedded manifold of $\mathbb{R}^{n\times p}$. In this context, we refer to the space $\mathbb{R}^{n\times p}$ as the *embedding space*. The dimension of $\mathrm{St}(p,n)$ is $np - \frac{1}{2}p(p+1)$. The embedding space is a Euclidean space which provides the differentiable structure for the embedded space $\mathrm{St}(p,n)$.

Let $X \in \mathrm{St}(p,n)$ and consider a curve γ on $\mathrm{St}(p,n)$, $\gamma(0) = X$. We previously identified the tangent vector $\dot\gamma(0)$ as a mapping from the differentiable functions on $\mathrm{St}(p,n)$ to their directional derivatives. However, the definition

$$\gamma'(0) = \lim_{h\to 0} \frac{\gamma(h) - \gamma(0)}{h}$$

now is well-defined, because the embedding space provides a vector space for the operation $\gamma(h) - \gamma(0)$. We now identify tangent vectors as the mappings $\dot\gamma$ from $\mathcal{F}_x$ to $\mathbb{R}$ (as in Definition 2) as well as the derivatives $\gamma'(0) \in \mathbb{R}^{p\times p}$ of curves on $\mathrm{St}(p,n)$.

We can use this to explore the tangent space $T_X\mathrm{St}(p,n)$. At each point t along the curve, $\gamma(t) \in \mathrm{St}(p,n)$, so that

$$\gamma(t)^T\gamma(t) = I_p \ .$$

Differentiating this equation yields

$$\dot\gamma(t)^T\gamma(t) + \gamma(t)^T\dot\gamma(t) = 0 \ ,$$

so that $\dot\gamma(t)^T\gamma(t)$ is skew-symmetric. Then tangent vectors $\dot\gamma(t) \in T_{\gamma(t)}\mathrm{St}(p,n)$ can be written

$$\dot\gamma(t) = \left\{\gamma(t)\Omega + B(t) : \Omega = -\Omega^T, \gamma(t)^T B(t) = 0_{n\times p}\right\} \ .$$

An embedded manifold inherits the Riemannian metric of its embedding space. For the Stiefel manifold $\mathrm{St}(p,n)$, this is

$$\langle \eta, \xi\rangle_X = \mathrm{trace}\left(\eta^T\xi\right) \ .$$

Recall that the tangent space $T_X\mathrm{St}(p,n)$ is a subspace of $T_X\mathbb{R}^{n\times p} \cong \mathbb{R}^{n\times p}$. The inner product can be used to decompose each vector in $\mathbb{R}^{n\times p}$ into components in $T_X\mathrm{St}(p,n)$ and orthogonal to $T_X\mathrm{St}(p,n)$. The latter is the *normal space to* $\mathrm{St}(p,n)$ *at* X, and it takes the form

$$(T_X\mathrm{St}(p,n))^\perp = \left\{XS : S \in R^{p\times p}, S = S^T\right\} \ .$$

Let P_X and $P_X^\perp$ denote the projectors from $\mathbb{R}^{n\times p}$ onto $T_X\mathrm{St}(p,n)$ and $(T_X\mathrm{St}(p,n))^\perp$, respectively:

$$\begin{aligned} P_X\eta &= X\mathrm{skew}(X^T\eta) + (I - XX^T)\eta \\ P_X^\perp\eta &= X\mathrm{sym}(X^T\eta) \ , \end{aligned}$$

where $\mathrm{skew}(A)$ and $\mathrm{sym}(A)$ are the decomposition of a matrix into symmetric and skew-symmetric parts:

$$\begin{aligned} \mathrm{skew}(A) &= \frac{1}{2}(A - A^T) \\ \mathrm{sym}(A) &= \frac{1}{2}(A + A^T) \ . \end{aligned}$$

There are a number of retractions that can be described for the Stiefel. The exponential retraction is defined by the geodesic:

$$\gamma(t, X, \eta) = \begin{bmatrix} X & \eta \end{bmatrix} \exp\left(t \begin{bmatrix} A & -S \\ I & A \end{bmatrix}\right) \begin{bmatrix} I \\ 0 \end{bmatrix} \exp(-At) \ , \tag{1.15}$$

where $A = X^T\eta$ and $S = \eta^T\eta$. The exponential retraction then is

$$\begin{aligned} R_X^{\exp}\eta &= \gamma(1; X, \eta) \\ &= \begin{bmatrix} X & \eta \end{bmatrix} \exp\left(\begin{bmatrix} A & -S \\ I & A \end{bmatrix}\right) \begin{bmatrix} I \\ 0 \end{bmatrix} \exp(-A) \ . \end{aligned}$$

Another retraction for the Stiefel is

$$R_X^{\mathrm{qf}}\eta = \mathrm{qf}(X + \eta) \ , \tag{1.16}$$

where $\mathrm{qf}(B)$ returns the Q factor of a thin QR factorization of B. This retraction does not satisfy the zero initial acceleration condition (1.14). However, it is less expensive to apply than the exponential retraction.

Defining the gradient of a function on the Stiefel can be eased by its embedding in $\mathbb{R}^{n\times p}$. Given a function $\bar{f}$ defined $\mathbb{R}^{n\times p}$, let f be the restriction of $\bar{f}$ to $\mathrm{St}(p,n)$. Then the gradient of f is given by

$$\mathrm{grad}\, f(X) = P_X \mathrm{grad}\, \bar{f}(X) \ . \tag{1.17}$$

Defining the Riemannian Hessian requires an affine connection. As with the Riemannian metric, the Stiefel manifold inherits the Riemannian connection of the embedding space $\mathbb{R}^{n\times p}$. Because the embedding space is a vector space, this is easily computed:

$$\nabla_\eta \xi = P_X \left(\mathrm{D}\, \xi(X)[\eta]\right) \ . \tag{1.18}$$

The projection P_X serves to ensure that ∇ is a connection, as the classical derivative $\mathrm{D}\, \xi(X)[\eta]$ does not necessarily produce a vector in $T_X\mathrm{St}(p,n)$.

Using this formula, the Riemannian Hessian of f is given by

$$\mathrm{Hess}\, f(X)[\eta] = P_X \mathrm{D}\, \left(\mathrm{grad}\, f(X)\right)[\eta] \ . \tag{1.19}$$

1.5.2 Geometry of the Grassmann Manifold

The Grassmann manifold will be treated here as the quotient manifold $\mathbb{R}_*^{n\times p}/\mathrm{GL}_p$. The space $\mathbb{R}_*^{n\times p}$ is the set of all full-rank $n\times p$ matrices and is the *total space* of the quotient. We define the *canonical projection*

$$\pi : \mathbb{R}_*^{n\times p} \to \mathrm{Grass}(p,n) : X \mapsto \mathrm{colsp}\,(X) \ .$$

We denote by $\pi(X)$ a point in $\mathrm{Grass}(p,n)$, and by $\pi^{-1}(\pi(X))$ a set of points

$$\pi^{-1}(\pi(X)) = \left\{Y \in \mathbb{R}_*^{n\times p} : \mathrm{colsp}\,(X) = \mathrm{colsp}\,(Y)\right\} \subset \mathbb{R}_*^{n\times p} \ .$$

A point $\mathcal{X} \in \mathrm{Grass}(p,n)$ is represented by any matrix $X \in \mathbb{R}_*^{n\times p}$ such that $\mathrm{colsp}\,(X) = \pi(X) = \mathcal{X}$. The benefit of this approach is that it allows elements on the Grassmann manifold to be easily represented on a computer. However, the quotient manifold treatment of the Grassmann manifold comes with some subtle problems not present for the orthogonal Stiefel manifold. The freedom in representing a particular element $\mathcal{X}$ as one of an infinite number of suitable bases means that the representation of tangent vectors requires extra consideration.

Tangent vectors were motivated by two different purposes: directional derivatives of functions and elementary variations of manifolds points. Consider the latter. Understanding the variations of a subspace in $\mathrm{Grass}(p,n)$ first requires understanding the variations of bases in the total space $\mathbb{R}_*^{n\times p}$. The tangent plane $T_X\mathbb{R}_*^{n\times p}$ is $\mathbb{R}^{n\times p}$. When choosing an element of $T_X\mathbb{R}_*^{n\times p}$ to represent a tangent vector $\eta \in T_{\mathcal{X}}\mathrm{Grass}(p,n)$, any element $\bar{\eta} \in T_X\mathbb{R}_*^{n\times p}$ satisfying

$$\mathrm{D}\,\pi(X)[\bar{\eta}] = \eta$$

will suffice. However, there are infinitely many $\bar{\eta}$ which satisfy this. When implementing numerical algorithms, it is useful to have a unique representation for each tangent vector. In this way, a unique vector in $T_X\mathbb{R}_*^{n\times p}$ can unambiguously represent a tangent vector in $T_{\mathcal{X}}\mathrm{Grass}(p,n)$.

The solution is to note that the set $\pi^{-1}(\mathcal{X})$ is an embedded submanifold of $\mathbb{R}_*^{n\times p}$. This manifold therefore has a tangent space which is a subspace of $T_X\mathbb{R}_*^{n\times p}$. This tangent space is called the *vertical space at* X, and is denoted

$$\mathcal{V}_X = T_X\left(\pi^{-1}(\mathcal{X})\right) \ .$$

A mapping $\mathcal{H}$ that assigns to each $X \in \mathbb{R}_*^{n\times p}$ a subspace $\mathcal{H}_X$ of $T_X\mathbb{R}_*^{n\times p}$ such that $\mathcal{V}_X \oplus \mathcal{H}_X = T_X\mathbb{R}_*^{n\times p}$ is called a *horizontal distribution*, and the subspace $\mathcal{H}_X$ is the *horizontal space at* X.

For a tangent vector $\eta \in T_{\mathcal{X}}\mathrm{Grass}(p,n)$, there is a unique vector $\bar{\eta} \in T_X\mathbb{R}_*^{n\times p}$ in $\mathcal{H}_X$ satisfying $\mathrm{D}\,\pi(X)[\bar{\eta}] = \eta$. This vector is denoted $\eta_{\uparrow X}$ and is referred to as the *horizontal lift of* η *at* X. For two different representations $X_1, X_2 \in \mathbb{R}_*$, satisfying

$$\mathrm{colsp}\,(X_1) = \mathcal{X} = \mathrm{colsp}\,(X_2) \ ,$$

there are two different horizontal lifts $\eta_{\uparrow X_1}$ and $\eta_{\uparrow X_2}$ representing η at X_1 and X_2, respectively. That is, both of these satisfy

$$\mathrm{D}\,\pi(X_1)[\eta_{\uparrow X_1}] = \eta = \mathrm{D}\,\pi(X_2)[\eta_{\uparrow X_2}] \ ,$$

and for any smooth function f on $\mathrm{Grass}(p,n)$,

$$\mathrm{D}\,f(\pi(X_1))[\eta_{\uparrow X_1}] = \eta f = \mathrm{D}\,f(\pi(X_2))[\eta_{\uparrow X_2}] \ .$$

In this regard, the horizontal space $\mathcal{H}_X$ represents the tangent space $T_{\mathcal{X}}\mathrm{Grass}(p,n)$.

This invariance to representation is required to extend to the Riemannian structure as well. The tangent space is a property of a differentiable manifold; we have shown how $\mathrm{Grass}(p,n)$ is represented as a differentiable quotient manifold of the total space $\mathbb{R}_*^{n\times p}$. The Grassmann manifold can also be defined as a Riemannian quotient manifold of the Riemannian manifold $\mathbb{R}_*^{n\times p}$. Let the total space $\mathbb{R}_*^{n\times p}$ have Riemannian metric $\bar{g}$; i.e., $(\mathbb{R}_*^{n\times p}, \bar{g})$ is a Riemannian manifold. Suppose that for every $\mathcal{X} \in \mathrm{Grass}(p,n)$ and $\xi, \eta \in T_{\mathcal{X}}\mathrm{Grass}(p,n)$, the expression $\bar{g}_X(\xi_{\uparrow X}, \eta_{\uparrow X})$ does not depend on the choice of representation $X \in \pi^{-1}(\mathcal{X})$. That is, for $X_1, X_2 \in \pi^{-1}(\mathcal{X})$,

$$\bar{g}_{X_1}\left(\xi_{\uparrow X_1}, \eta_{\uparrow X_1}\right) = \bar{g}_{X_2}\left(\xi_{\uparrow X_2}, \eta_{\uparrow X_2}\right) \ .$$

Then

$$g_{\mathcal{X}}(\xi,\eta) \doteq \bar{g}_X(\xi_{\uparrow X}, \eta_{\uparrow X})$$

defines a Riemannian metric on $\mathrm{Grass}(p,n)$, and $(\mathrm{Grass}(p,n), g)$ is a *Riemannian quotient manifold of* $\mathbb{R}_*^{n\times p}$.

Consider the Riemannian metric for $\mathbb{R}_*^{n\times p}$ is given by

$$\bar{g}_X(\bar{\eta}, \bar{\xi}) = \mathrm{trace}\left((X^TX)^{-1}\bar{\eta}^T\bar{\xi}\right) \ . \tag{1.20}$$

for $X \in \mathbb{R}_*^{n\times p}$ and $\bar{\eta}, \bar{\xi} \in T_X\mathbb{R}_*^{n\times p}$. It can be shown [AMS04, AMS08] that this metric provides a Riemannian quotient structure for $\mathrm{Grass}(p,n)$:

$$g_{\mathrm{colsp}(X)}(\eta,\xi) = \mathrm{trace}\left((X^TX)^{-1}\eta_{\uparrow X}^T\xi_{\uparrow X}\right) \ . \tag{1.21}$$

For the Grassmann manifold, the set $\pi^{-1}(\mathcal{X})$ is given by

$$\pi^{-1}(\mathcal{X}) = \{XM : M \in \mathrm{GL}_p\} \ .$$

The tangent space of this submanifold identifies the vertical space as

$$\mathcal{V}_X = \left\{XM : M \in \mathbb{R}^{p\times p}\right\} \ .$$

The horizontal distribution is canonically chosen so that elements of $\mathcal{H}_X$ are orthogonal to elements of $\mathcal{V}_X$ with respect to the Riemannian metric. Therefore, the canonical choice for the horizontal distribution, using the canonical Riemannian metric (1.21), is

$$\mathcal{H}_X = \left\{Z \in \mathbb{R}^{n\times p} : Z^TX = 0\right\} \ .$$

As with all Riemannian manifolds, the Grassmann manifold admits a retraction in the form of the exponential retraction. This retraction is defined by the geodesic, which is defined with respect to the Riemannian metric. For the canonical metric (1.21), the geodesic is given by

$$\gamma(t; \operatorname{colsp}(X), \eta) = \operatorname{colsp}\left(X(X^TX)^{-1/2}V\cos(\Sigma t) + U\sin(\Sigma t)\right) , \tag{1.22}$$

where $U\Sigma V^T = \eta_{\uparrow X}$ is a thin singular value decomposition [GV96] of $\eta_{\uparrow X}$. An alternative retraction is

$$R_{\operatorname{colsp}(X)}\eta = \operatorname{colsp}(X + \eta_{\uparrow X}) .$$

As with the qf retraction for the Stiefel manifold, this retraction is inexpensive to apply, at the expense of not satisfying the zero initial acceleration condition. However, unlike the qf retraction for the Stiefel manifold, this retraction does not specify an orthonormal basis as output. In fact, any basis may be used to represent $\operatorname{colsp}(X + \eta_{\uparrow X})$, though in practice, an orthonormal basis may be desired for numerical properties.

Given a function $\bar{f}$, defined on $R_*^{n\times p}$ and invariant to change of basis, consider its restriction $f(\operatorname{colsp}(X)) = \bar{f}(X)$ to $\operatorname{Grass}(p, n)$. If the horizontal distribution is such that $\mathcal{V}_X$ and $\mathcal{H}_X$ are orthogonal with respect to the Riemannian metric, then it holds that

$$\operatorname{grad} f(\mathcal{X})_{\uparrow X} = \operatorname{grad}\bar{f}(X) .$$

Similarly, the Riemannian connection is inherited from the total space as well:

$$(\nabla_\eta \xi)_{\uparrow X} = P_X^h\left(\mathrm{D}\,\eta_{\uparrow X}[\xi_{\uparrow X}]\right) ,$$

where P_X^h is the projection from $T_X\mathbb{R}_*^{n\times p}$ onto $\mathcal{H}_X$. This yields the Riemannian Hessian:

$$(\operatorname{Hess} f(\mathcal{X})[\eta])_{\uparrow X} = P_X^h\left(\mathrm{D}\left(\operatorname{grad}\bar{f}(X)\right)[\eta_{\uparrow X}]\right) .$$

These results relied on the assumption that the horizontal space and the vertical space are complementary under the Riemannian metric. If this is not the case, then the Riemannian gradient and Riemannian Hessian must be computed in some other manner.

CHAPTER 2

THE RIEMANNIAN TRUST-REGION METHOD

This chapter describes a trust-region method for Riemannian optimization. Like the generic Riemannian Optimization method (Algorithm 3), the trust-region method described here utilizes a retraction R_x to lift the cost function f from the manifold $\mathcal{M}$ to the tangent space. The pullback $\hat{f}_x = f \circ R_x$ is defined on $T_x\mathcal{M}$, an abstract Euclidean space. This enables Euclidean optimization methods to be easily applied to the optimization of $\hat{f}$. Section 2.1 describes a Riemannian Trust-Region (RTR) Method under the paradigm of retraction-based Riemannian optimization, and Section 2.2 moves the Steihaug-Toint conjugate gradient algorithm to the tangent plane.

It is the "lift-solve-retract" procedure that distinguishes the proposed RTR approach from the Euclidean trust-region methods; the Euclidean methods, since they live in $\mathbb{R}^n$, only require the "solve" part. On a manifold, the "lift" step creates the pullback $\hat{f}$ defined on a friendly Euclidean world (the tangent space $T_x\mathcal{M}$), where classical techniques can be applied; the "retract" step brings the result back to the manifold. A difficulty, from an analysis perspective, is that the RTR method does not deal with a unique cost function (as in the classical case) at each "solve" step, but rather with a succession of different lifted cost functions $\hat{f}_{x_k}$, where x_k is the kth iterate. This will comprise most of the effort in the global and local convergence proofs that follow, which otherwise follow from those of their Euclidean counterparts.

The motivation for developing the Riemannian Trust-Region method is the same as for the Euclidean Trust-Region. Similar to its Euclidean counterpart, Riemannian Steepest Descent has only a linear rate of convergence, albeit with a strong global convergence behavior. Riemannian Newton's Method, similar to Euclidean Newton's Method, has a superlinear rate of local convergence. However, this comes at the expense of global convergence, and

the Newton iteration requires the exact solution of linear system at each step, a potentially expensive operation. Euclidean Trust-Region methods admit a strong global convergence theory, while also exhibiting superlinear rates of local convergence. Section 2.3 shows that the Riemannian Trust-Region method retains the pleasant convergence properties of the Euclidean Trust-Region method.

2.1 RTR algorithm

Recalling Algorithm 1 from Section 1.2, the basic trust-region method in $\mathbb{R}^n$ for a cost function f consists of adding to the current iterate $x \in \mathbb{R}^n$ the update vector $\eta \in \mathbb{R}^n$ solving *the trust-region subproblem*:

$$\min_{\eta \in \mathbb{R}^n} m_x(\eta) = f(x) + \eta^T \nabla f(x) + \frac{1}{2}\eta^T \nabla^2 f(x)\eta \qquad \|\eta\| \leq \Delta \ , \tag{2.1}$$

where $\nabla f(x) = (\partial_1 f(x), \ldots, \partial_n f(x))$ is the gradient of f at x, $(\nabla^2 f(x))_{ij} = \partial^2_{ij} f(x)$ is the Hessian matrix of f at x[1] and Δ is the trust-region radius. The quality of the model m_x is assessed by forming the quotient

$$\rho = \frac{f(x) - f(x+\eta)}{m(0) - m(\eta)}.$$

Depending on the value of ρ, the new iterate will be accepted or discarded and the trust-region radius Δ will be updated. Major references on trust-region methods include the early work of Powell [Pow74] and Moré and Sorensen [Sor82, MS84], and the textbook [CGT00]; see also [WD05] and references therein for recent work on trust-region update strategies.

With a view towards extending the concept of the trust-region subproblem to manifolds, we first consider the case of an abstract *Euclidean space*, i.e., a vector space endowed with an inner product (i.e., a symmetric, bilinear, positive-definite form). This generalization to a Euclidean space $\mathcal{E}$ of dimension d requires little effort since $\mathcal{E}$ may be identified with $\mathbb{R}^d$ once a basis of $\mathcal{E}$ is chosen (we refer to [Boo75, Section I.2] for a discussion on the distinction between $\mathbb{R}^n$ and abstract Euclidean spaces). Let $g(\cdot, \cdot)$ denote the inner product on $\mathcal{E}$. Given a function $f : \mathcal{E} \to \mathbb{R}$ and a current iterate $x \in \mathcal{E}$, one can choose a basis $(e_i)_{i=1,\ldots,d}$ of $\mathcal{E}$ (not necessarily orthonormal with respect to the inner product) and write a classical G-norm

[1] Some convergence results allow for $\nabla^2 f(x)$ in (2.1) to be replaced by any symmetric matrix, but we postpone this relaxation until later in the development.

trust-region subproblem (see, e.g., [GLRT99, Section 2])

$$\min_{\bar{\eta}\in\mathbb{R}^d} m(\bar{\eta}) := \bar{f}(\bar{x}) + \nabla\bar{f}(\bar{x})\bar{\eta} + \frac{1}{2}\bar{\eta}^T\nabla^2\bar{f}(\bar{x})\bar{\eta}, \quad \bar{\eta}^T G\bar{\eta} \leq \Delta_x^2 \tag{2.2}$$

where $x = \sum_i \bar{x}_i e_i$, $\eta = \sum_i \bar{\eta}_i e_i$, $\bar{f}(\bar{x}) = f(\sum_i \bar{x}_i e_i)$ and $G_{ij} = g(e_i, e_j)$. It can be shown that $m(\eta)$ does not depend on the choice of basis $(e_i)_{i=1,\dots,d}$; therefore (2.2) can be written as a coordinate-free expression

$$\begin{aligned}\min_{\eta\in\mathcal{E}} m(\eta) &= f(x) + \mathrm{D}f(x)[\eta] + \frac{1}{2}\mathrm{D}^2 f(x)[\eta,\eta] \\ &= f(x) + g(\operatorname{grad} f(x), \eta) + \frac{1}{2} g(\operatorname{Hess} f[\eta], \eta) \quad \text{s.t. } g(\eta,\eta) \leq \Delta_x^2\end{aligned} \tag{2.3}$$

for the trust-region subproblem in the Euclidean space $\mathcal{E}$.

An important observation is that, for the purpose of defining a trust-region method, the choice of a basis $\{e_i\}$ in $T_x\mathcal{M}$ is immaterial, since trust-region subproblems on a Euclidean space (in particular, $T_x\mathcal{M}$) admit a coordinate-free expression (2.3). Therefore, an arbitrary retraction makes it possible to uniquely define trust-region subproblems on Riemannian manifolds by locally mapping the manifold to the Euclidean space $T_x\mathcal{M}$.

We can now lay out the structure of a trust-region method on a Riemannian manifold $(\mathcal{M}, g)$ with retraction R. Given a cost function $f : \mathcal{M} \to \mathbb{R}$ and a current iterate $x_k \in \mathcal{M}$, we use R_{x_k} to locally map the minimization problem for f on $\mathcal{M}$ into a minimization problem for the pullback

$$\hat{f}_{x_k} : T_{x_k}\mathcal{M} \to \mathbb{R} : \xi \mapsto f(R_{x_k}(\xi)). \tag{2.4}$$

The Riemannian metric g turns $T_{x_k}\mathcal{M}$ into a Euclidean space endowed with the inner product $g_{x_k}(\cdot,\cdot)$, and, following (2.3), the trust-region subproblem on $T_{x_k}\mathcal{M}$ reads

$$\begin{aligned}\min_{\eta\in T_{x_k}\mathcal{M}} m_{x_k}(\eta) &= \hat{f}_{x_k}(0_{x_k}) + \mathrm{D}\hat{f}_{x_k}(0_{x_k})[\eta] + \frac{1}{2}\mathrm{D}^2\hat{f}_{x_k}(0_{x_k})[\eta,\eta] \\ &= \hat{f}_{x_k}(0_{x_k}) + g_{x_k}\left(\operatorname{grad}\hat{f}_{x_k}(0_{x_k}), \eta\right) + \frac{1}{2} g_{x_k}\left(\operatorname{Hess}\hat{f}_{x_k}(0_{x_k})[\eta], \eta\right) \quad \text{s.t. } g_{x_k}(\eta,\eta) \leq \Delta_k^2 \ .\end{aligned} \tag{2.5}$$

For the global convergence theory it is only required that the second-order term in the model be some symmetric form. Therefore, instead of (2.5), we will consider the following more general formulation

$$\min_{\eta\in T_{x_k}\mathcal{M}} m_{x_k}(\eta) = f(x_k) + g_{x_k}(\operatorname{grad} f(x_k), \eta) + \frac{1}{2} g_{x_k}(H_{x_k}\eta, \eta) \quad \text{s.t. } g_{x_k}(\eta,\eta) \leq \Delta_k^2 \ , \tag{2.6}$$

where $H_{x_k} : T_{x_k}\mathcal{M} \to T_{x_k}\mathcal{M}$ is some symmetric linear operator, i.e., $g_{x_k}(H_{x_k}\xi, \chi) = g_{x_k}(\xi, H_{x_k}\chi)$, $\xi, \chi \in T_{x_k}\mathcal{M}$. This is called the *trust-region subproblem*.

Next, an (approximate) solution η_k of the Euclidean trust-region subproblem (2.6) is computed using any available method: if an iterative method is used, its iterations are called inner iterations of the overall algorithm (see Section 2.2). The candidate for the new iterate is then given by $x_+ = R_{x_k}(\eta_k)$.

The decision to accept or not the candidate and to update the trust-region radius is based on the quotient

$$\rho_k = \rho_{x_k}(\eta_x) = \frac{f(x_k) - f(R_{x_k}(\eta_k))}{m_{x_k}(0_{x_k}) - m_{x_k}(\eta_k)} = \frac{\hat{f}_{x_k}(0_{x_k}) - \hat{f}_{x_k}(\eta_k)}{m_{x_k}(0_{x_k}) - m_{x_k}(\eta_k)}. \tag{2.7}$$

If ρ_k is exceedingly small, then the model is very inaccurate: the step must be rejected and the trust-region radius must be reduced. If ρ_k is small but less dramatically so, then the step is accepted but the trust-region radius is reduced. If ρ_k is close to 1, then there is a good agreement between the model and the function over the step, and the trust-region radius can be expanded.

This procedure can be formalized as the following algorithm. It should be noted that this algorithm is equivalent to the classical $\mathbb{R}^n$ trust-region algorithm. That is, when the manifold $\mathcal{M}$ is Euclidean space $\mathbb{R}^n$, Algorithm 4 becomes [NW99, Alg. 4.1]. This can be seen by considering the following canonical choices: $\mathcal{M} = \mathbb{R}^n$, $T_x\mathbb{R}^n = \mathbb{R}^n$, $g(\xi, \zeta) = \xi^T\zeta$, and $R_x(\xi) = \mathrm{Exp}_x\xi = x + \xi$.

In general, there is no assumption on the operator H_{x_k} in (2.6) other than being a symmetric linear operator. Definition 3 requires that

$$\hat{f}_{x_k}(0_{x_k}) = f(x_k) \ ,$$
$$\mathrm{grad}\hat{f}_{x_k}(0_{x_k}) = \mathrm{grad}f(x_k) \ ,$$

regardless of the choice of retraction R. Consequently, even though m_{x_k} was initially presented as a model of $f \circ R_{x_k}$, the choice of the retraction R_{x_k} does not impose any constraint on m_{x_k}. In order to achieve superlinear convergence, however, H_{x_k} will be required to be an "approximate" Hessian (Theorem 16). Obtaining an appropriate approximate Hessian in practice is addressed in Section 2.4. A possible way of choosing H_{x_k} is to define m_x as the quadratic model of $f \circ \tilde{R}_{x_k}$, where $\tilde{R}_x$ is a retraction, not necessarily equal to R_{x_k}; a similar point of view was adopted in [HT04] in the framework of Newton's method.

Algorithm 4 Basic Riemannian Trust-Region Algorithm

Require: Complete Riemannian manifold $(\mathcal{M}, g)$; scalar field f on $\mathcal{M}$; retraction R from $T\mathcal{M}$ to $\mathcal{M}$.

Input: $\bar{\Delta} > 0$, $\Delta_0 \in (0, \bar{\Delta})$, and $\rho' \in [0, \frac{1}{4})$, initial iterate $x_0 \in \mathcal{M}$.

for $k = 0, 1, 2, \ldots$ **do**
  {Model-based minimization}
  Obtain η_k by approximately solving (2.6)
  Evaluate $\rho_k = \rho_{x_k}(\eta_k)$ as in (2.7)
  {Adjust trust region}
  if $\rho_k < \frac{1}{4}$ **then**
    Set $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  else if $\rho_k > \frac{3}{4}$ and $\|\eta_k\| = \Delta_k$ **then**
    Set $\Delta_{k+1} = \min(2\Delta_k, \bar{\Delta})$
  else
    Set $\Delta_{k+1} = \Delta_k$
  end if
  {Compute next iterate}
  if $\rho_k > \rho'$ **then**
    Set $x_{k+1} = R_{x_k}(\eta_k)$
  else
    Preserve $x_{k+1} = x_k$
  end if
end for

Output: Sequences of iterates $\{x_k\}$.

We conclude this section by pointing out more explicitly the link between Algorithm 4 and the Riemannian Newton method. Assume that H_{x_k} in (2.6) is the exact Hessian of f at x_k, and assume that the exact solution η^* of the trust-region subproblem (2.6) lies in the interior of the trust region. Then η^* satisfies

$$\operatorname{grad} f + \nabla_{\eta^*} \operatorname{grad} f = 0,$$

which is the Riemannian Newton equation of Smith [Smi93, Smi94] and Udrişte [Udr94, Ch. 7, §5]. Note that both authors propose to apply the update vector η^* using the Riemannian exponential retraction; namely, the new iterate is defined as $x_+ = \operatorname{Exp}_x \eta^*$. As shown by Smith [Smi93, Smi94], the Riemannian Newton algorithm converges locally quadratically to the nondegenerate stationary points of f. A cubic rate of convergence is even observed in frequently encountered cases where some symmetry condition holds [Smi93, Smi94, EAS98, AMS04]. We will see in Section 2.3 that the superlinear convergence

property of Newton's method is preserved by the trust-region modification, while the global convergence properties are improved: the accumulation points are guaranteed to be stationary points regardless of the initial conditions, and among the stationary points only the local minima can be local attractors.

2.2 Solving the model minimization

We have seen in Section 2.1 that the use of retractions yields trust-region subproblems expressed in Euclidean spaces $T_x\mathcal{M}$. Therefore, all the classical methods for solving the trust-region subproblem can be applied, including:

- exact solution [MS83]
- truncated Lanczos [GLRT99]
- truncated conjugate gradient [Toi81, Ste83]

As mentioned in the introduction, it is assumed here that for some reason, usually related to the large size of the problem under consideration or to the computational efficiency required to outperform alternative methods, it is impractical to check positive-definiteness of H_{x_k}; rather, H_{x_k} is only available via its application to a vector.

The *truncated conjugate-gradient method* of Steihaug [Ste83] and Toint [Toi81] is particularly appropriate in these circumstances. The following algorithm is a straightforward adaptation of the method of [Ste83] to the trust-region subproblem (2.6). This algorithm is an *inner iteration* as it is an iteration used within the RTR framework (Algorithm 4) to compute an approximate solution of the trust-region subproblems. Note that we use indices in superscript to denote the evolution of η within the inner iteration, while subscripts are used in the outer iteration.

The simplest stopping criterion for Algorithm 5 is to truncate after a fixed number of iterations. In order to improve the convergence rate, a possibility is to stop as soon as an iteration j is reached for which

$$\|r_j\| \leq \|r_0\| \min(\|r_0\|^\theta, \kappa). \tag{2.8}$$

Concerning the computation of τ, it can be shown that when $\langle \delta_j, H_{x_k}\delta_j \rangle \leq 0$, $\arg\min_{\tau\in\mathbb{R}} m_{x_k}(\eta^j + \tau\delta_j)$ is equal to the positive root of $\|\eta^j + \tau\delta_k\|_{g_x} = \Delta$, which is explicitly

Algorithm 5 Preconditioned Truncated CG for RTR

Require: Iterate $x \in \mathcal{M}$, $\operatorname{grad} f(x) \neq 0$; trust-region radius Δ; convergence criteria $\kappa \in (0,1)$, $\theta > 0$; model m_x as in (2.5); symmetric/positive definite preconditioner $N^{-1} : T_x\mathcal{M} \to T_x\mathcal{M}$

Set $\eta^0 = 0_x$, $r_0 = \operatorname{grad} f(x)$, $z_0 = N^{-1} r_0$, $d_0 = -z_0$
for $j = 0, 1, 2, \ldots$ **do**
  if $\|r_j\| \leq \|r_0\| \min\left\{\kappa, \|r_0\|^\theta\right\}$ **then**
    return η^j
  end if
  if $g_x\left(H_x[d_j], d_j\right) \leq 0$ **then**
    Compute $\tau > 0$ such that $\eta = \eta^j + \tau d_j$ satisfies $\|\eta\|_N = \Delta$
    return η
  end if
  Set $\alpha_j = g_x\left(z_j, r_j\right) / g_x\left(H_x[d_j], d_j\right)$
  Set $\eta^{j+1} = \eta^j + \alpha_j d_j$
  if $\|\eta^{j+1}\|_N > \Delta$ **then**
    Compute $\tau > 0$ such that $\eta = \eta^j + \tau d_j$ satisfies $\|\eta\|_N = \Delta$
    return η
  end if
  Set $r_{j+1} = r_j + \alpha_j H_x[d_j]$
  Set $z_{j+1} = N^{-1} r_{j+1}$
  Set $\beta_{j+1} = g_x\left(z_{j+1}, r_{j+1}\right) / g_x\left(z_j, r_j\right)$
  Set $d_{j+1} = -z_{j+1} + \beta_{j+1} d_j$
end for

given by

$$\frac{-\langle \eta^j, \delta_j \rangle + \sqrt{\langle \eta^j, \delta_j \rangle^2 - (\Delta^2 - \langle \eta^j, \eta^j \rangle)\langle \delta_j, \delta_j \rangle}}{\langle \delta_j, \delta_j \rangle}.$$

Note that Algorithm 5 presents a preconditioned version of the Steihaug-Toint conjugate-gradient method for use in solving the Riemannian trust-region subproblem. Note that this presentation exploits the preconditioner for measuring the trust-region radius, via the norm

$$\|\eta\|_N = \sqrt{\langle \eta, N\eta \rangle_x}\ .$$

The benefit of this modification is that it can be proven that after the iteration leaves the trust-region, it will not re-enter the trust-region [CGT00, Section 7.5.1]. However, computing the N-norm requires applying the operator N; it is often the case that we have access only to the operator N^{-1}. Fortunately, it is possible to compute each $\|\eta^j + \alpha_j d_j\|_N$ from information

at hand. This is done via the following recurrences, from [CGT00, pg. 206]:

$$\begin{aligned}
\|\eta^j + \alpha d_j\|_N^2 &= \|\eta^j\|_N^2 + 2\alpha \langle \eta^j, N d_j \rangle + \alpha^2 \|d_j\|_N^2 \\
\langle \eta^j, N d_j \rangle &= \beta_{j-1} \left(\langle \eta^{j-1}, N d_{j-1} \rangle + \alpha_{j-1} \|d_{j-1}\|_N^2 \right) \\
\|d_j\|_N^2 &= \langle r_j, z_j \rangle + \beta_{j-1}^2 \|d_{j-1}\|_N^2 \ ,
\end{aligned}$$

and initial values

$$\begin{aligned}
\|\eta^0\|_N &= 0 \\
\langle \eta^0, N d_0 \rangle &= 0 \\
\|d_0\|_N^2 &= \langle r_0, z_0 \rangle \ .
\end{aligned}$$

Notice that the tCG algorithm only requires the following:

- An evaluation of grad $f(x)$.
- A routine that performs line minimizations for the model m.
- A routine that returns $H_{x_k}\delta$ given $\delta \in T_x\mathcal{M}$.
- (Optional) A routine a returns $N^{-1}\eta$ for the symmetric positive definite preconditioner N

Thus the tCG algorithm is "inverse-free", as it uses H_{x_k} in the computation of $H_{x_k}\delta_j$ only. The reader interested in the underlying principles of the Steihaug-Toint truncated CG method should refer to one of [Ste83, NW99, CGT00]. Alternatives to tCG as an inner iteration for RTR (Algorithm 4) include the dogleg method of Powell [Pow70b], the double-dogleg method of Dennis and Mei [DM79], the method of Moré and Sorensen [MS83], the two-dimensional subspace minimization strategy of Byrd *et al.* [BSS88], the truncated Lanczos approach of Gould *et al.* [GLRT99], and the sequential subspace method of Hager [Hag01].

2.3 Convergence analysis for RTR

In this section, we first study the global convergence properties of the RTR scheme (Algorithm 4), without any assumption on the way the trust-region subproblems (2.6) are solved, except that the approximate solution η_k must produce a decrease of the model that is at least a fixed fraction of the so-called Cauchy decrease. Under mild additional assumptions on the retraction and the cost function, it is shown that the sequences $\{x_k\}$ produced by

Algorithm 4 converge to the set of stationary points of the cost function. This result is well known in the $\mathbb{R}^n$ case; in the case of manifolds, the convergence analysis has to address the fact that a different lifted cost function $\hat{f}_{x_k}$ is considered at each iterate x_k.

We then analyze the local convergence of Algorithm 4-5 around nondegenerate local minima. Algorithm 4-5 refers to the RTR framework where the trust-region subproblems are approximately solved using the tCG algorithm with stopping criterion (2.8). It is shown that the iterates of the algorithm converge to nondegenerate stationary points with an order of convergence $\min\{\theta + 1, 2\}$ (at least).

2.3.1 Global convergence

The objective of this section is to show that, under appropriate assumptions, the sequence $\{x_k\}$ generated by Algorithm 4 satisfies $\lim_{k\to\infty} \|\text{grad}\, f(x_k)\| = 0$; this generalizes a classical convergence property of trust-region methods in $\mathbb{R}^n$, see [NW99, Theorem 4.8]. In what follows, $(\mathcal{M}, g)$ is a complete Riemannian manifold of dimension d, and R is a retraction on $\mathcal{M}$ (Definition 3). We define

$$\hat{f} : T\mathcal{M} \mapsto \mathbb{R} : \xi \mapsto f(R(\xi)) \tag{2.9}$$

and, in accordance with (2.4), $\hat{f}_x$ denotes the restriction of $\hat{f}$ to $T_x\mathcal{M}$. We denote by $B_\delta(0_x) = \{\xi \in T_x\mathcal{M} : \|\xi\| < \delta\}$ the open ball in $T_x\mathcal{M}$ of radius δ centered at 0_x, and $B_\delta(x)$ stands for the set $\{y \in M : \text{dist}(x, y) < \delta\}$ where dist denotes the Riemannian distance (1.10). We denote by $P_\gamma^{t\leftarrow t_0} v$ the vector of $T_{\gamma(t)}\mathcal{M}$ obtained by parallel transporting the vector $v \in T_{\gamma(t_0)}\mathcal{M}$ along a curve γ.

As in the classical $\mathbb{R}^n$ case, we first show that at least one accumulation point of $\{x_k\}$ is stationary. The convergence result requires that $m_{x_k}(\eta_k)$ be a sufficiently good approximation of $\hat{f}_{x_k}(\eta_k)$. In [CGT00, Thm 6.4.5] this is guaranteed by the assumption that the Hessian of the cost function is bounded. It is however possible to weaken this assumption, which leads us to consider the following definition.

Definition 6 (radially L-C^1 function)**.** *Let $\hat{f} : T\mathcal{M} \to \mathbb{R}$ be as in (2.9). We say that $\hat{f}$ is* radially Lipschitz continuously differentiable *if there exist reals $\beta_{RL} > 0$ and $\delta_{RL} > 0$ such that, for all $x \in \mathcal{M}$, for all $\xi \in T_x\mathcal{M}$ with $\|\xi\| = 1$, and for all $t < \delta_{RL}$, it holds*

$$\left| \frac{d}{d\tau} \hat{f}_x(\tau\xi)|_{\tau=t} - \frac{d}{d\tau} \hat{f}_x(\tau\xi)|_{\tau=0} \right| \leq \beta_{RL} t. \tag{2.10}$$

For the purpose of Algorithm 4, which is a descent algorithm, this condition need only be imposed for all x in the level set

$$\{x \in \mathcal{M} : f(x) \leq f(x_0)\}. \tag{2.11}$$

A key assumption in the classical global convergence result in $\mathbb{R}^n$ is that the approximate solution η_k of the trust-region subproblem (2.6) produces at least as much decrease in the model function as a fixed fraction of the Cauchy decrease; see [NW99, Section 4.3]. Since the trust-region subproblem (2.6) is expressed on a Euclidean space, the definition of the Cauchy point is adapted from $\mathbb{R}^n$ without difficulty, and the bound

$$m_{x_k}(0) - m_{x_k}(\eta_k) \geq c_1 \|\mathrm{grad} f(x_k)\| \min\left(\Delta_k, \frac{\|\mathrm{grad} f(x_k)\|}{\|H_{x_k}\|}\right), \tag{2.12}$$

for some constant $c_1 > 0$, is readily obtained from the $\mathbb{R}^n$ case, where $\|H_{x_k}\|$ is defined as

$$\|H_{x_k}\| := \sup\{\|H_{x_k}\zeta\| : \zeta \in T_{x_k}\mathcal{M},\ \|\zeta\| = 1\}. \tag{2.13}$$

In particular, the truncated CG method (Algorithm 5) satisfies this bound (with $c_1 = \frac{1}{2}$, see [NW99, Lemma 4.5]) since it first computes the Cauchy point and then attempts to improve the model decrease.

With these preliminaries in place, we can state and prove the first global convergence result. Note that this theorem is presented under weak assumptions; stronger but arguably easier to check assumptions are given in Proposition 10.

Theorem 7. *Let $\{x_k\}$ be a sequence of iterates generated by Algorithm 4 with $\rho' \in [0, \frac{1}{4})$. Suppose that f is C^1 and bounded below on the level set (2.11), that $\hat{f}$ is radially L-C^1 (Definition 6), and that $\|H_{x_k}\| \leq \beta$ for some constant β. Further suppose that all approximate solutions η_k of (2.6) satisfy the Cauchy decrease inequality (2.12) for some positive constant c_1. We then have*

$$\liminf_{k\to\infty} \|\mathrm{grad}\, f(x_k)\| = 0.$$

Proof. First, we perform some manipulation of ρ_k from (2.7). Notice that

$$\begin{aligned} |\rho_k - 1| &= \left|\frac{(f(x_k) - \hat{f}_{x_k}(\eta_k)) - (m_{x_k}(0) - m_{x_k}(\eta_k))}{m_{x_k}(0) - m_{x_k}(\eta_k)}\right| \\ &= \left|\frac{m_{x_k}(\eta_k) - \hat{f}_{x_k}(\eta_k)}{m_{x_k}(0) - m_{x_k}(\eta_k)}\right|. \end{aligned} \tag{2.14}$$

Direct manipulations on the function $t \mapsto \hat{f}_{x_k}(t\frac{\eta_k}{\|\eta_k\|})$ yield

$$\begin{aligned}\hat{f}_{x_k}(\eta_k) &= \hat{f}_{x_k}(0_{x_k}) + \|\eta_k\| \tfrac{d}{d\tau}\hat{f}_{x_k}(\tau\tfrac{\eta_k}{\|\eta_k\|})|_{\tau=0} \\ &\quad + \int_0^{\|\eta_k\|} \left(\tfrac{d}{d\tau}\hat{f}_{x_k}(\tau\tfrac{\eta_k}{\|\eta_k\|})|_{\tau=t} - \tfrac{d}{d\tau}\hat{f}_{x_k}(\tau\tfrac{\eta_k}{\|\eta_k\|})|_{\tau=0}\right) dt \\ &= f(x_k) + g_{x_k}\left(\operatorname{grad} f(x_k), \eta_k\right) + \epsilon'\end{aligned}$$

where $|\epsilon'| < \int_0^{\|\eta_k\|} \beta_{RL} t \; dt = \frac{1}{2}\beta_{RL}\|\eta_k\|^2$ whenever $\|\eta_k\| < \delta_{RL}$, and β_{RL} and δ_{RL} are the constants in the radially L-C^1 property (2.10). Therefore, it follows from the definition (2.6) of m_{x_k} that

$$\begin{aligned}|m_{x_k}(\eta_k) - \hat{f}_{x_k}(\eta_k)| &= \left|\frac{1}{2} g_{x_k}\left(H_{x_k}\eta_k, \eta_k\right) - \epsilon'\right| \\ &\leq \frac{1}{2}\beta\|\eta_k\|^2 + \frac{1}{2}\beta_{RL}\|\eta_k\|^2 \leq \beta'\|\eta_k\|^2 \end{aligned} \tag{2.15}$$

whenever $\|\eta_k\| < \delta_{RL}$, where $\beta' = \max(\beta, \beta_{RL})$.

Assume for purpose of contradiction that the theorem does not hold; that is, assume there exist $\epsilon > 0$ and a positive index K such that

$$\|\operatorname{grad} f(x_k)\| \geq \epsilon, \quad \text{for all } k \geq K. \tag{2.16}$$

From (2.12), for $k \geq K$, we have

$$m_{x_k}(0) - m_{x_k}(\eta_k) \geq c_1 \|\operatorname{grad} f(x_k)\| \min\left(\Delta_k, \frac{\|\operatorname{grad} f(x_k)\|}{\|H_{x_k}\|}\right) \geq c_1 \epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta'}\right). \tag{2.17}$$

Substituting (2.15), and (2.17) into (2.14), we have that

$$|\rho_k - 1| \leq \frac{\beta'\|\eta_k\|^2}{c_1\epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta'}\right)} \leq \frac{\beta'\Delta_k^2}{c_1\epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta'}\right)} \tag{2.18}$$

whenever $\|\eta_k\| < \delta_{RL}$. We can choose a value of $\hat{\Delta}$ that allows us to bound the right-hand-side of the inequality (2.18), when $\Delta_k \leq \hat{\Delta}$. Choose $\hat{\Delta}$ as follows:

$$\hat{\Delta} \leq \min\left(\frac{c_1\epsilon}{2\beta'}, \frac{\epsilon}{\beta'}, \delta_{RL}\right).$$

This gives us $\min\left(\Delta_k, \frac{\epsilon}{\beta'}\right) = \Delta_k$. We can now write (2.18) as follows:

$$|\rho_k - 1| \leq \frac{\beta'\hat{\Delta}\Delta_k}{c_1\epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta'}\right)} \leq \frac{\Delta_k}{2\min\left(\Delta_k, \frac{\epsilon}{\beta'}\right)} = \frac{1}{2}.$$

Therefore, $\rho_k \geq \frac{1}{2} > \frac{1}{4}$ whenever $\Delta_k \leq \hat{\Delta}$, so that by the workings of Algorithm 4, it follows (from the argument above) that $\Delta_{k+1} \geq \Delta_k$ whenever $\Delta_k \leq \hat{\Delta}$. It follows that a reduction of Δ_k (by a factor of $\frac{1}{4}$) can occur in Algorithm 4 only when $\Delta_k > \hat{\Delta}$. Therefore, we conclude that

$$\Delta_k \geq \min\left(\Delta_K, \hat{\Delta}/4\right), \quad \text{for all } k \geq K. \tag{2.19}$$

Suppose now that there is an infinite subsequence $\mathcal{K}$ such that $\rho_k \geq \frac{1}{4} > \rho'$ for $k \in \mathcal{K}$. If $k \in \mathcal{K}$ and $k \geq K$, we have from (2.17) that

$$\begin{aligned} f(x_k) - f(x_{k+1}) &= f_{x_k} - \hat{f}_{x_k}(\eta_k) \\ &\geq \frac{1}{4}(m_{x_k}(0) - m_{x_k}(\eta_k)) \\ &\geq \frac{1}{4} c_1 \epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta'}\right). \end{aligned}$$

Since f is bounded below on the level set containing these iterates, it follows from this inequality that

$$\lim_{k \in \mathcal{K}, k \to \infty} \Delta_k = 0,$$

clearly contradicting (2.19). Then such an infinite subsequence as $\mathcal{K}$ cannot exist. If follows that we must have $\rho_k < \frac{1}{4}$ for all k sufficiently large, so that Δ_k will be reduced by a factor of $\frac{1}{4}$ on every iteration. Then we have, $\lim_{k \to \infty} \Delta_k = 0$, which again contradicts (2.19). Then our original assumption (2.16) must be false, giving us the desired result. □

To further show that all accumulation points of $\{x_k\}$ are stationary points, we need to make an additional regularity assumption on the cost function f. The global convergence result in $\mathbb{R}^n$, as stated in [NW99, Theorem 4.8], requires that f be Lipschitz continuously differentiable. That is to say, for any $x, y \in \mathbb{R}^n$,

$$\|\mathrm{grad} f(y) - \mathrm{grad} f(x)\| \leq \beta_1 \|y - x\|. \tag{2.20}$$

A key to obtaining a Riemannian counterpart of this global convergence result is to adapt the notion of Lipschitz continuous differentiability to the Riemannian manifold $(\mathcal{M}, g)$. The expression $\|x - y\|$ in the right-hand side of (2.20) naturally becomes the Riemannian distance $\mathrm{dist}(x, y)$. For the left-hand side of (2.20), observe that the operation $\mathrm{grad} f(x) - \mathrm{grad} f(y)$ is not well-defined in general on a Riemannian manifold since $\mathrm{grad}\, f(x)$ and $\mathrm{grad}\, f(y)$ belong to two different tangent spaces, namely $T_x\mathcal{M}$ and $T_y M$. However, if y belongs to a

normal neighborhood of x, then there is a unique geodesic $\alpha(t) = \mathrm{Exp}_x(t\mathrm{Exp}_x^{-1}y)$ such that $\alpha(0) = x$ and $\alpha(1) = y$, and we can parallel transport $\mathrm{grad}\, f(y)$ along α to obtain the vector $P_\alpha^{0\leftarrow 1}\mathrm{grad}\, f(y)$ in $T_x\mathcal{M}$, to yield the following definition.

Definition 8 (Lipschitz continuous differentiability)**.** *Assume that $(\mathcal{M}, g)$ has an injectivity radius $i(\mathcal{M}) > 0$. A real function f on $\mathcal{M}$ is* Lipschitz continuous differentiable *if it is differentiable and if, for all x, y in $\mathcal{M}$ such that $dist(x, y) < i(\mathcal{M})$, it holds that*

$$\|P_\alpha^{0\leftarrow 1}\mathrm{grad}\, f(y) - \mathrm{grad}\, f(x)\| \leq \beta_1\, dist(y, x), \tag{2.21}$$

where α is the unique geodesic with $\alpha(0) = x$ and $\alpha(1) = y$.

Note that (2.21) is symmetric in x and y; indeed, since the parallel transport is an isometry, it follows that

$$\|P_\alpha^{0\leftarrow 1}\mathrm{grad}\, f(y) - \mathrm{grad} f(x)\| = \|\mathrm{grad} f(y) - P_\alpha^{1\leftarrow 0}\mathrm{grad} f(x)\|.$$

Moreover, we place one additional requirement on the retraction R, that there exists some $\mu > 0$ and $\delta_\mu > 0$ such that

$$\|\xi\| \geq \mu\, d(x, R_x(\xi)), \quad \text{for all } x \in \mathcal{M}, \text{for all } \xi \in T_x\mathcal{M},\ \|\xi\| \leq \delta_\mu \tag{2.22}$$

Note that for the exponential retraction discussed in this paper, (2.22) is satisfied as an equality, with $\mu = 1$. The bound is also satisfied when R is smooth and $\mathcal{M}$ is compact (Corollary 11).

We are now ready to show that under some additional assumptions, the gradient of the cost function converges to zero on the whole sequence of iterates. Here again we refer to Proposition 10 for a simpler (but slightly stronger) set of assumptions that yield the same result.

Theorem 9. *Let $\{x_k\}$ be a sequence of iterates generated by Algorithm 4. Suppose that all the assumptions of Theorem 7 are satisfied. Further suppose that $\rho' \in (0, \frac{1}{4})$, that f is Lipschitz continuously differentiable (Definition 8), and that (2.22) is satisfied for some $\mu > 0$, $\delta_\mu > 0$. It then follows that*

$$\lim_{k\to\infty} \mathrm{grad}\, f(x_k) = 0.$$

Proof. Consider any index m such that $\operatorname{grad} f(x_m) \neq 0$. The Lipschitz property (2.21) yields

$$\|P_\alpha^{1\leftarrow 0}\operatorname{grad} f(x) - \operatorname{grad} f(x_m)\| \leq \beta_1 \operatorname{dist}(x, x_m)$$

for all x. Define scalars

$$\epsilon = \frac{1}{2}\|\operatorname{grad} f(x_m)\|, \quad r = \min\left(\frac{\|\operatorname{grad} f(x_m)\|}{2\beta_1}, i(\mathcal{M})\right) = \min\left(\frac{\epsilon}{\beta_1}, i(\mathcal{M})\right)$$

Define the ball $B_r(x_m) := \{x : \operatorname{dist}(x, x_m) < r\}$.

Then for any $x \in B_r(x_m)$, we have

$$\begin{aligned}
\|\operatorname{grad} f(x)\| &= \|P_\alpha^{0\leftarrow 1}\operatorname{grad} f(x)\| \\
&= \|P_\alpha^{0\leftarrow 1}\operatorname{grad} f(x) + \operatorname{grad} f(x_m) - \operatorname{grad} f(x_m)\| \\
&\geq \|\operatorname{grad} f(x_m)\| - \|P_\alpha^{0\leftarrow 1}\operatorname{grad} f(x) - \operatorname{grad} f(x_m)\| \\
&\geq 2\epsilon - \beta_1 \operatorname{dist}(x, x_m) \\
&> 2\epsilon - \beta_1 \min\left(\frac{\|\operatorname{grad} f(x_m)\|}{2\beta_1}, i(\mathcal{M})\right) \\
&\geq 2\epsilon - \frac{1}{2}\|\operatorname{grad} f(x_m)\| \\
&= \epsilon.
\end{aligned}$$

If the entire sequence $\{x_k\}_{k\geq m}$ stays inside of the ball $B_r(x_m)$, then we would have $\|\operatorname{grad} f(x_k)\| > \epsilon$ for all $k \geq m$, which contradicts the results of Theorem 7. Then the sequence eventually leaves the ball $B_r(x_m)$.

Let the index $l \geq m$ be such that x_{l+1} is the first iterate after x_m outside of $B_r(x_m)$. Since $\|\operatorname{grad} f(x_k)\| > \epsilon$ for $k = m, m+1, \ldots, l$, we have

$$\begin{aligned}
f(x_m) - f(x_{l+1}) &= \sum_{k=m}^{l} f(x_k) - f(x_{k+1}) \\
&\geq \sum_{k=m, x_k \neq x_{k+1}}^{l} \rho'\left(m_{x_k}(0) - m_{x_k}(\eta_k)\right) \\
&\geq \sum_{k=m, x_k \neq x_{k+1}}^{l} \rho' c_1 \|\operatorname{grad} f(x_k)\| \min\left(\Delta_k, \frac{\|\operatorname{grad} f(x_k)\|}{\|H_{x_k}\|}\right) \\
&\geq \sum_{k=m, x_k \neq x_{k+1}}^{l} \rho' c_1 \epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta}\right).
\end{aligned}$$

We distinguish two cases. If $\Delta_k > \epsilon/\beta$ in at least one of the terms of the sum, then

$$f(x_m) - f(x_{l+1}) \geq \rho' c_1 \epsilon \frac{\epsilon}{\beta}. \tag{2.23}$$

In the other case, we have

$$f(x_m) - f(x_{l+1}) \geq \rho' c_1 \epsilon \sum_{k=m, x_k \neq x_{k+1}}^{l} \Delta_k \geq \rho' c_1 \epsilon \sum_{k=m, x_k \neq x_{k+1}}^{l} \|\eta_k\|. \tag{2.24}$$

If $\|\eta_k\| > \delta_\mu$ in at least one term in the sum, then

$$f(x_m) - f(x_{l+1}) \geq \rho' c_1 \epsilon \delta_\mu. \tag{2.25}$$

Otherwise, (2.24) yields

$$\begin{aligned} f(x_m) - f(x_{l+1}) &\geq \rho' c_1 \epsilon \sum_{k=m, x_k \neq x_{k+1}}^{l} \mu d(x_k, R_{x_k}(\eta_k)) \\ &= \rho' c_1 \epsilon \mu \sum_{k=m, x_k \neq x_{k+1}}^{l} d(x_k, x_{k+1}) \\ &\geq \rho' c_1 \epsilon \mu r = \rho' c_1 \epsilon \mu \min\left(\frac{\epsilon}{\beta_1}, i(\mathcal{M})\right). \end{aligned} \tag{2.26}$$

It follows from (2.23), (2.25) and (2.26) that

$$f(x_m) - f(x_{l+1}) \geq \rho' c_1 \epsilon \min\left(\frac{\epsilon}{\beta}, \delta_\mu, \frac{\epsilon\mu}{\beta_1}, i(\mathcal{M})\mu\right). \tag{2.27}$$

Because $\{f(x_k)\}_{k=0}^{\infty}$ is decreasing and bounded below, we have

$$f(x_k) \downarrow f^*, \tag{2.28}$$

for some $f^* > -\infty$. It then follows from (2.27) that

$$\begin{aligned} f(x_m) - f^* &\geq f(x_m) - f(x_{l+1}) \\ &\geq \rho' c_1 \epsilon \min\left(\frac{\epsilon}{\beta}, \delta_\mu, \frac{\epsilon\mu}{\beta_1}, i(\mathcal{M})\mu\right) \\ &= \frac{1}{2}\rho' c_1 \|\mathrm{grad} f(x_m)\| \min\left(\frac{\|\mathrm{grad} f(x_m)\|}{2\beta}, \delta_\mu, \frac{\|\mathrm{grad} f(x_m)\|\mu}{2\beta_1}, i(\mathcal{M})\mu\right). \end{aligned}$$

Assume for the purpose of contradiction that it is not the case that

$$\lim_{m\to\infty} \|\mathrm{grad} f(x_m)\| = 0 \ .$$

Then there exists $\omega > 0$ and an infinite sequence $\mathcal{K}$ such that

$$\|\mathrm{grad} f(x_k)\| > \omega, \qquad \text{for all } k \in \mathcal{K}.$$

Then for $k \in \mathcal{K}, k \geq m$, we have

$$\begin{aligned} f(x_k) - f^* &\geq \frac{1}{2}\rho' c_1 \|\mathrm{grad} f(x_k)\| \min\left(\frac{\|\mathrm{grad} f(x_k)\|}{2\beta}, \frac{\|\mathrm{grad} f(x_k)\|\mu}{2\beta_1}, i(\mathcal{M})\mu\right) \\ &> \frac{1}{2}\rho' c_1 \omega \min\left(\frac{\omega}{2\beta}, \frac{\omega\mu}{2\beta_1}, i(\mathcal{M})\mu\right) \end{aligned}$$

which is a positive constant. This contradicts $\lim_{k\to\infty}(f(x_k) - f^*) = 0$, so that our hypothetical assumption must be false, and

$$\lim_{m\to\infty} \|\mathrm{grad} f(x_m)\| = 0.$$

□

Note that this theorem reduces gracefully to the classical $\mathbb{R}^n$ case, taking $\mathcal{M} = \mathbb{R}^n$ endowed with the classical inner product and $R_x(\xi) := x + \xi$. Then $i(\mathcal{M}) = +\infty > 0$, R satisfies (2.22), and the Lipschitz condition (2.21) reduces to the classical expression, which subsumes the radially L-C^1 condition.

The following proposition shows that the regularity conditions on f and $\hat{f}$ required in the previous theorems are satisfied under stronger but possibly easier to check conditions. These conditions impose a bound on the Hessian of f and on the "acceleration" along curves $t \mapsto R(t\xi)$. Note also that all these conditions need only be checked on the level set $\{x \in M : f(x) \leq f(x_0)\}$.

Proposition 10. *Suppose that* $\|\mathrm{grad}\, f(x)\| \leq \beta_g$ *and* $\|\mathrm{Hess}\, f(x)\| \leq \beta$ *for some constants* β_g, β, *and all* $x \in \mathcal{M}$. *Moreover suppose that*

$$\|\tfrac{D}{dt}\tfrac{d}{dt} R(t\xi)\| \leq \beta_D \tag{2.29}$$

for some constant β_D, *for all* $\xi \in T\mathcal{M}$ *with* $\|\xi\| = 1$ *and all* $t < \delta_D$, *where* $\frac{D}{dt}$ *denotes the covariant derivative along the curve* $t \mapsto R(t\xi)$ *(see [dC92, Ch. 2, Prop. 2.2]).*
Then the Lipschitz-C^1 *condition on* f *(Definition 8) is satisfied with* $\beta_L = \beta$; *the radially Lipschitz-*C^1 *condition on* $\hat{f}$ *(Definition 6) is satisfied for* $\delta_{RL} < \delta_D$ *and* $\beta_{RL} = \beta(1+\beta_D\delta_D)+\beta_g\beta_D$; *and the condition* (2.22) *on* R *is satisfied for values of* μ *and* δ_μ *satisfying* $\delta_\mu < \delta_D$ *and* $\frac{1}{2}\beta_D\delta_\mu < \frac{1}{\mu} - 1$.

Proof. By a standard Taylor argument (see Lemma 12), boundedness of the Hessian of f implies the Lipschitz-C^1 property of f.

For (2.22), define $u(t) = R(t\xi)$ and observe that

$$\operatorname{dist}(x, R(t\xi)) \leq \int_0^t \|u'(\tau)\| \, d\tau$$

where $\int_0^t \|u'(\tau)\| \, d\tau$ is the length of the curve u between 0 and t. Using the Cauchy-Schwarz inequality and the invariance of the metric by the connection, we have

$$\left| \tfrac{d}{d\tau} \|u'(\tau)\| \right| = \left| \tfrac{d}{d\tau} \sqrt{g_{u(\tau)}\left(u'(\tau), u'(\tau)\right)} \right| = \left| \frac{g_{u(\tau)}\left(\frac{D}{dt}u'(\tau), u'(\tau)\right)}{\|u'(\tau)\|} \right| \leq \frac{\beta_D \|u'(\tau)\|}{\|u'(\tau)\|} \leq \beta_D$$

for all $t < \delta_D$. Therefore

$$\int_0^t \|u'(\tau)\| \, d\tau \leq \int_0^t \|u'(0)\| + \beta_D \tau \; d\tau = \|\xi\| t + \tfrac{1}{2}\beta_D t^2 = t + \tfrac{1}{2}\beta_D t^2,$$

which is smaller than $\frac{t}{\mu}$ if $\frac{1}{2}\beta_D t < \frac{1}{\mu} - 1$.

For the radially Lipschitz-C^1 condition, let

$$u(t) = R(t\xi) \quad \text{and} \quad h(t) = f(u(t)) = \hat{f}(t\xi)$$

with $\xi \in T_x\mathcal{M}$, $\|\xi\| = 1$. Then

$$h'(t) = g_{u(t)}\left(\operatorname{grad} f(u(t)), u'(t)\right)$$

and

$$\begin{aligned} h''(t) &= \tfrac{D}{dt} g_{u(t)}\left(\operatorname{grad} f(u(t)), u'(t)\right) \\ &= g_{u(t)}\left(\tfrac{D}{dt}\operatorname{grad} f(u(t)), u'(t)\right) + g_{u(t)}\left(\operatorname{grad} f(u(t)), \frac{D}{dt}u'(t)\right). \end{aligned}$$

Now, $\frac{D}{dt}\operatorname{grad} f(u(t)) = \nabla_{u'(t)}\operatorname{grad} f(u(t)) = \operatorname{Hess} f(u(t))[u'(t)]$. It follows that $|h''(t)|$ is bounded on $t \in [0, \delta_D)$ by the constant $\beta_{RL} = \beta(1 + \beta_D\delta_D) + \beta_g\beta_D$. Then

$$|h'(t) - h'(0)| \leq \int_0^t |h''(\tau)| \, d\tau \leq t\beta_{RL}.$$

□

In many practical cases, the cost function and the retraction are smooth and the Riemannian manifold is compact. This is a comfortable situation, as the next result shows.

Corollary 11 (smoothness and compactness)**.** *If the cost function f and the retraction R are smooth and the Riemannian manifold $\mathcal{M}$ is compact, then all the conditions in Proposition 10 are satisfied.*

Proof. This result comes from the fact that every continuous real function on a compact space is bounded. The only nontrivial part is to show that the set $\{\xi \in T\mathcal{M} : \|\xi\| = 1\}$ is compact. Compactness is a topological notion; recall that the topology of a manifold is the topology induced by the topological basis formed by the collection of coordinate domains. First we need to prove that every compact Riemannian manifold has a finite covering by compact subsets of coordinate domains. This can be done via a result on paracompactness found in [Mun00, Lem. 41.3]. Here we give a direct argument. Let $\mathcal{M}$ be a compact Riemannian manifold. Since $\mathcal{M}$ is Riemannian, it is metrizable [O'N83, §5.18], thus it is a regular topological space. Let $\{A_i\}_{i=1,\dots,n}$ be an (open) covering of $\mathcal{M}$ by coordinate domains. Let $C_1 = \mathcal{M} - \cup_{i=2}^n A_i$. It is a closed—thus compact—set contained in A_1. Since $\mathcal{M}$ is a regular space, it follows that for all x in C_1, there is an open set V_x containing x such that $\overline{V}_x \subset A_1$. The collection $\{V_x\}_{x\in\mathcal{M}}$ is an open covering of the compact set C_1, thus it admits a finite subcovering $\{V_j\}_{j=1,\dots,m}$. Let $B_1 = \cup_{j=1}^m V_j$. Then B_1 is open, and $\overline{B_1} \subset A_1$, and $\{B_1, A_2, \dots, A_n\}$ is still an open covering of $\mathcal{M}$. The same operation can now be performed on A_2, to obtain an open covering $\{B_1, B_2, \dots, A_n\}$. Finally, we have an open covering $\{B_i\}_{i=1,\dots,n}$ such that $\overline{B_i} \subset A_i$, $i = 1, \dots, n$. The collection $\{\overline{B_i}\}_{i=1,\dots,n}$ is a covering of $\mathcal{M}$ by compacts such that $\overline{B_i} \subset A_i$, $i = 1, \dots, n$, as required.

Now let $S := \{\xi \in T\mathcal{M} : \|\xi\| = 1\}$. We want to show that S is compact. Using the construction above, let $S_i := \{\xi \in T\mathcal{M} : \|\xi\| = 1, \pi\xi \in \overline{B_i}\}$, where $\pi\xi$ denotes the foot of ξ, and notice that $S = \cup_{i=1}^n S_i$. Since each S_i is included in a coordinate domain, it is sufficient to show that the coordinate expression of each S_i is compact. Abusing notation, S_i has the coordinate expression $\{(x, \xi) : x \in \overline{B_i}, \xi^i g_{ij}(x)\xi^i = 1\}$. It is closed as the inverse image of a closed set by a continuous function, and it is bounded by continuity and non-degeneracy of the metric, hence it is compact, which completes the proof. □

2.3.2 Local convergence

We now state local convergence properties of Algorithm 4-5 (i.e., Algorithm 4 where the trust-region subproblem (2.6) is solved approximately with Algorithm 5). We first state a

few preparation lemmas.

As before, $(\mathcal{M}, g)$ is a complete Riemannian manifold of dimension d, and R is a retraction on $\mathcal{M}$ (Definition 3). The first lemma is a first-order Taylor formula for tangent vector fields. (Similar Taylor developments on manifolds can be found in [Smi94].)

Lemma 12 (Taylor)**.** *Let $x \in \mathcal{M}$, let V be a normal neighborhood of x, and let ζ be a C^1 tangent vector field on $\mathcal{M}$. Then, for all $y \in V$,*

$$P_\gamma^{0\leftarrow 1}\zeta_y = \zeta_x + \nabla_\xi \zeta + \int_0^1 \left(P_\gamma^{0\leftarrow\tau}\nabla_{\gamma'(\tau)}\zeta - \nabla_\xi\zeta\right) d\tau, \tag{2.30}$$

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

Proof. Start from

$$P_\gamma^{0\leftarrow 1}\zeta_y = \zeta_x + \int_0^1 \frac{d}{d\tau} P_\gamma^{0\leftarrow\tau}\zeta \; d\tau = \zeta_x + \nabla_\xi\zeta + \int_0^1 \left(\frac{d}{d\tau}P_\gamma^{0\leftarrow\tau}\zeta - \nabla_\xi\zeta\right) d\tau$$

and use the formula for the connection in terms of the parallel transport, see [dC92, Ch. 2, Ex. 2], to obtain

$$\frac{d}{d\tau}P_\gamma^{0\leftarrow\tau}\zeta = \frac{d}{d\epsilon}P_\gamma^{0\leftarrow\tau}P_\gamma^{\tau\leftarrow\tau+\epsilon}\zeta\Big|_{\epsilon=0} = P_\gamma^{0\leftarrow\tau}\nabla_{\gamma'}\zeta.$$

□

We use this lemma to show that in some neighborhood of a nondegenerate local minimizer v of f, the norm of the gradient of f can be taken as a measure of the Riemannian distance to v.

Lemma 13. *Let $v \in \mathcal{M}$ and let f be a C^2 cost function such that $\operatorname{grad} f(v) = 0$ and $\operatorname{Hess} f(v)$ is positive definite with maximal and minimal eigenvalues $\lambda_{\max}$ and $\lambda_{\min}$. Then, given $c_0 < \lambda_{\min}$ and $c_1 > \lambda_{\max}$, there exists a neighborhood V of v such that, for all $x \in V$, it holds that*

$$c_0\, dist(v,x) \leq \|\operatorname{grad} f(x)\| \leq c_1\, dist(v,x). \tag{2.31}$$

Proof. From Taylor (Lemma 12), it follows that

$$P_\gamma^{0\leftarrow 1}\operatorname{grad} f(v) = \operatorname{Hess} f(v)[\gamma'(0)] + \int_0^1 \left(P_\gamma^{0\leftarrow\tau}\operatorname{Hess} f(\gamma(\tau))[\gamma'(\tau)] - \operatorname{Hess} f(v)[\gamma'(0)]\right) d\tau. \tag{2.32}$$

Since f is C^2 and since $\|\gamma'(\tau)\| = \mathrm{dist}(v,x)$ for all $\tau \in [0,1]$, we have the following bound for the integral in (2.32):

$$\begin{aligned}\| \int_0^1 P_\gamma^{0\leftarrow\tau} \mathrm{Hess}\, f(\gamma(\tau))[\gamma'(\tau)] - \mathrm{Hess}\, f(v)[\gamma'(0)]\, d\tau \| \\ = \| \int_0^1 \left(P_\gamma^{0\leftarrow\tau} \circ \mathrm{Hess}\, f(\gamma(\tau)) \circ P_\gamma^{\tau\leftarrow 0} - \mathrm{Hess}\, f(v) \right) [\gamma'(0)]\, d\tau \| \leq \epsilon(\mathrm{dist}(v,x))\mathrm{dist}(v,x)\end{aligned}$$

where $\lim_{t\to 0} \epsilon(t) = 0$. Since $\mathrm{Hess}\, f(v)$ is nonsingular, it follows that $|\lambda_{min}| > 0$. Take V sufficiently small so that $\lambda_{\min} - \epsilon(\mathrm{dist}(v,x)) > c_0$ and $\lambda_{\max} + \epsilon(\mathrm{dist}(v,x)) < c_1$ for all x in V. Then, using the fact that the parallel translation is an isometry, (2.31) follows from (2.32). □

We need a relation between the gradient of f at $R_x(\xi)$ and the gradient of $\hat{f}_x$ at ξ.

Lemma 14. *Let R be a retraction on $\mathcal{M}$ and let f be a C^1 cost function on $\mathcal{M}$. Then, given $v \in \mathcal{M}$ and $c_5 > 1$, there exists a neighborhood V of v and $\delta > 0$ such that*

$$\|\mathrm{grad}\, f(R(\xi))\| \leq c_5 \|\mathrm{grad}\, \hat{f}(\xi)\|$$

for all $x \in V$ and all $\xi \in T_x\mathcal{M}$ with $\|\xi\| \leq \delta$, where $\hat{f}$ is as in (2.9).

Proof. Consider a parameterization of $\mathcal{M}$ at v, and consider the corresponding parameterization of $T\mathcal{M}$ (see [dC92, Ch. 0, Example 4.1]). Using Einstein's convention (see, e.g., [Sak96]), and denoting $\partial_i f$ by $f_{,i}$, we have

$$\hat{f}_{x,i}(\xi) = f_{,j}(R(\xi))A_i^j(\xi),$$

where $A(\xi)$ stands for the differential of R_x at $\xi \in T_x\mathcal{M}$. Then,

$$\|\mathrm{grad}\, \hat{f}_x(\xi)\|^2 = \hat{f}_{x,i}(\xi) g^{ij}(x) \hat{f}_{x,j}(\xi) = f_{,k}(R_x(\xi))A_i^k(\xi)g^{ij}(x)A_j^\ell(\xi)f_{,\ell}(R_x(\xi))$$

and

$$\|\mathrm{grad}\, f(R_x(\xi))\|^2 = f_{,j}(R_x(\xi))g^{ij}(R_x(\xi))f_{,j}(R_x(\xi)).$$

The conclusion follows by a real analysis argument, invoking the smoothness properties of R and g, compactness of the set $\{(x,\xi) : x \in V,\ \xi \in T_x\mathcal{M},\ \|\xi\| \leq \delta\}$, and using $A(0_x) = \mathrm{id}$. □

We now state and prove the local convergence results. The first result states that the nondegenerate local minima are attractors of Algorithm 4-5. The principle of the argument is closely related to the Capture Theorem, see [Ber95, Theorem 1.2.5].

Theorem 15 (local convergence to local minima)**.** *Consider Algorithm 4-5—i.e., the Riemannian trust-region algorithm where the trust-region subproblems (2.6) are solved using the truncated CG algorithm with stopping criterion (2.8)—with all the assumptions of Theorem 7. Let v be a nondegenerate local minimizer of f, i.e.,* $\operatorname{grad} f(v) = 0$ *and* $\operatorname{Hess} f(v)$ *is positive definite. Assume that $x \mapsto \|H_x^{-1}\|$ is bounded on a neighborhood of v and that (2.22) holds for some $\mu > 0$ and $\delta_\mu > 0$. Then there exists a neighborhood V of v such that, for all $x_0 \in V$, the sequence $\{x_k\}$ generated by Algorithm 4-5 converges to v.*

Proof. Take $\delta_1 > 0$ with $\delta_1 < \delta_\mu$ such that $B_{\delta_1}(v)$ is a neighborhood of v, which contains only v as stationary point, and such that $f(x) > f(v)$ for all $x \in \bar{B}_{\delta_1}(v)$.

Take δ_2 small enough that for all $x \in B_{\delta_2}(v)$, it holds that $\|\eta^*(x)\| \leq \mu(\delta_1 - \delta_2)$, where η^* is the (unique) solution of $H\eta^* = -\operatorname{grad} f(x)$; such a δ_2 exists because of Lemma 13 and the bound on $\|H_x^{-1}\|$. Consider a level set $\mathcal{L}$ of f such that $V := \mathcal{L} \cap B_{\delta_1}(v)$ is a subset of $B_{\delta_2}(v)$; invoke that $f \in C^1$ to show that such a level set exists. Then, V is a neighborhood of v and for all $x \in V$, we have

$$\operatorname{dist}(x, x_+) \leq \frac{1}{\mu}\|\eta^{tCG}(x, \Delta)\| \leq \frac{1}{\mu}\|\eta^*\| \leq (\delta_1 - \delta_2),$$

where we used the fact that $\|\eta\|$ is increasing along the truncated CG process [Ste83, Thm 2.1]. It follows from the equation above that x_+ is in $B_{\delta_1}(v)$. Moreover, since $f(x_+) \leq f(x)$, it follows that $x_+ \in V$. Thus V is invariant. But the only stationary point of f in V is v, so $\{x_k\}$ goes to v whenever x_0 is in V. □

Now we study the order of convergence of the sequences that converge to a nondegenerate local minimizer.

Theorem 16 (order of convergence)**.** *Consider Algorithm 4-5 with stopping criterion (2.8). Suppose that R is a C^2 retraction, that f is a C^2 cost function on $\mathcal{M}$, and that*

$$\|H_{x_k} - \operatorname{Hess} \hat{f}_{x_k}(0_k)\| \leq \beta_H \|\operatorname{grad} f(x_k)\|, \tag{2.33}$$

that is, H_{x_k} is a sufficiently good approximation of $\operatorname{Hess} \hat{f}_{x_k}(0_{x_k})$. Let $v \in \mathcal{M}$ be a nondegenerate local minimizer of f, (i.e., $\operatorname{grad} f(v) = 0$ *and* $\operatorname{Hess} f(v)$ *is positive definite). Further assume that* $\operatorname{Hess} \hat{f}_x$ *is Lipschitz-continuous at 0_x uniformly in x in a neighborhood*

of v, i.e., there exist $\beta_{L2} > 0$, $\delta_1 > 0$ and $\delta_2 > 0$ such that, for all $x \in B_{\delta_1}(v)$ and all $\xi \in B_{\delta_2}(0_x)$, there holds

$$\|\mathrm{Hess}\, \hat{f}_x(\xi) - \mathrm{Hess}\, \hat{f}_x(0_x)\| \leq \beta_{L2}\|\xi\|, \tag{2.34}$$

where $\|\cdot\|$ in the left-hand side denotes the operator norm in $T_x\mathcal{M}$ defined as in (2.13). Then there exists $c > 0$ such that, for all sequences $\{x_k\}$ generated by the algorithm converging to v, there exists $K > 0$ such that for all $k > K$,

$$dist(x_{k+1}, v) \leq c\,(dist(x_k, v))^{\min\{\theta+1,2\}} \tag{2.35}$$

with $\theta > 0$ as in (2.8).

Proof. We will show below that there exist $\tilde{\Delta}, c_0, c_1, c_2, c_3, c_3', c_4, c_5$ such that, for all sequences $\{x_k\}$ satisfying the conditions asserted, all $x \in \mathcal{M}$, all ξ with $\|\xi\| < \tilde{\Delta}$, and all k greater than some K, there holds

$$c_0 \mathrm{dist}(v, x_k) \leq \|\mathrm{grad}\, f(x_k)\| \leq c_1 \mathrm{dist}(v, x_k), \tag{2.36}$$

$$\|\eta_k\| \leq c_4 \|\mathrm{grad}\, m_{x_k}(0)\| \leq \tilde{\Delta}, \tag{2.37}$$

$$\rho_k > \rho', \tag{2.38}$$

$$\|\mathrm{grad}\, f(R_{x_k}(\xi))\| \leq c_5 \|\mathrm{grad}\, \hat{f}_{x_k}(\xi)\|, \tag{2.39}$$

$$\|\mathrm{grad}\, m_{x_k}(\xi) - \mathrm{grad}\, \hat{f}_{x_k}(\xi)\| \leq c_3\|\xi\|^2 + c_3'\|\mathrm{grad}\, f(x_k)\|\,\|\xi\|, \tag{2.40}$$

$$\|\mathrm{grad}\, m_{x_k}(\eta_k)\| \leq c_2 \|\mathrm{grad}\, m_{x_k}(0)\|^{\theta+1}, \tag{2.41}$$

where $\{\eta_k\}$ is the sequence of update vectors corresponding to $\{x_k\}$. With these results at hand the proof is concluded as follows. For all $k > K$, it follows from (2.36) and (2.38) that

$$c_0 \mathrm{dist}(v, x_{k+1}) \leq \|\mathrm{grad}\, f(x_{k+1})\| = \|\mathrm{grad}\, f(R_{x_k}(\eta_k))\|,$$

from (2.39) and (2.37) that

$$\|\mathrm{grad}\, f(R_{x_k}(\eta_k))\| \leq c_5 \|\mathrm{grad}\, \hat{f}_{x_k}(\eta_k)\|,$$

from (2.37) and (2.40) and (2.41) that

$$\begin{aligned}\|\mathrm{grad}\, \hat{f}_{x_k}(\eta_k)\| &\leq \|\mathrm{grad}\, m_{x_k}(\eta_k) - \mathrm{grad}\, \hat{f}_{x_k}(\eta_k)\| + \|\mathrm{grad}\, m_{x_k}(\eta_k)\| \\ &\leq (c_3 c_4^2 + c_3' c_4)\|\mathrm{grad}\, m_{x_k}(0)\|^2 + c_2\|\mathrm{grad}\, m_{x_k}(0)\|^{1+\theta},\end{aligned}$$

and from (2.36) that

$$\|\text{grad}\, m_{x_k}(0)\| = \|\text{grad}\, f(x_k)\| \leq c_1 \text{dist}(v, x_k).$$

Consequently, taking K larger if necessary so that $\text{dist}(v, x_k) < 1$ for all $k > K$, it follows that

$$\begin{aligned}
& c_0 \text{dist}(v, x_{k+1}) \\
\leq\ & \|\text{grad}\, f(x_{k+1})\| && (2.42) \\
\leq\ & c_5(c_3 c_4^2 + c_3' c_4)\|\text{grad}\, f(x_k)\|^2 + c_5 c_2 \|\text{grad}\, f(x_k)\|^{\theta+1} && (2.43) \\
\leq\ & c_5((c_3 c_4^2 + c_3' c_4) c_1^2 (\text{dist}(v, x_k))^2 + c_2 c_1^{1+\theta} (\text{dist}(v, x_k))^{1+\theta}) \\
\leq\ & c_5((c_3 c_4^2 + c_3' c_4) c_1^2 + c_2 c_1^{1+\theta}) (\text{dist}(v, x_k))^{\min\{2, 1+\theta\}}
\end{aligned}$$

for all $k > K$, which is the desired result.

It remains to prove the bounds (2.36)-(2.41).

Equation (2.36) comes from Lemma 13 and is due to the fact that v is a nondegenerate critical point.

We prove (2.37). Since $\{x_k\}$ converges to the nondegenerate local minimizer v where $\text{Hess}\, \hat{f}_v(0_v) = \text{Hess}\, f(v)$ (see Lemma 5) and since $\text{Hess}\, f(v)$ is positive definite with $f \in C^2$, it follows follows from the approximation condition (2.33) and from (2.36) that there exist $c_4 > 0$ such that $\|H_{x_k}^{-1}\| < c_4$ for all k greater than some K. Given a $k > K$, let η^* be the solution of $H_{x_k}\eta^* = -\text{grad}\, m_{x_k}(0)$. If follows that $\|\eta^*\| \leq c_4 \|\text{grad}\, m_{x_k}(0)\|$. Then, since the sequence of η_k^j's constructed by the tCG inner iteration (Algorithm 5) is strictly increasing in norm (see [Ste83, Theorem 2.1]) and would eventually reach η^* at $j = d$, it follows that (2.37) holds. The second inequality in (2.37) comes for any given $\tilde{\Delta}$ by choosing K larger if necessary.

We prove (2.38). Let γ_k denote $\|\text{grad}\, f(x_k)\|$. It follows from the definition of ρ_k that

$$\rho_k - 1 = \frac{m_{x_k}(\eta_k) - \hat{f}_{x_k}(\eta_k)}{m_{x_k}(0_{x_k}) - m_{x_k}(\eta_k)}. \tag{2.44}$$

From Taylor's theorem, there holds

$$\hat{f}_{x_k}(\eta_k) = \hat{f}_{x_k}(0_{x_k}) + g_{x_k}\left(\text{grad}\, f(x_k), \eta_k\right) + \int_0^1 g_{x_k}\left(\text{Hess}\, \hat{f}_{x_k}(\tau\eta_k)[\eta_k], \eta_k\right)(1-\tau)d\tau.$$

It follows that

$$\begin{aligned}\left|m_{x_k}(\eta_k) - \hat{f}_{x_k}(\eta_k)\right| &= \left|\int_0^1 \left(g_{x_k}\left(H_{x_k}[\eta_k], \eta_k\right) - g_{x_k}\left(\operatorname{Hess} \hat{f}_{x_k}(\tau\eta_k)[\eta_k], \eta_k\right)\right)(1-\tau)d\tau\right| \\ &\leq \int_0^1 \left|g_{x_k}\left((H_{x_k} - \operatorname{Hess} \hat{f}_{x_k}(0_{x_k}))[\eta_k], \eta_k\right)\right|(1-\tau)d\tau \\ &\quad + \int_0^1 \left|g_{x_k}\left((\operatorname{Hess} \hat{f}_{x_k}(0_{x_k}) - \operatorname{Hess} \hat{f}(\tau\eta_k))[\eta_k], \eta_k\right)\right|(1-\tau)d\tau \\ &\leq \frac{1}{2}\beta_H\gamma_k\|\eta_k\|^2 + \frac{1}{6}\beta_{L2}\|\eta_k\|^3.\end{aligned}$$

It then follows from (2.44), using the Cauchy bound (2.12), that

$$|\rho_k - 1| \leq \frac{(3\beta_H\gamma_k + \beta_{L2}\|\eta_k\|)\,\|\eta_k\|^2}{6\gamma_k \min\{\Delta_k, \gamma_k/\beta\}},$$

where β is an upper bound on the norm of H_{x_k}. Since $\|\eta_k\| \leq \Delta_k$ and $\|\eta_k\| \leq c_4\gamma_k$, it follows that

$$|\rho_k - 1| \leq \frac{(3\beta_H + \beta_{L2}c_4)\,(\min\{\Delta_k, c_4\gamma_k\})^2}{6\min\{\Delta_k, \gamma_k/\beta\}}. \tag{2.45}$$

Either, Δ_k is active in the denominator of (2.45), in which case we have

$$|\rho_k - 1| \leq \frac{(3\beta_H + \beta_{L2}c_4)\,\Delta_k c_4\gamma_k}{6\Delta_k} = \frac{(3\beta_H + \beta_{L2}c_4)\,c_4}{6}\gamma_k.$$

Or, γ_k/β is active in the denominator of (2.45), in which case we have

$$|\rho_k - 1| \leq \frac{(3\beta_H + \beta_{L2}c_4)\,(c_4\gamma_k)^2}{6\gamma_k/\beta} = \frac{(3\beta_H + \beta_{L2}c_4)\,c_4^2\beta}{6}\gamma_k.$$

In both cases, since $\lim_{k\to\infty}\gamma_k = 0$ in view of (2.36), if follows that $\lim_{k\to\infty}\rho_k = 1$.

Equation (2.39) comes from Lemma 14.

We prove (2.40). It follows from Taylor's formula (Lemma 12, where the parallel translation becomes the identity since the domain of $\hat{f}_{x_k}$ is the Euclidean space $T_{x_k}\mathcal{M}$) that

$$\operatorname{grad} \hat{f}_{x_k}(\xi) = \operatorname{grad} \hat{f}_{x_k}(0_{x_k}) + \operatorname{Hess} \hat{f}_{x_k}(0_{x_k})[\xi] + \int_0^1 \left(\operatorname{Hess} \hat{f}_{x_k}(\tau\xi) - \operatorname{Hess} \hat{f}_{x_k}(0_{x_k})\right)[\xi]\; d\tau.$$

The conclusion comes by the Lipschitz condition (2.34) and the approximation condition (2.33).

Finally, equation (2.41) comes from the stopping criterion (2.8) of the inner iteration. More precisely, the truncated CG loop (Algorithm 5) terminates if either $g(\delta_j, H_{x_k}\delta_j) \leq 0$,

or $\|\eta_{j+1}\| \geq \Delta$, or the criterion (2.8) is satisfied. Since $\{x_k\}$ converges to v and Hess $f(v)$ is positive-definite, if follows that H_{x_k} is positive-definite for all k greater than a certain K. Therefore, for all $k > K$, the criterion $g(\delta_j, H_{x_k}\delta_j) \leq 0$ is never satisfied. In view of (2.37) and (2.38), it can be shown that the trust-region is eventually inactive. Therefore, increasing K if necessary, the criterion $\|\eta_{j+1}\| \geq \Delta$ is never satisfied for all $k > K$. In conclusion, for all $k > K$, the stopping criterion (2.8) is satisfied each time a computed η_k is returned by the tCG loop. Therefore, the tCG loop behaves as a classical linear CG method; see, e.g., [NW99, Section 5.1]. Consequently, grad $m_{x_k}(\eta_j) = r_j$ for all j. Choose K such that for all $k > K$, $\|\text{grad}\, f(x_k)\| = \|\text{grad}\, m_{x_k}(0)\|$ is so small—it converges to zero in view of (2.36)—that the stopping criterion (2.8) yields

$$\|\text{grad}\, m_{x_k}(\eta_j)\| = \|r_j\| \leq \|r_0\|^{1+\theta} = \|\text{grad}\, m_{x_k}(0)\|^{1+\theta} \text{ or } \quad k \geq d. \tag{2.46}$$

If the second condition in (2.46) is active, then it means that the linear CG process has been completed, so grad $m_{x_k}(\eta_k^j) = 0$, and (2.41) trivially holds. On the other hand, if the first condition in (2.46) is active, then we obtain (2.41) with $c_2 = 1$. □

The constants in the proof of Theorem 16 can be chosen as $c_0 < \lambda_{min}$, $c_1 > \lambda_{max}$, $c_4 > 1/\lambda_{min}$, $c_5 > 1$, $c_3 \geq \beta_{L2}$, $c_3' \geq \beta_H$, $c_2 \geq 1$, where λ_{min} and λ_{max} are the smallest and largest eigenvalue of Hess $f(v)$ respectively. Consequently, the constant c in the convergence bound (2.35) can be chosen as

$$c > \frac{1}{\lambda_{min}} \left(\left(\beta_{L2}/\lambda_{min}^2 + \beta_H/\lambda_{min} \right) \lambda_{max}^2 + \lambda_{max}^{1+\theta} \right). \tag{2.47}$$

A nicer-looking bound holds when convergence is evaluated in terms of the norm of the gradient, as expressed in the theorem below which is a direct consequence of (2.42)-(2.43).

Theorem 17. *Under the assumptions of Theorem 16, if $\theta + 1 < 2$, then given $c_g > 1$ and $\{x_k\}$ generated by the algorithm, there exists $K > 0$ such that*

$$\|\text{grad}\, f(x_{k+1})\| \leq c_g \|\text{grad}\, f(x_k)\|^{\theta+1}$$

for all $k > K$.

Nevertheless, (2.42)-(2.43) suggests that the algorithm may not perform well when the relative gap $\lambda_{max}/\lambda_{min}$ is large. In spite of this, numerical experiments on eigenvalue problems have shown that the method tends to behave as well, or even better than other methods in the presence of a small relative gap [ABG06].

2.3.3 Discussion of Convergence

The main global convergence result (Theorem 9) shows that RTR/tCG (Algorithm 4-5) converges to a set of stationary points of the cost function for *all* initial conditions. This is an improvement on the pure Newton method, for which only local convergence results exist. However, the convergence theory falls short of showing that the algorithm always converges to a local minimizer. This is not surprising: since we have ruled out the possibility of checking positive-definiteness of the Hessian of the cost function, we have no way of testing whether a stationary point is a local minimizer or not (note that even checking positive-definiteness of the Hessian is not always sufficient for determining if a stationary point is a local minimizer or not: if the Hessian is singular and nonnegative definite, then no conclusion can be drawn). In fact, for the vast majority of optimization methods, only convergence to stationary points can be secured unless some specific assumptions (like convexity) are made; see, e.g., [Pol97, Ch. 1]. Nevertheless, it is observed in numerical experiments with random initial conditions that the algorithm systematically converges to a local minimizer; convergence to a saddle point is only observed on specifically crafted problems, for example when the iteration is started on a point that is a saddle point in computer arithmetic. This is due to the fact that the algorithm is a descent method, i.e., $f(x_{k+1}) < f(x_k)$ whenever $x_{k+1} \neq x_k$. Therefore, convergence to saddle points or local minima is unstable under perturbations.

Concerning the order of convergence to local minima, we point out that there are cases where the bound (2.35) also holds with "$\min\{\theta+1, 2\}$" replaced by "$\min\{\theta+1, 3\}$", i.e., cubic convergence can be achieved. This is related to the cubic convergence of the Riemannian Newton method when the cost function is symmetric around the local minimizer v, that is, $f(\mathrm{Exp}_x(\xi)) = f(\mathrm{Exp}_x(-\xi))$. This issue is of theoretical importance in applications where state-of-the-art methods converge cubically. Notice however that a cubic method may be less efficient than a quadratic method, even as k goes to infinity (as pointed out in [DV00], concatenating two steps of a quadratic method yields a quartic method).

Randomization

To conclude this discussion, we mention the possibility of using a stochastic version of RTR/tCG. The initial condition η^0 in the tCG algorithm is selected from a uniform distribution in some small neighborhood of 0_{x_k} in $T_{x_k}\mathcal{M}$—this can be thought of as artificial

numerical noise. Then the output η of the tCG algorithm is compared against the Cauchy point η^C (which can be computed at little additional cost). If $m_{x_k}(\eta) < m_{x_k}(\eta^C)$, then η is returned; otherwise, η^C is returned.

This stochastic version is interesting both from a theoretical and practical point of view. From a theoretical viewpoint, it yields convergence to a local minimizer with probability one for all initial conditions. Indeed, convergence to stationary points still holds since the algorithm improves on the Cauchy point; but accumulation points must be local minima because the instability of the saddle points or local maxima will eventually be revealed with probability one by the random perturbations. From a practical point of view, the randomized version is efficient in kicking the iteration away from saddle points: due to the trust-region approach, the iterates quickly escape from the saddle points.

The randomization we have described applies to the general RTR/tCG method. Practical applications may lend themselves to other forms of randomization. For example, if a Rayleigh quotient has to be minimized on a Grassmann manifold, then a possibility mentioned in [ST00, Section 3.2] is to use Rutishauser's randomization technique [Rut70]; it consists in appending a random vector to the current subspace, computing the Ritz pairs and discarding the one with largest Ritz value.

Note that this randomization technique may be difficult when using a preconditioned truncated CG solver. The reason is that a truncated CG using the preconditioner N^{-1} defines the trust-region according the norm induced by N. In the case that N it is not available, the norms of the subproblem iterates can be computed using the recurrences discussed in Section 2.2. However, these recurrences (as presented) require that the subproblem is initialized to zero in order to easily initialize the recurrences:

$$\begin{aligned} \|\eta^0\|_N = \langle\, \eta^0, N\eta^0 \rangle = \langle\, 0, 0 \rangle = 0 \\ \langle\, N\eta^0, d_0 \rangle = \langle\, 0, d_0 \rangle = 0 \\ \|d_0\|_N^2 = \langle\, r_0, z_0 \rangle\ . \end{aligned}$$

A randomized start therefore requires the ability to apply the operator N at least once per outer iteration. In the event that the operator N is not available—only N^{-1} is available—it's effect on η^0 can be observed by performing a linear solve with respect to N^{-1}. The implicit Riemannian trust-region method discussed in the next section does not suffer from this complication.

2.4 Implementing the RTR

In this section, we briefly review the essential "ingredients" necessary for applying the RTR/tCG method (Algorithm 4-5). For the problem of computing extreme eigenspaces of matrices, numerical experiments show that the RTR/tCG algorithm can match and sometimes dramatically outperform existing algorithms, as demonstrated in Chapter 4 and [ABG07, ABG06, ABGS05]. Other applications that lend themselves nicely to an RTR approach include reduced-rank approximation to matrices, the Procrustes problem, nearest-Jordan structure, trace minimization with a nonlinear term, simultaneous Schur decomposition, and simultaneous diagonalization; see, e.g., [HM94, LE00].

The following elements are required for applying the RTR method to optimizing a cost function f on a Riemannian manifold $(\mathcal{M}, g)$:

1. a tractable numerical representations for points x on $\mathcal{M}$, for tangent vectors in $T_x\mathcal{M}$, and for the inner products $g_x(\cdot,\cdot)$ on $T_x\mathcal{M}$;

2. choice of a retraction $R_x : T_x\mathcal{M} \to \mathcal{M}$ (Definition 3)

3. formulas for $f(x)$, $\operatorname{grad} f(x)$ and the approximate Hessian H_x that satisfies the properties required for the convergence results in Section 2.3.

Choosing a good retraction amounts to finding an approximation of the exponential mapping that can be computed with low computational cost. Guidelines can be found in [CI01, DN04]. This is an important open research topic.

Formulas for $\operatorname{grad} f(x)$ and $\operatorname{Hess} \hat{f}_x(0_x)$ can be obtained by identification in a Taylor expansion of the lifted cost function $\hat{f}_x$, namely

$$\hat{f}_x(\eta) = f(x) + g_x(\operatorname{grad} f(x), \eta) + \frac{1}{2} g_x\left(\operatorname{Hess} \hat{f}_x(0_x)[\eta], \eta\right) + O(\|\eta\|^3),$$

where $\operatorname{grad} f(x) \in T_x\mathcal{M}$ and $\operatorname{Hess} \hat{f}_x(0_x)$ is a linear transformation of $T_x\mathcal{M}$. In order to obtain an "approximate Hessian" H_x that satisfies the approximation condition (2.33), one can pick $H_x := \operatorname{Hess}(f \circ \tilde{R}_x)(0_x)$ where $\tilde{R}_x$ is any retraction. Then, assuming sufficient smoothness of f, R and $\tilde{R}$, the bound (2.33) follows from Lemmas 13 and 5. In particular, the choice $\tilde{R}_x = \operatorname{Exp}_x$ yields $H_x = \nabla \operatorname{grad} f(x)$. If $\mathcal{M}$ is an embedded submanifold of a Euclidean space, then $\nabla_\eta \operatorname{grad} f(x) = \pi \operatorname{Dgrad} f(x)[\eta]$ where π denotes the orthogonal projector onto $T_x\mathcal{M}$.

CHAPTER 3

THE IMPLICIT RIEMANNIAN TRUST-REGION METHOD

Similar to Euclidean trust-region methods, the Riemannian Trust-Region (RTR) method ensures strong global convergence properties while allowing superlinear local convergence. The trust-region mechanism is a heuristic, whereby the performance of the last update dictates the constraints on the next update. The trust-region mechanism makes it possible to disregard the (potentially expensive) objective function during the inner iteration by relying instead on a model restricted to a trust region, i.e., a region where the model is tentatively trusted to be a sufficiently accurate approximation of the objective function. A downside lies in the difficulty of adjusting the trust-region size. When the trust-region radius is too large, valuable time may be spent proposing a new iterate that may be rejected. Alternatively, when the trust-region radius is too small, the algorithm progresses unnecessarily slowly.

The inefficiencies resulting from the trust-region mechanism can be addressed by disabling the trust-region mechanism in such a way as to preserve the desired convergence properties. For example, in [GST05], the authors describe a filter-trust-region method, where a modified acceptance criterion seeks to encourage convergence to first-order critical points. Other approaches adjust the trust-region radius according to dynamic measures such as objective function improvement and step size lengths; see [CGT00].

Instead of relaxing the acceptance criterion, we consider redefining the trust-region as that set of points that would have been accepted under the classical mechanism. Therefore, as long as the update returned from the model minimization is feasible (i.e., it belongs to the trust-region), then acceptance is automatic. In addition to avoiding the discarding of valuable updates, this method eliminates the explicit trust-region radius and its heuristic mechanism, in exchange for a meaningful measure of performance. We refer to this new

trust-region concept as the *implicit trust-region* and to the resulting method as the Implicit Riemannian Trust-Region (IRTR) method.

Like the RTR in Chapter 2, the description of the algorithm and the analysis of convergence consider the optimization of a smooth real function f whose domain is a differentiable manifold $\mathcal{M}$ with Riemannian metric g, i.e., a Riemannian manifold $(\mathcal{M}, g)$. Also like the RTR, the description of this methods will occur in a retraction-based optimization setting.

Section 3.1 reviews the workings of the RTR and describes the IRTR modification. The modification of the trust-region naturally results in a modification to the trust-region subproblem. The effect of this on the solution of the trust-region subproblem is addressed in Section 3.2. Section 3.3 presents the global and local convergence properties for the IRTR method. Section 3.4 reviews the conditions required for an efficient application of the IRTR method.

3.1 IRTR Algorithm

This section briefly reviews the workings of the Riemannian Trust-Region (RTR) method and introduces the Implicit Riemannian Trust-Region (IRTR) method. A subset of the material in this chapter appeared in [BAG08].

We assume that $\mathcal{M}$ is a differentiable manifold and g is a Riemannian metric on $\mathcal{M}$. Together, $(\mathcal{M}, g)$ describes a Riemannian manifold. For all $x \in \mathcal{M}$, the restriction

$$g_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R} \ .$$

of g to the tangent plane $T_x\mathcal{M}$ defines an inner product on the vector space $T_x\mathcal{M}$. Assume that f is a real-valued differentiable function defined on $\mathcal{M}$, and $\operatorname{grad} f(x)$ and $\operatorname{Hess} f(x)$ denote the Riemannian gradient and Hessian, respectively.

The goal of the IRTR, like that of the RTR, is to find a local minimizer of the objective function

$$f : \mathcal{M} \to \mathbb{R} \ .$$

Recall that the RTR method computes iterates by solving a minimization problem on a model of the objective function. This model is typically a quadratic approximation of the "lifted" cost function

$$\hat{f} = f \circ R : T\mathcal{M} \to \mathbb{R}.$$

The RTR method constructs the model m_x of $\hat{f}_x$ and solves the *trust-region subproblem*:

$$\text{minimize } m_x(\xi), \quad \text{subject to } g_x(\xi, \xi) \leq \Delta^2, \tag{3.1}$$

where Δ is the *trust-region radius*. As before, we assume that the model m_x is a quadratic model of $\hat{f}_x$ which approximates $\hat{f}_x$ to at least the first order:

$$m_x(\xi) = \hat{f}_x(0_x) + g_x\left(\xi, \operatorname{grad} \hat{f}_x(0_x)\right) + \frac{1}{2} g_x\left(\xi, H_x[\xi]\right), \tag{3.2}$$

where $H_x[\xi]$ is some symmetric operator on $T_x\mathcal{M}$.

The tangent vector ξ is used to generate a new iterate, which is accepted depending on the value of the quotient

$$\rho_x(\xi) = \frac{\hat{f}_x(0_x) - \hat{f}_x(\xi)}{m_x(0_x) - m_x(\xi)}. \tag{3.3}$$

This quantity measures the ratio between decrease in the objective function and the decrease predicted by the model. In addition to accepting/rejecting proposed iterates, $\rho_x(\xi)$ is also used to expand or shrink the trust-region radius.

The trust-region heuristic is self-tuning, such that an appropriate trust-region radius will eventually be discovered by the algorithm. In practice, however, this adjustment can result in wasted iterations, as proposed iterates are rejected do to poor scores under ρ.

We propose a modification to the trust-region method. This modification bypasses the step size heuristic and directly addresses the model performance. The *implicit trust-region at x* is defined as a superlevel set of ρ_x:

$$\left\{\xi \in T_x\mathcal{M} : \rho_x(\xi) \geq \rho'\right\}. \tag{3.4}$$

The model minimization now consists of

$$\text{minimize } m_x(\xi), \quad \text{subject to } \rho_x(\xi) \geq \rho'. \tag{3.5}$$

The implicit trust-region contains exactly those points that would have been accepted by the classical trust-region mechanism. The result is that there is no trust-region radius to adjust and no explicit acceptance/rejection scheme. The IRTR algorithm is stated in Algorithm 6.

Remark 18. *A more careful examination reveals that a satisfactory value of ρ does not ensure that the next iterate produces a decrease in the objective function: an update η which*

increases the objective function is in the implicit trust-region as long it produces a similar increase in the model. This is in keeping with the classical trust-region presentation, which delayed the guarantee of model decrease to the discussion of global convergence, at which point it becomes necessary. Note that the implicit trust-region mechanism does ensure a decrease of any point in the trust-region, as long as there is also decrease in the model. Furthermore, the truncated conjugate gradient recommended in this paper always produces a decrease in the model.

Algorithm 6 Basic Implicit Riemannian Trust-Region Algorithm

Require: Complete Riemannian manifold $(\mathcal{M}, g)$; scalar field f on $\mathcal{M}$; retraction R from $T\mathcal{M}$ to $\mathcal{M}$.

Input: $\rho' \in (0, 1)$, initial iterate $x_0 \in \mathcal{M}$

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: {Model-based minimization}
3: Obtain η_k by approximately solving (3.5)
4: {Compute next iterate}
5: Set $x_{k+1} = R_{x_k}(\eta_k)$
6: **end for**

Output: Sequences of iterates $\{x_k\}$

The benefit of the classical trust-region definition is that trust-region membership is easily determined, requiring only a norm calculation. The implicit trust-region, on the other hand, requires checking the value of the update vector under ρ. Furthermore, there are two occasions in the truncated CG method that require following a search direction to the edge of the trust-region. In the case of the implicit trust-region, this will not in general admit an analytical solution, and may require a search of ρ along the direction of interest. In general, each evaluation of ρ will require evaluating the objective function f, which will be unallowable in many applications.

In the case that ρ admits an analytical solution, it may be possible to easily and efficiently search for a satisfactory value of ρ along a tangent vector, in order to evaluate step 16 and satisfy steps 9 and 17 of Algorithm 7. If there is simply an efficient method for testing or even bounding below ρ, a backtracking or binary search may be used to satisfy steps 9 and 17. Therefore, it is technically possible to apply the IRTR method to any objective function; it bears restating that the efficiency of the method is tied to the efficiency of evaluating and searching ρ.

We show in Chapter 4 that in a specific but very important application—computing the leftmost eigenvector of a generalized eigenvalue problem—the IRTR algorithm can be implemented in a remarkably efficient way, and yields an algorithm that outperforms state-of-the-art methods on certain instances of the problem. In addition to providing an efficient application of the IRTR, this analysis will provide a new look at an existing eigensolver, the Trace Minimization method [SW82] and [ST00]. Before this, Section 3.3 will show that the IRTR inherits all of the convergence properties of the RTR.

3.2 Solving the model minimization

The general algorithm does not state how (3.1) should be solved. Section 2.2 advocated the use of the truncated conjugate gradient method of Steihaug and Toint [Ste83, Toi81, CGT00]. This method has the benefit of requiring very little memory and returning a point inside the trust-region. It also benefits in the ability to exploit a preconditioner when solving the model minimization.

The new trust-region definition modifies the model minimization, and these modifications must be reflected in the truncated conjugate gradient solver. The trust-region definition occurs in the solver in two cases: when testing that the CG iterates remain inside the trust-region and when moving along search direction to the edge of the trust-region. In the case of a trust-region collision during model minimization, the truncated CG for RTR (Algorithm 5, lines 7 and 13) would move along the prescribed search direction to the edge of the trust-region, performing the search:

$$\text{find } \tau \geq 0 \text{ such that } \eta = \eta^j + \tau d_j \text{ satisfies } \|\eta\|_N \ .$$

Due to the simple description of the trust-region, this search is easily performed. It requires only the solution of a quadratic equation in one variable (namely, τ).

The analogous operation for the implicit trust-region is the following:

$$\text{find } \tau \geq 0 \text{ such that } \eta = \eta^j + \tau d_j \text{ satisfies } \rho_x(\eta) = \rho' \ .$$

In general, this may not be as easily accomplished. Later theorems and lemmas prove the existence of satisfactory points. However, finding them may require a search of ρ_x along directions of interest. In such a case, it is desirable to relax the search, so that we require

only a point inside the trust-region (not necessarily on its edge). Therefore, our presentation of truncated CG for the IRTR assumes only this. The updated truncated conjugate gradient algorithm is displayed in Algorithm 7.

Algorithm 7 Preconditioned Truncated CG for IRTR

Input: Iterate $x \in \mathcal{M}$, $\operatorname{grad} f(x) \neq 0$; trust-region parameter $\rho' \in (0,1)$; convergence criteria $\kappa \in (0,1)$, $\theta \geq 0$; model m_x as in (3.2); symmetric/positive definite preconditioner $N^{-1} : T_x\mathcal{M} \to T_x\mathcal{M}$

Set $\eta^0 = 0_x$, $r_0 = \operatorname{grad} f(x)$, $z_0 = N^{-1} r_0$, $d_0 = -z_0$
for $j = 0, 1, 2, \ldots$ **do**
  {Check κ/θ stopping criterion}
  if $\|r_j\| \leq \|r_0\| \min\left\{\kappa, \|r_0\|^\theta\right\}$ **then**
    return η^j
  end if
  {Check curvature of current search direction}
  if $g_x\left(H_x[d_j], d_j\right) \leq 0$ **then**
    Compute $\tau > 0$ such that $\eta = \eta^j + \tau d_j$ satisfies $\rho_x(\eta) \geq \rho'$
    return η
  end if
  {Generate next inner iterate}
  Set $\alpha_j = g_x\left(z_j, r_j\right) / g_x\left(H_x[d_j], d_j\right)$
  Set $\eta^{j+1} = \eta^j + \alpha_j d_j$
  {Check trust-region}
  if $\rho_x(\eta^{j+1}) < \rho'$ **then**
    Compute $\tau > 0$ such that $\eta = \eta^j + \tau d_j$ satisfies $\rho_x(\eta) \geq \rho'$
    return η
  end if
  {Use CG recurrences to update residual and search direction}
  Set $r_{j+1} = r_j + \alpha_j H_x[d_j]$
  Set $z_{j+1} = N^{-1} r_{j+1}$
  Set $\beta_{j+1} = g_x\left(z_{j+1}, r_{j+1}\right) / g_x\left(z_j, r_j\right)$
  Set $d_{j+1} = -z_{j+1} + \beta_{j+1} d_j$
end for

As with RTR, we have previously elected to stop as soon as an iteration j is reached where

$$\|r_j\| \leq \|r_0\| \min\left\{\kappa, \|r_0\|^\theta\right\} \ . \tag{3.6}$$

As will be discussed in Section 3.3.2, this strategy allows for an improved rate of convergence, by seeking linear convergence early on and superlinear convergence as the algorithm progresses.

Note that, unlike for the Euclidean and Riemannian trust-region methods already discussed, the definition of the trust-region is independent of the use of a preconditioner. This decoupling means that there is no need to observe the effect of N on a vector, so that in the absence of this ability, we are still capable of using a random initialization for the model subproblem.

3.3 Convergence Analysis for IRTR

The mechanisms of the IRTR method are sufficiently different from those of the RTR method that we must construct a separate convergence theory. We first study the global convergence properties of the IRTR method (Algorithm 6). As in Chapter 2, we assume mild conditions on the cost function and the retraction; no assumptions are made concerning the method used to solve the model minimization (3.5), except that there is a "sufficient decrease" on the model. For the RTR and classical trust-region methods in $\mathbb{R}^n$, this is tied to the so-called Cauchy decrease. The modification of the trust-region definition in the IRTR scheme requires revisiting the concept of the Cauchy point, and this endeavor constitutes most of the effort in the global convergence analysis of Section 3.3.1.

We then analyze the convergence of the proposed method around nondegenerate local minima. Specifically, this analysis is conducted in the context of Algorithm 6/7, referring to the IRTR method where the trust-region subproblems are solved using the tCG algorithm with stopping criterion (3.6). It is shown that the iterates of the algorithm converge to nondegenerate stationary points with an order of convergence $\min(\theta + 1, 2)$.

3.3.1 Global Convergence

The main objective of this section is to show that the sequence $\{x_k\}$ generated by Algorithm 6 satisfies $\lim_{k\to\infty} \|\mathrm{grad} f(x_k)\| = 0$. This is the stronger of two global convergence results shown for the RTR presented in Chapter 2 and [ABG07].

In the discussion that follows, $(\mathcal{M}, g)$ is a complete Riemannian manifold of dimension d and R is a retraction on $\mathcal{M}$, as defined in Chapter 2. We assume that the domain of R is the whole of $T\mathcal{M}$. We denote by $P_\gamma^{\tau\leftarrow 0} v$ the vector of $T_{\gamma(\tau)}\mathcal{M}$ obtained by parallel transporting the vector $v \in T_{\gamma(0)}\mathcal{M}$ along the curve γ. We denote by ∇ the Riemannian connection on

$\mathcal{M}$ and by $\operatorname{dist}(x, y)$ the distance between two points on the manifold:

$$\operatorname{dist}(x, y) = \inf_{\gamma} \left\{ \int_0^1 \|\dot{\gamma}(t)\| \mathrm{d}\, t \right\} ,$$

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

We define

$$\hat{f} : T\mathcal{M} \to \mathbb{R} : \xi \mapsto f(R(\xi)) , \tag{3.7}$$

and denote by $\hat{f}_x$ the restriction of $\hat{f}$ to $T_x\mathcal{M}$, with gradient $\operatorname{grad}\hat{f}_x(0_x)$ abbreviated $\operatorname{grad}\hat{f}_x$. Recall from (3.2) that m_x has the form

$$m_x(\xi) = \hat{f}_x(0_x) + g_x\left(\xi, \operatorname{grad}\hat{f}_x\right) + \frac{1}{2} g_x\left(\xi, H_x[\xi]\right) ,$$

with a direction of steepest descent at the origin given by

$$p_x^{\mathrm{S}} = -\frac{\operatorname{grad}\hat{f}_x}{\|\operatorname{grad}\hat{f}_x\|} . \tag{3.8}$$

The first-order convergence results for trust-region methods typically assume that $m_{x_k}(\eta_k)$ is a sufficiently good approximation of $\hat{f}_{x_k}(\eta_k)$. In [CGT00, Theorem 6.4.6], this is guaranteed by the assumption that the Hessian of the cost function is bounded. As for the RTR in Chapter 2, we will weaken this assumption and assume that the cost function is radially Lipschitz continuously differentiable (Definition 6).

The main effort here regards the concept of the Cauchy point. Introduced by Powell in his early papers on the convergence of trust-region methods ([Pow70a, Pow70b, Pow75]), the Cauchy point is defined as the point inside the current trust-region which minimizes the quadratic model m_x along the direction of steepest descent of m_x. In trust-region methods employing a spherical or elliptical definitions of the trust-region, the Cauchy point is easily computed. This follows from the fact that moving along a tangent vector (in this case, the gradient of m_x) will cause you exit the trust-region only once and never re-enter it. However, for the IRTR method, depending on the function ρ_x, it may be possible to move along a tangent vector, exiting and re-entering the trust-region numerous times. Therefore, it may be difficult to compute the Cauchy point; in some cases, the Cauchy point may be at infinity.

One solution is to restrict consideration to a local trust region. Definition 19 defines the relevant segment along the direction of steepest descent, and Definition 20 defines the *local Cauchy point*. Theorem 21 describes the form of the local Cauchy point, while Theorem 22

gives a bound on its decrease under the model m_x. All of these results are analogous to theorems and concepts from classical trust-region theory; see [NW99, CGT00].

Definition 19 (Local Trust-Region)**.** *Consider an iterate $x \in \mathcal{M}$, $\operatorname{grad}\hat{f}_x \neq 0$, and a model m_x as in (3.2). Let ρ_x be defined as in (3.3) and let p_x^{S} be the direction of steepest descent of m_x, given in (3.8). The* local trust-region *along p_x^{S} is given by the following set:*

$$\left\{\tau p_x^{\mathrm{S}} : 0 < \tau \leq \Delta_x\right\},$$

where Δ_x specifies the distance to the edge of the trust-region along p_x^{S}, given by

$$\Delta_x = \inf\left\{\tau > 0 : \rho_x\left(\tau p_x^{\mathrm{S}}\right) < \rho'\right\} \ . \tag{3.9}$$

The local Cauchy point will fulfill the same role as the Cauchy point, except that it is confined to the local trust-region instead of the entirety of the trust-region. The formal definition follows.

Definition 20 (Local Cauchy Point)**.** *Consider an iterate $x \in \mathcal{M}$, $\operatorname{grad}\hat{f}_x \neq 0$, and a model m_x. The* local Cauchy point *p_x^{L} is the point*

$$p_x^{\mathrm{L}} = \tau_x p_x^{\mathrm{S}} \ , \tag{3.10}$$

where

$$\tau_x = \operatorname*{argmin}_{0 \leq \tau \leq \Delta_x} \ m_x(\tau p_x^{\mathrm{S}}) \ ,$$

and where Δ_x and p_x^{S} are from Definition 20.

The local Cauchy point can be computed without leaving the trust-region. This makes it an attractive target when solving the trust-region subproblem using a feasible point method. In fact, the truncated conjugate gradient described earlier in the paper (Algorithm 7) begins with the local Cauchy point and makes reduction from there. As such, the global convergence result for IRTR will require that every solution to the trust-region subproblem produce at least as much decrease in m_x as the local Cauchy point. Therefore, we wish to describe this decrease. Before that, we present some helpful properties of the local Cauchy point.

Theorem 21. *Consider an iterate $x \in \mathcal{M}$, $\operatorname{grad}\hat{f}_x \neq 0$, and $\rho' \in (0, 1)$. Then the local Cauchy point takes the form*

$$p_x^{\mathrm{L}} = \tau_x p_x^{\mathrm{S}},$$

where

$$\begin{aligned} \tau_x &= \begin{cases} \Delta_x, & \text{if } \gamma_x \leq 0 \\ \min\left\{\Delta_x, \frac{\|\text{grad}\hat{f}_x\|^3}{\gamma_x}\right\} & \text{otherwise} \end{cases} \\ \gamma_x &= g_x\left(\text{grad}\hat{f}_x, H_x[\text{grad}\hat{f}_x]\right) . \end{aligned}$$

Furthermore, if $\hat{f}_x$ is bounded below, then $\tau_x < \infty$.

Proof. Assume first that $\gamma_x \leq 0$. Then m_x monotonically decreases as we move along p_x^S, so that the minimizer along p_x^S inside $[0, \Delta_x]$ is $\tau_x p_x^S = \Delta_x p_x^S$.

Assume instead that $\gamma_x > 0$. Then m_x has a global minimizer along p_x^S at $\tau_* p_x^S$, where

$$\tau_* = \frac{g_x\left(-p_x^S, \text{grad}\hat{f}_x\right)}{g_x\left(p_x^S, H_x[p_x^S]\right)} = \frac{\|\text{grad}\hat{f}_x\|^3}{\gamma_x} .$$

If $\tau_* \in (0, \Delta_x)$, then $\tau_x = \min\{\Delta_x, \tau_*\} = \tau_*$ is the minimizer of m_x along p_x^S in the local trust-region, and $\tau_x p_x^S$ is the local Cauchy point. Otherwise, $\Delta_x \leq \tau_*$. Note that m_x monotonically decreases along p_x^S between $[0, \tau_*]$, so that the minimizer of m_x along p_x^S between $[0, \Delta_x]$ occurs at $\Delta_x = \min\{\Delta_x, \tau_*\} = \tau_x$, and $\tau_x p_x^S$ is the local Cauchy point.

Assume now that $\hat{f}$ is bounded below. We will show that $\tau_x < \infty$. First consider when $\gamma > 0$. We have that $\tau_x = \min\{\tau_*, \Delta_x\}$. But τ_* is finite, so that τ_x is finite as well.

Consider now that $\gamma \leq 0$. Assume for the purpose of contradiction that $\tau_x = \infty$. Then $\Delta_x = \infty$, and for all $\tau > 0$, $\rho_x(\tau p_x^S) \geq \rho'$. Then

$$\begin{aligned} \lim_{\tau\to\infty} \hat{f}_x(0) - \hat{f}_x(\tau p_x^S) &= \lim_{\tau\to\infty} \rho_x(\tau p_x^S)\left(m_x(0) - m_x(\tau p_x^S)\right) \\ &\geq \lim_{\tau\to\infty} \rho'\left(m_x(0) - m_x(\tau p_x^S)\right) \\ &= \infty. \end{aligned}$$

But this contradicts the assumption that $\hat{f}$ is bounded below. Therefore, our initial assumption is false and τ_x is finite. □

The next theorem concerns the decrease in m_x associated with the local Cauchy point, as described above. The proof is a straightforward modification of the classical result; see [NW99, Lemma 4.5] or [CGT00, Theorem 6.3.1].

Theorem 22. *Take an iterate $x \in \mathcal{M}$, $\operatorname{grad}\hat{f}_x \neq 0$, and $\rho' \in (0,1)$. Then the local Cauchy point p_x^{L} (as given in Theorem 21) has a decrease in m_x satisfying*

$$m_x(0) - m_x(p_x^{\mathrm{L}}) \geq \frac{1}{2}\|\operatorname{grad}\hat{f}_x\| \min\left\{\Delta_x, \frac{\|\operatorname{grad}\hat{f}_x\|}{\|H_x\|}\right\} .$$

The last result needed before presenting the global convergence result proves that, under the radially Lipschitz continuous assumption on $\hat{f}$, our local trust-region in the direction of steepest descent always maintains a certain size. This property is necessary because the decrease in the local Cauchy point is tied to the size of the local trust-region. The local trust-region cannot be allowed to shrink to zero if we are to obtain a sufficient decrease of the model under the local Cauchy point. The following lemmas guarantee that this situation does not occur.

Lemma 23. *Assume that $\hat{f}$ is radially L-C^1. Assume that there exists $\beta \in (0,\infty)$ such that $\|H_x\| \leq \beta$ for all $x \in \mathcal{M}$. Then for all $\rho' \in (0,1)$, there exists $\beta_\Delta > 0$ such that, for all $x \in \mathcal{M}$, $\operatorname{grad}\hat{f}_x \neq 0$, and all $t \in (0,1]$,*

$$\rho_x\left(t \min\left\{\beta_\Delta \|\operatorname{grad}\hat{f}_x\|, \delta_{RL}\right\} p_x^{\mathrm{S}}\right) \geq \rho'.$$

Proof. As a consequence of the radially L-C^1 property, we have that

$$\left|\hat{f}_x(\xi) - \hat{f}_x(0) - g_x\left(\operatorname{grad}\hat{f}_x, \xi\right)\right| \leq \frac{1}{2}\beta_{\mathrm{RL}}\|\xi\|^2 , \tag{3.11}$$

for all $x \in \mathcal{M}$ and all $\xi \in T_x\mathcal{M}$ such that $\|\xi\| \leq \delta_{\mathrm{RL}}$.

Note that

$$\rho_x(\xi) = \frac{\hat{f}_x(0) - \hat{f}_x(\xi)}{m_x(0) - m_x(\xi)} = 1 - \frac{\hat{f}_x(\xi) - m_x(\xi)}{m_x(0) - m_x(\xi)}.$$

Let $t \in (0,1]$. Let ξ be defined

$$\xi = t \min\left\{\beta_\Delta \|\operatorname{grad}\hat{f}_x\|, \delta_{\mathrm{RL}}\right\} p_x^{\mathrm{S}} .$$

Then

$$\rho_x(\xi) = 1 - \frac{\hat{f}_x(\xi) - m_x(\xi)}{m_x(0) - m_x(\xi)} . \tag{3.12}$$

Since

$$\hat{f}_x(\xi) - m_x(\xi) = \hat{f}_x(\xi) - \hat{f}_x(0) - g_x\left(\operatorname{grad}\hat{f}_x, \xi\right) - \frac{1}{2}g_x\left(\xi, H_x[\xi]\right)$$

it follows from (3.11) and from the bound on $\|H_x\|$ that

$$\begin{aligned}\left|\hat{f}_x(\xi) - m_x(\xi)\right| \leq & \frac{1}{2}\beta_{\mathrm{RL}} t^2 \min{}^2\left\{\beta_\Delta \|\mathrm{grad}\hat{f}_x\|, \delta_{\mathrm{RL}}\right\} \\ & + \frac{1}{2}\beta t^2 \min{}^2\left\{\beta_\Delta \|\mathrm{grad}\hat{f}_x\|, \delta_{\mathrm{RL}}\right\} \ .\end{aligned} \tag{3.13}$$

Also note that

$$m_x(0) - m_x(\xi) = t \min\left\{\beta_\Delta \|\mathrm{grad}\hat{f}_x\|, \delta_{\mathrm{RL}}\right\} \|\mathrm{grad}\hat{f}_x\| - g_x\left(\xi, H_x[\xi]\right)$$

and

$$\begin{aligned}|m_x(0) - m_x(\xi)| \geq & t \min\left\{\beta_\Delta \|\mathrm{grad}\hat{f}_x\|, \delta_{\mathrm{RL}}\right\} \|\mathrm{grad}\hat{f}_x\| \\ & - t^2 \min{}^2\left\{\beta_\Delta \|\mathrm{grad}\hat{f}_x\|, \delta_{\mathrm{RL}}\right\} \beta \ .\end{aligned} \tag{3.14}$$

Then combining (3.13) and (3.14), we have

$$\begin{aligned}\frac{\left|\hat{f}_x(\xi) - m_x(\xi)\right|}{|m_x(0) - m_x(\xi)|} &\leq \frac{1}{2}\frac{(\beta_{\mathrm{RL}} + \beta) t \min\left\{\beta_\Delta \|\mathrm{grad}\hat{f}_x\|, \delta_{\mathrm{RL}}\right\}}{\|\mathrm{grad}\hat{f}_x\| - t \min\left\{\beta_\Delta \|\mathrm{grad}\hat{f}_x\|, \delta_{\mathrm{RL}}\right\} \beta} \\ &\leq \frac{1}{2}\frac{(\beta_{\mathrm{RL}} + \beta)\beta_\Delta \|\mathrm{grad}\hat{f}_x\|}{\|\mathrm{grad}\hat{f}_x\| - \beta_\Delta \|\mathrm{grad}\hat{f}_x\| \beta} \\ &= \frac{1}{2}\frac{(\beta_{\mathrm{RL}} + \beta)\beta_\Delta}{1 - \beta_\Delta \beta} \ ,\end{aligned}$$

because $t \min\left\{\beta_\Delta \|\mathrm{grad}\hat{f}_x\|, \delta_{\mathrm{RL}}\right\} \leq \beta_\Delta \|\mathrm{grad}\hat{f}_x\|$. Then it is easy to see that there exists $\beta_\Delta > 0$ such that

$$\frac{1}{2}\frac{(\beta_{\mathrm{RL}} + \beta)\beta_\Delta}{1 - \beta_\Delta \beta} < 1 - \rho' \ .$$

□

Corollary 24 (Bound on Δ_x)**.** *It follows from Lemma 23 that, under the conditions required for the lemma,* $\Delta_x \geq \min\left\{\beta_\Delta \|\mathrm{grad}\hat{f}_x\|, \delta_{RL}\right\}$.

The convergence theory of the RTR method in Section 2.3.1 provides two results on global convergence. The stronger of these results (Theorem 9 states that the accumulation points of any series generated by the algorithm are critical points of the objective function. The definition of the implicit trust-region allows to immediately prove this result, without passing first via the weaker result. The result and approach are analogous to a classical

result from Euclidean trust-region theory originally given in [SSB85]; see [NW99, Theorem 4.8] or [CGT00, Theorem 6.4.6] for modern representations. Theorem 25 proves this for the IRTR method described in Algorithm 6; the proof is much simpler than that for the RTR method.

Theorem 25 (Global Convergence)**.** *Let $\{x_k\}$ be a sequence of iterates produced by Algorithm 6, each $\operatorname{grad}\hat{f}_x \neq 0$, with $\rho' \in (0,1)$. Suppose that there exists $\beta \in (0,\infty)$ such that each $\|H_{x_k}\| \leq \beta$. Suppose that each $\hat{f}_{x_k}$ is C^1, and that $\hat{f}$ is radially L-C^1 and bounded below on the level set*

$$\{x : f(x) \leq f(x_0)\}.$$

Further suppose that each update η_k produces at least as much decrease in m_{x_k} as a fixed fraction of the local Cauchy point. That is, for some constant $c_1 > 0$,

$$m_{x_k}(0) - m_{x_k}(\eta_k) \geq c_1 \|\operatorname{grad}\hat{f}_{x_k}\| \min\left\{\Delta_{x_k}, \frac{\|\operatorname{grad}\hat{f}_{x_k}\|}{\beta}\right\},$$

where the terms in this inequality are from Theorem 22.

Then

$$\lim_{k\to\infty} \|\operatorname{grad} f(x_k)\| = 0.$$

Proof. Assume for the purpose of contradiction that the theorem does not hold. Then there exists $\epsilon > 0$ such that, for all $K > 0$, there exists $k \geq K$ such that

$$\|\operatorname{grad} f(x_k)\| > \epsilon.$$

From the workings of Algorithm 6,

$$\begin{aligned}
f(x_k) - f(x_{k+1}) &= \hat{f}_{x_k}(0) - \hat{f}_{x_k}(\eta_k) = \rho_{x_k}(\eta_k)\left(m_{x_k}(0) - m_{x_k}(\eta_k)\right) \\
&\geq \rho'\left(m_{x_k}(0) - m_{x_k}(\eta_k)\right) \\
&\geq \rho' c_1 \|\operatorname{grad}\hat{f}_{x_k}\| \min\left\{\Delta_{x_k}, \frac{\|\operatorname{grad}\hat{f}_{x_k}\|}{\beta}\right\} \\
&\geq \rho' c_1 \|\operatorname{grad}\hat{f}_{x_k}\| \min\left\{\beta_\Delta \|\operatorname{grad}\hat{f}_x\|, \delta_{\mathrm{RL}}, \frac{\|\operatorname{grad}\hat{f}_{x_k}\|}{\beta}\right\},
\end{aligned}$$

where the last inequality results from Corollary 24. Then for all $K > 0$, there exists $k \geq K$ such that

$$f(x_k) - f(x_{k+1}) \geq \rho' c_1 \epsilon \min\left\{\beta_\Delta \epsilon, \delta_{\mathrm{RL}}, \frac{\epsilon}{\beta}\right\} > 0.$$

But because f is bounded below and decreases monotonically with the iterates produced by the algorithm, we know that

$$\lim_{k\to\infty} (f(x_k) - f(x_{k+1})) = 0,$$

and we have reached a contradiction. Hence, our original assumption must be false, and the desired result is achieved. □

Remark 26. *The axioms for Theorem 25 require that each solution of the model subproblem produce some fixed fraction of the decrease produced by the local Cauchy point. For the truncated CG solver, this is not a problem, as the solver typically begins with the local Cauchy point and makes improvement with further iterations. However, in the case that the truncated CG solver is randomly initialized, it becomes necessary to ensure that the resulting subproblem solution satisfies the Cauchy decrease condition. The most straightforward way to ensure this is to compute the local Cauchy point and compare its decrease against that produced by the subproblem solution, keeping whichever of the two points produces the most decrease.*

3.3.2 Local Convergence

The local convergence results for the IRTR require significantly less modification from the RTR than did the global convergence results. For the sake of brevity, only original proofs will be provided. Neglected proofs may be found in Section 2.3.2.

As in Section 2.3.2, we ask one additional constraint be placed upon the retraction, in addition to the definition of retraction. This is that that there exists some $\mu > 0$ and δ_μ such that

$$\|\xi\| \geq \mu \text{dist}(x, R_x(\xi)), \quad \text{for all } x \in \mathcal{M}, \text{ for all } \xi \in T_x\mathcal{M}, \|\xi\| \leq \delta_\mu. \tag{3.15}$$

In particular, the exponential retraction satisfies (3.15) as an equality, with $\mu = 1$. The bound is also satisfied when R is smooth and $\mathcal{M}$ is compact.

The first local convergence result states that the nondegenerate local minima are attractors of Algorithm 6/7. This theorem is unmodified from the same result for the RTR; see Theorem 15, which itself is closely related to the Capture Theorem [Ber95, Theorem 1.2.5].

Theorem 27 (Local Convergence to Local Minimima)**.** *Consider Algorithm 6/7–i.e., the Implicit Riemannian Trust-Region algorithm where the trust-region subproblem* (3.1) *is solved using the modified truncated CG algorithm–with all the assumptions of Theorem 25 (Global Convergence). Let v be a nondegenerate local minimizer of f, i.e.,* $\operatorname{grad} f(v) = 0$ *and* $\operatorname{Hess} f(v)$ *is positive definite. Assume that $x \to \|H_x^{-1}\|$ is bounded on a neighborhood of v and that* (3.15) *holds for some $\mu > 0$ and $\delta_\mu > 0$. Then there exists a neighborhood V of v such that, for all $x_0 \in V$, the sequence $\{x_k\}$ generated by Algorithm 6/7 converges to v.*

Now we study the order of convergence of the sequences that converge to a nondegenerate local minimizer. This result is the same as for the RTR. However, the proof is slightly modified. The previous proof showed that the trust-region eventually becomes inactive as a stopping condition on the truncated CG; this requires review under the new trust-region definition.

Theorem 28 (Order of Local Convergence)**.** *Consider Algorithm 6/7. Suppose that R is C^2 retraction, that f is a C^2 cost function on $\mathcal{M}$, and that*

$$\|H_{x_k} - \operatorname{Hess} \hat{f}_{x_k}(0_{x_k})\| \leq \beta_H \|\operatorname{grad} f(x_k)\|, \tag{3.16}$$

that is, H_{x_k} is a sufficiently good approximation of $\operatorname{Hess} \hat{f}_{x_k}(0_{x_k})$*. Let $v \in \mathcal{M}$ be a nondegenerate local minimizer of f, (i.e.,* $\operatorname{grad} f(v) = 0$ *and* $\operatorname{Hess} f(v)$ *is positive definite). Further assume that* $\operatorname{Hess} \hat{f}_x(0_x)$ *is Lipschitz-continuous at 0_x uniformly in a neighborhood of v, i.e., there exist β_{L2}, $\delta_1 > 0$ and $\delta_2 > 0$ such that, for all $x \in B_{\delta_1}(v)$ and all $\xi \in B_{\delta_2}(0_x)$, there holds*

$$\|\operatorname{Hess} \hat{f}_x(\xi) - \operatorname{Hess} \hat{f}_x(0_x)\| \leq \beta_{\mathrm{L2}} \|\xi\|. \tag{3.17}$$

Then there exists $c \geq 0$ such that, for all sequences $\{x_k\}$ generated by the algorithm converging to v, there exists $K > 0$ such that for all $k > K$,

$$dist(x_{k+1}, v) \leq c \left(dist(x_k, v)\right)^{\min\{\theta+1, 2\}}.$$

Proof. We will show below that there exist $\tilde{\Delta}$, c_0, c_1, c_2, c_3, c_3', c_4, and c_5 such that, for all sequences $\{x_k\}$ satisfying the conditions asserted, all $x \in \mathcal{M}$, all ξ with $\|\xi\| \leq \tilde{\Delta}$, and all k

greater than some K, there holds

$$c_0 \text{dist}(v, x_k) \leq \|\text{grad} f(x_k)\| \leq c_1 \text{dist}(v, x_k), \tag{3.18}$$

$$\|\eta_k\| \leq c_4 \|\text{grad}\, m_{x_k}(0_{x_k})\| \leq \tilde{\Delta}, \tag{3.19}$$

$$\|\text{grad} f(R_{x_k}(\xi))\| \leq c_5 \|\text{grad}\, \hat{f}_{x_k}(\xi)\|, \tag{3.20}$$

$$\|\text{grad}\, m_{x_k}(\xi) - \text{grad}\, \hat{f}_{x_k}(\xi)\| \leq c_3 \|\xi\|^2 + c_3' \|\text{grad} f(x_k)\| \|\xi\|, \tag{3.21}$$

$$\|\text{grad}\, m_{x_k}(\eta_k)\| \leq c_2 \|\text{grad}\, m_{x_k}(0)\|^{\theta+1}, \tag{3.22}$$

where $\{\eta_k\}$ is the sequence of update vectors corresponding to $\{x_k\}$. With these results at hand, the proof is concluded as follows. For all $k > K$, it follows from (3.18) that

$$c_0 \text{dist}(v, x_{k+1}) \leq \|\text{grad} f(x_{k+1})\| = \|\text{grad} f(R_{x_k}(\eta_k))\|,$$

and from (3.20) that

$$\|\text{grad} f(R_{x_k}(\eta_k))\| \leq c_5 \|\text{grad}\, \hat{f}_{x_k}(\eta_k)\|,$$

and from (3.19) and (3.21) and (3.22) that

$$\begin{aligned} \|\text{grad}\, \hat{f}_{x_k}(\eta_k)\| &\leq \|\text{grad}\, m_{x_k}(\eta_k) - \text{grad}\, \hat{f}_{x_k}(\eta_k)\| + \|\text{grad}\, m_{x_k}(\eta_k)\| \\ &\leq (c_3 c_4^2 + c_3' c_4) \|\text{grad}\, m_{x_k}(0)\|^2 + c_2 \|\text{grad}\, m_{x_k}(0)\|^{\theta+1}, \end{aligned}$$

and from (3.18) that

$$\|\text{grad}\, m_{x_k}(0)\| = \|\text{grad} f(x_k)\| \leq c_1 \text{dist}(v, x_k).$$

Consequently, taking K larger if necessary so that $\text{dist}(v, x_k) < 1$ for all $k > K$, it follows that

$$\begin{aligned} c_0 \text{dist}(v, x_{k+1}) &\leq \|\text{grad} f(x_{k+1})\| \\ &\leq c_5 (c_3 c_4^2 + c_3' c_4) \|\text{grad} f(x_k)\|^2 + c_5 c_2 \|\text{grad} f(x_k)\|^{\theta+1} \\ &\leq c_5 ((c_3 c_4^2 + c_3' c_4) c_1^2 (\text{dist}(v, x_k))^2 + c_2 c_1^{\theta+1} (\text{dist}(v, x_k))^{\theta+1}) \\ &\leq c_5 ((c_3 c_4^2 + c_3' c_4) c_1^2 + c_2 c_1^{\theta+1}) (\text{dist}(v, x_k))^{\min\{2, \theta+1\}} \end{aligned}$$

for all $k > K$, which is the desired result. It remains to prove the bounds (3.18)-(3.22).

Equation (3.18) comes from Lemma 13 and is due to the fact that v is a nondegenerate critical point. Equations (3.19)-(3.21) are proved in Theorem 15.

It remains only to prove (3.22). Let γ_k denote $\|\text{grad} f(x_k)\|$. It follows from the definition of ρ_k that

$$\rho_k - 1 = \frac{m_{x_k}(\eta_k) - \hat{f}_{x_k}(\eta_k)}{m_{x_k}(0_{x_k}) - m_{x_k}(\eta_k)}. \tag{3.23}$$

From Taylor's theorem (12), there holds

$$\hat{f}_{x_k}(\eta_k) = \hat{f}_{x_k}(0_{x_k}) + g_{x_k}\left(\text{grad} f(x_k), \eta_k\right) + \int_0^1 g_{x_k}\left(\text{Hess}\, \hat{f}_{x_k}(\tau\eta_k)[\eta_k], \eta_k\right)(1-\tau)d\tau.$$

It follows that

$$\begin{aligned}
\left|m_{x_k}(\eta_k) - \hat{f}_{x_k}(\eta_k)\right| &= \left|\int_0^1 \left(g_{x_k}\left(H_{x_k}[\eta_k], \eta_k\right) - g_{x_k}\left(\text{Hess}\, \hat{f}_{x_k}(\tau\eta_k)[\eta_k], \eta_k\right)\right)(1-\tau)d\tau\right| \\
&\leq \int_0^1 \left|g_{x_k}\left((H_{x_k} - \text{Hess}\, \hat{f}_{x_k}(0_{x_k}))[\eta_k], \eta_k\right)\right|(1-\tau)d\tau \\
&\quad + \int_0^1 \left|g_{x_k}\left((\text{Hess}\, \hat{f}_{x_k}(0_{x_k}) - \text{Hess}\, \hat{f}(\tau\eta_k))[\eta_k], \eta_k\right)\right|(1-\tau)d\tau \\
&\leq \frac{1}{2}\beta_H \gamma_k \|\eta_k\|^2 + \frac{1}{6}\beta_{\text{L2}}\|\eta_k\|^3.
\end{aligned}$$

It then follows from (3.23), using the bound on the Cauchy decrease, that

$$\|\rho_k - 1\| \leq \frac{(3\beta_H\gamma_k + \beta_{\text{L2}}\|\eta_k\|)\|\eta_k\|^2}{6\gamma_k \min\{\Delta_k, \gamma_k/\beta\}},$$

where β is an upper bound on the norm of H_{x_k}. Since $\Delta_k \geq \min\{\beta_\Delta \gamma_k, \delta_{\text{RL}}\}$ (Corollary 24) and $\lim_{k\to\infty}\gamma_k = 0$ (in view of Theorem 25), we can choose K large enough that $\Delta_k \geq \beta_\Delta\gamma_k$, for all $k > K$. This and $\|\eta_k\| \leq c_4\gamma_k$ yield

$$\|\rho_k - 1\| \leq \frac{(3\beta_H + \beta_{\text{L2}}c_4)c_4^2\gamma_k^3}{6\min\left\{\beta_\Delta, \frac{1}{\beta}\right\}\gamma_k^2}.$$

Since $\lim_{k\to\infty}\gamma_k = 0$, it follows that $\lim_{k\to\infty}\rho_k = 1$.

Therefore, the trust-region eventually becomes inactive as a stopping criterion for the truncated CG. Furthermore, because $\{x_k\}$ converges to v and Hess $f(v)$ is positive definite, it follows that H_{x_k} is positive definite for all k greater than a certain K. This eliminates negative curvature of the Hessian as a stopping criterion for truncated CG.

This means that the truncated CG loop terminates only after sufficient reduction has been made in $\|\text{grad}\, m_{x_k}(\eta_k)\|$ with respect to $\|\text{grad}\, m_{x_k}(0_{x_k})\|$:

$$\|\text{grad}\, m_{x_k}(\eta_k)\| \leq \|\text{grad}\, m_{x_k}(0_{x_k})\|^{\theta+1},$$

(choosing K large enough that $\|\text{grad}\, m_{x_k}(0_{x_k})\|^\theta < \kappa$ for all $k > K$), or the model minimization has been solved exactly, in which case $\text{grad} m_{x_k}(\eta_k) = 0$. In either case, we have satisfied (3.22). □

3.4 Implementing the IRTR

The following ingredients are required for applying the IRTR method to optimizing a cost function f on a Riemannian manifold $(\mathcal{M}, g)$:

1. a tractable numerical representation for points x on $\mathcal{M}$, for tangent vectors in $T_x\mathcal{M}$, and for the inner products $g_x(\cdot, \cdot)$ on $T_x\mathcal{M}$,
2. a tractable retraction $R_x : T_x\mathcal{M} \to \mathcal{M}$,
3. formulas for $f(x)$, $\text{grad} f(x)$, and an approximate Hessian $H_x[\xi]$ that satisfy the properties required for convergence in Section 3.3,
4. an efficient formula for evaluating or bounding $\rho_x(\xi)$ and an efficient method for searching along $\rho_x(t\xi)$, as needed by steps 9, 16 and 17 of Algorithm 7.

The first three of these are requirements of the RTR method, with the third being required of trust-region methods in general. The fourth requirement is unique to the IRTR.

All trust-region methods assume the ability to evaluate ρ for the purpose of accepting/rejecting candidate iterates, as well as updating the trust-region radius. However, the occurrence of this evaluation is relatively rare, occurring once per outer iteration. Each evaluation of ρ in general requires evaluating the objective function f. The rarity of evaluating the objective function is one of the attractions of trust-region methods; for many problems, evaluating the objective function is significantly more expensive than evaluating the surrogate m.

However, in the case that ρ can be efficiently computed, either directly via f and m or indirectly via some other formula or bound, it is possible to implement the IRTR. With simply an efficient formula for ρ, Armijo-style backtracking searches can be employed to find a point in the implicit trust-region which satisfy sufficient decrease conditions. This is similar to the technique employed in the Generic RTR package ([BAG07]). This is an important point, as the global convergence result Theorem 25 requires a sufficient decrease with respect to the local Cauchy point.

CHAPTER 4

COMPUTING EXTREME SYMMETRIC GENERALIZED EIGENSPACES

Eigenvalues problem are often used as examples for Riemannian optimization because of their familiarity, their importance in numerous application, and their position as problems over non-trivial Riemannian manifolds; see [HM94, EAS98, LE02, AMSV02, AMS04, ABG07, AMS08]. This chapter examines the application of Riemannian optimization techniques to the problem of computing extreme eigenvectors and eigenvalues of a symmetric generalized eigenvalue problem. Specialized methods from the literature are analyzed in the context of Riemannian optimization, and theoretical results from Riemannian optimization shed light onto the performance of these methods. Furthermore, implementations of the RTR and IRTR methods are described for the generalized eigenvalue problem, providing novel iterative eigenvalue solvers with robust convergence theory. These methods are shown to be competitive with specialized eigenvalue solvers on a variety of numerical examples.

4.1 The Symmetric Generalized Eigenvalue Problem

Given two $n \times n$ matrices, A and B, it is possible to define a *generalized eigenvalue problem*. The problem is to find a scalar λ and a non-zero vector v satisfying the *generalized eigenvalue equation*

$$Av = Bv\lambda \ . \tag{4.1}$$

The scalar λ is a *generalized eigenvalue*; the corresponding vector is a *generalized eigenvector*. Together, an eigenvalue λ and an associated eigenvector v comprise an *eigenpair* (λ, v). The set of all eigenvalues is the *spectrum*, denoted $\lambda(A, B)$.

4.1.1 Characterization of Eigensolutions

Given two matrices A and B, a linear *matrix pencil* is a matrix-valued function defined over the complex numbers:

$$P(\lambda) = A - \lambda B \ .$$

We will use the abbreviation (A, B) to denote the linear matrix pencil P.

Recalling the definition of the generalized eigenvalue problem (4.1), an eigenpair (λ, v) satisfies the following:

$$Av = Bv\lambda \ .$$

By redistributing the terms of this equation, we see that

$$(A - \lambda B)v = 0 \ .$$

Because v is non-zero by definition, we see that evaluating the pencil (A, B) at a generalized eigenvalue yields a singular matrix. For this reason, the generalized eigenvalues are sometimes referred to as eigenvalues of the pencil (A, B). These are the values which satisfy the equation

$$\det(A - \lambda B) = 0 \ .$$

This equation is referred to as the *characteristic* equation, and the left hand side is the *characteristic polynomial*. Therefore, it is possible to identify the eigenvalues of (A, B) as the roots of characteristic polynomial. This indicates that an $n \times n$ pencil has only n eigenvalues, where non-real eigenvalues come in conjugate pairs.

If both matrices are symmetric then the matrix pencil is symmetric. If in addition one of the matrices is positive-definite, then the pencil is said to be symmetric/positive-definite. This chapter is concerned with those eigenvalue problems defined by symmetric/positive-definite matrix pencils, and we assume from here forward that the pencil (A, B) is symmetric/positive-definite. We further assume, without loss of generality[1], that the operator B is positive-definite. In this case, the generalized eigenvalue problem becomes analogous to a symmetric standard eigenvalue problem: the eigenvalues are all real, and the eigenvectors are B-orthogonal (and are typically scaled to be B-orthonormal). We can

[1] If instead, only A is positive-definite, then a new generalized eigenvalue problem is created using the pencil (B, A): $Av\lambda^{-1} = Bv$. This requires only that we consider $1/0 = \infty$ a valid eigenvalue.

therefore order and label the eigenvalues as follows:

$$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \ .$$

We will refer to the smallest eigenvalue λ_1 as the *leftmost* eigenvalue. Likewise, the largest eigenvalue λ_n will be referred to as the *rightmost* eigenvalue. These are referred to as *extreme eigenvalues*. We assume from this point forward that the eigenvalues $\{\lambda_j\}$ have been ordered in such a manner as above.

These properties of the symmetric/positive-definite generalized eigenvalue problem can easily be shown by converting the generalized eigenvalue problem defined by (A, B) into a symmetric standard eigenvalue problem, by utilizing the Cholesky decomposition $B = R^T R$:

$$Av = Bv\lambda = R^T Rv\lambda$$

so that

$$\tilde{A}\tilde{v} \doteq (R^{-T}AR^{-1})(Rv) = R^{-T}Av = (Rv)\lambda = \tilde{v}\lambda \ .$$

While it may be mathematically possible to transform a generalized eigenvalue problem into a standard eigenvalue problem in this way, a factorization of B may not be feasible due to the size of the eigenvalue problem. For this reason, the methods discussed in this chapter solve the generalized eigenvalue problem without transforming it to a standard eigenvalue problem. Clearly, these methods can be applied to the solution of symmetric standard eigenvalue problems by taking $B = I$.

For the standard symmetric eigenvalue problem, another characterization of the problem is to find an orthogonal matrix V that diagonalizes the matrix A:

$$V^T AV = \Lambda \ .$$

Such a matrix V has as its columns the eigenvectors of A, with the eigenvalues of A identifiable as the entries of the diagonal matrix Λ. A similar relationship between eigenvectors and diagonalization exists for the symmetric/positive-definite generalized eigenvalue problem as well.

Consider a non-singular matrix V which simultaneously diagonalizes A and B:

$$V^{-1}AV = \Phi = \mathrm{diag}(\phi_1, \ldots, \phi_n)$$
$$V^{-1}BV = \Psi = \mathrm{diag}(\psi_1, \ldots, \psi_n) \ .$$

Then we have

$$\Psi^{-1}\Phi = (V^{-1}B^{-1}V)(V^{-1}AV) = V^{-1}B^{-1}AV$$

and

$$AV = BV(\Psi^{-1}\Phi) \ . \tag{4.2}$$

This identifies the generalized eigenvalues as the elements $\lambda_j = \frac{\phi_j}{\psi_j}$ and the generalized eigenvectors as the columns of V.

It follows from Equation (4.2) that for any eigenvector v_j of the pencil (A, B), the following holds:

$$\text{grq}(v) \doteq \frac{v^T A v}{v^T B v} = \lambda_j \ . \tag{4.3}$$

The left-hand side of this equation is the *generalized Rayleigh quotient with respect to the pencil* (A, B). It should be noted that, in general, the converse is not true: if the generalized Rayleigh quotient of an arbitrary (non-zero) vector v is equal to an eigenvalue, it does not necessarily hold that v is an eigenvector. It is well-known that, for any non-zero vector v, its generalized Rayleigh quotient satisfies the bounds

$$\lambda_1 \leq \text{grq}(v) \leq \lambda_n \ , \tag{4.4}$$

where λ_1 and λ_n are the leftmost and rightmost eigenvalues, respectively. In the special case that $grq(v) = \lambda_1$ or $\text{grq}(v) = \lambda_n$, i.e., the generalized Rayleigh quotient evaluates to an extreme eigenvalue, then the vector v must correspond to the associated extreme eigenvector v_1 or v_n. This significant characteristic of the generalized Rayleigh quotient will be discussed in greater detail in Section 4.1.3.

The generalized Rayleigh quotient above takes a single vector as input. There exists a block version of the generalized Rayleigh quotient that is critical to the material that follows in this chapter. Given an $n \times p$ matrix of full column rank, the *block generalized Rayleigh quotient with respect to the pencil* (A, B) is defined as follows:

$$\text{GRQ}(V) \doteq \text{trace}\left((V^T B V)^{-1} V^T A V\right) \ , \tag{4.5}$$

where the function $\text{trace}\,(M)$ returns the sum of the diagonal entries of the matrix M. An analogous result to Equation (4.3) holds for the function GRQ. For a matrix containing a subset of distinct generalized eigenvectors, the generalized Rayleigh quotient evaluates to

the sum of the associated eigenvalues:

$$\mathrm{GRQ}\left(\begin{bmatrix} v_{i_1} & \dots & v_{i_p} \end{bmatrix}\right) = \sum_{j=1}^{p} \lambda_{i_j} \ . \tag{4.6}$$

Similar to the scalar ($p = 1$) case (4.4), bounds can be placed on the Rayleigh quotient of an $n \times p$ matrix:

$$\lambda_1 + \ldots + \lambda_p \leq \mathrm{GRQ}(V) \leq \lambda_{n-p+1} + \ldots + \lambda_n \ . \tag{4.7}$$

As before, an equality in this bound indicates that V is composed of the extreme eigenvectors, $\mathrm{colsp}\,(V) = \mathrm{colsp}\left(\begin{bmatrix} v_1 & \dots & v_p \end{bmatrix}\right)$ or $\mathrm{colsp}\,(V) = \mathrm{colsp}\left(\begin{bmatrix} v_{n-p+1} & \dots & v_n \end{bmatrix}\right)$.

4.1.2 Applications of Eigenvalue Problems

In many applications, it is necessary only to compute a small number of eigenvectors and eigenvalues, relative to the size of the eigenvalue problem. For example, the finite element discretization of structure results in a generalized eigenvalue problem, where the eigenvalues correspond to the natural frequencies (*modal frequencies*) of the structure and the associated eigenvectors correspond to the nodal displacements (*modal shapes*) associated with a particular frequency. The modes of a system depend on its stiffness and mass characteristics, and the computed modes can be used in a number of ways. They can be used to analyze the response of the structure when exposed to forces at a particular frequency. In this case, it is customary to compute those eigenvalues inside a particular frequency range of interest. Another use for the modes is as a basis for the transient analysis of the system. In this case, it is customary to use the mode shapes corresponding to the lowest frequencies of vibration, as they dominate the behavior of the structure. In this case, the desired eigenvalues are those with the smallest magnitude. These are the same as the leftmost eigenvalues, because the mass and stiffness matrices at positive semi-definite (and are often strictly positive-definite).

Other occurrences of (standard and generalized) symmetric eigenvalue problems arise in informatics applications. When performing empirical principle component analysis (PCA), we wish to compute the largest eigenvalues and eigenvectors of the covariance matrix $C = XX^T$, where X contains the mean-subtracted samples. A related eigenproblem comes from latent semantic analysis (LSA), where the largest eigenvalues and eigenvectors of the matrix XX^T are desired, where X is the document-term matrix. Still another application

comes from linear discriminant analysis (LDA). Given a set of objects partitioned into different classes, the goal in LDA is to compute a subspace which, upon projection of the objects, maximizes the distance between classes while simultaneously minimizing the distance inside of each class. One way to compute this linear subspace is via the eigenvectors corresponding to the largest generalized eigenvalues of the pencil (B, W), where B and W contain, respectively, information corresponding to the separation between classes and the separation within classes before projection.

These applications often result in eigenvalue problems with extremely large matrices (on the order of millions of elements or more). As described above, the usage of the computed eigenpairs, in addition to limitations in compute time and storage, may require that only relatively few eigenpairs be computed. In this case, iterative methods are employed. An iterative eigenvalue solver (*eigensolver*) typically produces a sequence of approximate eigenpairs of improving quality. This is typically done in a *matrix-free* manner. This phrase implies that the operators A and B must be known only via their effect on vectors. This is useful in the scenarios where a matrix representation of the linear operators is not available, or where the size of the problem does not admit any direct factorization of these matrices in the case that they are available.

4.1.3 Rayleigh-Ritz Approximation

The Rayleigh-Ritz process is critical to the implementation of several eigensolver methods. A listing for the algorithm is provided in Algorithm 8.

Given a basis Q for a subspace $\mathcal{Q}$, the Rayleigh-Ritz process computes the "best" approximate eigenvectors of (A, B) in the subspace $\mathcal{Q}$. The algorithm uses the basis to map the large-scale generalized eigenproblem (A, B) into a projected generalized eigenproblem that can be handled with dense methods. The solutions of this dense problem are used to construct approximate eigenpairs for the original eigenproblem. Examining Algorithm 8, the eigenvectors W of the projected problem are referred to as the *primitive Ritz vectors*. The values Θ are the *Ritz values*, and the vectors X are the *Ritz vectors*. The optimality of the process can be described according to three different measures.

The first indicator of optimality regards the eigenvalues computed by the Rayleigh-Ritz process. Given a subspace $\mathcal{Q}$ of dimension m, the p Ritz vectors corresponding to the *leftmost* Ritz values (with respect to $\mathcal{Q}$) *minimize* the generalized Rayleigh quotient over all

Algorithm 8 Rayleigh-Ritz Process

Require: Symmetric operator A, symmetric/positive-definite operator B, both of dimension $n \times n$

Input: Basis Q for subspace $\mathcal{Q}$ of dimension $m \leq n$

1: Form projected matrices:

$$\hat{A} = Q^T A Q \qquad\qquad \hat{B} = Q^T B Q$$

2: Compute $k \leq m$ eigenpairs (Θ, W) of the pencil $(\hat{A}, \hat{B})$:

$$\hat{A} W = \hat{B} W \Theta$$

3: Compute Ritz vectors:

$$X = QW$$

Output: k Ritz pairs (θ_j, x_j), $j = 1, \ldots, k$

p-dimensional subspaces of $\mathcal{Q}$. Similarly, the p Ritz vectors corresponding to the *rightmost* Ritz values (with respect to $\mathcal{Q}$) *maximize* the generalized Rayleigh quotient over all p-dimensional subspaces of $\mathcal{Q}$. This makes the Rayleigh-Ritz process an important part of algorithms that seek to compute extreme eigenspaces by minimizing the generalized Rayleigh quotient (as in Section 4.2).

The second measure of optimality regards the error in the eigenvalue Equation (4.1). Given approximate eigenpairs $(\mathrm{diag}(\theta_j), X)$, the residual of the generalized eigenvalue problem is

$$R = AX - BX\mathrm{diag}(\theta_j) \ .$$

The Ritz pairs produced by Algorithm 8 with respect to a subspace $\mathcal{Q}$ are optimal in reducing the norm of the error over all other B-orthogonal vectors in $\mathcal{Q}$. In particular, if the subspace $\mathcal{Q}$ contains an eigenvector, then this eigenvector will be recovered by the Rayleigh-Ritz process.

The third measure of optimality states that the Ritz vectors and Ritz values are exact eigenvectors and eigenvalues for a related problem. Let Π be the orthogonal projector onto the subspace $\mathcal{Q}$. Then any Ritz pair (θ, x) of (A, B) with respect to $\mathcal{Q}$ satisfies

$$\Pi A \Pi x = \Pi B \Pi x \theta \ .$$

That is, the Ritz vectors and Ritz values with respect to $\mathcal{Q}$ are exact solutions to the eigenproblem defined by restricting (A, B) to the subspace $\mathcal{Q}$.

It is prescient to point out some other properties of the Rayleigh-Ritz process. The process depends only on the relevant subspace, i.e., it is invariant with respect to the choice of basis. Given a subspace $\mathcal{Q}$ and two different bases for $\mathcal{Q}$, $\text{colsp}\,(Q_1) = \mathcal{Q} = \text{colsp}\,(Q_2)$, the Ritz vectors and Ritz values produced from Q_1 will be same as those produced from Q_2. In this way, the Rayleigh-Ritz process can be thought of as a mechanism for mapping a subspace into a set of approximate eigenpairs.

Another important property of the Rayleigh-Ritz process is that it is better suited to computing extreme eigenpairs than interior eigenpairs. Recall from Section 4.1.1 that the GRQ maps eigenvectors to eigenvalues, but that a vector mapped to an eigenvalue is not necessarily an eigenvector. The exception to this rule is for the extreme eigenvalues. The reason for this is that, given a basis Q, the Rayleigh-Ritz process returns Ritz vectors that are linear combinations of the vectors in Q. Because the eigenvectors of (A, B) provide a complete linear basis, then the Ritz vectors are also linear combinations of the eigenvectors of (A, B), so that the Rayleigh quotient of a Ritz vector (i.e., its associated Ritz value) is a linear combination of the eigenvalues of of (A, B). As a result, there are in general an infinite number of ways to construct an interior eigenvalue; however, this not possible with an extreme eigenvalue. As a result, the Rayleigh-Ritz process is used only to compute extreme eigenpairs of (A, B). To compute interior eigenvalues typically requires a spectral transformation.

4.1.4 Spectral Transformations

In order to facilitate the convergence of an eigensolver to the desired part of the spectrum, a common technique is to employ a spectral transformation. *Spectral transformations* are used to modify the spectrum of the eigenvalue problem so that the chosen solver will find the desired eigenvalues. The desired eigenvalues may be, for example:

- smallest (largest) magnitude eigenvalues: those eigenvalues whose absolute value is minimal (maximal), i.e., those eigenvalues closest to (furthest from) zero;
- smallest (largest) algebraic eigenvalues: those eigenvalues furthest to the left (right) on the real line. Also called the leftmost (rightmost) eigenvalues, or extreme eigenvalues;
- interior eigenvalues: those eigenvalues on the interior of the spectrum, i.e., those eigenvalues closest to some value σ.

A spectral transformation defines a new eigenvalue problem whose spectrum may be more amenable to a particular eigensolver. For example, when using an Arnoldi solver (which naturally seeks the largest magnitude eigenvalues) to find other eigenvalues, it is necessary to use a shift-invert spectral transformation. This is accomplished by writing a new eigenvalue problem:

$$(A - \sigma B)^{-1} B v = v\theta \ ,$$

where σ is the shift and $\theta = 1/(\lambda - \sigma)$ is a transformed eigenvalue. By choosing $\sigma = 0$, the smallest magnitude λ become the largest magnitude θ, allowing the Arnoldi eigensolver to easily compute those eigenvalues closest to zero. Choosing a nonzero shift transforms those λ closest to σ into the largest magnitude θ, allowing the Arnoldi eigensolver to easily compute interior eigenvalues. Note that a shift-invert transformation requires the ability to solve linear systems of $A - \sigma B$. While this can be done using iterative linear solvers in a matrix-free manner, it must be done accurately, as the eigenvalues θ that are computed are defined by the accuracy of the linear solve.

Spectral transformations are also useful in expanding the reach of eigensolvers which naturally compute the leftmost or rightmost eigenvalues. Consider the following transformed eigenvalue problem:

$$(A - \sigma B) B^{-1} (A - \sigma B) v = B v \theta \ ,$$

where σ is the shift and $\theta = (\lambda - \sigma)^2$ is a transformed eigenvalue. Choosing $\sigma = 0$ maps the smallest magnitude λ to the leftmost θ and the largest magnitude λ to the rightmost θ. Choosing a nonzero shift maps the λ closest to σ into the leftmost θ and maps the λ furthest from σ into the rightmost θ. As with the shift-invert spectral transformation discussed above, the accuracy of the linear operator $(A - \sigma B) B^{-1} (A - \sigma B)$ defines the computed θ. Note that this spectral transformation also requires a linear solve. However, this linear solve is for the matrix B, which is likely to be less cumbersome than the solution of $A - \sigma B$.

4.2 Specialized Generalized Eigensolvers

The choice among iterative eigensolvers depends on the eigenvalues that are desired. In this case, two classes of eigensolvers present themselves: those solvers that converge to the largest magnitude eigenvalues, and those solvers that converge to the leftmost or rightmost eigenvalues. The former class includes solvers related to the power method, such as the

subspace iteration method and the inverse iterations methods. The latter class includes the methods that will be discussed in this chapter: the Trace Minimization method, the Jacobi-Davidson method, and the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) method. The Riemannian optimization approaches that will be proposed fall into the latter category as well.

The eigenvectors associated with the leftmost (rightmost) eigenvalues are minimizers (maximizers) of the generalized Rayleigh quotient. Computing these eigenvectors is tantamount to optimizing the generalized Rayleigh quotient. The methods described in the following sections do this in one of two ways.

The Jacobi-Davidson method attempts to compute stationary points of the generalized Rayleigh quotient. It does this by looking for corrections to the current approximate eigenpairs, and implementations of this method typically incorporate the Rayleigh-Ritz process to encourage convergence to the extreme eigenvectors. The Trace Minimization and LOBPCG methods compute eigenpairs by explicitly minimizing the generalized Rayleigh quotient. The LOBPCG method relies on the optimizing characterization of the Rayleigh-Ritz procedure, while the Trace Minimization method may utilize the Rayleigh-Ritz procedure to accelerate the convergence of the iteration. The following subsections describe each of these specialized methods in greater detail.

4.2.1 Jacobi-Davidson Method

The Jacobi-Davidson (JD) method [SV96] identifies the eigenvectors and eigenvalues of the pencil (A, B) as the stationary points of the Rayleigh quotient. Given an approximate eigenvector x with Rayleigh quotient $\theta = \mathrm{grq}(x)$, the method computes a correction t such that $x+t$ is an eigenvector of the pencil (A, B). While the iteration can proceed by adding the correction to the current vector (a *simplified JD method*), the correction vector is typically used to expand a search subspace $\mathcal{Q}$, over which the next iterate is generated via the Rayleigh-Ritz process.

The Jacobi-Davidson method takes its name from the combination of approaches that it utilizes: "Jacobi" follows Jacobi's approach of constraining the correction vector to be orthogonal to the current vector, while "Davidson" refers to the subspace acceleration technique wrapped around this approach.

A generic simplified JD method is listed in Algorithm 9. The algorithm operates as

Algorithm 9 Simplified Jacobi-Davidson Eigensolver

Require: Symmetric operator A, symmetric/positive-definite operator B, both of dimension $n \times n$

Input: B-orthonormal matrix X_0 containing p Ritz vectors, diagonal matrix Θ_0 containing p Ritz values

1: **for** $k = 0, 1, \ldots$ until convergence **do**

2: Approximately solve the correction equations for S_k

$$P_k A S_k - P_k B S_k T = -(AX_k - BX_k\Theta) \ , \quad S_k^T B X_k = 0$$

where

$$P_k = I - BX_k X_k^T$$

3: Compute the next basis

$$\hat{X}_{k+1} = X_k + S_k$$

4: Compute the Ritz vectors X_{k+1} and Ritz values Θ_{k+1} of (A, B) with respect to $\hat{X}_{k+1}$

5: **end for**

Output: k Ritz pairs (θ_j, x_j), $j = 1, \ldots, p$

follows. Recall the formula for the single-vector generalized Rayleigh quotient:

$$\operatorname{grq}(x) = \frac{x^T A x}{x^T B x} \ .$$

The simplified Jacobi-Davidson method takes a Newton-like approach to solving the eigenvalue problem. Instead of explicitly minimizing the generalized Rayleigh quotient, the JD method attempts to find an update s to the current iterate x such that $x + s$ is a critical point of grq, i.e., such that the gradient of grq at $x + s$ is zero.

Consider the gradient of grq:

$$\nabla \operatorname{grq}(x) = 2(x^T B x)^{-1}(Ax - Bx \operatorname{grq}(x)) \ .$$

Recall from Chapter 1 that Newton's method operates by computing the Newton step, i.e., that point s satisfying the equation

$$\nabla^2 \operatorname{grq}(x) s = -\nabla \operatorname{grq}(x) \ .$$

In this case, the equation takes the form

$$(A - B \operatorname{grq}(x))\, s = -\left(Ax - Bx \operatorname{grq}(x)\right) \ .$$

The JD method adds an additional constraint on the Newton step, that it satisfy $x^T Bs$. Defining the projector $P = I - Bxx^T$ and noting that $P^T s = s$ and $P\left(Ax - Bx\,\mathrm{grq}(x)\right) = Ax - Bx\,\mathrm{grq}(x)$, we have the following:

$$P\left(A - B\,\mathrm{grq}(x)\right) P^T s = -\left(Ax - Bx\,\mathrm{grq}(x)\right) \ .$$

This equation is referred to as the *Jacobi correction equation*.

Line 2 in Algorithm 9 specifies a block form of this equation, where we wish to compute the correction for p Ritz vectors. The listed equation also leaves unspecified the exact form of the equation, by neglecting to define the matrix T in the left-hand side. The reason is that different implementations of JD specify different values for this matrix. In [Not02], Notay suggests using $T = \tau I$, where $\tau \leq \lambda_1$. This has the benefit of guaranteeing that the matrix $P_k(A - BT)P_k^T$ is positive semi-definite[2] (in addition to being symmetric), so that it can be easily solved using a conjugate gradient linear solver. Once the solver has moved sufficiently close to the leftmost eigenvalue, Notay suggests substituting T with a diagonal matrix containing the current Ritz values. This allows the simplified JD algorithm to enjoy the cubic convergence characteristic of Newton methods on symmetric problems. This fast local convergence can be explained because of ties to Newton's method and the fact that the exact solution to the correction equation results in an iteration which is equivalent to the Rayleigh quotient iteration [GV96, Ste01].

Other descriptions of the JD method employ the Ritz values in T throughout the entire algorithm. In addition to creating a problem that is potentially difficult to solve, the algorithm is no longer guaranteed to converge to the leftmost eigenvectors. The solution to the latter problem comes via the Davidson-type subspace acceleration mechanism. Instead of choosing the next iterate as $X + S$, as in Newton's method, the correction step S is used to expand a subspace from which the next iterate will be chosen. The subspace-accelerated Newton method is the standard presentation of the Jacobi-Davidson method. It is listed in Algorithm 10.

4.2.2 Trace Minimization Method

The Trace Minimization (TRACEMIN) method [SW82, ST00] attempts to compute the leftmost eigenspace of the pencil (A, B) by explicitly minimizing the generalized Rayleigh

[2] In fact, the operator is strictly positive-definite in the space of vectors M-orthogonal to X, where many properly initialized iterative solvers are guaranteed to remain.

Algorithm 10 Jacobi-Davidson Eigensolver

Require: Symmetric operator A, symmetric/positive-definite operator B, both of dimension $n \times n$

Input: B-orthonormal matrix X_0 containing p Ritz vectors, diagonal matrix Θ_0 containing p Ritz values

1: **for** $k = 0, 1, \ldots$ until convergence **do**
2: Compute S_k as in Algorithm 9
3: **if** V_k has reached some maximum size **then**
4: B-orthonormalize $\begin{bmatrix} X_k & S_k \end{bmatrix}$ into V_{k+1}
5: **else**
6: B-orthonormalize $\begin{bmatrix} V_k & S_k \end{bmatrix}$ into V_{k+1}
7: **end if**
8: Compute the Ritz vectors X_{k+1} and Ritz values Θ_{k+1} of (A, B) with respect to V_{k+1}
9: **end for**

Output: k Ritz pairs (θ_j, x_j), $j = 1, \ldots, p$

quotient:

$$\mathrm{GRQ}(X) = \mathrm{trace}\left((X^T B X)^{-1}(X^T A X)\right) \ .$$

This method is defined only for problems where both A and B are symmetric/positive-definite. In this case, we know that all of the generalized eigenvalues are strictly positive. For problems where A is not positive-definite, it is recommend to shift the problem by some value ν, such that $A - \nu B$ is positive-definite. This results in a generalized eigenvalue problem with shifted eigenvalues:

$$(A - \nu B)v = (\lambda - \nu)v \ .$$

Clearly, any $\nu < \lambda_1$ achieves this goal.

A basic version of the TRACEMIN algorithm is stated in Algorithm 11. This algorithm was presented by Sameh and Wisniewski in [SW82]. The algorithm operates as follows. The goal is to solve the minimization problem

$$\begin{aligned} \text{minimize} \quad & \mathrm{trace}\left(X^T A X\right) \\ \text{subject to} \quad & X^T B X = I \ . \end{aligned}$$

Given an iterate X_k satisfying $X_k^T B X_k = I$, the approach is to compute an update S_k which decreases the objective function:

$$\begin{aligned} \text{minimize} \quad & \mathrm{trace}\left((X_k - S_k)^T A (X_k - S_k)\right) \\ \text{subject to} \quad & X^T B S_k = 0 \ . \end{aligned} \tag{4.8}$$

Algorithm 11 Basic TRACEMIN Eigensolver

Require: Symmetric/positive-definite operators A and B, both of dimension $n \times n$

Input: B-orthonormal matrix X_0 containing p Ritz vectors, diagonal matrix Σ_0 containing p Ritz values

1: **for** $k = 0, 1, \ldots$ until convergence **do**

2: Approximately solve the equation for S_k

$$(P_k A P_k) S_k = -P_k A X_k$$

where

$$P_k = I - BX_k \left(X_k^T B^2 X_k\right)^{-1} X_k^T B$$

3: Compute the next basis

$$\hat{X}_{k+1} = X_k - S_k$$

4: Compute the Ritz vectors X_{k+1} and Ritz values Σ_{k+1} of (A, B) with respect to $\hat{X}_{k+1}$

5: **end for**

Output: k Ritz pairs (σ_j, x_j), $j = 1, \ldots, p$

Choosing the update S_k to be B-orthogonal to the current iterate X_k places it in the tangent space to the constraints, i.e., for infinitesimally small S_k, $X_k - S_k$ satisfies the B-orthonormality constraint. Another motivating factor (which will be discussed more in the next section) is that the generalized Rayleigh quotient is invariant to the choice of subspace, so that components of X_k in S_k do not affect the objective function, and therefore can be excluded.

By considering the first-order optimality conditions of the constrained optimization problem (4.8), we can characterize the minimizer S_k as the solution of the KKT system

$$\begin{bmatrix} A & BX_k \\ X_k^T B & 0 \end{bmatrix} \begin{bmatrix} S_k \\ L_k \end{bmatrix} = \begin{bmatrix} AX_k \\ 0 \end{bmatrix} ,$$

where the matrix L_k contains the Lagrange multipliers which enforce the constraints [NW99]. This can be rewritten as the positive semi-definite system

$$(P_k A P_k) S_k = (P_k A X_k), \quad \text{subject to } X_k^T B S_k = 0 , \tag{4.9}$$

where P_k is the orthogonal projector onto the space B-orthogonal to X_k:

$$P_k = I - BX_k (X_k^T B^2 X_k)^{-1} X_k^T B .$$

Note that this system is positive-definite for all vectors B-orthogonal to X_k.

The authors show that if the update S_k satisfies $X_k^T B S_k = 0$ and

$$\text{trace}\left((X_k - S_k)^T A (X_k - S_k)\right) \leq \text{trace}\left(X_k^T A X_k\right) \ ,$$

then the next iterate X_{k+1} formed by normalizing $X_k - S_k$ (such as via the Rayleigh-Ritz process) automatically satisfies

$$\text{trace}\left(X_{k+1}^T A X_{k+1}\right) \leq \text{trace}\left((X_k - S_k)^T A (X_k - S_k)\right) \leq \text{trace}\left(X_k^T A X_k\right) \ .$$

The authors suggest approximately solving this system via a conjugate gradient method. Such a solver, properly initialized, guarantees that intermediate solutions satisfy the orthogonality condition and that successive solutions monotonically decrease the objective function. Also, the solution of this linear solver admits the use of a preconditioner, which admits the basic TRACEMIN method into the class of preconditioned eigensolvers.

The basic TRACEMIN iteration is similar to the simplified JD iteration. Both are attempting to minimize the generalized Rayleigh quotient. Both choose the next iterate is $X + S$, where S is computed as the solution to some linear system. The difference is that the TRACEMIN method computes the update S to produce a decrease in the generalized Rayleigh quotient, while JD computes the update S to move the iteration to a critical point of the generalized Rayleigh quotient (and hopefully, also a minimizer.)

The authors of TRACEMIN show that the update S_k determined by the exact solution of Equation (4.8) is

$$S_k = X_k - A^{-1} B X_k \left(X_k^T B A^{-1} B X_k\right)^{-1} \ .$$

This means that the subspace spanned by the next iterate $X_{k+1} = X_k - S_k$ is the same subspace as that spanned by $A^{-1} B X_k$, the iteration produced by the method of inverse iterations. This is a classical iterative eigensolver known to have robust global convergence to the leftmost eigenvectors, albeit at a linear rate of convergence. However, the inverse iterations method dictates an exact linear solve against A, while the basic TRACEMIN does not. In this way, the method can be thought of as an inexact inverse iteration method, which the authors show is able to retain the convergence theory of inverse iterations [GV96, Ste01].

In [ST00], Sameh and Tong expand the functioning of the basic TRACEMIN method. One modification of the method seeks to accelerate the convergence by employing a subspace acceleration strategy like that used in the Jacobi-Davidson method. Under this strategy, the

Algorithm 12 Subspace-Accelerated TRACEMIN Eigensolver

Require: Symmetric/positive-definite operators A and B, both of dimension $n \times n$

Input: B-orthonormal matrix X_0 containing p Ritz vectors, diagonal matrix Σ_0 containing p Ritz values

1: **for** $k = 0, 1, \ldots$ until convergence **do**
2: Compute S_k as in Algorithm 11
3: **if** V_k has reached some maximum size **then**
4: B-orthonormalize $\begin{bmatrix} X_k & S_k \end{bmatrix}$ into V_{k+1}
5: **else**
6: B-orthonormalize $\begin{bmatrix} V_k & S_k \end{bmatrix}$ into V_{k+1}
7: **end if**
8: Compute the Ritz vectors X_{k+1} and Ritz values Σ_{k+1} of (A, B) with respect to V_{k+1}
9: **end for**

Output: k Ritz pairs (σ_j, x_j), $j = 1, \ldots, p$

update step S_k computed in Line 2 is used to expand the current approximation subspace, from which the next iterate X_{k+1} is computed via the Rayleigh-Ritz process.

The authors also seek to address the linear rate of convergence observed for the basic TRACEMIN method. As noted above, TRACEMIN's convergence rate is due to its relationship with the inverse iteration method. Just as the Rayleigh quotient iteration improves the convergence rate of the inverse iteration method by shifting the system by the current Ritz values, the *dynamically shifted TRACEMIN method* proposed in [ST00] seeks to improve the rate of convergence by shifting the system solved at Line 2 of the basic TRACEMIN algorithm. The shifts are intended to increase the accuracy of the model (4.8) in approximating the generalized Rayleigh quotient. These strategies attempt to set this model equal to the quadratic Taylor expansion of the generalized Rayleigh quotient. However, such an approach is similar to a Newton method, and as such, it struggles with the problems introduced by a possibly non-convex quadratic model—namely, the effect on the global convergence of the method. These issues are addressed by the trust-region methods discussed in Sections 4.3.3 and 4.3.4.

4.2.3 LOBPCG

The locally optimal block preconditioned conjugate gradient (LOBPCG) eigensolver applies a nonlinear preconditioned conjugate gradient method to the problem of minimizing the

block generalized Rayleigh quotient:

$$\mathrm{GRQ}(V) = \mathrm{trace}\left(\left(V^T BV\right)^{-1}\left(V^T AV\right)\right) \ .$$

The LOBPCG method is listed in Algorithm 13.

Algorithm 13 Locally Optimal Block Preconditioned Conjugate Gradient Method

Require: Symmetric operator A, symmetric/positive-definite operator B, both of dimension $n \times n$, symmetric/positive-definite preconditioner N

Input: B-orthonormal matrix X_0 containing p Ritz vectors, diagonal matrix Θ_0 containing p Ritz values

1: Set $P_0 = []$

2: **for** $k = 0, 1, \ldots$ until convergence **do**

3: Compute the residuals

$$R_k = AX_k - BX_k\Theta_k$$

4: Apply the preconditioner

$$H_k = NR_k$$

5: Compute the primitive Ritz vectors W_k and Ritz values Θ_k with respect to the basis $\begin{bmatrix} X_k & H_k & P_k \end{bmatrix}$

6: Compute the Ritz vectors:

$$X_{k+1} = \begin{bmatrix} X_k & H_k & P_k \end{bmatrix} W_k$$

7: Compute the CG direction:

$$P_{k+1} = \begin{bmatrix} 0 & H_k & P_k \end{bmatrix} W_k$$

8: **end for**

Output: k Ritz pairs (θ_j, x_j), $j = 1, \ldots, p$

In [Kny01], Knyazev proposes a nonlinear CG iteration for the minimization of the generalized Rayleigh quotient. Instead of the traditional CG iteration:

$$x_{k+1} = x_k + \alpha_k p_k \ ,$$

where p_k is the CG search direction, Knyazev recalls a three-term recurrence variation of CG [Saa92, Algorithm 6.19]:

$$x_{k+1} = x_k + \alpha_k h_k + \beta_k (x_k - x_{k-1}) \ ,$$

where h_k is the preconditioned residual. Numerous rules for nonlinear CG dictate the coefficients for the next iterate. However, Knyazev points out that the Rayleigh-Ritz process

provides an efficient way for computing the locally optimal iterate x_{k+1} with respect to the subspace spanned by $\begin{bmatrix} x_k & h_k & (x_k - x_{k-1}) \end{bmatrix}$.

When implementing the LOBPCG method, it is prudent to make some slight modifications to the recurrence to avoid potential problems. In [Kny01], Knyazev notes that in the asymptotic convergence of the method, the update $x_k - x_{k-1}$ between consecutive iterates approaches zero, which causes problems when forming the projected eigenvalue problem in the Rayleigh-Ritz process. Knyazev's suggestion is to replace the explicit difference $x_k - x_{k-1}$ with an implicitly computed difference p_k, as in Line 7 of Algorithm 13. Algorithm 13 presents a block version of LOBPCG implementing this suggestion.

Another issue that was addressed in [HL06] concerns rank deficiency that may occur in the matrix $\begin{bmatrix} X_k & H_k & P_k \end{bmatrix}$, causing the algorithm to stagnate. The authors of [HL06] recommend that the Rayleigh-Ritz process be performed on an orthonormal basis, where the preconditioned residual H_k is explicitly made orthogonal to X_k and the CG direction P_k is implicitly constructed orthogonal to X_k.

4.3 Riemannian Optimization and ESGEV

This section introduces a Riemannian optimization characterization of the extreme symmetric generalized eigenvalue (ESGEV) problem. The specialized methods from Section 4.2 are re-examined in the context of Riemannian methods, and the Riemannian trust-region methods from Chapters 2 and 3 are applied to the problem.

4.3.1 Riemannian Optimization Characterization

Consider a generalized eigenvalue problem, with $n \times n$ real matrices A and B, A symmetric and B symmetric/positive-definite. Let the eigenvalues of the pencil (A, B) be $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Consider the p leftmost eigenvalues, $\lambda_1, \ldots, \lambda_p$, and corresponding eigenvectors, $v_1, \ldots, v_p$. We will assume below, though it is not strictly necessary, that $\lambda_p < \lambda_{p+1}$[3]. Recall from Section 4.1.1 that the $n \times p$ matrix containing the leftmost eigenvectors is a global minimizer of the *generalized Rayleigh quotient*

$$\text{GRQ} : \mathbb{R}_*^{n \times p} \to \mathbb{R} : X \mapsto \text{trace}\left((X^T B X)^{-1}(X^T A X)\right),$$

[3]This assumption allows us to identify a unique global minimizer for the subspace optimization problem that follows. If the assumption does not hold, then there are a collection of globally minimizing subspaces.

where $\mathbb{R}_*^{n\times p}$ is the set of $n \times p$ real matrices of full column rank. Recall also that the converse holds: any global minimizer of this function has captured the subspace associated with the leftmost eigenvectors.

One notable feature of the generalized Rayleigh quotient is that the function is invariant to the choice of basis. Consider two bases, V and VM, where M is a non-singular matrix. The generalized Rayleigh quotient of both bases is the same:

$$\begin{aligned}
\mathrm{GRQ}(VM) &= \mathrm{trace}\left((M^TV^TBVM)^{-1}(M^TV^TAVM)\right) \\
&= \mathrm{trace}\left(M^{-1}(V^TBV)^{-1}M^{-T}M^T(V^TAV)M\right) \\
&= \mathrm{trace}\left(M^{-1}(V^TBV)^{-1}(V^TAV)M\right) \\
&= \mathrm{trace}\left((V^TBV)^{-1}(V^TAV)MM^{-1}\right) \\
&= \mathrm{trace}\left((V^TBV)^{-1}(V^TAV)\right) \\
&= \mathrm{GRQ}(V) \ .
\end{aligned}$$

An explanation for this is that the generalized Rayleigh quotient of a basis is the sum of the Ritz values associated with that basis. However, the Ritz values are dependent only on the subspace spanned by a given basis, a fact which we have already exploited in order to map approximate eigenspaces into Ritz vectors.

Note, the eigenvalues of the pencil (A, B) are the negated eigenvalues of the pencil $(-A, B)$, so that the generalized Rayleigh quotient of (A, B) is the negated generalized Rayleigh quotient of $(-A, B)$. Therefore, computing the rightmost eigenvectors can be done either by minimizing $-$GRQ or by minimizing the generalized Rayleigh quotient of the pencil $(-A, B)$. Therefore, as is traditional with optimization methods, we will discuss only the minimization of GRQ.

Clearly, the generalized Rayleigh quotient induces a real-valued function on the set of p-dimensional subspaces of $\mathbb{R}^n$. This set is the Grassmann manifold $\mathrm{Grass}(p, n)$. Chapter 1.5 described how a Riemannian structure can be put on this set. In order to make precise the notation in the rest of this chapter, we will allow GRQ to continue to represent the generalized Rayleigh quotient on $\mathbb{R}_*^{n\times p}$, and we will denote by f the generalized Rayleigh quotient on the Grassmann manifold:

$$f : \mathrm{Grass}(p, n) \to \mathbb{R} : \mathrm{colsp}\,(X) \mapsto \mathrm{GRQ}(X) \ .$$

Section 1.5 suggested a Riemannian structure for the Grassmann manifold using the Riemannian metric inherited by the total space:

$$g_{\text{colsp}(X)}(\eta, \xi) = \text{trace}\left((X^T X)^{-1} \eta_{\uparrow X}^T \xi_{\uparrow X}\right) \ ,$$

where the basis X is the representation for the subspace $\text{colsp}\,(X)$, and where $\eta_{\uparrow X}$ and $\xi_{\uparrow X}$ are the horizontal lifts of tangent vectors η and ξ with respect to X. The canonical choice for the horizontal distribution requires that each horizontal space $\mathcal{H}_X$ is orthogonal to the corresponding vertical space $\mathcal{V}_X$.

The choice of Riemannian metric and horizontal distribution dictate the formulas for the gradient and Hessian. As such, the flexibility in choosing these structures can be used to reduce the complexity of the resulting gradient and Hessian or to illustrate connections between different solvers. The following subsections desribe two characterizations of this manifold that will be utilized in this chapter.

GRQ with Metric A

With the goal of computing the leftmost eigenspaces of the pencil (A, B), consider the following Riemannian metric:

$$g_{\text{colsp}(X)}(\eta, \xi) = \text{trace}\left((X^T B X)^{-1} \eta_{\uparrow X}^T B \xi_{\uparrow X}\right) \ . \tag{4.10}$$

Recall that the vertical space of the Grassmann manifold is

$$\mathcal{V}_X = \left\{XM : M \in R^{p \times p}\right\} \ .$$

Choosing the horizontal distribution so that the horizontal space $\mathcal{H}_X$ is orthogonal to the vertical space $\mathcal{V}_X$ implies a number of useful properties, regarding the formulas for the gradient and Hessian of f. In this case, the horizontal distribution dictates that each horizontal space at X is

$$\mathcal{H}_X = \left\{Z \in R^{n \times p} : X^T B Z = 0\right\} \ . \tag{4.11}$$

The orthogonal projector onto $\mathcal{H}_X$ is

$$P_X^h = P_{X,BX} = I - X(X^T B X)^{-1} X^T B \ , \tag{4.12}$$

where

$$P_{U,V} = I - U(V^T U)^{-1} V^T \tag{4.13}$$

denotes the projection parallel to colsp (U) onto the space orthogonal to colsp (V).

Let $\mathcal{X} = \text{colsp}\,(X)$. Because the horizontal distribution is orthogonal to the vertical space with respect to g, the gradient of f with respect to g is equal to the gradient of GRQ in the total space:

$$\begin{aligned}
\text{grad} f(\mathcal{X})_{\uparrow X} &= \nabla \text{GRQ}(X) \\
&= 2B^{-1}AX - X(X^TBX)^{-1}X^TAX \\
&= 2P_{X,BX}B^{-1}AX \\
&= 2B^{-1}P_{BX,X}AX \ . \qquad (4.14)
\end{aligned}$$

The choice of horizontal distribution also guarantees that the Riemannian connection is inherited as well, so that the Riemannian Hessian of f is

$$\begin{aligned}
(\text{Hess} f(\mathcal{X})[\eta])_{\uparrow X} &= P_X^h \left(\text{D}\left(\nabla \text{GRQ}(X)\right)[\eta_{\uparrow X}]\right) \\
&= 2P_{X,BX}\left(B^{-1}A\eta_{\uparrow X} - \eta_{\uparrow X}(X^TBX)^{-1}X^TAX\right) \\
&= 2B^{-1}P_{BX,X}\left(A\eta_{\uparrow X} - B\eta_{\uparrow X}(X^TBX)^{-1}X^TAX\right) \ . \qquad (4.15)
\end{aligned}$$

With the Riemannian gradient and Riemannian Hessian in hand, we are equipped to describe a Riemannian steepest descent method for the generalized Rayleigh quotient on the Grassmann manifold. Such an approach is related to the JD eigensolver, as will be shown in Section 4.3.2.

GRQ with Metric B

In order to apply the Riemannian trust-region methods to the optimization of the generalized Rayleigh quotient over the Grassmann manifold, it is useful to have a Taylor expansion of the lifted cost function. Here, we present another characterization of the Grassmann manifold that will later simplify this expansion. Define the horizontal distribution so that the horizontal space at X has the form

$$\mathcal{H}_X = \left\{Z \in \mathbb{R}^{n\times p} : Z^TBX = 0\right\} \qquad (4.16)$$

and employ the non-canonical Riemannian metric

$$g_{\mathcal{X}}\left(\xi, \zeta\right) = \text{trace}\left((X^TBX)^{-1}\xi_{\uparrow X}^T\zeta_{\uparrow X}\right) \ . \qquad (4.17)$$

We denote by P_X^h the orthogonal projector onto the horizontal space $\mathcal{H}_X$:

$$P_X^h = P_{BX} = I - BX(X^T B^2 X)^{-1} X^T B \ , \tag{4.18}$$

where P_{BX} is a shortened notation for $P_{BX,BX}$, defined as in Equation (4.13):

$$P_{BX,BX} = I - BX \left(X^T B^2 X\right)^{-1} X^T B \ .$$

Consider the retraction

$$R_{\mathcal{X}}(\xi) = \operatorname{colsp}\left(X + \xi_{\uparrow X}\right) \ . \tag{4.19}$$

The retraction is used to lift this function from the manifold to the tangent bundle, as follows:

$$\hat{f} : T\operatorname{Grass}(p,n) \to \mathbb{R} : \xi \mapsto f(R(\xi)), \tag{4.20}$$

As before, $\hat{f}_{\mathcal{X}}$ denotes the restriction of $\hat{f}$ to $T_{\mathcal{X}}\operatorname{Grass}(p,n)$.

A second-order expansion of $\hat{f}_{\mathcal{X}}$ yields:

$$\begin{aligned}
\hat{f}_{\mathcal{X}}(\xi) &= \operatorname{trace}\left(\left((X+\xi_{\uparrow X})^T B(X+\xi_{\uparrow X})\right)^{-1}(X+\xi_{\uparrow X})^T A(X+\xi_{\uparrow X})\right) \\
&= \operatorname{trace}\left((X^T BX)^{-1} X^T AX\right) + 2\operatorname{trace}\left((X^T BX)^{-1}\xi_{\uparrow X}^T AX\right) \\
&\quad + \operatorname{trace}\left((X^T BX)^{-1}\xi_{\uparrow X}^T\left(A\xi_{\uparrow X} - B\xi_{\uparrow X}(X^T BX)^{-1}X^T AX\right)\right) + HOT \\
&= \operatorname{trace}\left((X^T BX)^{-1} X^T AX\right) + 2\operatorname{trace}\left((X^T BX)^{-1}\xi_{\uparrow X}^T P_X^h AX\right) \\
&\quad + \operatorname{trace}\left((X^T BX)^{-1}\xi_{\uparrow X}^T P_X^h\left(A\xi_{\uparrow X} - B\xi_{\uparrow X}(X^T BX)^{-1}X^T AX\right)\right) + HOT,
\end{aligned}$$

where the introduction of the projectors does not modify the expression since $P_X^h \xi_{\uparrow X} = \xi_{\uparrow X}$.

Using the Riemannian metric (4.17), we can make the following identifications:

$$\left(\operatorname{grad} f(\mathcal{X})\right)_{\uparrow X} = \left(\operatorname{grad}\hat{f}_{\mathcal{X}}(0_{\mathcal{X}})\right)_{\uparrow X} = 2P_{BX}AX, \tag{4.21}$$

$$\left(\operatorname{Hess}\hat{f}_{\mathcal{X}}(0_{\mathcal{X}})[\xi]\right)_{\uparrow X} = 2P_{BX}\left(A\xi_{\uparrow X} - B\xi_{\uparrow X}(X^T BX)^{-1}X^T AX\right) \ . \tag{4.22}$$

4.3.2 Analysis of Specialized solvers

This section analyzes the specialized eigensolvers from Section 4.2 in the context of the optimization of the generalized Rayleigh quotient over the Grassmann manifold.

Jacobi-Davidson Method

The connection between the Jacobi-Davidson eigensolver and Newton's method has been analyzed numerous times in the literature; see [SBFV96] and references therein. This section discusses an implementation of Newton-on-Grassmann for the generalized Rayleigh quotient which identically evokes the form of the simplified JD eigensolver.

Consider the choice of Riemannian metric and horizontal metric from Equations (4.10) and (4.11). This results in the equations for the Riemannian gradient and Riemannian Hessian as in Equations (4.14) and (4.15):

$$\begin{aligned}\text{grad} f(\mathcal{X})_{\uparrow X} &= 2B^{-1}P_{BX,X}AX \ , \\ (\text{Hess} f(\mathcal{X})[\eta])_{\uparrow X} &= 2B^{-1}P_{BX,X}\left(A\eta_{\uparrow X} - B\eta_{\uparrow X}(X^TBX)^{-1}X^TAX\right) \ .\end{aligned}$$

Solving the Newton equation then consists of finding $\eta_{\uparrow X}$ such that

$$2B^{-1}P_{BX,X}\left(A\eta_{\uparrow X} - B\eta_{\uparrow X}(X^TBX)^{-1}X^TAX\right) = -2B^{-1}P_{BX,X}AX \ ,$$

or, equivalently,

$$P_{BX,X}\left(A\eta_{\uparrow X} - B\eta_{\uparrow X}(X^TBX)^{-1}X^TAX\right) = -P_{BX,X}AX \ .$$

If we assume that X contains the Ritz values with respect to $\text{colsp}\,(X)$, then $X^TBX = I$ and $X^TAX = \Theta$ is diagonal. In this case, the Newton equation simplifies to

$$(I - BXX^T)\,(A\eta_{\uparrow X} - B\eta_{\uparrow X}\Theta) = -(AX - BX\Theta) \ .$$

Recalling the discussion in Section 4.2.1, it is clear that the simplified Jacobi-Davidson method is equivalent to a Newton-on-Grassmann for this choice of inner product and horizontal space and with retraction $R_{\text{colsp}(X)}(\eta) = \text{colsp}\,(X + \eta_{\uparrow X})$. Note that the Jacobi condition that the update is orthogonal to the current iterate is enforced here by the requirement that $\eta_{\uparrow X}$ is a member of the horizontal space $\mathcal{H}_X$.

Trace Minimization Method

In the examination of the Trace Minimization method, we will analyze the second Riemannian characterization of the generalized Rayleigh quotient minimization. Assuming the

Riemannian metric and horizontal distribution from Equations (4.10) and (4.11) and the retraction from Equation (4.19), then the pullback $\hat{f}$ has the following gradient and Hessian:

$$\begin{aligned}\left(\text{grad} f(\mathcal{X})\right)_{\uparrow X} &= 2P_{BX}AX \ , \\ \left(\text{Hess}\hat{f}_{\mathcal{X}}(0_{\mathcal{X}})[\eta]\right)_{\uparrow X} &= 2P_{BX}\left(A\eta_{\uparrow X} - B\eta_{\uparrow X}(X^TBX)^{-1}X^TAX\right) \ .\end{aligned}$$

Two possibilities exist for analyzing the Trace Minimization method. One approaches the method as a quasi-Newton method. A retraction-based Riemannian Newton method solves the equation

$$\left(\text{Hess}\hat{f}_{\mathcal{X}}(0_{\mathcal{X}})[\xi]\right)_{\uparrow X} = -\left(\text{grad} f(\mathcal{X})\right)_{\uparrow X}$$

and chooses the next iterate

$$X_+ = R_{\mathcal{X}}(\eta_{\uparrow X}) = X + \eta \ .$$

By substituting the Hessian of $\hat{f}_{\mathcal{X}}$ with approximate Hessian $2P_{BX}AP_{BX}$, the correction equation becomes

$$P_{BX}AP_{BX}\eta_{\uparrow X} = -P_{BX}AX \ .$$

This system is the same as Equation (4.9) solved in the Trace Minimization method. Furthermore, because TRACEMIN is only described for positive-definite A, this linear system is positive-definite for all vectors in $\mathcal{H}_X$. As a result, the system has the property that the Newton step is well-defined, a condition which requires extra effort by most quasi-Newton schemes.

Recall, however, that the authors of the TRACEMIN method established a global convergence theory for the method, which is not captured by the characterization of TRACEMIN as a quasi-Newton method. In order to capture this convergence theory in our Riemannian characterization of TRACEMIN, we shall reanalyze the method as an implementation of the IRTR method.

The IRTR method constructs a model of the pullback and minimizes this model subject to the implicit trust-region. The definition of the implicit trust-region is those tangent vectors η for which the decrease in the model is some specified fraction of the decrease in the lifted objective function, $\rho_{\mathcal{X}}(\eta) \geq \rho'$.

Recall the form of the model from Equation (3.2):

$$m_{\mathcal{X}}(\eta) = \hat{f}_{\mathcal{X}}(0_{\mathcal{X}}) + g_{\mathcal{X}}\left(\eta, \text{grad}\hat{f}_{\mathcal{X}}(0_{\mathcal{X}})\right) + \frac{1}{2}g_{\mathcal{X}}\left(\eta, H_{\mathcal{X}}[\eta]\right) \ .$$

Substituting the formulas for the generalized Rayleigh quotient yields the following:

$$\begin{aligned} m_{\mathcal{X}}(\eta) = &\operatorname{trace}\left((X^TBX)^{-1}X^TAX\right) + 2\operatorname{trace}\left((X^TBX)^{-1}\eta_{\uparrow X}^TAX\right) \\ &+ \operatorname{trace}\left((X^TBX)^{-1}\eta_{\uparrow X}^T\left(H_{\mathcal{X}}[\eta]\right)_{\uparrow\eta}\right) \ . \end{aligned}$$

The TRACEMIN method assumes that both A and B are positive-definite.[4] It remains to choose the operator that will act as the model Hessian. Consider the following:

$$\left(H_{\mathcal{X}}[\eta]\right)_{\uparrow X} = 2P_{BX}AP_{BX}\eta_{\uparrow X}.$$

This operator is symmetric/positive definite, hence the model admits a unique unconstrained minimizer. For simplicity, assume also that the basis X representing $\mathcal{X}$ is B-orthonormal. The implicit trust-region subproblem (3.5) now consists of the following:

$$\begin{aligned} &\text{minimize } \operatorname{trace}\left(X^TAX + 2\eta_{\uparrow X}^TP_{BX}AX + \eta_{\uparrow X}^TP_{BX}AP_{BX}\eta_{\uparrow X}\right), \\ &\text{such that } \eta_{\uparrow X}^TBX = 0 \text{ and } \rho_{\mathcal{X}}(\eta) \geq \rho'. \end{aligned} \tag{4.23}$$

If we neglect the trust-region requirement, then we are left with the following problem:

$$\begin{aligned} &\text{minimize } \operatorname{trace}\left(X^TAX + 2\eta_{\uparrow X}^TP_{BX}AX + \eta_{\uparrow X}^TP_{BX}AP_{BX}\eta_{\uparrow X}\right), \\ &\text{such that } \eta_{\uparrow X}^TBX = 0. \end{aligned} \tag{4.24}$$

This minimization problem is precisely that in Equation (4.8). In [SW82, ST00], the authors prove the following inequality:

$$\begin{aligned} \hat{f}_{\mathcal{X}}(\xi) &= \operatorname{trace}\left(\left(I + \xi_{\uparrow X}^TB\xi_{\uparrow X}\right)^{-1}(X+\xi_{\uparrow X})^TA(X+\xi_{\uparrow X})\right) \\ &\leq \operatorname{trace}\left((X+\xi_{\uparrow X})^TA(X+\xi_{\uparrow X})\right) \\ &= m_{\mathcal{X}}(\xi) \ . \end{aligned} \tag{4.25}$$

Recall from the model definition that $m_{\mathcal{X}}(0_{\mathcal{X}}) = \hat{f}_{\mathcal{X}}(0_{\mathcal{X}})$. Inserting this into Equation (4.25) yields the following:

$$\hat{f}_{\mathcal{X}}(0_{\mathcal{X}}) - \hat{f}_{\mathcal{X}}(\xi) \geq m_{\mathcal{X}}(0_{\mathcal{X}}) - m_{\mathcal{X}}(\xi) \ .$$

Then any $\xi \in T_{\mathcal{X}}\text{Grass}(p,n)$ produces at least as much decrease in the objective function as in the model. Returning to the context of the implicit trust-region, this means that ρ satisfies the following:

$$\rho_{\mathcal{X}}(\xi) = \frac{\hat{f}_{\mathcal{X}}(0_{\mathcal{X}}) - \hat{f}_{\mathcal{X}}(\xi)}{m_{\mathcal{X}}(0_{\mathcal{X}}) - m_{\mathcal{X}}(\xi)} \geq 1 \ .$$

[4] For what follows, A need strictly be only positive semi-definite.

As a result of this and the assumption that $\rho' \leq 1$, the implicit trust region (3.5) is the whole $T_{\mathcal{X}}\text{Grass}(p,n)$, and the solution of (4.24) is the unique solution of (4.23). In this way, the Trace Minimization method is equivalent to the Implicit Riemannian Trust-Region method for a particular choice of the model Hessian. One consequence of this is that the Trace Minimization method inherits the convergence analysis of the IRTR, in addition to that provided by its authors.

Note that because the entirety of the tangent plane lies inside the implicit trust-region and because the model Hessian is positive definite, the truncated CG Algorithm 7 will terminate only when it has sufficiently reduced the gradient of the model, i.e., the residual of the linear system, $H_{\mathcal{X}}[\xi] = -\text{grad} f(\mathcal{X})$. However, the global convergence Theorem 25 for the IRTR method requires only that the solution provide as much decrease in $m_{\mathcal{X}}$ as some fixed fraction of that provided by the local Cauchy point. The truncated CG algorithm generates the local Cauchy point on the first iterate, so that the algorithm need not solve the equation to a high-accuracy in order to yield superlinear convergence. Therefore, while the TRACEMIN method can be described as an inexact, quasi-Newton method, it has the added benefit that the implicit trust-region mechanism provides stable convergence only to local minimizers. Again, the convergence properties of the method were not lost on its authors.

However, an unfortunate consequence of the Hessian choice $(H_{\mathcal{X}}[\xi])_{\uparrow X} = 2P_{BX}AP_{BX}\xi_{\uparrow X}$ is that it does not adequately approximate the actual Hessian of $\hat{f}$. As a result, the method yields only a linear rate of asymptotic convergence. This result was known by the authors of TRACEMIN, due to the relationship between optimal TRACEMIN and the subspace iteration method; see [SW82] or [ST00].

LOBPCG Method

Analyzing the LOBPCG method in the context of Riemannian optimization requires coming to terms with the Rayleigh-Ritz process at the heart of the LOBPCG method. The authors of [AG06, AMS08] describe the Rayleigh-Ritz process as an example of an acceleration technique capable of augmenting Riemannian line-search or trust-region methods. This ability comes via the process's ability to find a minimizer of the generalized Rayleigh quotient subject to a larger subspace. The authors show that, under circumstances relating to the gradient information contained in the search subspaces, the acceleration mechanism (i.e., Rayleigh-Ritz) does not compromise the convergence properties.

In the case of the LOBPCG method, recall that a new iterate X_{k+1} is chosen as the Ritz vectors corresponding to the smallest Ritz values with respect to the subspace $\operatorname{colsp}\left(\begin{bmatrix} X_k & H_k & (X_k - X_{k-1}) \end{bmatrix}\right)$. This is done by choosing X_{k+1} as

$$X_{k+1} = \begin{bmatrix} X_k & H_k & (X_k - X_{k-1}) \end{bmatrix} W \ ,$$

where W are the primitive Ritz vectors. These vectors W have the benefit of making X_{k+1} into Ritz vectors; this is a particular choice of basis with some nice properties, but any other basis for the same subspace has the same (minimal) value under the generalized Rayleigh quotient. Re-normalize the vectors W instead so that they takes the form

$$W = \begin{bmatrix} I \\ W_2 \\ W_3 \end{bmatrix} \ .$$

This can be done so long as $X_k^T X_{k+1}$ is non-singular.

Then the next iterate has the form

$$X_{k+1} = \begin{bmatrix} X_k & H_k & (X_k - X_{k-1}) \end{bmatrix} W = X_k + H_k W_2 + (X_k - X_{k-1}) W_3 \ ,$$

where the coefficients in W_2 and W_3 are chosen by LOBPCG to optimize the generalized Rayleigh quotient over the "2-D" subspace spanning H_k and $(X_k - X_{k-1})$.

This can be related to a Riemannian optimization strategy. This strategy operates by using the retraction $R_{\operatorname{colsp}(X)}(\eta) = \operatorname{colsp}(X + \eta_{\uparrow X})$, so that the lifted cost function is

$$\hat{f}_{\operatorname{colsp}(X)}(\eta) = f(R_{\operatorname{colsp}(X)}(\eta)) = \operatorname{GRQ}(X + \eta_{\uparrow X}) \ .$$

The analogy with LOBPCG is made by choosing the tangent vector η as the minimizer of $\hat{f}_{\operatorname{colsp}(X)}$ in the "two dimensional" subspace spanned by the preconditioned gradient and the tangent vector $R^{-1}_{\operatorname{colsp}(X_k)}(X_{k-1})$. The ability to choose the optimal vector in this subspace is possible in this case because of the properties of the Rayleigh-Ritz process.

4.3.3 RTR and the ESGEV Problem

For the RTR implementation of the ESGEV problem, we will use the following characterization of the problem, as with TRACEMIN:

$$
\begin{aligned}
g_{\mathcal{X}}(\xi,\zeta) &= \operatorname{trace}\left((X^TBX)^{-1}\xi_{\uparrow X}^T\zeta_{\uparrow X}\right) \ , \\
\mathcal{H}_X &= \left\{Z \in \mathbb{R}^{n\times p} : Z^TBX = 0\right\} \ , \\
P_X^h = P_{BX} &= I - BX(X^TB^2X)^{-1}X^TB \ , \\
(\operatorname{grad} f(\mathcal{X}))_{\uparrow X} &= 2P_{BX}AX \ , \\
R_{\mathcal{X}}(\xi) &= \operatorname{colsp}(X + \xi_{\uparrow X}) \ , \\
\left(\operatorname{Hess}\hat{f}_{\mathcal{X}}(0_{\mathcal{X}})[\eta]\right)_{\uparrow X} &= 2P_{BX}\left(A\eta_{\uparrow X} - B\eta_{\uparrow X}(X^TBX)^{-1}X^TAX\right) \ .
\end{aligned}
$$

Choosing the iterates X as Ritz vectors allows some simplifications: $X^TBX = I$ somewhat simplifies the formula for the Riemannian metric and the Hessian of the pullback. Also, X^TAX is a diagonal matrix containing the Ritz values; this also simplifies the application of the Hessian operator. Substituting these equations into the RTR/tCG algorithm (Algorithm 4/5) and using standard linear algebra notation results in the RTR/tCG ESGEV algorithms (Algorithm 14/15).

It should be noted that, as before, the trust-region is evaluated using the norm derived from the preconditioner N (e.g., Line 13). In general, preconditioned eigensolvers can assume only the ability to apply N^{-1}; this operation may be the result of an iterative linear solve or a multi-level preconditioner, operations which do not easily admit inversion. As suggested in Section 2.2, recurrences from [CGT00, Section 7.5.1] allow the ability to compute the N-norm of all necessary terms via the available quantities, though these recurrences have been neglected from the algorithmic listing for the sake of brevity. However, one consequence of this approach is that it requires that the subproblem iteration be initialized to zero, preventing a random initialization which can be useful to guarantee escape even when initialized to a numerical critical point. The IRTR implementation for the ESGEV problem does not employ an explicit trust-region and therefore does not suffer from this drawback.

Implementing RTR/ESGEV

Line 3 of Algorithm 14 evaluates the improvement ratio, ρ. Substituting the model and objective function for the ESGEV problem into the definition of ρ (Equation 2.7) yields the

Algorithm 14 RTR/ESGEV

Require: Symmetric A, s.p.d. B, of dimension $n \times n$

Input: B-orthonormal matrix X_0 containing p Ritz vectors, diagonal matrix Θ_0 containing p Ritz values, $\bar{\Delta} > 0$, $\Delta_0 \in (0, \bar{\Delta})$, and $\rho' \in (0, \frac{1}{4})$

1: **for** $k = 0, 1, \ldots$ until convergence **do**
2: {Model-based minimization}
3: Approximately solve for S_k the model minimization using Algorithm 15:

$$\begin{aligned} \text{minimize} \quad & \operatorname{trace}\left(X_k^T A X_k\right) + 2\operatorname{trace}\left(S^T A X_k\right) + \operatorname{trace}\left(S^T A S - S^T B S \Theta_k\right) \\ \text{subject to} \quad & X^T B S = 0 \ . \end{aligned}$$

4: Evaluate $\rho_k = \rho_{X_k}(S_k)$
5: {Adjust trust region}
6: **if** $\rho_k < \frac{1}{4}$ **then**
7: Set $\Delta_{k+1} = \frac{1}{4}\Delta_k$
8: **else if** $\rho_k > \frac{3}{4}$ and $\|\eta_k\| = \Delta_k$ **then**
9: Set $\Delta_{k+1} = \min(2\Delta_k, \bar{\Delta})$
10: **else**
11: Set $\Delta_{k+1} = \Delta_k$
12: **end if**
13: {Compute next iterate}
14: **if** $\rho_k > \rho'$ **then**
15: Set X_{k+1} as the Ritz vectors with respect to $X_k + S_k$, with Ritz values in Θ_{k+1}
16: **else**
17: Preserve $X_{k+1} = X_k$, $\Theta_{k+1} = \Theta_k$
18: **end if**
19: **end for**

Output: k Ritz pairs (θ_j, x_j), $j = 1, \ldots, p$

following:

$$\begin{aligned} \rho_X(S) &= \frac{\operatorname{GRQ}(X) - \operatorname{GRQ}(X+S)}{\operatorname{GRQ}(X) - \left(\operatorname{GRQ}(X) + 2\operatorname{trace}\left(S^T A X\right) + \operatorname{trace}\left(S^T A S + S^T B S \Theta\right)\right)} \\ &= \frac{\operatorname{trace}\left(\Theta\right) - \operatorname{trace}\left((I + S^T B S)^{-1}(X+S)^T A (X+S)\right)}{\operatorname{trace}\left(S^T A S + S^T B S \Theta\right) - 2\operatorname{trace}\left(S^T A X\right)} \ . \end{aligned}$$

Because the terms in the equation require only the trace of a matrix, most of the matrices in question do not need be formed explicitly, allowing some savings with respect to floating point operations. For example, given S and AS, forming the matrix $S^T(AS)$ requires $O(np^2)$, whereas computing trace $\left(S^T(AS)\right)$ requires only $O(np)$.

Furthermore, it is also possible to avoid the applications of the operators A and B to the update S needed to evaluate $\rho_X(S)$. This is done by returning a copy of AS and BS from

Algorithm 15 Preconditioned Truncated CG for RTR/ESGEV

Require: Symmetric A, s.p.d. B, s.p.d. preconditioner N^{-1}

Input: Ritz vectors $X \in \mathbb{R}^{n \times p}$ with Ritz values Θ; trust-region radius Δ; convergence criteria $\kappa \in (0, 1)$, $\theta > 0$

1: Set $S_0 = 0$, $R_0 = 2P_{BX}AX$, $Z_0 = N^{-1}R_0$, $D_0 = -Z_0$
2: **for** $j = 0, 1, 2, \ldots$ **do**
3: **if** $\|R^j\| \leq \|R_0\| \min\left\{\kappa, \|R_0\|^\theta\right\}$ **then**
4: **return** S^j
5: **end if**
6: Compute $W^j = P_{BX}\left(AD^j - BD^j\Theta\right)$
7: Set $\alpha = \operatorname{trace}\left((Z^j)^T R^j\right) / \operatorname{trace}\left((W^j)^T D^j\right)$
8: **if** $\operatorname{trace}\left((W^j)^T D^j\right) \leq 0$ or $\operatorname{trace}\left((S^j + \alpha D^j)^T N (S^j + \alpha D^j)\right) > \Delta^2$ **then**
9: Compute $\tau > 0$ such that $S = S^j + \tau D^j$ satisfies $\operatorname{trace}\left(S^T N S\right) = \Delta^2$
10: **return** S
11: **end if**
12: Set $S^{j+1} = S^j + \alpha D^j$
13: Set $R^{j+1} = R^j + \alpha W^j$
14: Set $Z^{j+1} = N^{-1}R^{j+1}$
15: Set $\beta = \operatorname{trace}\left((Z^{j+1})^T R^{j+1}\right) / \operatorname{trace}\left((Z^j)^T R^j\right)$
16: Set $D^{j+1} = -Z^{j+1} + \beta D^j$
17: **end for**

Output: Matrix S satisfying $S^T BX = 0$ and Cauchy decrease under m_X.

Algorithm 15. Each update to the iterate S (Line 12) is a linear combination of the previous iterate and the matrix D. If the image of D under A and B is known, then we can easily update the cached copy of AS and BS as well:

$$\begin{aligned} S^{j+1} &= S^j + \alpha D^j \\ (AS)^{j+1} &= (AS)^j + \alpha (AD)^j \\ (BS)^{j+1} &= (BS)^j + \alpha (BD)^j \ . \end{aligned}$$

Because the quantity AD^j and BD^j are required for the application of the Hessian to D^j, these can be cached and used in the computation of S^{j+1}.

This approach trades off the additional storage for the cached results and additional floating point operations required to maintain them, against the number of times that the operators A and B must be applied. Caching and maintaining AD, BD, AS and BS means that AS and BS need not be computed on the outer iteration. Furthermore, if AX and BX are cached as well, then $A(X + S)$ and $B(X + S)$ can be computed without applying

the operators. However, the savings in operator applications occur once per outer iteration, while the additional memory requirements are continuous and the maintenance of the caches increases the floating point cost of each inner iteration. For inexpensive operators and/or memory constrained problems, the caching implementation will be less effective. Table 4.1 lists the memory requirements associated with both approaches. We refer to the versions with caching as "hefty" and to the version without caching as "skinny". Note, these memory requirements are the same as for the IRTR/ESGEV algorithm.

Table 4.1: Memory cost in $n \times p$ multivectors for caching ("Hefty") and non-caching ("Skinny") versions of RTR/ESGEV and IRTR/ESGEV (Algorithm 14 and Algorithm 16). **B** denotes that storage for the variable is required only if $B \neq I$, whereas **N** denotes that storage for the variable is required only if $N \neq I$.

Variable	X	AX	BX	S	AS	BS	R	Z	D	AD	BD	W	**Total**
Skinny	yes		**B**	yes			yes	yes	yes			yes	6–7
Hefty	yes	yes	**B**	yes	yes	yes	yes	**N**	yes	yes	yes	yes	10–12

In practice, it may not be known ahead of time which implementation—skinny or hefty—is the most efficient. Given a problem of dimension n and a block size p, the amount of storage required by the solver can easily be deduced, as can the number of floating point operations. However, the cost of the operator A and B applications relative to the floating point operation depends on a number of factors that may be difficult to anticipate *a priori*, including effects of memory hierarchy, the cost of opaque operators (perhaps resulting from a spectral transformation), and the average number of inner iterations per outer iteration. One solution is that an implementation may use timing information to switch to the more efficient of the two solver weights. By recording the amount of time necessary to perform the caching and weighing this amount against the time needed to apply the operators, a more intelligent solver may dynamically choose the more efficient implementation.

Another issue involved in the efficient implementation of the RTR/ESGEV regards the selection of the solver parameters: the trust-region acceptance parameter ρ', the maximum trust-region radius $\bar{\Delta}$, and the initial trust-region radius Δ_0. There is currently no suggestion for choosing ρ'; a study analyzing how this parameter affects the performance of the method is appropriate for the body of future work.

The upcoming analysis of ρ in Section 4.3.4 could be useful for choosing an initial radius Δ_0. Unfortunately, this seems only useful when there is no preconditioner. In the case where a preconditioner is used, the truncated CG uses the preconditioner induced norm for evaluating trust-region membership. As a results, the most effective choice of trust-region radius will depend on the preconditioner, information that is not likely to be available *a priori*.

The situation is a little better for $\bar{\Delta}$. The convergence theory requires only that $\bar{\Delta}$ is finite. Setting the value too small will limit the performance of the algorithm, and a "large enough" value is difficult to determine due to the influence of the preconditioner. In practice, there have been no ill-effects from setting $\bar{\Delta}$ to infinity.

Section 2.3.2 employed a convergence criterion for the inner iteration which seeks a linear reduction in the model gradient early on and a superlinear reduction asymptotically. This works because as the outer iteration approaches a stationary point, the norm of grad $f(x)$ approaches zero. The residual of the model is the same as the objective function, so that as it becomes smaller, the θ term of the stopping condition becomes active:

$$\|r_0\| \min \left\{ \kappa, \|r_0\|^\theta \right\} \ .$$

It is clear that the algorithm seeks θ-convergence (i.e., superlinear convergence) whenever $|r_0| = |\text{grad}\, f(x) \leq 1$. While this mechanism works fine for the purpose of analyzing the asymptotic converge, it may not be relevant in a finite-precision implementation. The reason is that the progress of the algorithm may stagnate or the algorithm may terminate before reaching the domain of θ-convergence. Alternatively, the scaling of the problem may be such that the superlinear component dominates the stopping condition, so that the algorithm seeks θ-convergence from the outset.

It is not immediately clear whether this situation requires treatment. However, in the case that one wishes the algorithm to attempt asymptotic convergence, one possibility is to scale the term in the superlinear component of the stopping condition:

$$\|r_0\| \min \left\{ \kappa, \left(\frac{\|r_0\|}{\gamma} \right)^\theta \right\} \ .$$

This requires choosing some scalar γ. Values $\gamma > 1$ encourage an earlier entry into the θ-convergence domain, while values $\gamma < 1$ encourage a later entry into this domain. Choices

for γ might consider the norm of the gradient of the initial outer iterate and the threshold for the outer stopping condition.

It should also be noted that superlinear convergence does entail some additional cost. Pursuing a smaller decrease in the model minimization increases the possibility of over-convergence in the outer iteration. Techniques for preventing this are: to avoid expending a large number of inner iterations before returning (at which time, the outer stopping condition is evaluated); or to monitor the outer stopping condition during the model minimization, terminating early in the case that it is satisfied.

Finally, another issue that must be addressed concerns the method for preconditioning the model minimization. It is customary in iterative eigensolvers to use an approximate inverse of the A matrix as a preconditioner. In the case of the model minimization for the RTR/ESGEV and IRTR/ESGEV solvers using the truncated CG solver, the preconditioner is required to be a symmetric/positive-definite operator on the tangent space. There are two approaches that are commonly used.

Given a symmetric/positive-definite operator $N^{-1} \approx A^{-1}$, on option is to apply the following preconditioner:

$$P_{BX,BX} N^{-1} P_{BX,BX} \ .$$

Note that the latter of these projectors does not need to be applied, as the input to this preconditioner should always be a vector in the tangent space. Note that this preconditioner is symmetric, in addition to being positive-definite for all vectors in the tangent plane.

Another possibility is to precondition the model minimization by solving the following system:

$$P_{BX,BX} N P_{BX,BX} \xi = \zeta \ ,$$

where ξ and ζ are both tangent vectors. In [OJS90], the authors show that this system can be solved (in a least-squares sense) as follows:

$$\xi = P_{N^{-1}BX,BX} N^{-1} \zeta \ .$$

At first glance, this preconditioner seems to require two applications of the operator N^{-1}. Note, however, that one of these applications produces $N^{-1}BX$. Because the value of X—and therefore, the value of BX—does not change during the model minimization, this product can be computed at the beginning of the model minimization and stored. However,

this does require a multivector for storing the product $N^{-1}BX$, in addition to the required storage listed in Table 4.1.

4.3.4 IRTR and the ESGEV Problem

The IRTR implementation for the ESGEV problem will use the same characterization as for the RTR implementation:

$$
\begin{aligned}
g_{\mathcal{X}}(\xi,\zeta) &= \operatorname{trace}\left((X^TBX)^{-1}\xi_{\uparrow X}^T\zeta_{\uparrow X}\right) \ , \\
\mathcal{H}_X &= \left\{Z \in \mathbb{R}^{n\times p} : Z^TBX = 0\right\} \ , \\
P_X^h = P_{BX} &= I - BX(X^TB^2X)^{-1}X^TB \ , \\
\left(\operatorname{grad} f(\mathcal{X})\right)_{\uparrow X} &= 2P_{BX}AX \ , \\
R_{\mathcal{X}}(\xi) &= \operatorname{colsp}\left(X + \xi_{\uparrow X}\right) \ , \\
\left(\operatorname{Hess}\hat{f}_{\mathcal{X}}(0_{\mathcal{X}})[\eta]\right)_{\uparrow X} &= 2P_{BX}\left(A\eta_{\uparrow X} - B\eta_{\uparrow X}(X^TBX)^{-1}X^TAX\right) \ .
\end{aligned}
$$

An efficient implementation of the implicit RTR requires an understanding of the improvement ratio ρ, repeated here:

$$\rho_{\mathcal{X}}(\xi) = \frac{\hat{f}_{\mathcal{X}}(0_{\mathcal{X}}) - \hat{f}_{\mathcal{X}}(\xi)}{m_{\mathcal{X}}(0_{\mathcal{X}}) - m_{\mathcal{X}}(\xi)},$$

where $m_{\mathcal{X}}$ is the quadratic model chosen to approximate $\hat{f}_{\mathcal{X}}$:

$$m_{\mathcal{X}}(\xi) = \hat{f}_{\mathcal{X}}(0_{\mathcal{X}}) + g_{\mathcal{X}}\left(\operatorname{grad}\hat{f}_{\mathcal{X}}(0_{\mathcal{X}}),\xi\right) + \frac{1}{2}g_{\mathcal{X}}\left(H_{\mathcal{X}}[\xi],\xi\right). \tag{4.26}$$

The Trace Minimization method was shown to satisfy an implementation of the IRTR method for a particular choice of model Hessian:

$$\left(H_{\mathcal{X}}\right)_{\uparrow X} = P_{BX}AP_{BX} \ .$$

In the case where A is positive-definite, this allows for a very strong statement to be proven regarding ρ, i.e., that $\rho_X(S) \geq 1$. Unfortunately, this approach suffers from a linear rate of convergence. The IRTR/ESGEV approach described in this section will use the Hessian of the pullback as the model Hessian, in order to facilitate higher rates of convergence.

Consider the case where the quadratic model $m_{\mathcal{X}}$ is chosen as the *Newton model*, i.e., the quadratic Taylor expansion of $\hat{f}_{\mathcal{X}}$:

$$m_{\mathcal{X}}(\xi) = \hat{f}_{\mathcal{X}}(0_{\mathcal{X}}) + g_{\mathcal{X}}\left(\operatorname{grad}\hat{f}_{\mathcal{X}}(0_{\mathcal{X}}),\xi\right) + \frac{1}{2}g_{\mathcal{X}}\left(\operatorname{Hess}\hat{f}_{\mathcal{X}}(0_{\mathcal{X}})[\xi],\xi\right).$$

We wish to perform an analysis of $\rho_{\mathcal{X}}$ for the Newton model just as we did for the TRACEMIN model. We will forgo the Riemannian optimization notation in favor of standard linear algebra notation. Assume as before that a subspace $\mathcal{X}$ is represented by a B-orthonormal basis X, i.e. $X^TBX = I$. Take a tangent vector represented by an $n \times p$ matrix S, $S^TBX = 0$. Consider the denominator of $\rho_X(S)$:

$$\begin{aligned} m_X(0) - m_X(S) &= -2\,\mathrm{trace}\left(S^TAX\right) - \mathrm{trace}\left(S^TAS - S^TBSX^TAX\right) \\ &= \mathrm{trace}\left(S^TBSX^TAX - 2S^TAX - S^TAS\right) \\ &= \mathrm{trace}\left(\hat{M}\right), \end{aligned} \tag{4.27}$$

for $\hat{M} \doteq S^TBSX^TAX - 2S^TAX - S^TAS$. Consider the numerator:

$$\begin{aligned} \hat{f}_X(0) - \hat{f}_X(S) &= \mathrm{GRQ}(X) - \mathrm{GRQ}(X+S) \\ &= \mathrm{trace}\left(X^TAX - \left(I + S^TBS\right)^{-1}(X+S)^TA(X+S)\right) \\ &= \mathrm{trace}\left(\left(I + S^TBS\right)^{-1}\left(S^TBSX^TAX - 2S^TAX - S^TAS\right)\right) \\ &= \mathrm{trace}\left((I + S^TBS)^{-1}\hat{M}\right). \end{aligned} \tag{4.28}$$

Combining equations (4.27) and (4.28) allows $\rho_X(S)$ to be written as follows:

$$\rho_X(S) = \frac{\mathrm{trace}\left((I + S^TBS)^{-1}\hat{M}\right)}{\mathrm{trace}\left(\hat{M}\right)}. \tag{4.29}$$

Note that in the specific case of $p = 1$, i.e., the solution for a single eigenpair, the Equation (4.29) simplifies to

$$\rho_x(s) = \frac{1}{1 + s^TBs}.$$

This formula provides both of the ingredients for an efficient implementation of Algorithm 6/7: a trivial evaluation of $\rho_x(s)$, and an efficient search along s for some $\rho_x(ts) = \rho'$.

$p > 1$: The Hard Case

Unfortunately, for $p > 1$, Equation (4.29) currently defies efficient evaluation and/or search. Here we present a heuristic solution to this problem.

Assume that the matrix X contains Ritz vectors, such that $X^TBX = I$ and that $X^TAX = \Theta$ is a diagonal matrix containing the associated Ritz values. Also assume that

$$s_j^TBs_j \leq \Delta^2, \quad j = 1, \ldots, p \tag{4.30}$$

where

$$\Delta^2 = \frac{1}{\rho'} - 1 \ . \tag{4.31}$$

As a result of X being Ritz vectors, the model m_X can be decoupled into p individual models:

$$\begin{aligned} m_X(S) &= \text{trace}\left(X^T A X\right) + 2\,\text{trace}\left(S^T A X\right) + \text{trace}\left(S^T A S - S^T B S X^T A X\right) \\ &= \text{trace}\left(X^T A X\right) + 2\,\text{trace}\left(S^T A X\right) + \text{trace}\left(S^T A S - S^T B S \Theta\right) \\ &= \sum_{j=1}^{p} x_j^T A x_j + 2 s_j^T A x_j + s_j^T A s_j - s_j^T B s_j \theta_j \\ &= \sum_{j=1}^{p} m_{x_j}(s_j) \ . \end{aligned}$$

We desire a lower bound on $\rho_X(S)$, in order to guarantee that $\rho_X(S) \geq \rho'$ and verify that S is in the implicit trust-region. However, there currently is not a simple formula for $\rho_X(S)$. Consider instead the function $\hat{\rho}_X$:

$$\hat{\rho}_X(S) \doteq \frac{\hat{f}_X(0) - \sum_{j=1}^{p} \hat{f}_{x_j}(s_j)}{m_X(0) - m_X(S)} \approx \frac{\hat{f}_X(0) - \hat{f}_X(S)}{m_X(0) - m_X(S)} = \rho_X(S) \ .$$

Then assuming (4.30) and $m_{x_j}(s_j) < m_{x_j}(0)$ guarantees that each s_j satisfies the following:

$$\rho_{x_j}(s_j) = \frac{\hat{f}_{x_j}(0) - \hat{f}_{x_j}(s_j)}{m_{x_j}(0) - m_{x_j}(s_j)} = \frac{1}{1 + s_j^T B s_j} \geq \rho' \ ,$$

or, equivalently,

$$\hat{f}_{x_j}(0) - \hat{f}_{x_j}(s_j) \geq \rho' \left(m_{x_j}(0) - m_{x_j}(s_j)\right) \ .$$

Substituting this into the formula for $\hat{\rho}_X$ yields

$$\begin{aligned} \hat{\rho}_X(S) &= \frac{\hat{f}_X(0) - \sum_{j=1}^{p} \hat{f}_{x_j}(s_j)}{m_X(0) - m_X(S)} \\ &= \frac{\sum_{j=1}^{p} \left(\hat{f}_{x_j}(0) - \hat{f}_{x_j}(s_j)\right)}{\sum_{j=1}^{p} \left(m_{x_j}(0) - m_{x_j}(s_j)\right)} \\ &\geq \frac{\rho' \sum_{j=1}^{p} \left(m_{x_j}(0) - m_{x_j}(s_j)\right)}{\sum_{j=1}^{p} \left(m_{x_j}(0) - m_{x_j}(s_j)\right)} \\ &= \rho' \ . \end{aligned}$$

This heuristic, approximating $\rho_X(S)$ by $\hat{\rho}_X(S)$, suggests an eigensolver approach based on IRTR for the ESGEV problem. Algorithms 16 and 17 list this approach. The method works by using the A-orthogonality of the Ritz vectors to decouple the model minimization into p separate models of rank one. These models are solved using an Implicit IRTR approach, guaranteeing a specified accuracy of the models with respect to the decoupled pullbacks. Solving these models independently seems necessary, as it guarantees that each model m_{x_j} sees a decrease under s_j; this is necessary for showing that $\hat{\rho}_X(S) \geq \rho'$ and is not necessarily the case with a coupled solution as in Algorithm 15. The cumulative solution of the models are used to update the current iterate. Furthermore, in the case that $p = 1$, the heuristic is exact, and the Algorithm 16 is exactly an implementation of the IRTR for the ESGEV problem.

Algorithm 16 IRTR/ESGEV

Require: Symmetric A, s.p.d. B, of dimension $n \times n$

Input: B-orthonormal matrix X_0 containing p Ritz vectors, diagonal matrix Θ_0 containing p Ritz values, and $\rho' \in (0, 1)$

1: **for** $k = 0, 1, \ldots$ until convergence **do**

2: {Model-based minimization}

3: Approximately solve for S_k the model minimization using Algorithm 17:

$$\begin{aligned} \text{minimize} \quad & \operatorname{trace}\left(X_k^T A X_k\right) + 2\operatorname{trace}\left(S^T A X_k\right) + \operatorname{trace}\left(S^T A S - S^T B S \Theta_k\right) \\ \text{subject to} \quad & X^T B S = 0 \ . \end{aligned}$$

4: {Compute next iterate}

5: Set X_{k+1} as the Ritz vectors with respect to $X_k + S_k$, with Ritz values in Θ_{k+1}

6: **end for**

Output: k Ritz pairs (θ_j, x_j), $j = 1, \ldots, p$

This approach has not been shown to satisfy the theoretical requirements of the IRTR, which insist that $\rho_X(S) \geq \rho'$. For instances of X and S where

$$\sum_{j=1}^{p} \hat{f}_{x_j}(s_j) < \hat{f}_X(S) \ ,$$

it is easily shown that $\hat{\rho}_X(S) < \rho_X(S)$. Note that while there are some examples where the value $\rho_X(S)$ is less than ρ', the numerical experiments in Section 4.4 suggest that the violations are minor. Furthermore, as long as there is some fixed constant $c > 0$ such that $\rho_{\mathcal{X}}(\eta) \geq c\rho'$, then the convergence theory of IRTR (namely, Theorem 25) still holds. The

Algorithm 17 Preconditioned Synchronized Truncated CG for IRTR/ESGEV

Require: Symmetric A, s.p.d. B, s.p.d. preconditioner N^{-1}

Input: Ritz vectors $X \in \mathbb{R}^{n\times p}$ with Ritz values Θ; trust-region radius Δ; convergence criteria $\kappa \in (0,1)$, $\theta > 0$

Set $\Delta^2 = 1/\rho' - 1$
Set $S^{(0)} = 0$, $R^{(0)} = 2P_{BX}AX$, $Z^{(0)} = N^{-1}R^{(0)}$, $D^{(0)} = -Z^{(0)}$
for $j = 0, 1, 2, \ldots$ **do**
  if $\|R^j\| \leq \|R^{(0)}\| \min\left\{\kappa, \|R^{(0)}\|^\theta\right\}$ **then**
    return S^j
  end if
  Compute $W^j = P_{BX}\left(AD^j - BD^j\Theta\right)$
  for $i = 1, \ldots, p$ **do**
    Set $\alpha_i = (z_i^j)^T r_i^j / (w_i^j)^T d_i^j$
  end for
  Let $\mathcal{I}$ contain the indices i such that α_i is negative or $(s_i^j + \alpha_i d_i^j)^T B(s_i^j + \alpha_i d_i^j) > \Delta^2$
  if $\mathcal{I}$ is not empty **then**
    for $i \in \mathcal{I}$ **do**
      Compute $\tau > 0$ such that $(s_i^j + \tau d_i^j)^T B(s_i^j + \tau d_i^j) = \Delta^2$
      Set $\alpha_i = \tau$
    end for
  end if
  Set $S^{j+1} = S^j + D^j \operatorname{diag}(\alpha_1, \ldots, \alpha_p)$
  if $\mathcal{I}$ is not empty **then**
    return S^{j+1}
  end if
  Set $R^{j+1} = R^j + W^j \operatorname{diag}(\alpha_1, \ldots, \alpha_p)$
  Set $Z^{j+1} = N^{-1}R^{j+1}$
  for $i = 1, \ldots, p$ **do**
    Set $\beta_i = (z_i^{j+1})^T r_i^{j+1} / (z_i^j)^T r_i^j$
  end for
  Set $D^{j+1} = -Z^{j+1} + D^j \operatorname{diag}(\beta_1, \ldots, \beta_p)$
end for

Output: Matrix S satisfying $S^T BX = 0$ and local Cauchy decrease under m_X.

other theoretical hurdle comes from the requirement that the decrease under m_X provided by this approach is some fixed fraction of the decrease produced by the local Cauchy point. Because each individual model m_{x_j} sees at least the Cauchy decrease and the cumulative model m_X is simply the sum of the local models, it should be fairly easy to demonstrate that the fractional decrease can be demonstrated for m_X. The issue there is negotiating the discrepancy between the individual implicit trust-regions (which are well understood)

and the cumulative implicit trust-region (which is less understood). Future work will be concerned with rigorously establishing these necessary results.

It bears noting that the solver approach in IRTR/ESGEV, like that in RTR/ESGEV, has many similarities to the Jacobi-Davidson method. In particular, there are striking similarities between IRTR/ESGEV and an approach suggested in [Not02]. There, Notay developed an analysis which (inexpensively) provides knowledge of the residual of the outer (eigenvalue) iteration based on the conjugate gradient coefficients used to solve the Jacobi-Davidson correction equation. Notay suggests exploiting this information as a stopping criterion for the inner iteration. His suggestion involves stopping the inner iteration when the marginal decrease in the outer residual norm is less than some fraction of the marginal decrease in the inner residual norm. The implicit trust-region, on the other hand, is comprised of strictly those points where the decrease under the objective function is at least some fraction of the decrease of the quadratic model. In this regard, both approaches strive to stop the inner iteration when it becomes inefficient or irrelevant with regard to the outer iteration, though the IRTR does this in such a way as to encourage global convergence to a minimizer.

Implementing IRTR/ESGEV

The IRTR removes much of the logic pertaining to the classical trust-region mechanism, specifically, the update of the trust-region radius and the accept/reject mechanism. As a result, the IRTR method is generally simpler to implement than a RTR method. In the case of IRTR/ESGEV, some of this simplicity is compromised by the decoupled model minimization routine (Algorithm 17).

Computationally, the routines are still very similar. The caching techniques suggestion in Section 4.3.3 can also be utilized to reduce the number of applications of A and B. The cost of the outer iteration is somewhat reduced by the relaxation of the need to evaluate $\rho_X(S)$. The cost of the inner iteration is very similar. Whether implementing a cached ("Hefty") or a non-cached ("Skinny") version of the algorithm, the memory requirements are the same as for the RTR/ESGEV. These requirements are listed in Table 4.1.

Algorithm 17 specifies a synchronized version of the truncated CG algorithm, operating at once on p decoupled models of rank one. This is not necessary for the theory of the method. However, by keeping these methods in lockstep, as opposed to solving one models before moving on to the next, one benefit is that the application of the operators A and B

and N become block operations, which may (depending on the structure of these operators) see a performance benefit. The drawback is that stopping all iterations when one element hits the trust-region prevents the rest from making any further progress.

4.3.5 A Note on Apparent Similarities

One significant difference between TRACEMIN and the RTR/tCG and IRTR/tCG algorithm described in Chapter 3 involves the solution of the model subproblems. This difference illustrates an easily overlooked characteristic of the IRTR/truncated CG approach.

Consider the case where the IRTR and RTR methods are applied to the model:

$$m_X(S) = \text{trace}\left(X^T AX\right) + 2\ \text{trace}\left(S^T AX\right) + \text{trace}\left(S^T AS\right)$$

The solution to the model minimization can be written as the solution to the following system of simultaneous linear equations:

$$P_{BX} A P_{BX} \eta_{\uparrow X} = -P_{BX} AX.$$

The TRACEMIN method approaches this problem by independently solving each of these equations. In contrast, the RTR and IRTR algorithms, combined with the truncated CG model subproblem solvers, may not exploit the structure present in this problem, by failing to note the independence of these equations. A straightforward implementation of RTR/tCG for the ESGEV problem resulting in Algorithm 14 will solve this equation in a coupled fashion. Because the solutions of the equations are decoupled, either the TRACEMIN approach or the tCG approach will find the same result when solved to completion. However, if the model subproblem is only solved approximately, the difference between the two approaches (coupled solve for RTR/ESGEV and decoupled solve for TRACEMIN) will result in different result, even if both methods employ a conjugate gradient iteration.

This is not to say that our knowledge of the problem could not be injected into the solvers. Indeed, the departure of the IRTR/ESGEV from the generic IRTR/tCG approach is a result of the decision to decouple the solution of the model minimization. Recall that this decision was made in order to utilize the much simpler $p = 1$ formula in the approximation of $\rho_X(S)$. However, this decoupling could be inserted into the RTR/ESGEV subproblem solver as well. This is one topic deserving further investigation, as the decoupled solver may allow increased efficiency.

4.3.6 Adaptive Model IRTR/ESGEV

Section 4.3.2 showed that the Trace Minimization method can be interpreted as an implementation of the IRTR for a particular choice of Hessian. Such a method has some strengths. It was shown earlier that the method can be implemented without monitoring the implicit trust-region, as $\rho_X(S) \geq 1$ for all tangent vectors S. Furthermore, due to the link with the method of inverse iterations, the TRACEMIN method quickly purges the eigenvectors whose eigenvalues are well separated from the p leftmost eigenvalues. This is particularly true when a good preconditioner is available, as is often the case in the very sparse problems encountered in structural mechanics. Consequently, the Basic Tracemin iteration is efficient when the iterates are still far away from the solution. However, as discussed earlier, one drawback is that TRACEMIN is limited to a linear rate of asymptotic convergence. This results in significant computational exertion when the eigenpairs are required to satisfy even moderate accuracy.

In [ABGS05], a hybrid approach is suggested. The authors propose using the TRACEMIN method early in the iteration, so that its strengths may be exploited. After some period, when the iteration has approached a neighborhood of the minimizer, the algorithm should switch to a solver capable of superlinear convergence. Many such solvers are candidates. The authors propose using the RTR/ESGEV method in the second stage. The reasons are two-fold. First, in the case that the iteration has not moved sufficiently close to the minimizer, the method utilized in the second stage should be a globally convergent method. Another reason is that the similarity between the two solvers (due to their shared heritage of model-based solvers) allows for a significant reuse of algorithmic investment.

Here, we propose using instead the IRTR/ESGEV solver in the second stage. This pairing, TRACEMIN followed by IRTR/ESGEV, has all of the strengths of the previous suggestion: the initial efficiency of TRACEMIN followed by the safe but fast asymptotic convergence of RTR/ESGEV. Replacing RTR/ESGEV by IRTR/ESGEV in this hybrid has some additional benefits. One benefit is that the algorithm is simpler: both TRACEMIN and IRTR/ESGEV share the feature that there is no computation of $\rho_X(S)$ or accept/reject mechanism. Furthermore, one of the motivations for the IRTR method is that it may be better able than the RTR to exploit a good preconditioner, so that if the switch is made too early from TRACEMIN, the impact may be less significant.

The latter introduces one of the difficulties in such a hybrid approach: the need for an effective mechanism for switching from the TRACEMIN stage to the IRTR/ESGEV stage. Currently, such a mechanism does not exist. Another problem with this approach is that the TRACEMIN method is valid only for pencils where both A and B are positive-definite. It seems, initially, that the hybrid would therefore be valid only on those problems as well.

4.4 Numerical Experiments

This section provides numerical experiments illustrating the performance of the trust-region based eigensolvers proposed in this chapter. The goal is to show the potential improvement of the IRTR approach over the RTR approach, as well as the performance of these solvers with respect to the specialized eigensolvers discussed in this chapter.

Table 4.2: Select Matrix Market benchmark problems.

Name	Size	nnz(K)	nnz(M)	Description
BCSST22	138	417	138	Textile loom frame
BCSST20	485	1810	485	Frame within a suspension bridge
BCSST19	817	3835	817	Part of a suspension bridge
BCSST08	1074	7017	1074	TV studio
BCSST10	1086	11578	11589	Buckling of a hot washer
BCSST11	1473	17857	1473	Ore car – lumped mass
BCSST26	1922	16129	1922	Seismic analysis, nuclear power station
BCSST13	2003	42943	11973	Fluid flow generalized eigenvalues
BCSST21	3600	15100	3600	Clamped square plate
BCSST23	3134	24156	3134	Part of a 3D globally triang. building
BCSST24	3562	81736	3562	Calgary Olympic Saddledome arena
BCSST25	15439	133840	15439	Columbia Center (Seattle) 76-story skyscraper

The numerical experiments are conducted on matrices from the Matrix Market[5] collection. The problems used for evaluating the solvers in this section are from the Harwell-Boeing collections BCSSTRUC1, BCSSTRUC3, and BCSSTRUC4. Each of these matrix pencils is composed of the stiffness and mass matrices from a structural engineering example. The

[5]The Matrix Market is a service of the Mathematical and Computational Sciences Division of the Information Technology Laboratory of the National Institute of Standards and Technology. http://math.nist.gov/MatrixMarket/

matrices result from the finite element analysis of a mechanical structure and are therefore sparse. The information for these pencils is summarized in Table 4.2.

The majority of the numerical experiments were conducted in MATLAB 7.4.0.287 on a 2.5 GHz PowerPC G5 processor. The following eigensolvers were tested:

- RTR/ESGEV - both "hefty" (denoted RTR) and "skinny" (denoted SRTR) versions of RTR/ESGEV, as described in Section 4.3.3;
- IRTR/ESGEV - both "hefty" (denoted IRTR) and "skinny" (denoted SIRTR) versions of IRTR/ESGEV, as described in Section 4.3.4;
- LOBPCG - the LOBPCG method with full orthogonalization, as described in [HL06] and implemented by the authors; and
- JDCG - the Jacobi-Davidson solver with preconditioned conjugate-gradient correction solver, as described in [Not02] and implemented by the author[6]. The subspace acceleration mechanism is limited to two blocks for comparison purposes. This is enough to globalize the convergence of the solver.

Preconditioners are used for each of these eigenvalue problems. For the MATLAB test problems, each problem is tested using an exact Cholesky (denoted "EC") factorization of A computed as follows:

```
P = colamd(A);
R = chol(A(P,P));
```

For some of the MATLAB tests, an incomplete Cholesky (denoted "IC") factorization of A is also used. It is computed as follows:

```
P = colamd(A);
R = cholinc(A(P,P),0.1);
```

Some of the numerical experiments are conducted using solvers implemented under the Anasazi [BHLT05] framework of the Trilinos project [HBH+03]. These solvers are implemented in C++ using an object-oriented programming paradigm. In these cases, exact preconditioners use an LU factorization (denoted "LU") from the sparse solver package Amesos, while inexact preconditioners use an incomplete Cholesky (denoted "IC") factorization from the IFPACK package.

[6] JDCD code `jdcd_gep` available at http://mntek3.ulb.ac.be/pub/docs/jdcg/

4.4.1 RTR/ESGEV vs. IRTR/ESGEV

The motivation behind the implicit trust-region was to improve the efficiency of a trust-region method by deactivating the classical trust-region mechanism at points where it impedes the progress of the algorithm. This happens primarily when the trust-region radius prevents the algorithm from making progress, as well as when updates are rejected.

Figure 4.1 illustrates the speedup of the IRTR-based methods relative to the RTR-based methods. For each problem the faster of the IRTR/ESGEV methods (skinny or hefty) is compared against the faster of the RTR/ESGEV methods (skinny or hefty). These best-of-class results are compared against each other, with speedup computed by dividing the clock time for RTR/ESGEV by that for IRTR/ESGEV. The full data is listed in Table 4.3.

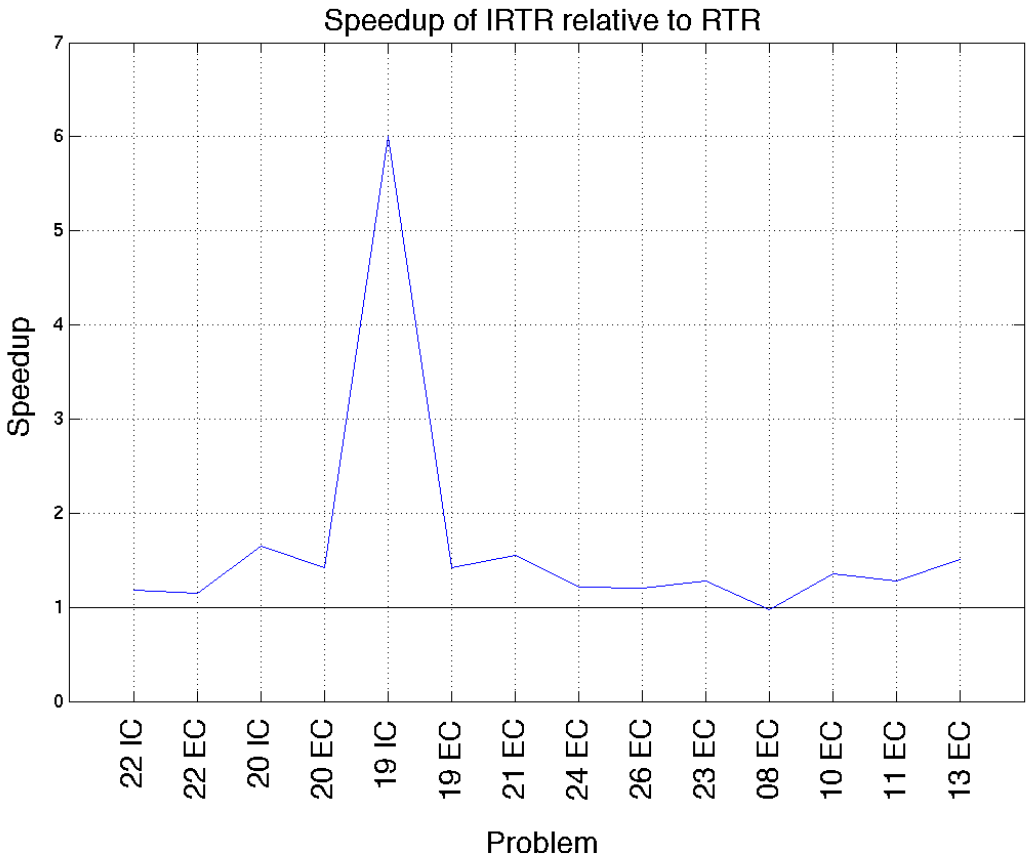

Figure 4.1: Figure illustrating the speedups of IRTR over RTR. Average speedup: 1.512.

Note that on the Harwell-Boeing benchmark problems, the IRTR/ESGEV approach outperforms the RTR/ESGEV approach on all but one problem, with a speed-up mostly between 1.25 and 1.5. The average speedup of IRTR/ESGEV over RTR/ESGEV is 1.5, skewed somewhat by the BCSST19 problem with incomplete Cholesky preconditioner (speed-up greater than 6). These problems show that, at least for the problem of computing extreme eigenspaces, the implicit trust-region mechanism is capable of improved efficiency over the classical trust-region mechanism.

Figure 4.2 presents a convergence curve which illustrates the improved efficiency of the implicit trust-region mechanism in a more qualitative manner. This plot compares the IRTR/ESGEV solver against the RTR/ESGEV solver for the problem of computing 5 eigenpairs of the BCSST24 problem with an exact Cholesky preconditioner. The top figure shows that the IRTR/ESGEV solver leads the RTR/ESGEV in accuracy with respect to iterations for the entire run. The bottom figure shows a zoomed and annotated plot. Note the two annotations on this figure. The leftmost circle illustrates where the explicit trust-region radius is holding back the progress of the method while it grows. The rightmost circle illustrates where rejected updates stall the progress of the algorithm while the trust-region shrinks. During all of this, the implicit trust-region solver makes continuous progress toward the solution, relatively unimpeded.

4.4.2 Skinny Solvers vs. Hefty Solvers

Figure 4.3 illustrates the speed-up of the skinny trust-region implementations over the hefty implementations. It is clear that, in MATLAB on these problems, the difference between skinny and hefty implementations is small. The discussion in Section 4.3.3 illustrated that the benefit of caching operator applications over explicitly performing them will depend on the cost of those operator applications relative to the effort in maintaining the caches, the associated memory cost, and the impact on the memory hierarchy.

For smaller problems such as those in the Harwell-Boeing benchmark suite, the effects on the memory hierarchy will be negligible. In this case, the difference in timings between a skinny implementation and a hefty implementation will come down to the effort in maintaining the cache in the inner iterations and the cost associated with applying the operators in the outer iteration. The former is a function of the problem size, the block size, and the number of inner iterations, while the latter is a function of the cost of the operator,

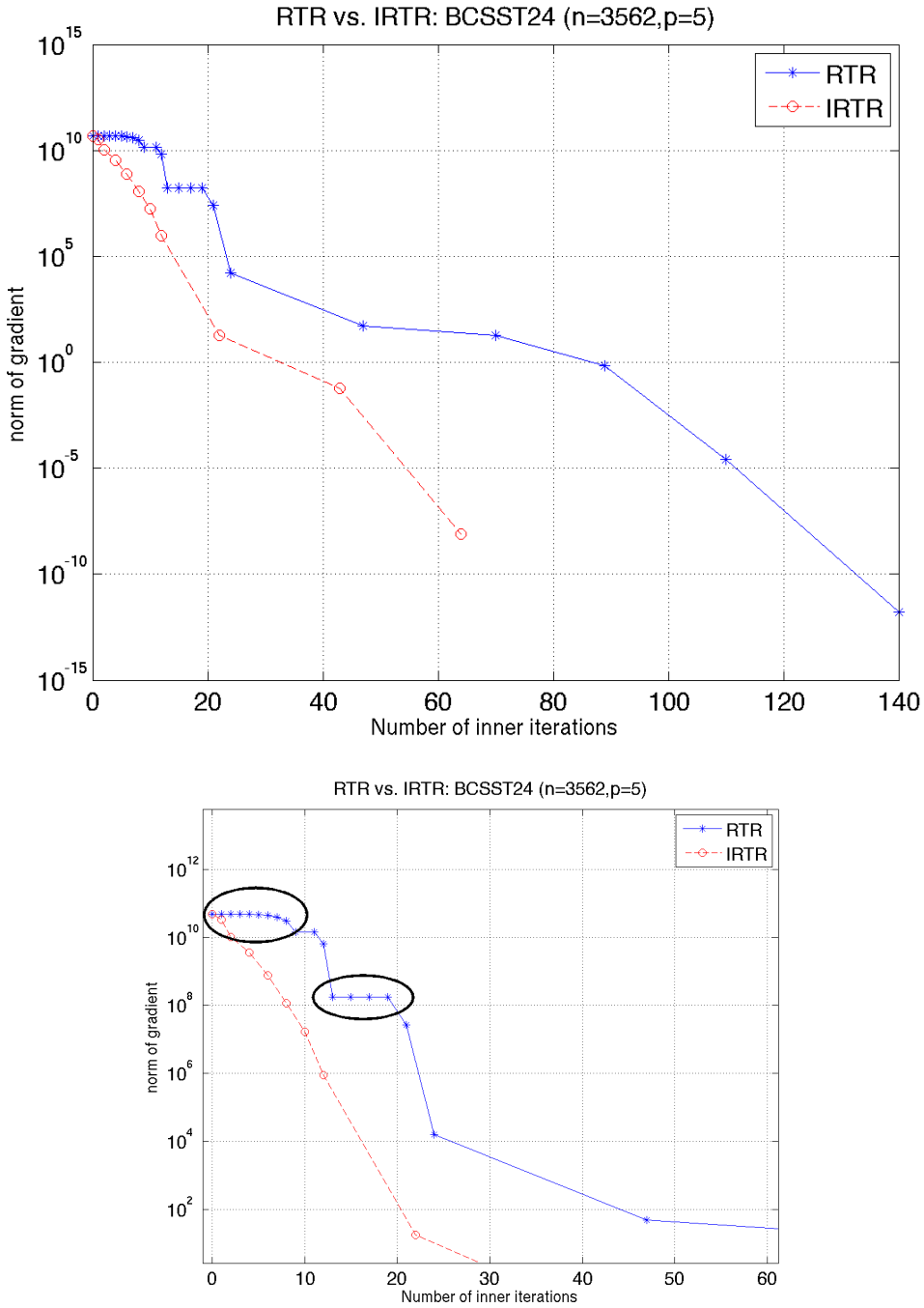

Figure 4.2: Figure illustrating the potential improvement of the IRTR/ESGEV solver over the RTR/ESGEV solver on Matrix Market problem BCSST24. The annotated version highlights the predicted drawbacks of the trust-region mechanism.

Table 4.3: MATLAB timings (in seconds) for Matrix Market problems. Each timing is the mean of three tests. "—" indicates no progress towards convergence. IC denotes incomplete Cholesky preconditioner, while EC denotes exact Cholesky preconditioner. Bold face indicates the fastest time for each problem.

Problem	Prec	nev	RTR	IRTR	SRTR	SIRTR	LOBPCG	JDCG
BCSST22	IC	5	1.448	**1.214**	1.435	1.296	6.194	—
BCSST22	EC	5	.6111	.5302	.6069	**.5268**	1.005	3.428
BCSST20	IC	5	594.9	385.9	630.2	**361.6**	1312	—
BCSST20	EC	5	.4814	.3380	.4884	.3395	**.2724**	.5173
BCSST19	IC	5	4623	803	4497	**749**	7858	—
BCSST19	EC	5	1.900	1.353	1.898	1.341	**.7782**	2.362
BCSST08	EC	25	70.69	72.98	70.53	72.65	**21.15**	—
BCSST10	EC	25	12.59	9.405	12.55	9.296	**4.194**	33.41
BCSST11	EC	25	21.68	17.17	21.68	16.99	**5.251**	—
BCSST26	EC	25	53.19	44.55	53.08	44.10	**10.17**	—
BCSST13	EC	25	108.8	72.05	108.8	71.93	**16.46**	184.8
BCSST21	EC	25	51.52	33.41	51.38	33.12	**12.60**	92.00
BCSST23	EC	25	377.3	296.0	377.1	295.0	**56.79**	377.4
BCSST24	EC	25	79.05	65.25	79.22	65.04	**20.49**	482.6

the block size, and the number of outer iterations.

It is not difficult to construct a problem where a skinny implementation is less efficient than the hefty implementation. Table 4.6 gives the timings and speedups of the RTR/ESGEV and IRTR/ESGEV methods when computing a single eigenvector of a dense eigenvalue problem. In this case, the operator application is sufficiently expensive so make the caching in the hefty implementations more efficient than the additional applications of the operator required by a skinny implementation.

4.4.3 IRTR/ESGEV Heuristic Approximation

The IRTR/ESGEV method presented in Section 4.3.4 was not demonstrated to be an implementation of the IRTR algorithm. This is because it was not rigorously shown that the point returned from the model minimization was inside the implicit trust-region. This is because the difficult formula for $\rho_X(S)$ defining the implicit trust-region was approximated by the simpler formula for $\hat{\rho}_X(S)$. As a result, the global convergence properties of the IRTR

Table 4.4: MATLAB number of iterations for Matrix Market problems. Each count is the mean of three tests. "—" indicates no progress towards convergence. IC denotes incomplete Cholesky preconditioner, while EC denotes exact Cholesky preconditioner. Bold face indicates the smallest number of iterations for each problem.

Problem	Prec	nev	RTR	IRTR	SRTR	SIRTR	LOBPCG	JDCG
BCSST22	IC	5	317	**221**	317	**238**	**660**	—
BCSST22	EC	5	119	91	119	**91**	**95**	538
BCSST20	IC	5	46,631	25,213	50,100	**24,108**	**56,468**	—
BCSST20	EC	5	25	18	25	18	11	48
BCSST19	IC	5	231,232	35,059	221,359	**31,513**	**269,359**	—
BCSST19	EC	5	70	47	70	47	24	118
BCSST08	EC	25	85	53	85	53	24	490
BCSST10	EC	25	63	51	63	51	24	1,000
BCSST11	EC	25	175	140	175	140	35	—
BCSST26	EC	25	213	165	213	165	51	483
BCSST13	EC	25	191	194	191	194	73	—
BCSST21	EC	25	101	72	101	72	25	707
BCSST23	EC	25	98	75	98	75	23	—
BCSST24	EC	25	96	63	96	63	23	420

method are not necessarily inherited by the IRTR/ESGEV algorithm as presented. However, it should be noted that the algorithm did enjoy global convergence on all of the problems tested to date (including the Harwell-Boeing problems used in this section). Furthermore, recalling the discussion from Section 4.3.4, the heuristic may only cause problems in the case that the violation of $\rho_X(S) \geq \hat{\rho}_X(S)$ is severe.

Table 4.7 lists the average ρ violations for the Harwell-Boeing test problems. The violation is measured by dividing the value $\rho_X(S)$ by the implicit trust-region parameter ρ'. The reported number is the mean of this value over all tangent vectors returned from the model minimization for which there was a violation. In each of these cases, the violation is mild, implying that the model m_X still has good agreement with the lifted objective $\hat{f}_X$ and justifying the approximation of ρ_X by $\hat{\rho}_X$.

Table 4.5: MATLAB flops counts (excluding operator applications) for Matrix Market problems. Each count is the mean of three tests. IC denotes incomplete Cholesky preconditioner, while EC denotes exact Cholesky preconditioner.

Problem	Prec	nev	RTR	IRTR	SRTR	SIRTR	LOBPCG
BCSST22	IC	5	1.29e+08	8.77e+07	1.23e+08	8.80e+07	6.62e+10
BCSST22	EC	5	6.23e+07	2.79e+07	5.99e+07	2.67e+07	1.42e+09
BCSST20	IC	5	1.13e+11	5.01e+10	1.13e+11	4.48e+10	1.68e+15
BCSST20	EC	5	6.97e+07	2.92e+07	6.76e+07	2.84e+07	7.11e+07
BCSST19	IC	5	1.93e+12	8.75e+10	1.75e+12	6.71e+10	6.16e+16
BCSST19	EC	5	2.66e+08	1.04e+08	2.56e+08	1.00e+08	5.10e+08
BCSST08	EC	25	8.99e+09	8.00e+09	8.88e+09	7.90e+09	1.60e+11
BCSST10	EC	25	4.70e+09	2.80e+09	4.65e+09	2.77e+09	1.88e+10
BCSST11	EC	25	9.25e+09	5.05e+09	9.15e+09	4.99e+09	2.16e+10
BCSST26	EC	25	2.62e+10	1.24e+10	2.59e+10	1.22e+10	6.56e+10
BCSST13	EC	25	1.89e+10	6.66e+09	1.87e+10	6.60e+09	2.87e+10
BCSST21	EC	25	3.30e+10	1.02e+10	3.27e+10	1.01e+10	5.78e+10
BCSST23	EC	25	5.58e+10	2.78e+10	5.52e+10	2.75e+10	2.22e+11
BCSST24	EC	25	2.00e+10	1.11e+10	1.99e+10	1.10e+10	5.57e+10

Table 4.6: MATLAB timings comparison skinny and hefty solvers for a $p = 1$ dense eigenvalue problem. Each count is the mean of three tests.

RTR	SRTR	RTR "speed-up"	IRTR	SIRTR	IRTR "speed-up"
1.1852	2.1141	.5606	1.0156	1.7572	.5780

4.4.4 Adaptive Model IRTR/ESGEV

The motivation for the adaptive model hybrid method combining TRACEMIN and IRTR/ESGEV was that the TRACEMIN method is able to make faster progress early with a good preconditioner, while the IRTR/ESGEV method was able to make faster progress asymptotically due to its superlinear rate of convergence. Figure 4.4 and 4.5 illustrate this property of TRACEMIN and IRTR/ESGEV and demonstrate the potential for speed-up from a hybrid approach.

Note that in each of these experiments, the TRACEMIN method initially outperforms the IRTR/ESGEV method, before succumbing to a linear rate of convergence. These figures

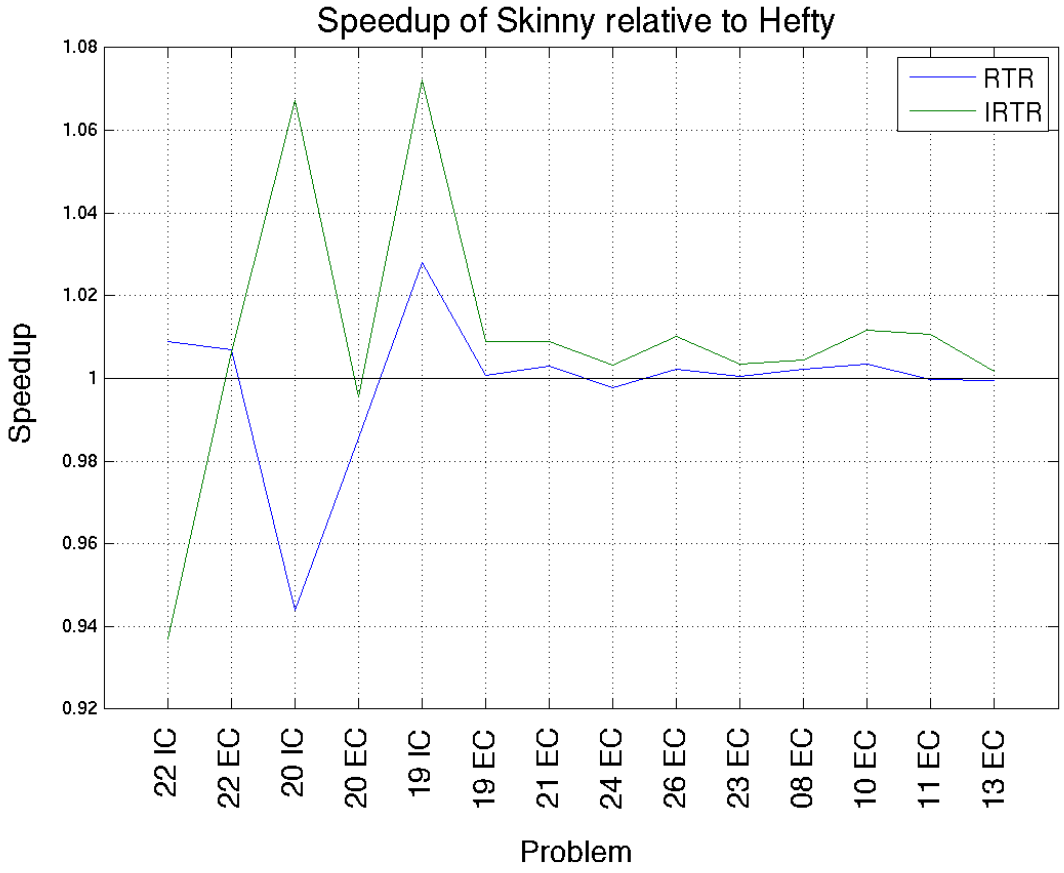

Figure 4.3: Figure illustrating the speedups of skinny solvers over hefty solvers for Harwell-Boeing benchmark problems. Average speedup is .999 for RTR and 1.01 for IRTR.

also illustrate the performance benefit that can the be achieved via the hybrid method. In order to illustrate the effect due to the timing of the switch in the hybrid method, each figure contains plots for a various number of switch points. The hybrid method is able to exploit the head start provided by the TRACEMIN method to reduce the number of iterations required for convergence, as compared against the pure IRTR/ESGEV approach. For example, in the particular case of the BCSST20 problem (Figure 4.4, bottom), the adaptive model solver switched after 5 outer iterations (denoted AM(5)) reduces the number of inner iterations by 33%.

The switching points for these experiments were determined manually after examining

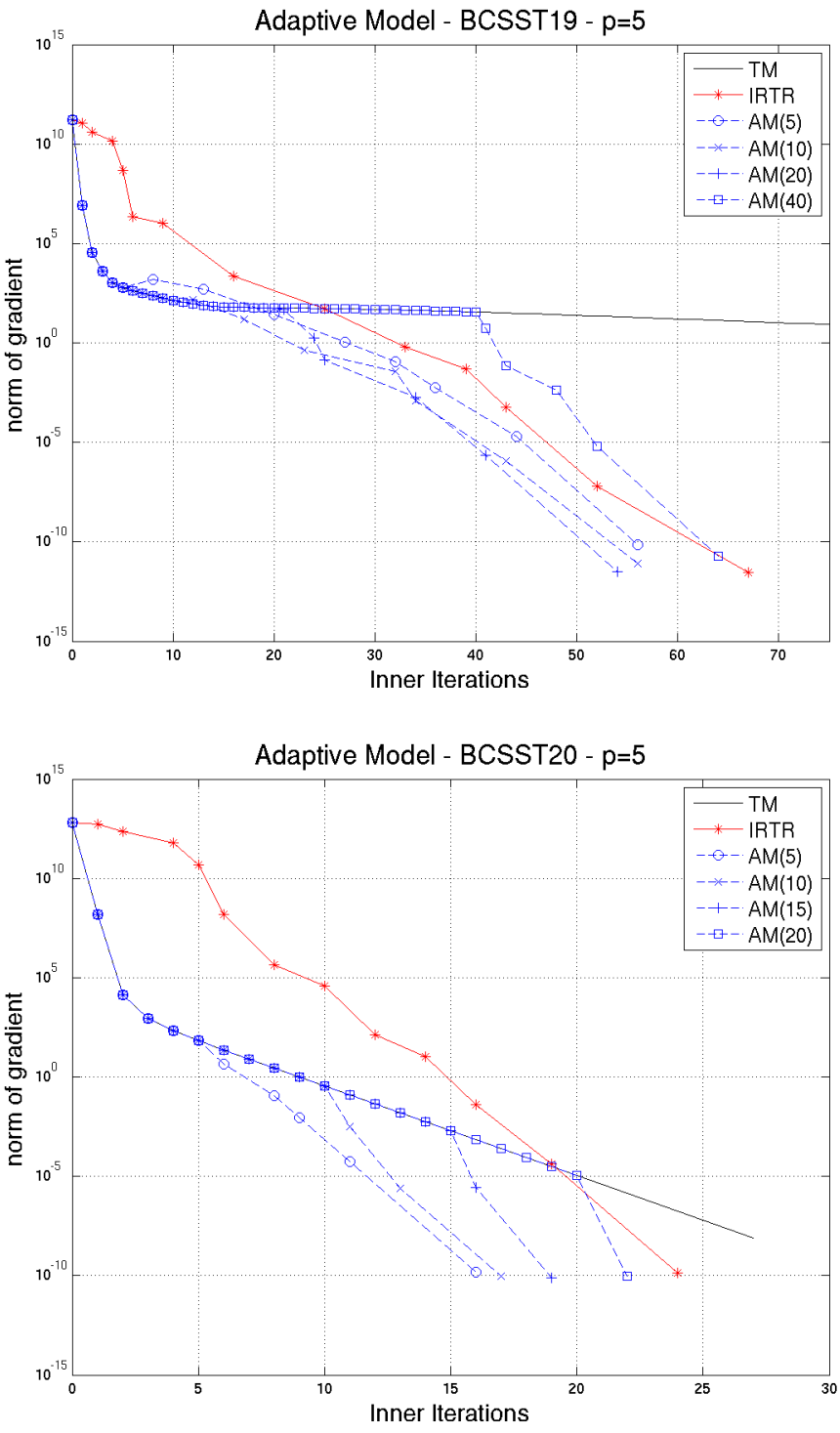

Figure 4.4: Figures comparing TRACEMIN, IRTR/ESGEV and the Adaptive Model hybrid.

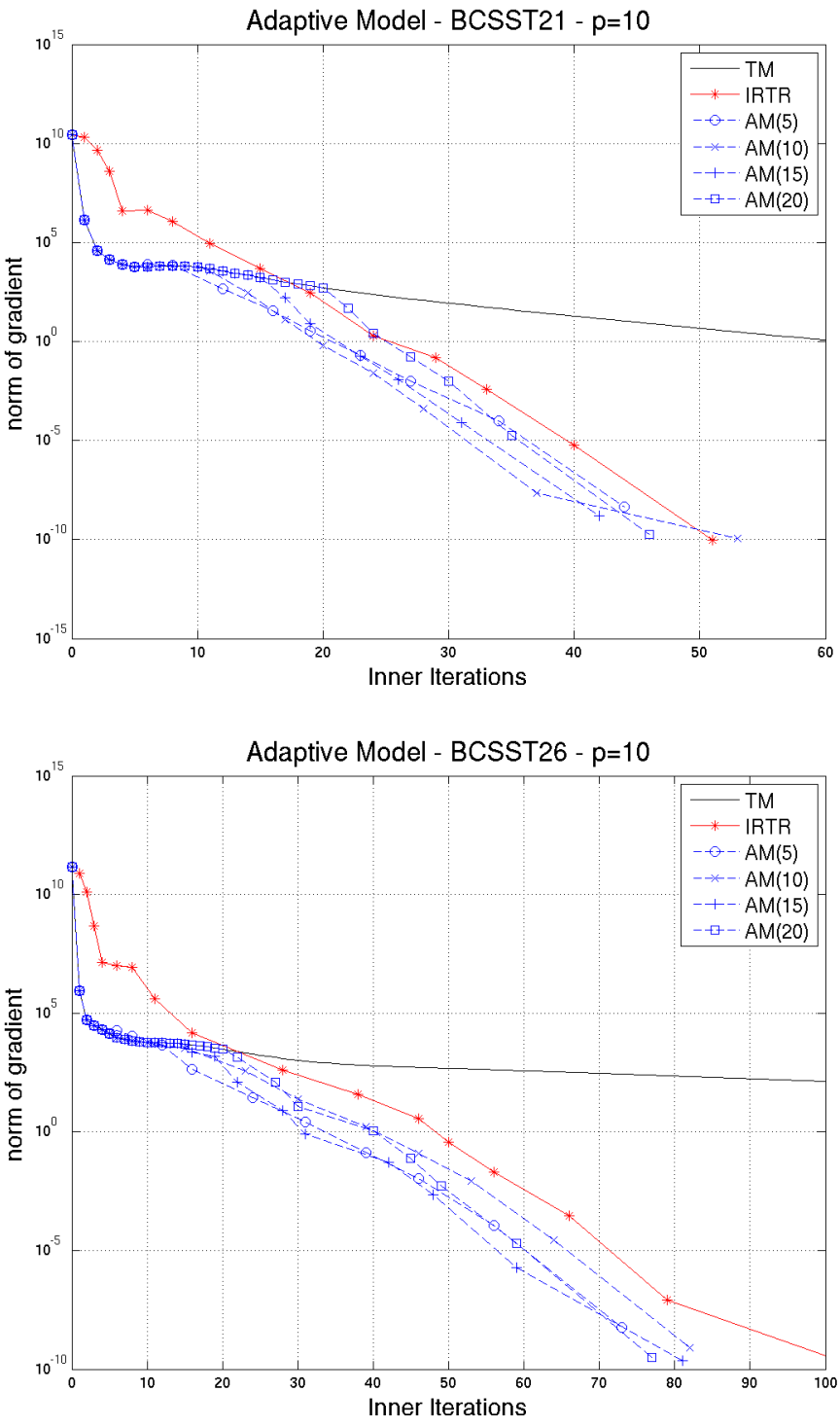

Figure 4.5: Figures comparing TRACEMIN, IRTR/ESGEV and the Adaptive Model hybrid.

Table 4.7: Average ρ violations for a subset of Harwell-Boeing problems for IRTR/ESGEV with $\rho' = 0.5$. EC denotes exact Cholesky preconditioner.

Problem	Prec	nev	$\rho_X(S)/\rho'$	# of violations
BCSST22	EC	5	0.95	2
BCSST20	EC	5	0.79	2
BCSST19	EC	5	0.89	2
BCSST08	EC	25	0.97	1
BCSST10	EC	25	0.97	3
BCSST11	EC	25	0.94	3
BCSST26	EC	25	0.92	3
BCSST13	EC	25	0.89	2
BCSST21	EC	25	0.93	1
BCSST23	EC	25	0.93	3
BCSST24	EC	25	0.93	3

the convergence curves for the IRTR/ESGEV and TRACEMIN methods. The effective use of the hybrid solver will require a mechanism for automatically determining the switch point without any *a priori* information. This is the subject of continuing research.

4.4.5 RTR/IRTR solvers vs. Specialized Solvers

Table 4.3 lists the MATLAB timings for the each of the solvers on the Harwell-Boeing test problems. It is clear from these timings that, for MATLAB implementations, the LOBPCG solver dominates JDCG and the trust-region solvers in the presence of a good preconditioner (with the exception of BCSST22). Examining Table 4.4, we see that the trust-region methods are able to outperform the LOBPCG method (with respect to clock time) only in those cases when the number of (inner) iterations for the trust-region solver is less than the number of outer iterations for the LOBPCG solver. However, examination of the floating point operations in Table 4.5 shows that the LOBPCG method performs more work than the trust-region methods in every case, even those where the number of iterations is less.

This is explained by noting that one iteration of the LOBPCG method is computationally more dense than one inner iteration of the trust-region methods. When solving an eigenvalue

problem of order n for p eigenvectors, the cost of one iteration[7] of the LOBPCG method is $O(np^2)$, while the cost of the trust-region methods is $O(np)$. In addition to explaining why the LOBPCG method has a significantly higher flop count than the trust-region methods, it also suggests that this difference will become more apparent if p is enlarged.

Table 4.8: Anasazi/C++ timings (in seconds) comparing RTR solvers and LOBPCG. Each timing is the mean of three tests. Average speedup of IRTR is 1.33 over RTR, 3.46 over LOBPCG. * denotes time-out before convergence.

IC denotes incomplete Cholesky preconditioner, while LU denotes exact LU preconditioner. Bold face indicates the fastest time for each problem.

Problem	nev	Prec	RTR	IRTR	SRTR	SIRTR	LOBPCG
BCSST22	5	none	2.73	2.34	2.64	**1.90**	39.03
BCSST22	5	IC	1.11	1.07	1.10	**1.03**	3.17
BCSST22	5	LU	0.29	0.27	0.30	**0.24**	0.45
BCSST20	5	IC	57.49	38.89	49.04	**34.40**	*151.00
BCSST20	5	LU	0.12	0.09	0.11	**0.08**	0.14
BCSST13	25	LU	12.86	7.87	13.25	7.81	**6.20**
BCSST13	100	LU	79.41	58.34	79.70	**56.95**	**56.12**
BCSST23	25	LU	28.71	22.35	28.25	22.10	**16.86**
BCSST23	100	LU	174.10	131.70	168.76	**129.06**	180.40
BCSST24	25	LU	9.34	8.17	10.09	8.23	**7.76**
BCSST24	100	LU	98.93	69.95	98.23	**69.83**	108.20
BCSST25	25	LU	720.93	**85.25**	361.40	97.64	*3218.00

However, in spite of the large flop counts, the LOBPCG method is still able to outperform the trust-region methods. One explanation for this is that the limitations of MATLAB programming environment favor the (relatively) simple LOBPCG iteration in terms of efficiency. In order to explore this, Table 4.8 lists timings of the LOBPCG and trust-region solvers as implemented in C++ in the Anasazi eigensolver framework [BHLT05]. This table lists the results of selected Harwell-Boeing problems.

Comparing Tables 4.3 and 4.8, we see that the performance advantage of the LOBPCG method in the MATLAB experiments is almost entirely lost. Furthermore, in the cases that LOBPCG still outperforms the trust-region methods, the speed-up of LOBPCG is reduced from as much as 4.3 in MATLAB to under 1.3 in Anasazi.

[7] neglecting the cost of the operation applications

As with the MATLAB tests, the trust-region solvers perform significantly better than LOBPCG in the presence of an inferior preconditioner. Also, as predicted, when the number of requested eigenvalues is increased, the time required for LOBPCG to compute the solutions increases much more than the time required for the trust-region methods.

These experiments show that the trust-region methods, particularly the IRTR/ESGEV method, enjoy competitive performance against specialized eigensolvers. The performance of these methods is expected to increase further as the impact of the algorithmic parameters becomes better understood.

CHAPTER 5

CONCLUDING REMARKS AND FUTURE RESEARCH

The major contributions of this dissertation are four-fold: the description of the retraction-based paradigm for Riemannian optimization; the description of the Riemannian trust-region methods and the analysis of their convergence properties; the description of a new trust-region mechanism, the implicit Riemannian trust-region method; and the application of the trust-region methods to the problem of computing extreme eigenspaces of a symmetric matrix pencil.

A majority of the previous literature on Riemannian optimization made exclusive use of the exponential map. While perfectly acceptable, this leaves little freedom for the customization of algorithms, to the detriment of algorithmic efficiency. We illustrate that general retractions as defined in Chapter 1 preserve the necessary analytical properties while allowing more efficient algorithms. Because the exponential map is a retraction, this paradigm encapsulates the previous exponential-only approaches. Furthermore, we describe a new paradigm for Riemannian optimization, that of retraction-based Riemannian optimization. By using a general retraction to explicitly move the optimization problem from the manifold to the tangent space, we allow for the easy adaptation of a multitude of Euclidean optimization algorithms for the Riemannian setting.

In particular, we presented two new optimization methods in this setting. The Riemannian trust-region method was described by employing a classical trust-region step for the optimization of the pullback at each step. We show that this algorithm retains the strong global convergence and fast local convergence properties that popularized Euclidean trust-region methods. In order to address some of the inefficiencies of the classical trust-region mechanism, we propose a novel optimization algorithm: the Implicit Riemannian Trust-

Region method. By removing the explicit trust-region radius and the rejection mechanism, the IRTR method is shown to allow significantly increased performance for the problem of computing extreme eigenspaces. Furthermore, we show that the IRTR method retains all of the beneficial convergence properties of the RTR and Euclidean trust-region methods.

Lastly, we consider the application of these Riemannian optimization techniques to the problem of computing extreme eigenpairs of a symmetric/positive-definite matrix pencil. Similar to previous works, we reconsider this problem as the optimization of the generalized Rayleigh quotient over the Grassmann manifold. Applying the Riemannian trust-region methods yields two new preconditioned eigenvalue solvers, which we demonstrate to be very competitive against the related LOBPCG and Jacobi-Davidson methods. We also provide some analysis of other eigenvalue solvers in the context of Riemannian optimization.

The success of Riemannian optimization—in particular, the Riemannian trust-region methods—suggests the possibility of impact on other problems. The following are a number of avenues of investigation that we look forward to pursuing in the future.

- The analysis we have conducted thus far illustrated that the retraction mechanism is sufficient to describe a wide range of efficient and robust algorithms. The connection between Riemannian optimization and Euclidean optimization in the face of smooth constraints suggests that some amount of technology transfer may prove successful.

- The success of the IRTR method over the RTR method for the eigenvalue problem fulfills the expectations of greater efficiency. However, as repeatedly noted in Chapter 3, an efficient application of the IRTR method to a problem requires a non-trivial understanding of the ρ ratio for the problem at hand. It is our hope that other problems in both Riemannian and Euclidean optimization can be identified where this promising method can be put to effective use.

- Because of the heuristic employed for the application of the IRTR method to the eigenvalue problem, the convergence theory of the IRTR method does not necessary follow for the IRTR/ESGEV eigensolver as presented here. Extension of the theory and its application to the RTR/ESGEV solver will comprise the body of future work.

- The RTR/ESGEV and IRTR/ESGEV eigensolvers presented in this dissertation are relatively straightforward applications of those methods, yet they still show great

promise.

- Widescale utilization of these methods will require a more comprehensive understanding of the algorithmic parameters: the parameter ρ' for both solvers, as well as the various trust-region parameters for the RTR/ESGEV solver.
- The algorithms as presented exploited very little knowledge of the underlying eigenvalue problem. More efficient algorithms should be possible by incorporating eigensolver-specific methods, such as subspace acceleration.
- Some preliminary testing has shown that the superlinear convergence of these methods is not always the most efficient route, with a cap on the number of inner iterations allowing for a fast linear rate of convergence. This deserves further investigation.
- The ability to balance the operation count of these methods against the storage requirements and the cost of operator applications, suggests the need for some criterion to decide when the former should be preferred over the latter.
- The adaptive model TRACEMIN-IRTR hybrid solver cannot be effectively deployed without some criterion for switching from the first to the second phase.

REFERENCES

[ABG04] P.-A Absil, C. G. Baker, and K. A. Gallivan, *Trust-region methods on Riemannian manifolds with applications in numerical linear algebra*, Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems (MTNS2004), Leuven, Belgium, 5–9 July 2004, 2004. 23

[ABG06] _____, *A truncated-CG style method for symmetric generalized eigenvalue problems*, J. Comput. Appl. Math. **189** (2006), no. 1–2, 274–285. 56, 59

[ABG07] P.-A. Absil, C. G. Baker, and K. A. Gallivan, *Trust-region methods on Riemannian manifolds*, Foundations of Computational Mathematics **7** (2007), no. 3, 303–330. 22, 24, 25, 59, 66, 78

[ABGS05] P.-A Absil, C. G. Baker, K. A. Gallivan, and A. Sameh, *Adaptive model trust region methods for generalized eigenvalue problems*, International Conference on Computational Science (Vaidy S. Sunderam, Geert Dick van Albada, and Peter M. A. Sloot, eds.), Lecture Notes in Computer Science, vol. 3514, Springer-Verlag, 2005, pp. 33–41. 59, 118

[ADM+02] R. L. Adler, J.-P. Dedieu, J. Y. Margulies, M. Martens, and M. Shub, *Newton's method on Riemannian manifolds and a geometric model for the human spine*, IMA J. Numer. Anal. **22** (2002), no. 3, 359–390. 1, 20, 21, 23

[AG06] P.-A Absil and K. A. Gallivan, *Accelerated line-search and trust-region methods*, Tech. Report FSU-SCS-2005-095, School of Computational Science, Florida State University, June 2006, http://scseprints.scs.fsu.edu. 103

[AMS04] P.-A Absil, R. Mahony, and R. Sepulchre, *Riemannian geometry of Grassmann manifolds with a view on algorithmic computation*, Acta Appl. Math. **80** (2004), no. 2, 199–220. 25, 30, 36, 78

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, Princeton, NJ, January 2008. 21, 24, 30, 78, 103

[AMSV02] P.-A Absil, R. Mahony, R. Sepulchre, and P. Van Dooren, *A Grassmann-Rayleigh quotient iteration for computing invariant subspaces*, SIAM Rev. **44** (2002), no. 1, 57–73 (electronic). MR MR1924546 (2003h:65045) 78

[BAG07] C. G. Baker, P.-A. Absil, and K. A. Gallivan, *Generic Riemannian trust-region package*, 2007, Website: http://www.scs.fsu.edu/~cbaker/GenRTR/. 77

[BAG08] C. G. Baker, P.-A Absil, and K. A. Gallivan, *Implicit trust-region methods on Riemannian manifolds*, IMA Journal of Numerical Analysis (2008), Accepted for publication. 25, 61

[Ber95] D. P. Bertsekas, *Nonlinear programming*, Athena Scientific, Belmont, Massachusetts, 1995. 51, 73

[BHLT05] C. G. Baker, U. L. Hetmaniuk, R. B. Lehoucq, and H. K. Thornquist, *The Anasazi block eigensolvers package*, 2005, See http://trilinos.sandia.gov/packages/anasazi/. 120, 131

[Boo75] W. M. Boothby, *An introduction to differentiable manifolds and Riemannian geometry*, Academic Press, New York-London, 1975. 13, 15, 17, 33

[BSS88] R. H. Byrd, R. B. Schnabel, and G. A Shultz, *Approximate solution of the trust region problem by minimization over two-dimensional subspaces*, Math. Programming **40** (1988), no. 3, (Ser. A), 247–263. 39

[CGT00] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-region methods*, MPS/SIAM Series on Optimization, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, and Mathematical Programming Society (MPS), Philadelphia, PA, 2000. 9, 33, 38, 39, 40, 60, 64, 67, 68, 69, 72, 105

[CI01] E. Celledoni and A Iserles, *Methods for the approximation of the matrix exponential in a Lie-algebraic setting*, IMA J. Numer. Anal. **21** (2001), no. 2, 463–488. 59

[dC92] M. P. do Carmo, *Riemannian geometry*, Mathematics: Theory & Applications, Birkhäuser Boston Inc., Boston, MA, 1992, Translated from the second Portuguese edition by Francis Flaherty. 13, 15, 17, 22, 24, 47, 50, 51

[DM79] J. E. Dennis, Jr. and H. H. W. Mei, *Two new unconstrained optimization algorithms which use function and gradient values*, J. Optim. Theory Appl. **28** (1979), no. 4, 453–482. 39

[DN04] J.-P. Dedieu and D. Novitsky, *Symplectic methods for the approximation of the exponential and the Newton sequence on Riemannian submanifolds*, submitted to the Journal of Complexity, 2004. 59

[DPM03] Jean-Pierre Dedieu, Pierre Priouret, and Gregorio Malajovich, *Newton's method on Riemannian manifolds: convariant alpha theory*, IMA J. Numer. Anal. **23** (2003), no. 3, 395–419. MR 2004e:65061 1, 20

[DV00] J. Dehaene and J. Vandewalle, *New Lyapunov functions for the continuous-time QR algorithm*, Proceedings CD of the 14th International Symposium on the Mathematical Theory of Networks and Systems (MTNS2000), Perpignan, France, July 2000, 2000. 57

[EAS98] A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl. **20** (1998), no. 2, 303–353. 1, 20, 23, 25, 36, 78

[Gab82] D. Gabay, *Minimizing a differentiable function over a differential manifold*, Journal of Optimization Theory and Applications **37** (1982), no. 2, 177–219. 20

[GLRT99] N. I. M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint, *Solving the trust-region subproblem using the Lanczos method*, SIAM J. Optim. **9** (1999), no. 2, 504–525 (electronic). 34, 37, 39

[GST05] N. I. M. Gould, C. Sainvitu, and Ph. L. Toint, *A filter-trust-region method for unconstrained optimization*, SIAM J. Optimization **16** (2005), no. 2, 341–357. 60

[GV96] G. H. Golub and C. F. Van Loan, *Matrix computations, third edition*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, 1996. 31, 89, 92

[Hag01] W. W. Hager, *Minimizing a quadratic over a sphere*, SIAM J. Optim. **12** (2001), no. 1, 188–208 (electronic). 39

[HBH+03] M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams, *An Overview of Trilinos*, Tech. Report SAND2003-2927, Sandia National Laboratories, Albuquerque, N.M., 2003. 120

[HL06] U. Hetmaniuk and R. Lehoucq, *Basis selection in LOBPCG*, J. Comput. Phys. **218** (2006), no. 1, 324–332. 95, 120

[HM94] U. Helmke and J. B. Moore, *Optimization and dynamical systems*, Springer-Verlag London Ltd., London, 1994. 25, 59, 78

[HT04] Knut Hüper and Jochen Trumpf, *Newton-like methods for numerical optimization on manifolds*, Proc. 38th IEEE Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, November 7-10, 2004, 2004. 1, 20, 35

[Kny01] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput. **23** (2001), no. 2, 517–541. 94, 95

[LE00] R. Lippert and A. Edelman, *Nonlinear eigenvalue problems with orthogonality constraints (Section 9.4)*, Templates for the Solution of Algebraic Eigenvalue Problems (Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst, eds.), SIAM, Philadelphia, 2000, pp. 290–314. 59

[LE02] E. Lundström and L. Eldén, *Adaptive eigenvalue computations using Newton's method on the Grassmann manifold*, SIAM J. Matrix Anal. Appl. **23** (2002), no. 3, 819–839. 78

[Lue72] David G. Luenberger, *The gradient projection method along geodesics*, Management Sci. **18** (1972), 620–631. MR MR0362899 (50 #15337) 20

[Mah96] R. E. Mahony, *The constrained Newton method on a Lie group and the symmetric eigenvalue problem*, Linear Algebra Appl. **248** (1996), 67–89. 1

[Man02] J. H. Manton, *Optimization algorithms exploiting unitary constraints*, IEEE Trans. Signal Process. **50** (2002), no. 3, 635–650. 1, 20, 21

[MM02] R. Mahony and J. H. Manton, *The geometry of the Newton method on non-compact Lie groups*, J. Global Optim. **23** (2002), no. 3, 309–327. 20

[MS83] J. J. Moré and D. C. Sorensen, *Computing a trust region step*, SIAM J. Sci. Statist. Comput. **4** (1983), 553–572. 37, 39

[MS84] J. J. Moré and D. C. Sorensen, *Newton's method*, Studies in numerical analysis, MAA Stud. Math., vol. 24, Math. Assoc. America, Washington, DC, 1984, pp. 29–82. 33

[MS85] A. Machado and I. Salavessa, *Grassmannian manifolds as subsets of Euclidean spaces*, Res. Notes in Math. **131** (1985), 85–102. 25

[Mun00] James R. Munkres, *Topology*, second ed. ed., Prentice Hall, Upper Saddle River, NJ, 2000. 49

[Nas56] John Nash, *The imbedding problem for Riemannian manifolds*, The Annals of Mathematics **63** (1956), no. 1, 20–63. 12

[Not02] Y. Notay, *Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl. **9** (2002), no. 1, 21–44. 89, 116, 120

[NS95] S. Nash and A. Sofer, *Linear and nonlinear programmming*, McGraw-Hill, New York, 1995. 11

[NW99] J. Nocedal and S. J. Wright, *Numerical optimization*, Springer Series in Operations Research, Springer-Verlag, New York, 1999. 7, 9, 11, 35, 39, 40, 41, 43, 56, 68, 69, 72, 91

[OJS90] J. Olsen, P. Jørgensen, and J. Simons, *Passing the one-billion limit in full configuration-interaction (FCI) calculations*, Chemical Physics Letters **169** (1990), 463–472. 110

[O'N83] B. O'Neill, *Semi-riemannian geometry: With applications to relativity*, Pure and Applied Mathematics, vol. 103, Academic Press, New York, U.S.A., 1983. 49

[OW00] B. Owren and B. Welfert, *The Newton iteration on Lie groups*, BIT **40** (2000), no. 1, 121–145. 1, 20

[Pol97] Elijah Polak, *Optimization*, Applied Mathematical Sciences, vol. 124, Springer-Verlag, New York, 1997, Algorithms and consistent approximations. MR 98g:49001 57

[Pow70a] M. J. D. Powell, *A hybrid method for nonlinear equations*, Numerical Methods for Nonlinear Algebraic Equations (P. Rabinowitz, ed.), Gordon and Breach, London, 1970, pp. 87–114. 67

[Pow70b] ———, *A new algorithm for unconstrained optimization*, Nonlinear Programming (J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds.), Academic Press, London, 1970, pp. 31–65. 39, 67

[Pow74] M. J. D. Powell, *Convergence properties of a class of minimization algorithms*, Nonlinear programming, 2 (Proc. Sympos. Special Interest Group on Math. Programming, Univ. Wisconsin, Madison, Wis., 1974) (New York), Academic Press, 1974. 33

[Pow75] M. J. D. Powell, *Convergence properties of a class of minimization algorithms*, Nonlinear Programming 2 (O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds.), Academic Press, London, 1975, pp. 1–27. 67

[Rut70] H. R. Rutishauser, *Simultaneous iteration method for symmetric matrices*, Numerische Mathematik **16** (1970), 205–223. 58

[Saa92] Y. Saad, *Numerical methods for large eigenvalue problems*, Halstead Press, New York, 1992. 94

[Sak96] T. Sakai, *Riemannian geometry*, Translations of Mathematical Monographs, no. 149, American Mathematical Society, 1996. 51

[SBFV96] Gerard L. G. Sleijpen, Albert G. L. Booten, Diederik R. Fokkema, and Henk A. Van der Vorst, *Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems*, BIT Numerical Mathematics **36** (1996), no. 3, 595–633. 100

[Shu86] M. Shub, *Some remarks on dynamical systems and numerical analysis*, Proc. VII ELAM. (L. Lara-Carrero and J. Lewowicz, eds.), Equinoccio, U. Simón Bolívar, Caracas, 1986, pp. 69–92. 1, 20, 21, 23

[Smi93] S. T. Smith, *Geometric optimization methods for adaptive filtering*, Ph.D. thesis, Division of Applied Sciences, Harvard University, Cambridge, Massachusetts, 1993. 20, 36

[Smi94] ———, *Optimization techniques on Riemannian manifolds*, Hamiltonian and gradient flows, algorithms and control (Anthony Bloch, ed.), Fields Inst. Commun., vol. 3, Amer. Math. Soc., Providence, RI, 1994, pp. 113–136. MR MR1297990 (95g:58062) 20, 23, 36, 50

[Sor82] D. C. Sorensen, *Newton's method with a model trust region modification*, SIAM J. Numer. Anal. **19** (1982), no. 2, 409–426. 33

[SSB85] Gerald A. Shultz, Robert B. Schnabel, and Richard H. Byrd, *A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties*, SIAM J. Numer. Anal. **22** (1985), no. 1, 47–67. 72

[ST00] A. Sameh and Z. Tong, *The trace minimization method for the symmetric generalized eigenvalue problem*, J. Comput. Appl. Math. **123** (2000), 155–175. 58, 64, 89, 92, 93, 102, 103

[Ste83] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal. **20** (1983), 626–637. 37, 39, 52, 54, 64

[Ste01] G. W. Stewart, *Matrix algorithms, vol II: Eigensystems*, Society for Industrial and Applied Mathematics, Philadelphia, 2001. 89, 92

[SV96] G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl. **17** (1996), no. 2, 401–425. 87

[SW82] A. H. Sameh and J. A. Wisniewski, *A trace minimization algorithm for the generalized eigenvalue problem*, SIAM J. Numer. Anal. **19** (1982), no. 6, 1243–1259. 64, 89, 90, 102, 103

[Toi81] Ph. L. Toint, *Towards an efficient sparsity exploiting Newton method for minimization*, Sparse Matrices and Their Uses (I. S. Duff, ed.), Academic Press, London, 1981, pp. 57–88. 37, 64

[Udr94] C. Udrişte, *Convex functions and optimization methods on Riemannian manifolds*, Kluwer Academic Publishers, Dordrecht, 1994. 20, 23, 36

[WD05] Jérome M. B. Walmag and Éric J. M. Delhez, *A note on trust-region radius update*, SIAM J. on Optimization **16** (2005), no. 2, 548–562. 33

[Yan07] Y. Yang, *Globally convergent optimization algorithms on Riemannian manifolds: Uniform framework for unconstrained and constrained optimization*, Journal of Optimization Theory and Applications **132** (2007), no. 2, 245–265. 23

BIOGRAPHICAL SKETCH

Christopher Grover Baker

Christopher Grover Baker, son of Frank and Lynn Baker, was born on September 5, 1979 in Marianna, FL. In the spring of 2002, he graduated *magna cum laude* from Florida State University, earning a bachelor's degree in Pure Mathematics and Computer Science. Under the advisement of Professor Kyle Gallivan, he obtained his Master's degree in the summer of 2004, also from the Department of Computer Science at Florida State University. He enrolled in the doctoral program in the Computer Science department. In the summer of 2005, Christopher began the first of three internships at Sandia National Laboratories in Albuquerque, NM. In the Fall of 2007, he transferred to the School of Computational Science. Under the advisement of Professor Kyle Gallivan, he completed his Ph.D. in the summer of 2008.

Christopher's research involves a number of topics in numerical linear algebra and high-performance computing. His master's thesis concerned low-rank incremental methods for computing singular subspaces, and his recent work has better illuminated the workings of that class of methods. With Absil and Gallivan, he made significant contributions to the field of Riemannian optimization, including the development of retraction-based Riemannian optimization and the family of Riemannian trust-region methods. One application of these methods resulted in the development of a novel family of high-performance, robust solvers for large-scale symmetric eigenvalue problems.

After his Ph.D., Christopher accepted a postdoctoral research position at Sandia National Laboratories. He currently lives in Albuquerque, N.M., with his spouse, Kelly Baker, a graduate of the doctoral program in the Department of Religious Studies at Florida State University. They have two dogs and one cat.