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

$$AXM^T + MXA^T = C \tag{1}$$

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

$$AXM^T + MXA^T = C \tag{1}$$

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

$A \succ 0, M \succ 0$ 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?

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

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]

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] $\Longrightarrow X = YY^T$

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] $\Longrightarrow X = YY^T$

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

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] $\Longrightarrow X = YY^T$

- 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] $\Longrightarrow X = YY^T$

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

Problem Reformulation

- Consider a cost function on the set of symmetric matrices:
  - Cost function: $F : \mathbb{S}^{n \times n} \to \mathbb{R} : X \mapsto \mathrm{trace}(XAXM) - \mathrm{trace}(XC)$;
  - Gradient: $AXM + MXA - C$;
  - The critical point is unique [Pen98].
  - Minimizer is the solution.

Problem Reformulation

- Consider a cost function on the set of symmetric matrices:
  - Cost function: $F : \mathbb{S}^{n\times n} \to \mathbb{R} : X \mapsto \mathrm{trace}(XAXM) - \mathrm{trace}(XC)$;
  - 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: $f : \mathbb{S}_r^{n\times n} \to \mathbb{R} : X \mapsto \mathrm{trace}(XAXM) - \mathrm{trace}(XC)$;
  - Gradient: $P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(AXM + MXA - C)$;
  - 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

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

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

Ingredients for Riemannian optimization

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

$$\begin{aligned}\mathrm{T}_X \mathbb{S}_r^{n\times n} &= \left\{ \begin{bmatrix} Y & Y_\perp \end{bmatrix} \begin{bmatrix} 2S & N^T \\ N & 0 \end{bmatrix} \begin{bmatrix} Y^T \\ Y_\perp^T \end{bmatrix} \mid S \in \mathbb{S}^{r\times r}, N \in \mathbb{R}^{(n-r)\times r} \right\} \\ &= \left\{ YZ^T + ZY^T \mid Z \in \mathbb{R}^{n\times r} \right\};\end{aligned}$$

Ingredients for Riemannian optimization

- Tangent space at $X = YY^T$ is $\left\{ YZ^T + ZY^T \mid Z \in \mathbb{R}^{n \times r} \right\}$;
- Riemannian metric:

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

for any $\eta_X, \xi_X \in \mathrm{T}_X \, \mathbb{S}_r^{n \times n}$;

Ingredients for Riemannian optimization

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

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

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

Ingredients for Riemannian optimization

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

$$\mathrm{grad}\, f(X) = P_{\mathrm{T}_X\, \mathbb{S}_r^{n\times n}}(AXM + MXA - C),$$

where $P_{\mathrm{T}_X\, \mathbb{S}_r^{n\times n}}(Z) = P_Y Z P_Y + P_Y^{\perp} Z P_Y + P_Y Z P_Y^{\perp}$, $P_Y^{\perp} = I - P_Y$ and $P_Y = Y(Y^T Y)^{-1} Y^T$;

Ingredients for Riemannian optimization

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

$$\begin{aligned}\mathrm{Hess}\, f(X)[\eta_X] =& P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A\eta_X M + M\eta_X A) \\ &+ P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}\left(\mathrm{D}\, P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}[\eta_X](AXM + MXA - C)\right)\end{aligned}$$

Riemannian Trust-region Newton method

for $k = 0, 1, 2, \ldots$ **do**
  Let $m_k(\eta) = f(X_k) + g_{X_k}(\mathrm{grad}\, f(X_k), \eta) + \frac{1}{2} g_{X_k}(\mathrm{Hess}\, f(X_k)[\eta], \eta)$;
  Obtain η_k by approximately solving $\min_{\eta \in \mathrm{T}_{X_k} \mathbb{S}_r^{n\times n}, \|\eta\| \leq \Delta_k} m_k(\eta)$;
  Compute $\rho_k = \frac{f(X_k) - f(R_{X_k}(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
  Set $X_{k+1} = R_{X_k}(\eta_k)$ if ρ_k is sufficient large, Otherwise $X_{k+1} = X_k$;
  Set $\Delta_{k+1} = 2\Delta_k$ if ρ_k is sufficient large;
  Set $\Delta_{k+1} = \Delta_k/4$ if ρ_k is small;
end for

Riemannian Trust-region Newton method

for $k = 0, 1, 2, \dots$ **do**
    Let $m_k(\eta) = f(X_k) + g_{X_k}(\mathrm{grad}\, f(X_k), \eta) + \frac{1}{2} g_{X_k}(\mathrm{Hess}\, f(X_k)[\eta], \eta)$;
    Obtain η_k by approximately solving $\min_{\eta \in \mathrm{T}_{X_k} \mathbb{S}_r^{n \times n}, \|\eta\| \leq \Delta_k} m_k(\eta)$;
    Compute $\rho_k = \frac{f(X_k) - f(R_{X_k}(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
    Set $X_{k+1} = R_{X_k}(\eta_k)$ if ρ_k is sufficient large, Otherwise $X_{k+1} = X_k$;
    Set $\Delta_{k+1} = 2\Delta_k$ if ρ_k is sufficient large;
    Set $\Delta_{k+1} = \Delta_k / 4$ if ρ_k is small;
end for

- Build a local quadratic model;

Riemannian Trust-region Newton method

for $k = 0, 1, 2, \ldots$ **do**
    Let $m_k(\eta) = f(X_k) + g_{X_k}(\mathrm{grad}\, f(X_k), \eta) + \frac{1}{2} g_{X_k}(\mathrm{Hess}\, f(X_k)[\eta], \eta)$;
    Obtain η_k by approximately solving $\min_{\eta \in \mathrm{T}_{X_k} \mathbb{S}_r^{n \times n}, \|\eta\| \leq \Delta_k} m_k(\eta)$;
    Compute $\rho_k = \frac{f(X_k) - f(R_{X_k}(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
    Set $X_{k+1} = R_{X_k}(\eta_k)$ if ρ_k is sufficient large, Otherwise $X_{k+1} = X_k$;
    Set $\Delta_{k+1} = 2\Delta_k$ if ρ_k is sufficient large;
    Set $\Delta_{k+1} = \Delta_k / 4$ if ρ_k is small;
end for

- Build a local quadratic model;
- Solve the local model approximately by truncated CG;

Riemannian Trust-region Newton method

for $k = 0, 1, 2, \dots$ **do**
  Let $m_k(\eta) = f(X_k) + g_{X_k}(\mathrm{grad}\, f(X_k), \eta) + \frac{1}{2} g_{X_k}(\mathrm{Hess}\, f(X_k)[\eta], \eta)$;
  Obtain η_k by approximately solving $\min_{\eta \in \mathrm{T}_{X_k} \mathbb{S}_r^{n\times n}, \|\eta\| \le \Delta_k} m_k(\eta)$;
  Compute $\rho_k = \frac{f(X_k) - f(R_{X_k}(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
  Set $X_{k+1} = R_{X_k}(\eta_k)$ if ρ_k is sufficient large, Otherwise $X_{k+1} = X_k$;
  Set $\Delta_{k+1} = 2\Delta_k$ if ρ_k is sufficient large;
  Set $\Delta_{k+1} = \Delta_k / 4$ if ρ_k is small;
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

for $k = 0, 1, 2, \ldots$ **do**
  Let $m_k(\eta) = f(X_k) + g_{X_k}(\mathrm{grad}\, f(X_k), \eta) + \frac{1}{2} g_{X_k}(\mathrm{Hess}\, f(X_k)[\eta], \eta)$;
  Obtain η_k by approximately solving $\min_{\eta \in \mathrm{T}_{X_k} \mathbb{S}_r^{n \times n}, \|\eta\| \leq \Delta_k} m_k(\eta)$;
  Compute $\rho_k = \frac{f(X_k) - f(R_{X_k}(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
  Set $X_{k+1} = R_{X_k}(\eta_k)$ if ρ_k is sufficient large, Otherwise $X_{k+1} = X_k$;
  Set $\Delta_{k+1} = 2\Delta_k$ if ρ_k is sufficient large;
  Set $\Delta_{k+1} = \Delta_k / 4$ if ρ_k is small;
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

for $k = 0, 1, 2, \ldots$ **do**
    Let $m_k(\eta) = f(X_k) + g_{X_k}(\mathrm{grad}\, f(X_k), \eta) + \frac{1}{2} g_{X_k}(\mathrm{Hess}\, f(X_k)[\eta], \eta)$;
    Obtain η_k by approximately solving $\min_{\eta \in \mathrm{T}_{X_k} \mathbb{S}_r^{n \times n}, \|\eta\| \leq \Delta_k} m_k(\eta)$;
    Compute $\rho_k = \frac{f(X_k) - f(R_{X_k}(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
    Set $X_{k+1} = R_{X_k}(\eta_k)$ if ρ_k is sufficient large, Otherwise $X_{k+1} = X_k$;
    Set $\Delta_{k+1} = 2\Delta_k$ if ρ_k is sufficient large;
    Set $\Delta_{k+1} = \Delta_k / 4$ if ρ_k is small;
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

$$\begin{aligned}\operatorname{Hess} f(X)[\eta_X] =& P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A\eta_X M + M\eta_X A) \\ &+ P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}\left(\mathrm{D}\, P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}[\eta_X](AXM + MXA - C)\right)\end{aligned}$$

Preconditioner

The action of the Riemannian Hessian is

$$\begin{aligned}\operatorname{Hess} f(X)[\eta_X] = & P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A\eta_X M + M\eta_X A) \\ & + P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}\left(\mathrm{D}\, P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}[\eta_X](AXM + MXA - C)\right)\end{aligned}$$

- Preconditioner for the first term in the Riemannian Hessian: for any $\xi_X \in \mathrm{T}_X \mathbb{S}_r^{n\times n}$, find η_X such that

$$P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A\eta_X M + M\eta_X A) = \xi_X \tag{2}$$

Preconditioner

The action of the Riemannian Hessian is

$$\begin{aligned}\operatorname{Hess} f(X)[\eta_X] =& P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A\eta_X M + M\eta_X A) \\ &+ P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}\left(\mathrm{D}\, P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}[\eta_X](AXM + MXA - C)\right)\end{aligned}$$

- Preconditioner for the first term in the Riemannian Hessian: for any $\xi_X \in \mathrm{T}_X \mathbb{S}_r^{n\times n}$, find η_X such that

$$P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A\eta_X M + M\eta_X A) = \xi_X \qquad (2)$$

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

$$P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A\otimes M + M\otimes A)P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}\operatorname{vec}(\eta_X) = \operatorname{vec}(\xi_X),$$

Preconditioner

Preconditioner:

$$P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A \otimes M + M \otimes A) P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}} \mathrm{vec}(\eta_X) = \mathrm{vec}(\xi_X)$$

Existing Preconditioner in [VV10]

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

Preconditioner

Preconditioner:

$$P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A \otimes M + M \otimes A) P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}} \mathrm{vec}(\eta_X) = \mathrm{vec}(\xi_X)$$

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

Preconditioner:

$$P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A \otimes M + M \otimes A) P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}} \mathrm{vec}(\eta_X) = \mathrm{vec}(\xi_X)$$

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

$$P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A \otimes M + M \otimes A) P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}} \mathrm{vec}(\eta_X) = \mathrm{vec}(\xi_X)$$

- Key idea: Let $X = YY^T$; Then for any $\zeta_X \in \mathrm{T}_X \mathbb{S}_r^{n\times n}$, ζ_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 $Y_{\perp_M}^T Y_{\perp_M} = I$;

New Preconditioner

$$P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A \otimes M + M \otimes A) P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}} \mathrm{vec}(\eta_X) = \mathrm{vec}(\xi_X)$$

- Key idea: Let $X = YY^T$; Then for any $\zeta_X \in \mathrm{T}_X \mathbb{S}_r^{n\times n}$, ζ_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 $O(n)$;

New Preconditioner

$$P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}}(A \otimes M + M \otimes A) P_{\mathrm{T}_X \mathbb{S}_r^{n\times n}} \mathrm{vec}(\eta_X) = \mathrm{vec}(\xi_X)$$

- Key idea: Let $X = YY^T$; Then for any $\zeta_X \in \mathrm{T}_X \mathbb{S}_r^{n\times n}$, ζ_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 η_X and ξ_X, one can solve for η_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

for $k = 0, 1, 2, \ldots$ **do**
    Approximately solving $\mathrm{Hess}\, f(X_k)[\eta_k] = -\,\mathrm{grad}\, f(X_k)$ for η_k;
    Set $\alpha = 1$;
    while $f(R_{X_k}(\alpha\eta_k)) > f(X_k) + 0.001 g_{X_k}(\alpha\eta_k, \mathrm{grad}\, f(X_k))$ **do**
        $\alpha = 0.25\alpha$;
    end while
    $X_{k+1} = R_{X_k}(\alpha\eta_k)$;
end for

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

Numerical Experiments

- $n = 50^2$; $r = 10$; Stop if $\|gradf(x_i)\|/\|gradf(x_0)\| < 10^{-10}$;
- A: the negative stiffness matrix of PDE $\nabla u(x, y) = f$ on unit square Ω 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: $M = I$

		No precon.	precon. [VV10]	New precon.
RTRNewton	iter. #	89	48	47
	precon. #	439	57	54
RNewton	iter. #	21	14	14
	precon. #	328	22	25

Numerical Experiments

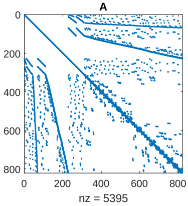
- $n = 50^2$; $r = 10$; Stop if $\|gradf(x_i)\|/\|gradf(x_0)\| < 10^{-10}$;
- A: the negative stiffness matrix of PDE $\nabla u(x, y) = f$ on unit square Ω and $u = 0$ on $\partial\Omega$ (Lyapack [Pen00a]);
- M: diagonal matrix;
- C: rank one matrix bb^T with entries of b from standard normal distribution;

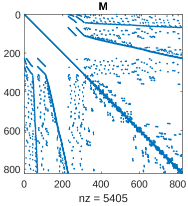
Table: $M = \mathrm{diag}([\mathrm{rand}(n-1, 1); 0] + 0.1)$

		No precon.	precon. [VV10]	New precon.
RTRNewton	iter. #	48	57	49
	precon. #	398	114	84
RNewton	iter. #	23	33	19
	precon. #	324	95	46

Numerical Experiments

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

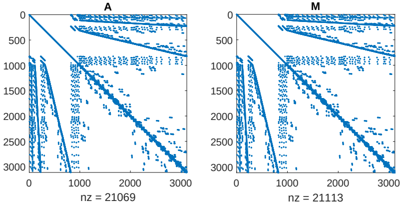




		No precon.	precon. [VV10]	New precon.
RTRNewton	iter. #	1476	68	83
	precon. #	3838	155	114
RNewton	iter. #	260	47	21
	precon. #	1160	129	51

Numerical Experiments

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



		No precon.	precon. [VV10]	New precon.
RTRNewton	iter. #	2000	79	79
	precon. #	5942	195	127
RNewton	iter. #	320	60	30
	precon. #	2015	267	91

Summary and Future Work

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.
Handbook of Linear Algebra, Chapman and Hall/CRC, 2006.

B. Moore.
Principal component analysis in linear systems: Controllability, observability, and model reduction.
IEEE Transactions on Automatic Control, 26(1):17–32, 2003.

Thilo Penzl.
Numerical solution of generalized lyapunov equations.
Advances in Computational Mathematics, 8(1-2):33–48, 1998.

Thilo Penzl.
A cyclic low-rank smith method for large sparse lyapunov equations.
Siam Journal on Scientific Computing, 21(4):1401–1418, 2000.

Thilo Penzl.
Eigenvalue decay bounds for solutions of lyapunov equations: the symmetric case.
Systems and Control Letters, 40(2):139–144, 2000.

Thilo Penzl.
Algorithms for model reduction of large dynamical systems.
Linear Algebra and Its Applications, 415(2):322–343, 2006.

V. Simoncini.
A new iterative method for solving large-scale Lyapunov matrix equations.
SIAM Journal on Scientific Computing, 29:1268–1288, 2007.

References II

Bart Vandereycken and Stefan Vandewalle.
A riemannian optimization approach for computing low-rank solutions of lyapunov equations.
Siam Journal on Matrix Analysis and Applications, 31(5):2553–2579, 2010.