Riemannian Optimization for Computing Low-rank Solutions of Lyapunov Equations with a New Preconditioner

Wen Huang

Xiamen University

April 15, 2019

This is joint work with Bart Vandereycken at University of Geneva.
Problem Statement

**Generalized Lyapunov equation:** Given matrix $A$, $M$ and $C$, find $X$ such that

$$AXM^T + MXA^T = C$$  \hspace{1cm} (1)

**Applications:** signal processing, model reduction, and system and control theory. [Moo03, Ben06]
Problem Statement

Generalized Lyapunov equation: Given matrix $A$, $M$ and $C$, find $X$ such that

$$AXM^T + MXA^T = C$$

(1)

Applications: signal processing, model reduction, and system and control theory. [Moo03, Ben06]

Problem: We focus on the problem:

- $A$, $M$, $C \in \mathbb{R}^{n \times n}$ are symmetric;
- $A \succ 0$, $M \succ 0$ (positive definite), $C \succeq 0$ (positive semidefinite);
- $A$, $M$ are sparse;
- medium- to large-scale problems;
Problem Statement

\[ A \succ 0, \ M \succ 0 \text{ and } C \succeq 0, \ A, \ M, \text{ and } C \text{ are symmetric:} \]
\[ AXM + MXA - C = 0 \]

- \( X \) is not sparse, even \( A \) and \( M \) are sparse;
- How to solve it for large-scale problems?
Problem Statement

\[ A \succ 0, \ M \succ 0 \ \text{and} \ C \succeq 0, \ A, \ M, \ \text{and} \ C \ \text{are symmetric:} \]

\[ AXM + MXA - C = 0 \]

- \( X \) is not sparse, even \( A \) and \( M \) are sparse;
- How to solve it for large-scale problems? Low rank solution
Problem Statement

$A \succ 0, M \succ 0$ and $C \succeq 0$, $A$, $M$, and $C$ are symmetric:

$$AXM + MXA - C = 0$$

- $X$ is not sparse, even $A$ and $M$ are sparse;
- How to solve it for large-scale problems? **Low rank solution**
- Reasonable: For low rank $C$, the solution $X$ has low numerical rank [Pen00b]
Problem Statement

Existing Methods

Riemannian Approach

New Preconditioner and Riemannian Methods

Numerical Experiments

Summary

Existing Methods

\[ A \succ 0, \quad M \succ 0 \quad \text{and} \quad C \succeq 0, \quad A, \quad M, \quad \text{and} \quad C \quad \text{are symmetric:} \]

\[ AXM + MXA - C = 0 \]

Unique solution \( X \) and \( X = X^T, \quad X \succeq 0 \) [Pen98] \( \iff \) \( X = YY^T \)
Existing Methods

$A \succ 0, M \succ 0$ and $C \succeq 0$, $A$, $M$, and $C$ are symmetric:

$$AXM + MXA - C = 0$$

Unique solution $X$ and $X = X^T, X \succeq 0$ \cite{Pen98} $\implies X = YY^T$

- Alternating Direction Implicit Iteration (ADI) or Smith method;
- Krylov subspace technique;
- Optimization method;
Existing Methods

\[ A \succ 0, \quad M \succ 0 \quad \text{and} \quad C \succeq 0, \quad A, \quad M, \quad \text{and} \quad C \text{ are symmetric:} \]

\[ AXM + MXA - C = 0 \]

Unique solution \( X \) and \( X = X^T, X \succeq 0 \) [Pen98] \( \iff \) \( X = YY^T \)

- Alternating Direction Implicit Iteration (ADI) or Smith method;
- Krylov subspace technique;

Reformulate well-known iterative method to a low-rank setting. Work on the factor \( Y \) of \( X = YY^T \).
Existing Methods

A \succ 0, M \succ 0 and C \succeq 0, A, M, and C are symmetric:

\[ AXM + MXA - C = 0 \]

Unique solution X and \( X = X^T, X \succeq 0 \) [Pen98] \( \iff X = YY^T \)

- Alternating Direction Implicit Iteration (ADI) or Smith method;
- Krylov subspace technique;
- Optimization method;
Consider a cost function on the set of symmetric matrices:
- Cost function: \( F : \mathbb{S}^{n \times n} \to \mathbb{R} : X \mapsto \text{trace}(XAXM) - \text{trace}(XC) \);
- Gradient: \( AXM + MXA - C \);
- The critical point is unique [Pen98].
- Minimizer is the solution.
Consider a cost function on the set of symmetric matrices:

- Cost function: $F : \mathbb{S}^{n \times n} \rightarrow \mathbb{R} : X \mapsto \text{trace}(XAXM) - \text{trace}(XC)$;
- Gradient: $AXM + MXA - C$;
- The critical point is unique [Pen98].
- Minimizer is the solution.

Add low-rank constraints by fixing the rank to be $r$:

- Cost function: $f : \mathbb{S}^{n \times n}_r \rightarrow \mathbb{R} : X \mapsto \text{trace}(XAXM) - \text{trace}(XC)$;
- Gradient: $P_{TX} \mathbb{S}^{n \times n}_r (AXM + MXA - C)$;
- Minimizer can be viewed as a low-rank approximation of the solution;
Existing Riemannian Optimization technique [VV10]

Optimization problem on the symmetric positive semidefinite with rank $r$

$$\min_{X \in \mathbb{S}^{n \times n}_r} f(X) = \text{trace}(XAXM) - \text{trace}(XC)$$

- Ingredients for Riemannian optimization;
- Trust-region Newton method
- Preconditioner
Ingredients for Riemannian optimization

- Tangent space at $X = YY^T$ is

\[
T_X \mathbb{S}_r^{n \times n} = \left\{ \begin{bmatrix} Y & Y^\perp \\ N & 0 \end{bmatrix} \begin{bmatrix} 2S \\ N^T \\ 0 \end{bmatrix} \begin{bmatrix} Y^T \\ Y^\perp \end{bmatrix} \mid S \in \mathbb{S}_r^{r \times r}, N \in \mathbb{R}^{(n-r) \times r} \right\} \\
= \left\{ YZ^T + ZY^T \mid Z \in \mathbb{R}^{n \times r} \right\};
\]
Ingredients for Riemannian optimization

- Tangent space at \( X = YY^T \) is \( \{ YZ^T + ZY^T \mid Z \in \mathbb{R}^{n \times r} \} \);
- Riemannian metric:
  \[
g_X(\eta_X, \xi_X) = \text{trace}(\eta_X^T \xi_X).
\]
  for any \( \eta_X, \xi_X \in T_X S_r^{n \times n} \);
Ingredients for Riemannian optimization

- Tangent space at $X = YY^T$ is $\{YZ^T + ZY^T \mid Z \in \mathbb{R}^{n \times r}\}$;
- Riemannian metric: $g_X(\eta_X, \xi_X) = \text{trace}(\eta_X^T \xi_X)$;
- Retraction:

$$R_X(\eta_X) = P_{\mathbb{S}_r^{n \times n}}(X + \eta_X),$$

where $P_{\mathbb{S}_r^{n \times n}}(Z) = \sum_{i=1}^{r} \sigma_i v_i v_i^T$, $Z = V\Sigma V$, $V = [v_1, \ldots, v_n]$, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$ and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$. 
Ingredients for Riemannian optimization

- Tangent space at $X = YY^T$ is $\{YZ^T + ZY^T \mid Z \in \mathbb{R}^{n \times r}\}$;
- Riemannian metric: $g_X(\eta_X, \xi_X) = \text{trace}(\eta_X^T \xi_X)$;
- Retraction: $R_X(\eta_X) = P_{S_{r \times n}}(X + \eta_X)$;
- Riemannian gradient:
  \[
  \text{grad} \ f(X) = P_{TX} S_{r \times n}(AXM + MXA - C),
  \]
  where $P_{TX} S_{r \times n}(Z) = P_Y ZP_Y + P_{P_Y} ZP_Y + P_Y ZP_{P_Y}$, $P_{P_Y} = I - P_Y$ and $P_Y = Y(Y^TY)^{-1}Y^T$. 
Ingredients for Riemannian optimization

- Tangent space at $X = YY^T$ is $\{YZ^T + ZY^T \mid Z \in \mathbb{R}^{n \times r}\}$;
- Riemannian metric: $g_X(\eta_X, \xi_X) = \text{trace}(\eta_X^T \xi_X)$;
- Retraction: $R_X(\eta_X) = P_{S_r^{n \times n}}(X + \eta_X)$;
- Riemannian gradient: $\text{grad} \ f(X) = P_{T_X S_r^{n \times n}}(AXM + MXA - C)$;
- Action of the Riemannian Hessian:

$$\text{Hess} \ f(X)[\eta_X] = P_{T_X S_r^{n \times n}}(A\eta_X M + M\eta_X A)$$
$$+ P_{T_X S_r^{n \times n}} \left( D P_{T_X S_r^{n \times n}}[\eta_X](AXM + MXA - C) \right)$$
Riemannian Trust-region Newton method

1: for $k = 0, 1, 2, \ldots$ do
2:   Let $m_k(\eta) = f(X_k) + g_{X_k}(\text{grad} f(X_k), \eta) + \frac{1}{2} g_{X_k}(\text{Hess} f(X_k)[\eta], \eta)$;
3:   Obtain $\eta_k$ by approximately solving $\min_{\eta \in \mathfrak{T}_{X_k} \mathbb{S}^{n \times n}, ||\eta|| \leq \Delta_k} m_k(\eta)$;
4:   Compute $\rho_k = \frac{f(X_k) - f(R_{X_k}(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
5:   Set $X_{k+1} = R_{X_k}(\eta_k)$ if $\rho_k$ is sufficient large, Otherwise $X_{k+1} = X_k$;
6:   Set $\Delta_{k+1} = 2\Delta_k$ if $\rho_k$ is sufficient large;
7:   Set $\Delta_{k+1} = \Delta_k/4$ if $\rho_k$ is small;
8: end for
Riemannian Trust-region Newton method

1: \textbf{for} \ k = 0, 1, 2, \ldots \ \textbf{do}
2: \hspace{1em} \text{Let} \ m_k(\eta) = f(X_k) + g_{X_k} (\text{grad} \ f(X_k), \eta) + \frac{1}{2} g_{X_k} (\text{Hess} \ f(X_k)[\eta], \eta);
3: \hspace{1em} \text{Obtain} \ \eta_k \ \text{by approximately solving} \ \min_{\eta \in T_{X_k} \mathbb{S}_n, \|\eta\| \leq \Delta_k} m_k(\eta);
4: \hspace{1em} \text{Compute} \ \rho_k = \frac{f(X_k) - f(R_{X_k}(\eta_k))}{m_k(0) - m_k(\eta_k)};
5: \hspace{1em} \text{Set} \ X_{k+1} = R_{X_k}(\eta_k) \ \text{if} \ \rho_k \ \text{is sufficient large, Otherwise} \ X_{k+1} = X_k;
6: \hspace{1em} \text{Set} \ \Delta_{k+1} = 2\Delta_k \ \text{if} \ \rho_k \ \text{is sufficient large};
7: \hspace{1em} \text{Set} \ \Delta_{k+1} = \Delta_k / 4 \ \text{if} \ \rho_k \ \text{is small};
8: \hspace{1em} \textbf{end for}

- Build a local quadratic model;
Riemannian Trust-region Newton method

1: for $k = 0, 1, 2, \ldots$ do
2:   Let $m_k(\eta) = f(X_k) + g_{X_k}(\text{grad} \, f(X_k), \eta) + \frac{1}{2}g_{X_k}(\text{Hess} \, f(X_k)[\eta], \eta)$;
3:   Obtain $\eta_k$ by approximately solving $\min_{\eta \in T_{X_k} \mathbb{S}^n, \|\eta\| \leq \Delta_k} m_k(\eta)$;
4:   Compute $\rho_k = \frac{f(X_k) - f(R_{X_k}(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
5:   Set $X_{k+1} = R_{X_k}(\eta_k)$ if $\rho_k$ is sufficient large, Otherwise $X_{k+1} = X_k$;
6:   Set $\Delta_{k+1} = 2\Delta_k$ if $\rho_k$ is sufficient large;
7:   Set $\Delta_{k+1} = \Delta_k/4$ if $\rho_k$ is small;
8: end for

- Build a local quadratic model;
- Solve the local model approximately by truncated CG;
Riemannian Trust-region Newton method

1: for $k = 0, 1, 2, \ldots$ do
2: Let $m_k(\eta) = f(X_k) + g_{X_k} \left( \text{grad} f(X_k), \eta \right) + \frac{1}{2} g_{X_k} \left( \text{Hess} f(X_k)[\eta], \eta \right)$;
3: Obtain $\eta_k$ by approximately solving $\min_{\eta \in T_{X_k} \mathbb{S}_n, \|\eta\| \leq \Delta_k} m_k(\eta)$;
4: Compute $\rho_k = \frac{f(X_k) - f(R_{X_k}(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
5: Set $X_{k+1} = R_{X_k}(\eta_k)$ if $\rho_k$ is sufficient large, Otherwise $X_{k+1} = X_k$;
6: Set $\Delta_{k+1} = 2\Delta_k$ if $\rho_k$ is sufficient large;
7: Set $\Delta_{k+1} = \Delta_k / 4$ if $\rho_k$ is small;
8: end for

- Build a local quadratic model;
- Solve the local model approximately by truncated CG;
- Accept the candidate if the local model is good enough;
Riemannian Trust-region Newton method

1: \textbf{for} $k = 0, 1, 2, \ldots$ \textbf{do}
2: \hspace{1em} Let $m_k(\eta) = f(X_k) + g_{X_k}(\text{grad } f(X_k), \eta) + \frac{1}{2} g_{X_k}(\text{Hess } f(X_k)[\eta], \eta)$;
3: \hspace{1em} Obtain $\eta_k$ by approximately solving $\min_{\eta \in T_{X_k} \mathbb{S}_n, \|\eta\| \leq \Delta_k} m_k(\eta)$;
4: \hspace{1em} Compute $\rho_k = \frac{f(X_k) - f(R_{X_k}(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
5: \hspace{1em} Set $X_{k+1} = R_{X_k}(\eta_k)$ if $\rho_k$ is sufficient large, Otherwise $X_{k+1} = X_k$;
6: \hspace{1em} Set $\Delta_{k+1} = 2\Delta_k$ if $\rho_k$ is sufficient large;
7: \hspace{1em} Set $\Delta_{k+1} = \Delta_k / 4$ if $\rho_k$ is small;
8: \textbf{end for}

- Build a local quadratic model;
- Solve the local model approximately by truncated CG;
- Accept the candidate if the local model is good enough;
- Update the radius of the trust region;
Riemannian Trust-region Newton method

1: for $k = 0, 1, 2, \ldots$ do
2: Let $m_k(\eta) = f(X_k) + gX_k(\text{grad } f(X_k), \eta) + \frac{1}{2} gX_k(\text{Hess } f(X_k)[\eta], \eta)$;
3: Obtain $\eta_k$ by approximately solving $\min_{\eta \in T_{X_k} S^{n \times n}, \|\eta\| \leq \Delta_k} m_k(\eta)$;
4: Compute $\rho_k = \frac{f(X_k) - f(RX_k(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
5: Set $X_{k+1} = RX_k(\eta_k)$ if $\rho_k$ is sufficient large, Otherwise $X_{k+1} = X_k$;
6: Set $\Delta_{k+1} = 2\Delta_k$ if $\rho_k$ is sufficient large;
7: Set $\Delta_{k+1} = \Delta_k/4$ if $\rho_k$ is small;
8: end for

- Build a local quadratic model;
- Solve the local model approximately by truncated CG;
- Accept the candidate if the local model is good enough;
- Update the radius of the trust region;

(1) RTR-Newton converges quadratically locally; (2) Solving the local model is expensive.
Preconditioner

The action of the Riemannian Hessian is

\[
\text{Hess } f(X)[\eta_X] = P_{T_X S_r^{n \times n}} (A \eta_X M + M \eta_X A) \\
+ P_{T_X S_r^{n \times n}} \left( D P_{T_X S_r^{n \times n}} [\eta_X] (AXM + MXA - C) \right)
\]
The action of the Riemannian Hessian is

\[
\text{Hess } f(X)[\eta_X] = P_{TX} S_r^{n \times n} (A\eta_X M + M\eta_X A) \\
+ P_{TX} S_r^{n \times n} \left( D P_{TX} S_r^{n \times n} [\eta_X] (AXM + MXA - C) \right)
\]

Preconditioner for the first term in the Riemannian Hessian: for any \( \xi_X \in TX S_r^{n \times n} \), find \( \eta_X \) such that

\[
P_{TX} S_r^{n \times n} (A\eta_X M + M\eta_X A) = \xi_X
\] (2)
Preconditioner

The action of the Riemannian Hessian is

\[
\text{Hess } f(X)[\eta_X] = P_{TX} S_r^{n \times n}(A\eta_X M + M\eta_X A)
\]
\[
+ P_{TX} S_r^{n \times n} \left( D P_{TX} S_r^{n \times n} [\eta_X] (AXM + MXA - C) \right)
\]

- Preconditioner for the first term in the Riemannian Hessian: for any \( \xi_X \in TX S_r^{n \times n} \), find \( \eta_X \) such that

\[
P_{TX} S_r^{n \times n} (A\eta_X M + M\eta_X A) = \xi_X
\]

- Is equation (2) solvable? Yes, it can be written as

\[
P_{TX} S_r^{n \times n} (A \otimes M + M \otimes A) P_{TX} S_r^{n \times n} \text{vec}(\eta_X) = \text{vec}(\xi_X),
\]
Preconditioner:

\[ P_{T_X \mathbb{S}_r^{n \times n}} (A \otimes M + M \otimes A) P_{T_X \mathbb{S}_r^{n \times n}} \text{vec}(\eta_X) = \text{vec}(\xi_X) \]

Existing Preconditioner in [VV10]

- The preconditioner need be solved in \( O(nr^c) \) with a reasonable constant \( c \);
Preconditioner

Preconditioner:

\[
P_{T_X S_r^{n \times n}}(A \otimes M + M \otimes A)P_{T_X S_r^{n \times n}} vec(\eta_X) = vec(\xi_X)
\]

Existing Preconditioner in [VV10]

- The preconditioner need be solved in \(O(nr^c)\) with a reasonable constant \(c\);
- The existing one
  - Assumption: solve \((A + \lambda I)x = b\) in \(O(n)\)
  - Only for \(M = I\);
Preconditioner:

$$P_{TX} S_{r \times n}^{n \times n} (A \otimes M + M \otimes A) P_{TX} S_{r \times n}^{n \times n} \text{vec}(\eta x) = \text{vec}(\xi x)$$

Existing Preconditioner in [VV10]

- The preconditioner need be solved in $O(nr^c)$ with a reasonable constant $c$;
- The existing one
  - Assumption: solve $(A + \lambda I)x = b$ in $O(n)$
  - Only for $M = I$;
- Solve the preconditioner without letting $M = I$ in order $O(nr^c)$;
New Preconditioner

\[ P_{T_X S_r^{n \times n}}(A \otimes M + M \otimes A) P_{T_X S_r^{n \times n}} \text{vec} (\eta_X) = \text{vec} (\xi_X) \]

- **Key idea:** Let \( X = YY^T \); Then for any \( \zeta_X \in T_X S_r^{n \times n} \), \( \zeta_X \) can be decomposed into
  \[ \zeta_X = YZ^T + ZY^T, \]
where \( Z = \begin{bmatrix} Y & Y_{\perp M} \end{bmatrix} \begin{bmatrix} S \\ K \end{bmatrix} \), \( S = S^T \), \( Y^T MY_{\perp M} = 0 \) and \( Y_{\perp M}^T Y_{\perp M} = I \);
New Preconditioner

\[ P_{T_X S_r^{n\times n}}(A \otimes M + M \otimes A)P_{T_X S_r^{n\times n}} \text{vec}(\eta_X) = \text{vec}(\xi_X) \]

- Key idea: Let \( X = YY^T \); Then for any \( \zeta_X \in T_X S_r^{n\times n} \), \( \zeta_X \) can be decomposed into

\[ \zeta_X = YZ^T + ZY^T, \]

where

\[ Z = \begin{bmatrix} Y & Y_{\perp M} \end{bmatrix} \begin{bmatrix} S \\ K \end{bmatrix}, \quad S = S^T, \quad Y^TMY_{\perp M} = 0 \text{ and } \]

\[ Y_{\perp M}^T Y_{\perp M} = I; \]

- Assumption: solve \((A + \lambda M)x = b\) in \( O(n) \);
New Preconditioner

\[
P_{T_X S_r^{n \times n}} (A \otimes M + M \otimes A) P_{T_X S_r^{n \times n}} \text{vec}(\eta_X) = \text{vec}(\xi_X)
\]

- Key idea: Let \( X = YY^T \); Then for any \( \zeta_X \in T_X S_r^{n \times n} \), \( \zeta_X \) can be decomposed into
  \[
  \zeta_X = YZ^T + ZY^T,
  \]
  where \( Z = \begin{bmatrix} Y & Y_{\perp M} \end{bmatrix} \begin{bmatrix} S \\ K \end{bmatrix}, \ S = S^T, \ Y^T M Y_{\perp M} = 0 \) and \( Y_{\perp M}^T Y_{\perp M} = I \);
- Assumption: solve \((A + \lambda M)x = b\) in \(O(n)\);
- Using such decomposition for \( \eta_X \) and \( \xi_X \), one can solve for \( \eta_X \) in \(O(nr^c)\);
Other Riemannian Algorithms

- Riemannian steepest descent method;
- Limited-memory Riemannian quasi-Newton methods (LRBFGS, LRTRSR1);
- Riemannian nonlinear CG methods;
- Riemannian Newton method;

Riemannian Newton method based on line search with preconditioned truncate CG works best.
Riemannian Line-search Newton method

1: for $k = 0, 1, 2, \ldots$ do
2:  Approximately solving $\text{Hess } f(X_k)[\eta_k] = -\nabla f(X_k)$ for $\eta_k$;
3:  Set $\alpha = 1$;
4:  while $f(R_{X_k}(\alpha \eta_k)) > f(X_k) + 0.001 g_{X_k}(\alpha \eta_k, \nabla f(X_k))$ do
5:    $\alpha = 0.25 \alpha$;
6:  end while
7:  $X_{k+1} = R_{X_k}(\alpha \eta_k)$;
8: end for

- Approximately solve the linear system by the preconditioned truncated CG;
- Search for appropriate step size, attempt 1 first;
- Converge quadratically locally;
Numerical Experiments

- \( n = 50^2; \ r = 10; \) Stop if \( \|\text{grad} f(x_i)\|/\|\text{grad} f(x_0)\| < 10^{-10}; \)
- \( A: \) the negative stiffness matrix of PDE \( \nabla u(x, y) = f \) on unit square \( \Omega \) and \( u = 0 \) on \( \partial \Omega \) (Lyapack [Pen00a]);
- \( M: \) diagonal matrix;
- \( C: \) rank one matrix \( bb^T \) with entries of \( b \) from standard normal distribution;

Table: \( M = I \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RTRNewton</td>
<td>89</td>
<td>48</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>RTRNewton</td>
<td>439</td>
<td>57</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>RNewton</td>
<td>21</td>
<td>14</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>RNewton</td>
<td>328</td>
<td>22</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>
Numerical Experiments

- \( n = 50^2; \ r = 10; \) Stop if \( \|\text{grad} f(x_i)\|/\|\text{grad} f(x_0)\| < 10^{-10}; \)
- \( A: \) the negative stiffness matrix of PDE \( \nabla u(x, y) = f \) on unit square \( \Omega \) and \( u = 0 \) on \( \partial \Omega \) (Lyapack [Pen00a]);
- \( M: \) diagonal matrix;
- \( C: \) rank one matrix \( bb^T \) with entries of \( b \) from standard normal distribution;

Table: \( M = \text{diag}([\text{rand}(n - 1, 1); 0] + 0.1) \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RTRNewton</td>
<td>iter. #</td>
<td>48</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>precon. #</td>
<td>398</td>
<td>114</td>
</tr>
<tr>
<td>RNewton</td>
<td>iter. #</td>
<td>23</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>precon. #</td>
<td>324</td>
<td>95</td>
</tr>
</tbody>
</table>
Numerical Experiments

- $A$, $M$ and $C$; from semidiscretization of a steel rail cooling problem [Pen06];
- Coarse discretization: $n = 821$; $r = 20$; Stop if $\|\nabla f(x_i)\|/\|\nabla f(x_0)\| < 10^{-10}$;

<table>
<thead>
<tr>
<th></th>
<th>No precon.</th>
<th>precon. [VV10]</th>
<th>New precon.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTRNewton</td>
<td>iter. #</td>
<td>1476</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>precon. #</td>
<td>3838</td>
<td>155</td>
</tr>
<tr>
<td>RNewton</td>
<td>iter. #</td>
<td>260</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>precon. #</td>
<td>1160</td>
<td>129</td>
</tr>
</tbody>
</table>
Numerical Experiments

- $A$, $M$ and $C$; from semidiscretization of a steel rail cooling problem [Pen06];
- Dense discretization: $n = 3113$; $r = 20$; Stop if $\|\text{grad}f(x_i)\| / \|\text{grad}f(x_0)\| < 10^{-10}$;

<table>
<thead>
<tr>
<th></th>
<th>No precon.</th>
<th>precon. [VV10]</th>
<th>New precon.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTRNewton</td>
<td>iter. #</td>
<td>2000</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>precon. #</td>
<td>5942</td>
<td>195</td>
</tr>
<tr>
<td>RNewton</td>
<td>iter. #</td>
<td>320</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>precon. #</td>
<td>2015</td>
<td>267</td>
</tr>
</tbody>
</table>
Summary:
- Briefly introduced the generalized Lyapunov equation;
- Propose a new efficient preconditioner for the subproblem;
- Use different Riemannian methods and propose Riemannian line-search Newton method;
- Compare different preconditioners by experiments;

Future Work:
- Add rank update strategy;
- Compare with other state-of-the-art methods, e.g., CF-ADI [Pen00a], KPIK [Sim07];
- Use large-scale real data;
Riemannian Manifold Optimization Library

- Most state-of-the-art methods;
- Commonly-encountered manifolds;
- Written in C++;
- Interfaces with Matlab, Julia and R;
- BLAS and LAPACK;
- www.math.fsu.edu/~whuang2/Indices/index_ROPTLIB.html

Users need only provide a cost function, gradient function, an action of Hessian (if a Newton method is used) in Matlab, Julia, R or C++ and parameters to control the optimization, e.g., the domain manifold, the algorithm, stopping criterion.
References I

P. Benner.
*Control Theory.*

B. Moore.
Principal component analysis in linear systems: Controllability, observability, and model reduction.

Thilo Penzl.
Numerical solution of generalized lyapunov equations.

Thilo Penzl.
A cyclic low-rank smith method for large sparse lyapunov equations.

Thilo Penzl.
Eigenvalue decay bounds for solutions of lyapunov equations: the symmetric case.

Thilo Penzl.
Algorithms for model reduction of large dynamical systems.

V. Simoncini.
A new iterative method for solving large-scale Lyapunov matrix equations.
Bart Vandereycken and Stefan Vandewalle.
A riemannian optimization approach for computing low-rank solutions of lyapunov equations. 