Riemannian Manifold Trust-Region Methods With Applications to Eigenproblems

Christopher G. Baker

Advisors:
Kyle Gallivan[1]
Pierre-Antoine Absil[2]

[1] School of Computational Science
Florida State University

[2] Département d'ingénierie mathématique
Université catholique de Louvain

May 22nd, 2008
10 AM EDT

(4 PM in Paris)

What is Riemannian optimization?

Definition

Riemannian Optimization refers to the optimization of an objective function over a Riemannian manifold.

Given a Riemannian manifold $\mathcal{M}$ and a smooth function

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

the goal is to find an extreme point:

$$\min_{x \in \mathcal{M}} f(x) \quad \text{or} \quad \max_{x \in \mathcal{M}} f(x)$$

Examples of Riemannian optimization problems

Riemannian optimization problems are best identified by the involvement of a Riemannian manifold.

The usual suspects

- set of linear subspaces:

$$\mathrm{Grass}(p, n, \mathbb{R}) = p \text{ dimensional subspaces of } \mathbb{R}^n$$

- set of (orthonormal) linear bases

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

- set of orthogonal matrices

$$\mathrm{O}(n, \mathbb{R}) = \{Q \in \mathbb{R}^{n \times n} : Q^T Q = QQ^T = I_n\}$$

Examples of Riemannian problems

Subspace Optimization

- H_2-optimal model reduction of MIMO systems [Absil, Gallivan, Van Dooren]
- Interpolation of linear ROMs across parameter changes [Amsallem, Farhat and Lieu]
- Optimal linear subspace for face recognition [Liu, Srivastava, Gallivan]
- Computing solutions of generalized eigenvalue problems: $KX = MX\Lambda$

Basis Optimization

- Computing dominant singular vectors/values
- Computing optimal rank tensor factorizations
- ICA

Orthogonal Group Optimization

- Pose estimation
- Motion recovery

Isn't this just constrained Euclidean optimization?

Why bother with manifolds?

- You have no choice.
  - There may be no efficient embedding.
- You don't like constrained optimization.
  - Riemannian optimization methods are feasible.
  - Unconstrained Riemannian optimization methods have "simpler" theory.

The difference

Riemannian optimization can be thought of as an unconstrained optimization in a constrained search space.

Outline

Iterations on the manifold

Consider the following generic update for an iterative Euclidean optimization algorithm:

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

It is implemented in numerous ways, e.g.:

- Newton's method: $x_{k+1} = x_k - \alpha_k \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$
- Steepest descent: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$

To Do

We need Riemannian concepts describing directions and movement on the manifold.

What is a Riemannian manifold?

Definition

A Riemannian manifold is a differentiable manifold endowed with a Riemannian metric.

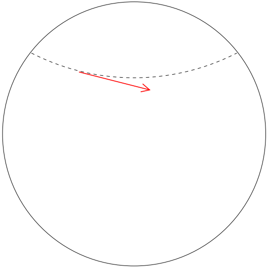
Why Riemannian Manifolds?

The combination of these provides the tools necessary to conduct optimization on a manifold:

- topology
- calculus
- geometry

Tangent vectors to the rescue

- The concept of direction is provided by tangent vectors.
- Intuitively, tangent vectors are tangent to curves on the manifold.
- Tangent vectors are an intrinsic property of a differentiable manifold.

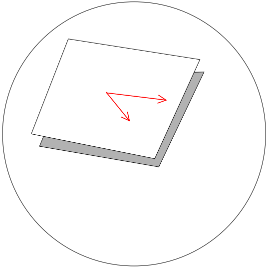


Definition

The tangent space $T_x\mathcal{M}$ is the vector space comprised of the tangent vectors at $x \in \mathcal{M}$.

Tangent vectors to the rescue

- The concept of direction is provided by tangent vectors.
- Intuitively, tangent vectors are tangent to curves on the manifold.
- Tangent vectors are an intrinsic property of a differentiable manifold.



Definition

The tangent space $T_x\mathcal{M}$ is the vector space comprised of the tangent vectors at $x \in \mathcal{M}$.

What is a Riemannian metric?

Definition

A Riemannian metric is a symmetric bi-linear mapping on the tangent spaces, which varies smoothly from point to point:

$$\begin{aligned} g_x &: T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R} \\ &: (\xi, \zeta) \mapsto g_x(\xi, \zeta) = \langle \xi, \zeta \rangle \end{aligned}$$

What is it good for?

The metric provides an inner product for the tangent spaces and a Riemannian geometric structure for the manifold $\mathcal{M}$.

Riemannian gradient and Riemannian Hessian

- We have calculus and geometry at our disposal.
- First step: the Riemannian gradient
- Second step: the Riemannian Hessian

Definition

The Riemannian gradient of f at x is the tangent vector in $T_x\mathcal{M}$ satisfying

$$\mathrm{D}\, f(x)[\eta] = \langle \operatorname{grad} f(x), \eta \rangle \ .$$

Definition

The Riemannian Hessian of f at x is a symmetric linear operator from $T_x\mathcal{M}$ to $T_x\mathcal{M}$ defined as

$$\operatorname{Hess} f(x)[\eta] = \mathrm{D} \operatorname{grad} f(x)[\eta] \ .$$

Geodesics: Straight lines in a curvy world

What are they good for?

- Tangent vectors describe directions on the manifold.
- Geodesics describe a mechanism for movement.

Definition

A geodesic is a curve on the manifold with zero acceleration.

- Geodesic γ embodies many ideal properties:
  - distance minimizing curve between points
  - uniquely defined w.r.t. a tangent vector
  - Homogeneous: $\gamma(t; x, \eta) = \gamma(1; x, t\eta)$
  - analogous to straight lines

The exponential map

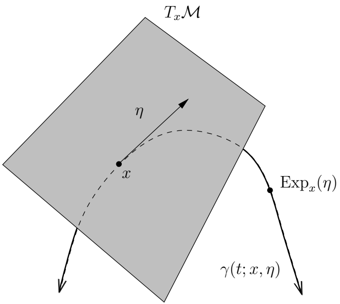
Definition

A point $x \in \mathcal{M}$, the exponential mapping Exp_x is a one-to-one mapping between a neighborhood of x and a subset of the tangent space $T_x\mathcal{M}$:

$$\mathrm{Exp}_x(\eta) = \gamma(1; x, \eta)$$

What is it good for?

- The exponential map allows us to map tangent vectors to nearby points on the manifold.
- We are now fully equipped to describe some iterations.



Back to the old school

Steepest descent

1. compute steepest descent direction $s = -\nabla f(x)$
2. find step size α
3. $x_+ = x + \alpha s$

[HM94][Udr94]

Newton's method

1. compute Newton update $s = -\left[\nabla^2 f(x)\right]^{-1} \nabla f(x)$
2. find step size α
3. $x_+ = x + \alpha s$

[Lue72, Gab82, Udr94, EAS98, MM02, ADM+02, DPM03, HT04]

Back to the old school

Steepest descent on a manifold.

1. compute steepest descent direction $\eta = -\mathrm{grad}\, f(x)$
2. find step size α
3. $x_+ = \mathrm{Exp}_x(\alpha\eta)$

[HM94][Udr94]

Newton's method

1. compute Newton update $s = -\left[\nabla^2 f(x)\right]^{-1} \nabla f(x)$
2. find step size α
3. $x_+ = x + \alpha s$

[Lue72, Gab82, Udr94, EAS98, MM02, ADM+02, DPM03, HT04]

Back to the old school

Steepest descent on a manifold.

1. compute steepest descent direction $\eta = -\mathrm{grad}\, f(x)$
2. find step size α
3. $x_+ = \mathrm{Exp}_x(\alpha\eta)$

[HM94][Udr94]

Newton's method

1. compute Newton update $s = -\left[\nabla^2 f(x)\right]^{-1} \nabla f(x)$
2. find step size α
3. $x_+ = x + \alpha s$

[Lue72, Gab82, Udr94, EAS98, MM02, ADM+02, DPM03, HT04]

Back to the old school

Steepest descent on a manifold.

1. compute steepest descent direction $\eta = -\mathrm{grad}\, f(x)$
2. find step size α
3. $x_+ = \mathrm{Exp}_x(\alpha\eta)$

[HM94][Udr94]

Newton's method on a manifold.

1. compute Newton update $\eta = -\left[\mathrm{Hess}\, f(x)\right]^{-1} \mathrm{grad}\, f(x)$
2. find step size α
3. $x_+ = \mathrm{Exp}_x(\alpha\eta)$

[Lue72, Gab82, Udr94, EAS98, MM02, ADM+02, DPM03, HT04]

Why Riemannian trust-region methods?

Beg the question

Why Euclidean trust-region methods?

Improved convergence theory and performance

- Robust global convergence of steepest descent
- Fast local convergence of Newton methods
- Avoids expensive linear solves of Newton methods

Trust-region methods

Operation of trust-region methods

Work on a model inside a region of tentative trust

1. At iterate x, construct (quadratic) model m_x of f around x
2. Find (approximate) solution to

$$s^* = \underset{\|s\| \leq \Delta}{\operatorname{argmin}}\ m_x(s)$$

3. Compute $\rho_x(s)$:

$$\rho_x(s) = \frac{f(x) - f(x+s)}{m_x(0) - m_x(s)}$$

4. Use $\rho_x(s)$ to adjust Δ and accept/reject proposed iterate:

$$x_+ = x + s$$

Needs for Riemannian trust-region method

Trust-region requirements

A Riemannian trust-region method needs:

- theoretical setting for constructing model
- tractable setting for conducting the model minimization
- preservation of convergence theory

How about the exponential map?

The exponential map provides some of this, with drawbacks:

- computationally expensive
- unnecessarily specific

Relaxing the exponential: Retractions

Definition

A retraction is a mapping R from $T\mathcal{M}$ to $\mathcal{M}$ satisfying the following:

- R is continuously differentiable
- $R_x(0) = x$
- $\mathrm{D}\, R_x(0)[\eta] = \eta$ "First-order rigidity"

What is it good for?

- mapping tangent vectors back to the manifold
- lifting the objective function f from $\mathcal{M}$ to $T_x\mathcal{M}$, via the pullback

$$\hat{f}_x = f \circ R_x$$

Retraction-based Riemannian optimization

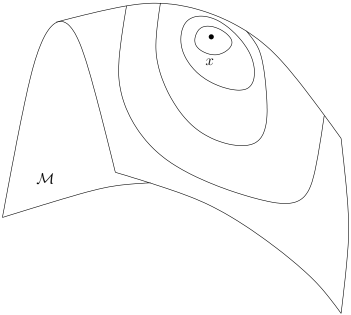
A novel optimization paradigm

Q: How do we conduct optimization on a manifold?

A: We do it in the tangent spaces.

Exponential vs. Retraction

- Previously: exponential map conducts movement on the manifold
- Instead: Use a general retraction to lift f to the tangent space
  - Can easily employ classical optimization techniques
  - Less expensive than the exponential map
  - Generality does not compromise the important theory



Retraction-based Riemannian optimization

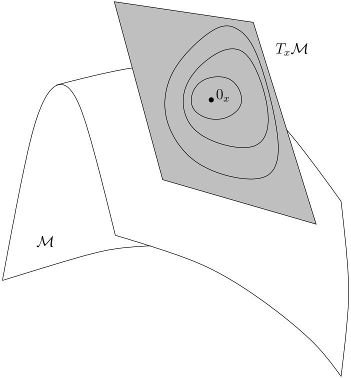
A novel optimization paradigm

Q: How do we conduct optimization on a manifold?

A: We do it in the tangent spaces.

Exponential vs. Retraction

- Previously: exponential map conducts movement on the manifold
- Instead: Use a general retraction to lift f to the tangent space
  - Can easily employ classical optimization techniques
  - Less expensive than the exponential map
  - Generality does not compromise the important theory



Optimality conditions

Equivalence of the pullback $\hat{f}_x = f \circ R_x$

	Exp_x	R_x
$\mathrm{grad}\, f(x) = \mathrm{grad}\, \hat{f}_x(0)$	yes	yes
$\mathrm{Hess}\, f(x) = \mathrm{Hess}\, \hat{f}_x(0)$	yes	no
$\mathrm{Hess}\, f(x) = \mathrm{Hess}\, \hat{f}_x(0)$ at critical points	yes	yes

Riemannian Sufficient Optimality Conditions

If $\mathrm{grad}\, \hat{f}_x(0) = 0$ and $\mathrm{Hess}\, \hat{f}_x(0) > 0$,
then $\mathrm{grad}\, f(x) = 0$ and $\mathrm{Hess}\, f(x) > 0$,
so that x is a local minimizer of f.

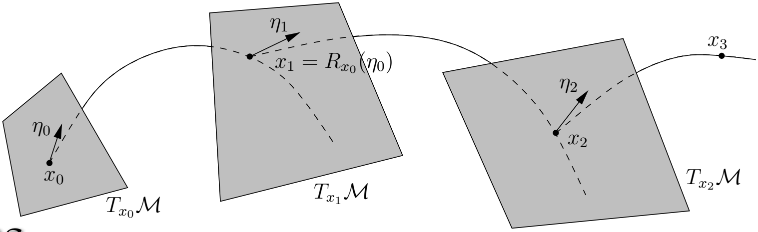
New approach

Generic Riemannian Optimization Algorithm

1. At iterate $x \in \mathcal{M}$, define $\hat{f}_x = f \circ R_x$
2. Find minimizer η of $\hat{f}_x$
3. Choose new iterate $x_+ = R_x(\eta)$
4. Goto step 1

A suitable setting

This paradigm is sufficient for describing trust-region methods.



Riemannian trust-region method

Operation of RTR

RTR operates in an analogous manner to Euclidean trust-region methods.

1a. At iterate x, define pullback $\hat{f}_x = f \circ R_x$

1. Construct quadratic model m_x of f around x
2. Find (approximate) solution to

$$\eta = \operatorname*{argmin}_{\|\eta\| \leq \Delta} m_x(\eta)$$

3. Compute $\rho_x(\eta)$:

$$\rho_x(\eta) = \frac{f(x) - f(x+\eta)}{m_x(0) - m_x(\eta)}$$

4. Use $\rho_x(\eta)$ to adjust Δ and accept/reject new iterate:

$$x_+ = x + \eta$$

Riemannian trust-region method

Operation of RTR

RTR operates in an analogous manner to Euclidean trust-region methods.

1a. At iterate x, define pullback $\hat{f}_x = f \circ R_x$

1b. Construct quadratic model m_x of $\hat{f}_x$

2. Find (approximate) solution to

$$\eta = \operatorname*{argmin}_{\eta \in T_x\mathcal{M},\, \|\eta\| \leq \Delta} m_x(\eta)$$

3. Compute $\rho_x(\eta)$:

$$\rho_x(\eta) = \frac{\hat{f}_x(0) - \hat{f}_x(\eta)}{m_x(0) - m_x(\eta)}$$

4. Use $\rho_x(\eta)$ to adjust Δ and accept/reject new iterate:

$$x_+ = R_x(\eta)$$

How to solve the model minimization?

$$\min_{\eta \in T_x\mathcal{M},\, \|\eta\| \leq \Delta} m_x(\eta)$$

Possible choices

Abstract Euclidean space supports many different algorithms:

- exact solution [Moré and Sorensen,1983]
- truncated conjugate gradient [Steihaug83][Toint81]
- truncated Lanczos [Gould *et al.*, 1999]
- ...

Truncated Conjugate Gradient

Simple modifications to the classical CG:

- trust-region membership is actively monitored
- directions of negative curvature are followed to the edge
- convergence tailored to the needs of the outer iteration

Convergence properties of RTR

Preserves convergence

Convergence of RTR is equivalent to that of Euclidean trust-region methods.

Global convergence

Under very mild smoothness assumptions:

- Global convergence to a stationary point.
- Stable convergence only to local minimizers.

Local convergence

For RTR/tCG:

- Every non-degenerate local minimizer $v \in \mathcal{M}$ has a neighborhood of attraction.
- If $m_x \approx \hat{f}_x$, asymptotic convergence is superlinear.

Classical TR mechanism

Drawbacks of classical trust-region mechanism

- Trust-region radius is heuristic
  - TR radius is based on the performance of the previous iterations.
    - radius too large $\longrightarrow$ rejected iterates
    - radius too small $\longrightarrow$ progress is impeded
  - can take some time to adjust
- Rejections are expensive!

Solutions

The solutions involve modifying the trust-region mechanism while preserving good convergence properties:

- more complicated TR radius updates [Conn, Gould, Toint, 2000]
- filter trust-region method of [Gould, Sainvitu, Toint, 2005]
- implicit trust-region [Baker, Absil, Gallivan, 2008]

Implicit Riemannian Trust-Region Method

A new optimization algorithm

The implicit Riemannian trust-region (IRTR) method uses a different trust-region definition:

$$\text{TR at } x = \{\eta \in T_x\mathcal{M} : \rho_x(\eta) \geq \rho'\}$$

where

$$\rho_x(s) = \frac{\hat{f}_x(0) - \hat{f}_x(s)}{m_x(0) - m_x(s)}$$

Effect

- TR mechanism replaced by a meaningful measure of model performance
- Accept/reject mechanism is discarded.
- Modification to trust-region requires revisiting model minimization and convergence theory.

IRTR Model Minimization

Interplay between trust-region and truncated CG

Trust-region definition comes into play when:

- checking that an iterate is in the trust-region

$$\rho_x(\eta) \geq \rho'$$

- following a search direction to the edge

$$\text{find } \tau > 0 \text{ s.t. } \rho_x(\eta + \tau\delta) = \rho'$$

The practical significance?

Requires an efficient relationship with $\rho_x(\eta)$:

- an analytical formula for $\rho_x(\eta)$, or
- an efficient evaluation of $\rho_x(\eta)$ combined with direct search

The latter assumes that evaluating f is not expensive.

Ingredients for RTR/IRTR

What do we need to apply RTR?

- Riemannian manifold $(\mathcal{M}, g)$, smooth function $f : \mathcal{M} \to \mathbb{R}$
- efficient representation for points $x \in \mathcal{M}$
- efficient representation for points $\eta \in T_x\mathcal{M}$
- tractable retraction R from $T_x\mathcal{M}$ to $\mathcal{M}$
- formula for $\operatorname{grad} f(x)$
- formula for $\operatorname{Hess} \hat{f}_x(0)$

Additional requirements for IRTR

- efficient formula for evaluating $\rho_x(\eta)$

Generalized eigenvalue problem

Generalized Eigenvalue Problem

Symmetric A, s.p.d. B, give rise to n eigenpairs:

$$Av_i = Bv_i\lambda_i, \qquad \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$$

Many application require only p extreme eigenpairs:

$$(v_1, \lambda_1), \ldots, (v_p, \lambda_p)$$

Generalized Eigenvalue Optimization Problem

$V = \begin{bmatrix} v_1 & \ldots & v_p \end{bmatrix}$ minimizes generalized Rayleigh quotient:

$$\mathrm{GRQ}(X) = \mathrm{trace}\left(\left(X^T B X\right)^{-1} X^T A X \right)$$

Generalized eigenvalue problem

Generalized Eigenvalue Problem

Symmetric A, s.p.d. B, give rise to n eigenpairs:

$$Av_i = Bv_i\lambda_i, \qquad \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$$

Many application require only p extreme eigenpairs:

$$(v_1, \lambda_1), \ldots, (v_p, \lambda_p)$$

Generalized Eigenvalue Optimization Problem

$V = \begin{bmatrix} v_1 & \ldots & v_p \end{bmatrix}$ minimizes generalized Rayleigh quotient:

$$\mathrm{GRQ}(X) = \mathrm{trace}\left(\left(X^T B X\right)^{-1} X^T A X\right)$$

Optimization approach to eigenvalue problem

Newton's method for GRQ

Consider solving the optimization problem with Newton's method.

- Newton's method fails!
- Newton update at X yields $2X$.
  - $X \mapsto 2X \mapsto 4X \mapsto \dots$
- This is because $\mathrm{GRQ}(X) = \mathrm{GRQ}(XM)$ for non-singular M.

Solutions?

A. Introduce constraints on the domain of GRQ.

B. Recognize the Riemannian optimization problem and apply an appropriate solver (e.g., RTR/IRTR).

Optimization approach to eigenvalue problem

Newton's method for GRQ

Consider solving the optimization problem with Newton's method.

- Newton's method fails!
- Newton update at X yields $2X$.
  - $X \mapsto 2X \mapsto 4X \mapsto \ldots$
- This is because $\mathrm{GRQ}(X) = \mathrm{GRQ}(XM)$ for non-singular M.

Solutions?

A. Introduce constraints on the domain of GRQ.

B. Recognize the Riemannian optimization problem and apply an appropriate solver (e.g., RTR/IRTR).

Riemannian Optimization Eigenvalue Problem

Riemannian setting

- GRQ is invariant to choice of basis, varies only with subspace
- Manifold is the set of p-dimensional subspaces of $\mathbb{R}^n$
  - This is the Grassmann manifold $\mathrm{Grass}(p, n)$
- $\mathrm{GRQ} : \mathrm{Grass}(p, n) \to \mathbb{R} : \mathrm{span}(X) \mapsto \mathrm{trace}\left(\left(X^T B X\right)^{-1} \left(X^T A X\right)\right)$
- $\mathrm{span}(X)$ represented by any basis X
- $R_{\mathrm{span}(X)}(S) = \mathrm{span}(X + S)$

Tangent vectors and Riemannian metric

The choice of representation allows significant variety in implementation:

- GRQ + Riemannian Newton $\Rightarrow$ Jacobi-Davidson [SVdV96]
- GRQ + Riemannian CG $\Rightarrow$ LOBPCG [Kny2001]
- GRQ + Riemannian TR $\Rightarrow$ Exciting new eigensolvers!!!

Riemannian Trust-Region + GRQ

Setting

- $T_{\text{span}(X)}\,\text{Grass}(p,n) = \left\{S \in \mathbb{R}^{n\times p} : S^T B X = 0\right\}$
- $g_{\text{span}(X)}(S,U) = \text{trace}\left((X^T B X)^{-1} S^T U\right)$
- $\text{grad}\,\hat{f}_{\text{span}(X)}(0) = 2 P_{BX} A X$
- $\text{Hess}\,\hat{f}_{\text{span}(X)}(0)[S] = 2 P_{BX}\left(AS - BSX^T AX\right)$

RTR

- Conditions on f and R are satisfied for global convergence.
- tCG solver enables superlinear (cubic!) rate of local convergence.

IRTR

Efficient implementation of IRTR requires analysis of $\rho_X(S)$

- This requires choosing a quadratic model.

A Tale of Two Models

Quadratic Model

Quadratic model $m_X \approx \hat{f}_X$ leaves freedom:

$$\begin{aligned} m_X(S) &= \mathrm{GRQ}(X) + \langle S, \operatorname{grad} \mathrm{GRQ}(X)\rangle + \frac{1}{2}\langle S, H_X[S]\rangle \\ &= \operatorname{trace}\left(X^T A X\right) + 2 \operatorname{trace}\left(S^T P_{BX} A X\right) + \frac{1}{2}\operatorname{trace}\left(S^T H_X[S]\right) \end{aligned}$$

Model Hessian

- Newton model:

$$H_X[S] = 2\, P_{BX}\left(AS - BSX^T AX\right)$$

- TRACEMIN model:

$$H_X[S] = 2\, P_{BX} A P_{BX} S$$

TRACEMIN Model Analysis

ρ Analysis: TRACEMIN

Authors of Trace Minimization method [SW82][ST00] show that

$$\mathrm{GRQ}(X+S) \leq m_X(S)$$

for all S producing a decrease in m_X. This implies

$$\rho_X(S) = \frac{\mathrm{GRQ}(X) - \mathrm{GRQ}(X+S)}{\mathrm{GRQ}(X) - m_X(S)} \geq 1$$

TRACEMIN Eigensolver

- TRACEMIN can be easily and efficiently implemented in the context of the IRTR.
- The minimization of this model is easily preconditioned.
- Asymptotic convergence is only linear.

Newton Model Analysis

ρ Analysis: Newton

$$\rho_X(S) = \frac{\operatorname{trace}\left(\left(I + S^T B S\right)^{-1}\left(\hat{M}\right)\right)}{\operatorname{trace}\left(\hat{M}\right)}$$

$$\hat{M} = S^T B S X^T A X - 2\, S^T A X - S^T A S$$

Two cases

- $p = 1$: Easy case

$$\rho_x(s) = (1 + s^T B s)^{-1}$$

so that

$$\rho_x(s) \geq \rho' \quad \Leftrightarrow \quad \|s\|_B^2 \leq \frac{1}{\rho'} - 1$$

- $p > 1$: Hard case yields no tractable formula

Newton Model IRTR

Approximation for $p > 1$

- If $X^T AX = \mathrm{diag}(\theta_1, \ldots, \theta_p)$, we can decouple model:

$$m_X(S) = \sum_j m_{x_j}(s_j)$$

- Then use $p = 1$ formula and approximate $\rho_X(S)$ via

$$\begin{aligned}\rho' &= \frac{\rho' \sum_j \left(m_{x_j}(0) - m_{x_j}(s_j)\right)}{\sum_j \left(m_{x_j}(0) - m_{x_j}(s_j)\right)} \leq \frac{\sum_j \left(\hat{f}_{x_j}(0) - \hat{f}_{x_j}(s_j)\right)}{\sum_j \left(m_{x_j}(0) - m_{x_j}(s_j)\right)} \\ &= \frac{\hat{f}_X(0) - \sum_j \hat{f}_{x_j}(s_j)}{m_X(0) - m_X(S)} \approx \frac{\hat{f}_X(0) - \hat{f}_X(S)}{m_X(0) - m_X(S)} = \rho_X(S)\end{aligned}$$

- This gives an IRTR-like eigensolver.

Newton Model IRTR

Approximation for $p > 1$

- If $X^T A X = \mathrm{diag}(\theta_1, \ldots, \theta_p)$, we can decouple model:

$$m_X(S) = \sum_j m_{x_j}(s_j)$$

- Then use $p = 1$ formula and approximate $\rho_X(S)$ via

$$\begin{aligned} \rho' &= \frac{\rho' \sum_j \left(m_{x_j}(0) - m_{x_j}(s_j)\right)}{\sum_j \left(m_{x_j}(0) - m_{x_j}(s_j)\right)} \leq \frac{\sum_j \left(\hat{f}_{x_j}(0) - \hat{f}_{x_j}(s_j)\right)}{\sum_j \left(m_{x_j}(0) - m_{x_j}(s_j)\right)} \\ &= \frac{\hat{f}_X(0) - \sum_j \hat{f}_{x_j}(s_j)}{m_X(0) - m_X(S)} \approx \frac{\hat{f}_X(0) - \hat{f}_X(S)}{m_X(0) - m_X(S)} = \rho_X(S) \end{aligned}$$

- This gives an IRTR-like eigensolver.

Adaptive Model IRTR

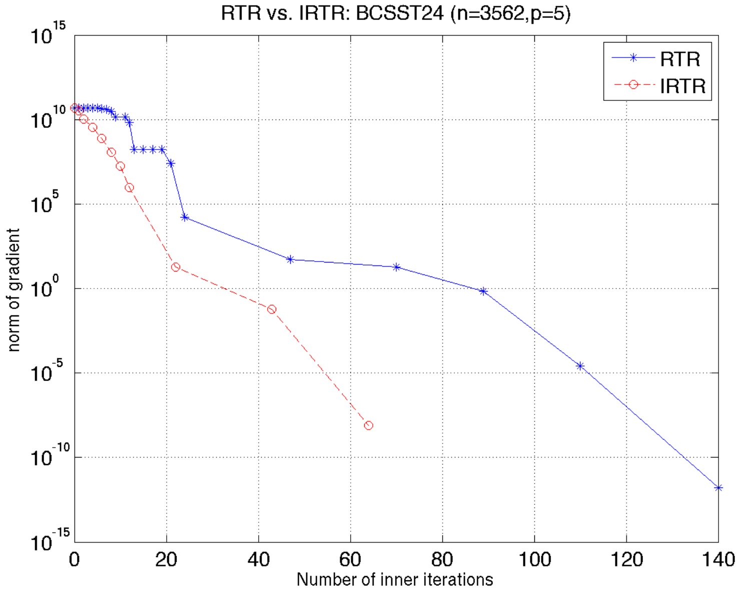
A Hybrid Method

Compute in two stages [ABGS05]:

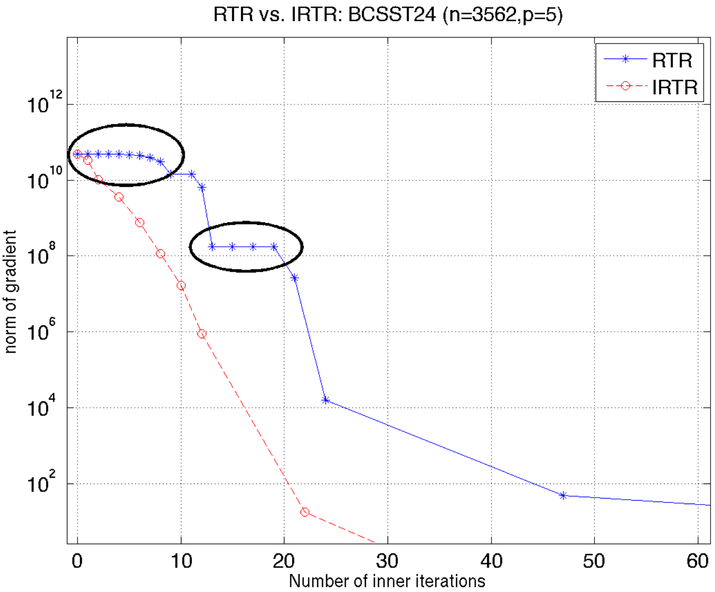
- Stage 1: Use TRACEMIN-model
  - quickly purge large eigenvalues
  - easily preconditioned
- Stage 2: Use Newton model
  - fast local convergence
  - heuristic-safe

The Best of Both...

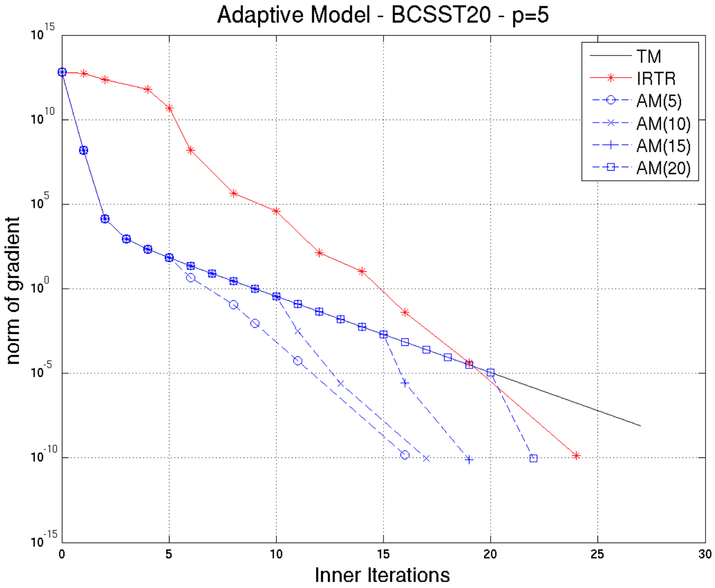
- easy and efficient iterations in stage 1
- stage 2 has fast convergence to the solution
- globally convergent by construction
- efficiency is tied to switching criteria



BCSST24 with Cholesky preconditioner



The trust-region radius can limit effectiveness of a good preconditioner, and rejections can stall progress.



BCSST20 with Cholesky preconditioner

Problem	Size	p	Prec	RTR	IRTR	LOBPCG
BCSST22	138	5	none	2.64	1.90	39.03
BCSST22	138	5	inexact	1.11	1.03	3.17
BCSST22	138	5	exact	0.29	0.24	0.45
BCSST20	485	5	inexact	49.04	34.40	*151.00
BCSST20	485	5	exact	0.11	0.08	0.14
BCSST13	2,003	25	exact	12.86	7.81	6.20
BCSST13	2,003	100	exact	79.41	56.95	56.12
BCSST23	3,134	25	exact	28.25	22.10	16.86
BCSST23	3,134	100	exact	168.76	129.06	180.40
BCSST24	3,562	25	exact	9.34	8.17	7.76
BCSST24	3,562	100	exact	98.23	69.83	108.20
BCSST25	15,439	25	exact	361.40	85.25	*3218.00

Timings (in seconds) in Trilinos/Anasazi (C++). Average speedup of IRTR w.r.t. RTR is 1.33; IRTR w.r.t. LOBPCG is 3.46.

Summary and Future work

Summary

- Described the retraction-based paradigm for Riemannian optimization
- Described the Riemannian trust-region method and its convergence properties
- Described the implicit Riemannian trust-region method and its convergence properties
- Applied the trust-region solvers to the computation of extreme eigenspaces

Future work

- Need more applications where IRTR can be put to efficient use
- Further analysis of $\rho_X(S)$ for eigenvalue problem
  - may yield workable formula
  - should show current approximation is sufficient for convergence

Impact

Software Efforts

- Generic RTR (GenRTR) package (MATLAB)
  http://www.scs.fsu.edu/~cbaker/GenRTR/
- RTR/ESGEV solvers (MATLAB and Anasazi/C++)
  http://www.scs.fsu.edu/~cbaker/RTRESGEV/
- RTR/TSVD solvers (RBGen/C++)
  http://trilinos.sandia.gov/

Papers

- Absil, Baker, Gallivan: *"A truncated-CG style method for symmetric generalized eigenvalue problems"* (JCAM,2006)
- Absil, Baker, Gallivan: *"Trust-region methods on Riemannian manifolds"* (FoCM,2007)
- Baker, Absil, Gallivan: *"An implicit trust-region method on Riemannian manifolds"* (IMAJNA,2008)

Acknowledgments

Thanks...

- The committee: Profs. Absil, Erlebacher, Gallivan, Hussaini, Krothapalli, Srivastava
- Clayton Webster, for the FSU/SCS Beamer template
- Dept. of Computer Science and School of Computational Science

Funding

- NSF Grants OCI0324944 and CCR9912415
- School of Computational Science, FSU
- 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.