An Implicit Riemannian Trust-Region Method

Christopher G. Baker[1,2] Pierre-Antoine Absil[3]

Kyle Gallivan[2]

[1]Computer Science Research Institute
Sandia National Laboratories

[2]School of Computational Science
Florida State University

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

May 2008 / SIOPT

Acknowledgments

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.

What is Riemannian optimization?

Definition

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

Objective

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

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

the goal is to find an extreme point:

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

Methods for Riemannian Optimization

- Riemannian Steepest Descent Method [HM94][Udr94]
- Riemannian Newton Method [Lue72, Gab82, Udr94, EAS98, MM02, ADM+02, DPM03, HT04]

Riemannian Steepest Descent Method

- Search along manifold in direction of steepest descent: $-\text{grad } f(x)$
- Robust global convergence
- Slow local convergence: linear

Riemannian Newton Method

- Search along manifold in Newton direction: $-[\text{Hess } f(x)]^{-1}\text{grad } f(x)$
- Fast local convergence: quadratic or even cubic
- Requires additional work for global convergence

Trust-region methods

Trust-region methods combine the benefits of Newton and steepest descent:

- the robust global convergence of steepest descent
- the fast local convergence of Newton's method
- no exact linear solves

Riemannian Trust-Region Method

- Riemannian trust-region (RTR) method [ABG07]
- Retains convergence properties of Euclidean trust-region methods
- Promising application to generalized eigenvalue problems [ABG06]

Drawbacks

- Trust-region radius is a heuristic
- Radius slow to adjust, rejections hurt efficiency

Outline

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:

$$x_+ = x + s$$

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

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

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?

- lift the objective function f from $\mathcal{M}$ to $T_x\mathcal{M}$, via the pullback $\hat{f}_x = f \circ R_x$
- map tangent vectors back to the manifold

Needs for Riemannian trust-region method

Trust-region requirements

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

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?

- lift the objective function f from $\mathcal{M}$ to $T_x\mathcal{M}$, via the pullback $\hat{f}_x = f \circ R_x$
- map tangent vectors back to the manifold

A novel optimization paradigm

Question: How do we conduct optimization on a manifold?

Answer: We do it in the tangent spaces.

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. Go to step 1

Exponential vs. Retraction

- Previously: exponential map conducts movement on the manifold
- Retraction-based Riemannian optimization uses a general retraction to lift the objective function to the tangent space
  - Can employ classical optimization techniques
  - Retraction is less expensive than the exponential map
  - The increased generality does not compromise the important theory

A novel optimization paradigm

Question: How do we conduct optimization on a manifold?
Answer: We do it in the tangent spaces.

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. Go to step 1

Exponential vs. Retraction

- Previously: exponential map conducts movement on the manifold
- Retraction-based Riemannian optimization uses a general retraction to lift the objective function to the tangent space
  - Can employ classical optimization techniques
  - Retraction is less expensive than the exponential map
  - The increased generality does not compromise the important theory

A novel optimization paradigm

Question: How do we conduct optimization on a manifold?
Answer: We do it in the tangent spaces.

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. Go to step 1

Exponential vs. Retraction

- Previously: exponential map conducts movement on the manifold
- Retraction-based Riemannian optimization uses a general retraction to lift the objective function to the tangent space
  - Can employ classical optimization techniques
  - Retraction is less expensive than the exponential map
  - The increased 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.

A suitable setting

This paradigm is sufficient for implementing trust-region methods.

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.

A suitable setting

This paradigm is sufficient for implementing trust-region methods.

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.

A suitable setting

This paradigm is sufficient for implementing trust-region methods.

Riemannian trust-region method

Operation of RTR

RTR operates in an analogous manner to ETR methods, replacing Euclidean concepts with Riemannian analogues.

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

$$\eta = \underset{\|\eta\| \leq \Delta}{\operatorname{argmin}} \; m_x(\eta)$$

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 ETR methods, replacing Euclidean concepts with Riemannian analogues.

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

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

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

Outline

Classical TR mechanism

Drawbacks of classical trust-region mechanism

- Trust-region radius is heuristic
  - Radius is based on performance of previous iterations
  - 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 radius updates [Conn, Gould, Toint, 2000]
- filter trust-region method of [Gould, Sainvitu, Toint, 2005]
- Implicit Trust-Region [Baker, Absil, Gallivan, 2008]

Classical TR mechanism

Drawbacks of classical trust-region mechanism

- Trust-region radius is heuristic
  - Radius is based on performance of previous iterations
  - 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 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

- Classical 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

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

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.

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

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

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.

Convergence properties of IRTR

Convergence equivalent to RTR and Euclidean trust-region methods

Global convergence

Under very mild smoothness assumptions:

- Global convergence to stationary points
- Stable convergence only to local minimizers

Local convergence

For IRTR/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

Efficiency

Improved efficiency on problems where ρ is inexpensive

Ingredients for RTR/IRTR

What did 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 $\rho_x(\eta)$

Application: Computing Extreme Eigenspaces

Symmetric Generalized Eigenvalue Problem

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

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

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:

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

Application: Computing Extreme Eigenspaces

Symmetric Generalized Eigenvalue Problem

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

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

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:

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

Riemannian Optimization Eigenvalue Problem

Basis invariance

Given $X \in \mathbb{R}_*^{n \times p}$ and $M \in \mathbb{R}^{p \times p}$ non-singular,

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

This can cause problems with Euclidean approaches (e.g., P-A's talk)

Natural 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 $\mathrm{Grass}(p, n)$
- $\mathrm{span}(X)$ represented by any basis X
- $T_{\mathrm{span}(X)} \mathrm{Grass}(p, n) = \left\{ S \in \mathbb{R}^{n \times p} : S^T B X = 0 \right\}$

A Tale of Two Models

TRACEMIN Model

$$m_X(S) = \operatorname{trace}\left(X^T A X\right) + 2\operatorname{trace}\left(S^T A X\right) + \operatorname{trace}\left(S^T A S\right)$$
$$\rho_X(S) \geq 1, \quad \forall S \in T_X \operatorname{Grass}(p,n)$$

Trace Minimization method [SW82][ST00] implements IRTR.

- Asymptotic convergence is linear

Newton Model

$$m_X(S) = \operatorname{trace}\left(X^T A X\right) + 2\operatorname{trace}\left(S^T A X\right) + \operatorname{trace}\left(S^T A S - S^T B S X^T A X\right)$$
$$\rho_X(S) = \frac{\operatorname{trace}\left(\left(I + S^T B S\right)^{-1}\left(\hat{M}\right)\right)}{\operatorname{trace}\left(\hat{M}\right)}$$
$$\hat{M} = S^T B S X^T A X - 2\, S^T A X - S^T A S$$

A Tale of Two Models

TRACEMIN Model

$$m_X(S) = \operatorname{trace}\left(X^T A X\right) + 2\operatorname{trace}\left(S^T A X\right) + \operatorname{trace}\left(S^T A S\right)$$
$$\rho_X(S) \geq 1, \quad \forall S \in T_X \operatorname{Grass}(p,n)$$

Trace Minimization method [SW82][ST00] implements IRTR.

- Asymptotic convergence is linear

Newton Model

$$m_X(S) = \operatorname{trace}\left(X^T A X\right) + 2\operatorname{trace}\left(S^T A X\right) + \operatorname{trace}\left(S^T A S - S^T B S X^T A X\right)$$
$$\rho_X(S) = \frac{\operatorname{trace}\left(\left(I + S^T B S\right)^{-1}\left(\hat{M}\right)\right)}{\operatorname{trace}\left(\hat{M}\right)}$$
$$\hat{M} = S^T B S X^T A X - 2\, S^T A X - S^T A S$$

Newton model easy case: $p = 1$

$$\rho_x(s) = \frac{1}{1 + s^T B s} \quad \text{so that} \quad \rho_x(s) \geq \rho' \Leftrightarrow \|s\|_B^2 \leq \frac{1}{\rho'} - 1$$

Hard case: $p > 1$

- No tractable formula (yet). Instead, decouple model:

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

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

$$\rho' = \frac{\rho' \sum_j \left(m_{x_j}(0) - m_{x_j}(s_j)\right)}{\sum_j \left(m_{x_j}(0) - m_{x_j}(s_j)\right)} \leq \frac{\sum_j \left(\hat{f}_{x_j}(0) - \hat{f}_{x_j}(s_j)\right)}{\sum_j \left(m_{x_j}(0) - m_{x_j}(s_j)\right)}$$

$$= \frac{\hat{f}_X(0) - \sum_j \hat{f}_{x_j}(s_j)}{m_X(0) - m_X(S)} \approx \frac{\hat{f}_X(0) - \hat{f}_X(S)}{m_X(0) - m_X(S)} = \rho_X(S)$$

Newton model easy case: $p = 1$

$$\rho_x(s) = \frac{1}{1 + s^T B s} \quad \text{so that} \quad \rho_x(s) \geq \rho' \Leftrightarrow \|s\|_B^2 \leq \frac{1}{\rho'} - 1$$

Hard case: $p > 1$

- No tractable formula (yet). Instead, decouple model:

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

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

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

Newton model easy case: $p = 1$

$$\rho_x(s) = \frac{1}{1 + s^T B s} \quad \text{so that} \quad \rho_x(s) \geq \rho' \Leftrightarrow \|s\|_B^2 \leq \frac{1}{\rho'} - 1$$

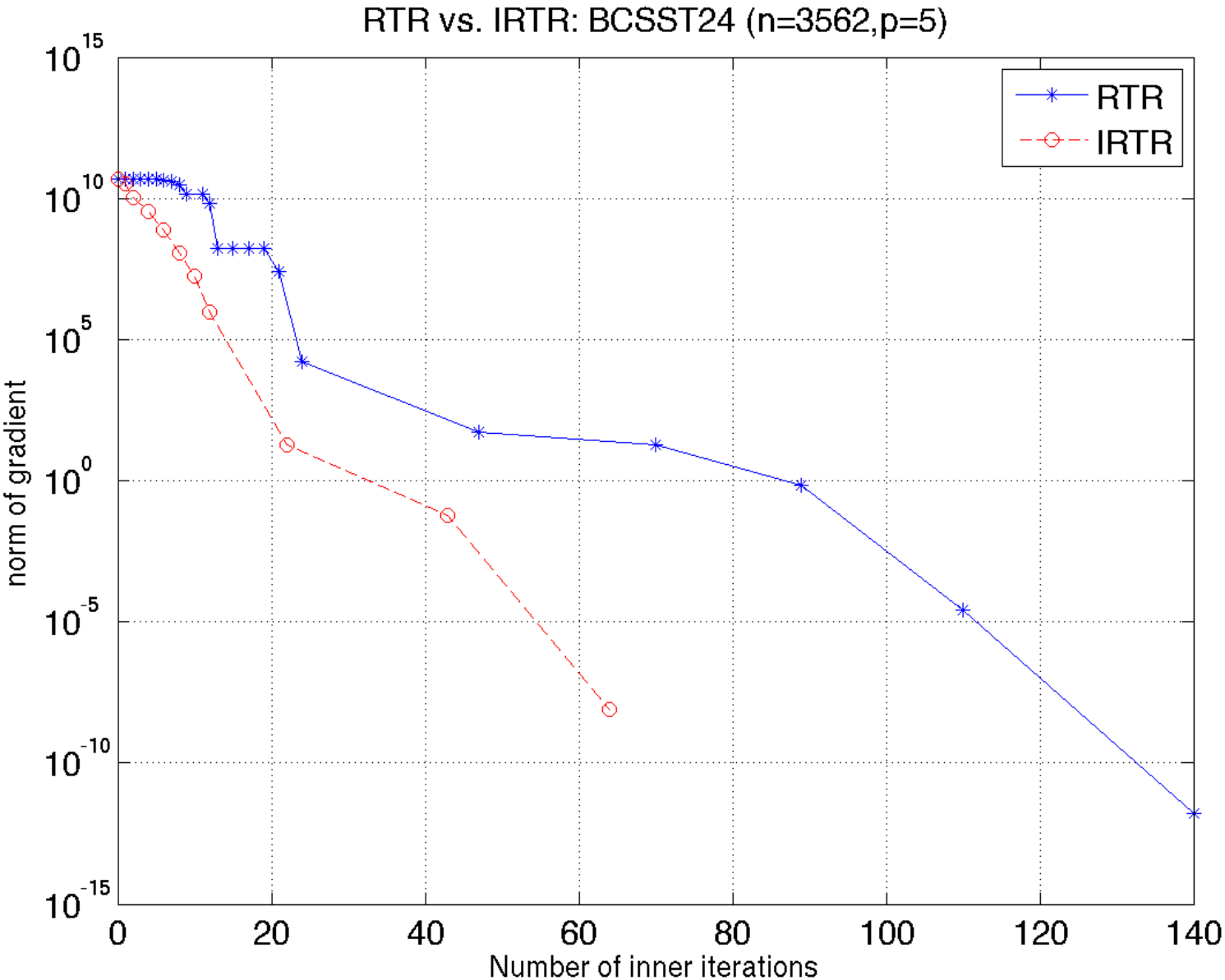
Hard case: $p > 1$

- No tractable formula (yet). Instead, decouple model:

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

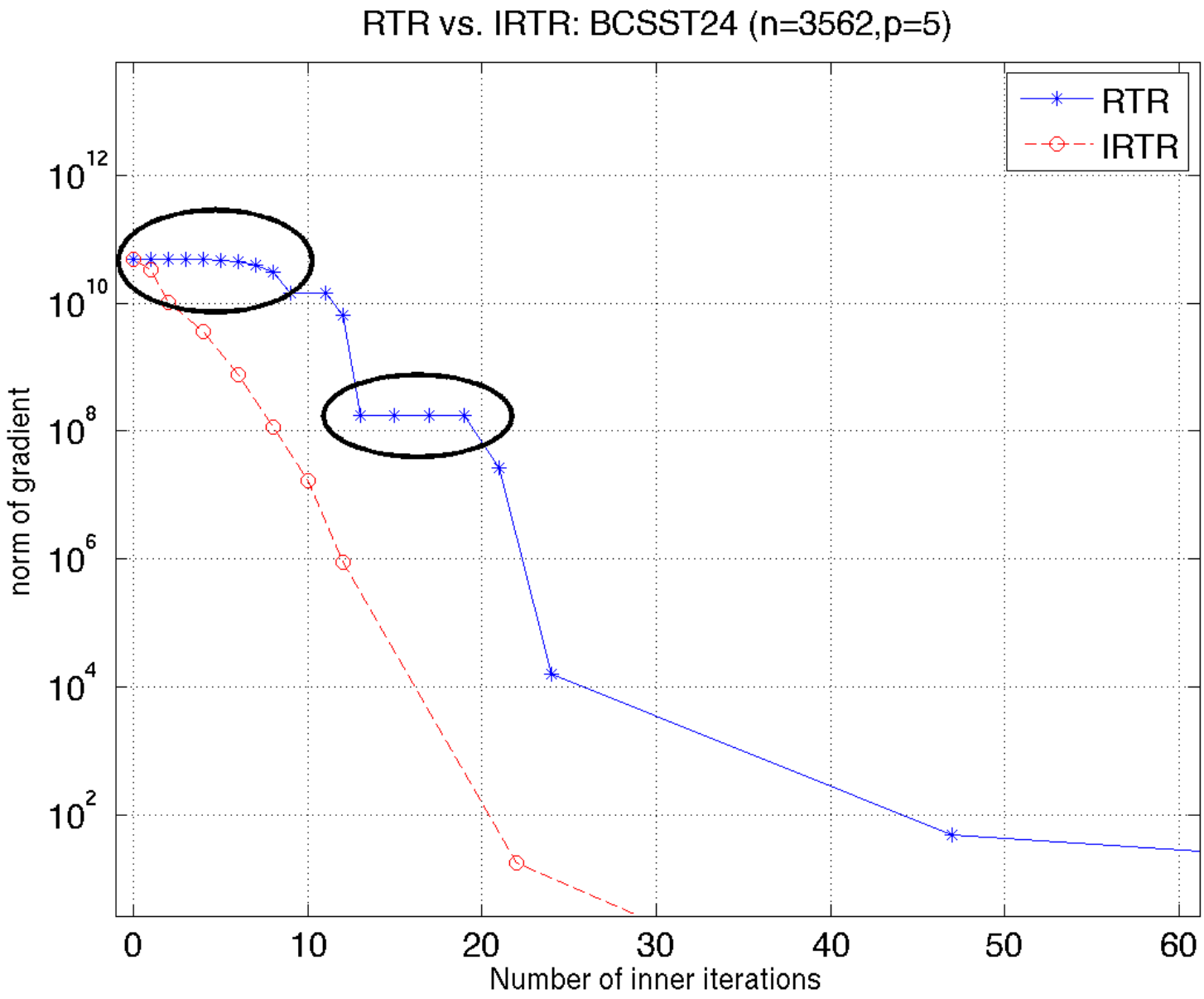
- Use $p = 1$ formula and approximate $\rho_X(S)$ via

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



BCSST24 (Calgary Olympic Saddledome) with Cholesky preconditioner

$\rho'_{\mathrm{rtr}} = 0.1,\ \rho'_{\mathrm{irtr}} = 0.5$



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

Problem	Size	p	Prec	RTR	IRTR
BCSST22	138	5	none	2.64	1.90
BCSST22	138	5	IC	1.11	1.03
BCSST22	138	5	LU	0.29	0.24
BCSST20	485	5	IC	49.04	34.40
BCSST20	485	5	LU	0.11	0.08
BCSST13	2003	25	LU	12.86	7.81
BCSST13	2003	100	LU	79.41	56.95
BCSST23	3134	25	LU	28.25	22.10
BCSST23	3134	100	LU	168.76	129.06
BCSST24	3562	25	LU	9.34	8.17
BCSST24	3562	100	LU	98.23	69.83
BCSST25	15439	25	LU	361.40	85.25

Timings (in seconds) in Trilinos/Anasazi (C++). Average speedup is 1.33.

Conclusions and Future work

Conclusion

- Necessary conditions for surrogate minimization
- Modification to the trust-region mechanism
- Resulting algorithm addresses efficiency, preserves convergence theory
- General description in the context of Riemannian optimization
- Novel algorithm for computing 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

Conclusions and Future work

Conclusion

- Necessary conditions for surrogate minimization
- Modification to the trust-region mechanism
- Resulting algorithm addresses efficiency, preserves convergence theory
- General description in the context of Riemannian optimization
- Novel algorithm for computing 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

Software Efforts

- Generic RTR (GenRTR) package (MATLAB)
  http://www.scs.fsu.edu/~cbaker/GenRTR
- RTR/ESGEV solvers (MATLAB and Anasazi)
  http://www.scs.fsu.edu/~cbaker/RTRESGEV

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)