Otto-von-Guericke Universität Magdeburg
Faculty of Mathematics
Summer term 2015

Model Reduction for Dynamical Systems

— Lecture 8 —

Peter Benner   Lihong Feng

Max Planck Institute for Dynamics of Complex Technical Systems
Computational Methods in Systems and Control Theory
Magdeburg, Germany

benner@mpi-magdeburg.mpg.de   feng@mpi-magdeburg.mpg.de

www.mpi-magdeburg.mpg.de/2909616/mor_ss15

Outline

1. Introduction
2. Mathematical Basics
3. Model Reduction by Projection
4. Modal Truncation
5. Balanced Truncation
6. Moment-Matching
7. Solving Large-Scale Matrix Equations
   - Linear Matrix Equations
   - Numerical Methods for Solving Lyapunov Equations
   - Solving Large-Scale Algebraic Riccati Equations
   - Software

Solving Large-Scale Matrix Equations

Large-Scale Algebraic Lyapunov and Riccati Equations

Algebraic Riccati equation (ARE) for $A, G = G^T, W = W^T \in \mathbb{R}^{n \times n}$ given and $X \in \mathbb{R}^{n \times n}$ unknown:

$$0 = \mathcal{R}(X) := A^T X + XA - XGX + W.$$

$G = 0 \Longrightarrow$ Lyapunov equation:

$$0 = \mathcal{L}(X) := A^T X + XA + W.$$

Typical situation in model reduction and optimal control problems for semi-discretized PDEs:

- $n = 10^3 - 10^6$ ($\Longrightarrow 10^6 - 10^{12}$ unknowns!),
- A has sparse representation ($A = -M^{-1}S$ for FEM),
- G, W low-rank with $G, W \in \{BB^T, C^T C\}$, where $B \in \mathbb{R}^{n \times m}$, $m \ll n$, $C \in \mathbb{R}^{p \times n}$, $p \ll n$.
- Standard (eigenproblem-based) $\mathcal{O}(n^3)$ methods are not applicable!

Solving Large-Scale Matrix Equations

Large-Scale Algebraic Lyapunov and Riccati Equations

Algebraic Riccati equation (ARE) for $A, G = G^T, W = W^T \in \mathbb{R}^{n \times n}$ given and $X \in \mathbb{R}^{n \times n}$ unknown:

$$0 = \mathcal{R}(X) := A^T X + XA - XGX + W.$$

$G = 0 \Longrightarrow$ Lyapunov equation:

$$0 = \mathcal{L}(X) := A^T X + XA + W.$$

Typical situation in model reduction and optimal control problems for semi-discretized PDEs:

- $n = 10^3 - 10^6$ ($\Longrightarrow 10^6 - 10^{12}$ unknowns!),
- A has sparse representation ($A = -M^{-1}S$ for FEM),
- G, W low-rank with $G, W \in \{BB^T, C^T C\}$, where $B \in \mathbb{R}^{n \times m}$, $m \ll n$, $C \in \mathbb{R}^{p \times n}$, $p \ll n$.
- Standard (eigenproblem-based) $\mathcal{O}(n^3)$ methods are not applicable!

Solving Large-Scale Matrix Equations

Large-Scale Algebraic Lyapunov and Riccati Equations

Algebraic Riccati equation (ARE) for $A, G = G^T, W = W^T \in \mathbb{R}^{n\times n}$ given and $X \in \mathbb{R}^{n\times n}$ unknown:

$$0 = \mathcal{R}(X) := A^T X + XA - XGX + W.$$

$G = 0 \Longrightarrow$ Lyapunov equation:

$$0 = \mathcal{L}(X) := A^T X + XA + W.$$

Typical situation in model reduction and optimal control problems for semi-discretized PDEs:

- $n = 10^3 - 10^6$ ($\Longrightarrow 10^6 - 10^{12}$ unknowns!),
- A has sparse representation ($A = -M^{-1}S$ for FEM),
- G, W low-rank with $G, W \in \{BB^T, C^TC\}$, where $B \in \mathbb{R}^{n\times m}$, $m \ll n$, $C \in \mathbb{R}^{p\times n}$, $p \ll n$.
- Standard (eigenproblem-based) $\mathcal{O}(n^3)$ methods are not applicable!

Solving Large-Scale Matrix Equations

Large-Scale Algebraic Lyapunov and Riccati Equations

Algebraic Riccati equation (ARE) for $A, G = G^T, W = W^T \in \mathbb{R}^{n\times n}$ given and $X \in \mathbb{R}^{n\times n}$ unknown:

$$0 = \mathcal{R}(X) := A^T X + XA - XGX + W.$$

$G = 0 \Longrightarrow$ Lyapunov equation:

$$0 = \mathcal{L}(X) := A^T X + XA + W.$$

Typical situation in model reduction and optimal control problems for semi-discretized PDEs:

- $n = 10^3 - 10^6$ ($\Longrightarrow 10^6 - 10^{12}$ unknowns!),
- A has sparse representation ($A = -M^{-1}S$ for FEM),
- G, W low-rank with $G, W \in \{BB^T, C^T C\}$, where $B \in \mathbb{R}^{n\times m}$, $m \ll n$, $C \in \mathbb{R}^{p\times n}$, $p \ll n$.
- Standard (eigenproblem-based) $\mathcal{O}(n^3)$ methods are not applicable!

Solving Large-Scale Matrix Equations

Large-Scale Algebraic Lyapunov and Riccati Equations

Algebraic Riccati equation (ARE) for $A, G = G^T, W = W^T \in \mathbb{R}^{n \times n}$ given and $X \in \mathbb{R}^{n \times n}$ unknown:

$$0 = \mathcal{R}(X) := A^T X + XA - XGX + W.$$

$G = 0 \Longrightarrow$ Lyapunov equation:

$$0 = \mathcal{L}(X) := A^T X + XA + W.$$

Typical situation in model reduction and optimal control problems for semi-discretized PDEs:

- $n = 10^3 - 10^6$ ($\Longrightarrow 10^6 - 10^{12}$ unknowns!),
- A has sparse representation ($A = -M^{-1}S$ for FEM),
- G, W low-rank with $G, W \in \{BB^T, C^T C\}$, where $B \in \mathbb{R}^{n \times m}$, $m \ll n$, $C \in \mathbb{R}^{p \times n}$, $p \ll n$.
- Standard (eigenproblem-based) $\mathcal{O}(n^3)$ methods are not applicable!

Solving Large-Scale Matrix Equations

Large-Scale Algebraic Lyapunov and Riccati Equations

Algebraic Riccati equation (ARE) for $A, G = G^T, W = W^T \in \mathbb{R}^{n \times n}$ given and $X \in \mathbb{R}^{n \times n}$ unknown:

$$0 = \mathcal{R}(X) := A^T X + XA - XGX + W.$$

$G = 0 \Longrightarrow$ Lyapunov equation:

$$0 = \mathcal{L}(X) := A^T X + XA + W.$$

Typical situation in model reduction and optimal control problems for semi-discretized PDEs:

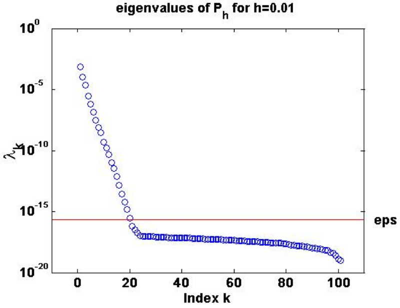
- $n = 10^3 - 10^6$ ($\Longrightarrow 10^6 - 10^{12}$ unknowns!),
- A has sparse representation ($A = -M^{-1}S$ for FEM),
- G, W low-rank with $G, W \in \{BB^T, C^T C\}$, where $B \in \mathbb{R}^{n \times m}$, $m \ll n$, $C \in \mathbb{R}^{p \times n}$, $p \ll n$.
- Standard (eigenproblem-based) $\mathcal{O}(n^3)$ methods are not applicable!

Solving Large-Scale Matrix Equations
Low-Rank Approximation

Consider spectrum of ARE solution (analogous for Lyapunov equations).

Example:

- Linear 1D heat equation with point control,
- $\Omega = [0, 1]$,
- FEM discretization using linear B-splines,
- $h = 1/100 \implies n = 101$.



Idea: $X = X^T \geq 0 \implies$

$$X = ZZ^T = \sum_{k=1}^{n} \lambda_k z_k z_k^T \approx Z^{(r)}(Z^{(r)})^T = \sum_{k=1}^{r} \lambda_k z_k z_k^T.$$

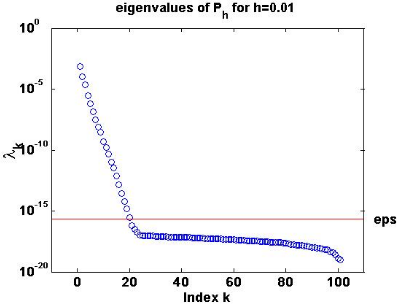
$\implies$ Goal: compute $Z^{(r)} \in \mathbb{R}^{n \times r}$ directly w/o ever forming X!

Solving Large-Scale Matrix Equations
Low-Rank Approximation

Consider spectrum of ARE solution (analogous for Lyapunov equations).

Example:

- Linear 1D heat equation with point control,
- $\Omega = [0, 1]$,
- FEM discretization using linear B-splines,
- $h = 1/100 \implies n = 101$.



Idea: $X = X^T \geq 0 \implies$

$$X = ZZ^T = \sum_{k=1}^{n} \lambda_k z_k z_k^T \approx Z^{(r)}(Z^{(r)})^T = \sum_{k=1}^{r} \lambda_k z_k z_k^T.$$

$\implies$ Goal: compute $Z^{(r)} \in \mathbb{R}^{n \times r}$ directly w/o ever forming X!

Solving Large-Scale Matrix Equations

Linear Matrix Equations

Equations without symmetry

Sylvester equation

$$AX + XB = W$$

discrete Sylvester equation

$$AXB - X = W$$

with data $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$, $W \in \mathbb{R}^{n\times m}$ and unknown $X \in \mathbb{R}^{n\times m}$.

Equations with symmetry

Lyapunov equation

$$AX + XA^T = W$$

Stein equation (discrete Lyapunov equation)

$$AXA^T - X = W$$

with data $A \in \mathbb{R}^{n\times n}$, $W = W^T \in \mathbb{R}^{n\times n}$ and unknown $X \in \mathbb{R}^{n\times n}$.

Here: focus on (Sylvester and) Lyapunov equations; analogous results and methods for discrete versions exist.

Solving Large-Scale Matrix Equations

Linear Matrix Equations

Equations without symmetry

Sylvester equation

$$AX + XB = W$$

discrete Sylvester equation

$$AXB - X = W$$

with data $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$, $W \in \mathbb{R}^{n\times m}$ and unknown $X \in \mathbb{R}^{n\times m}$.

Equations with symmetry

Lyapunov equation

$$AX + XA^T = W$$

Stein equation (discrete Lyapunov equation)

$$AXA^T - X = W$$

with data $A \in \mathbb{R}^{n\times n}$, $W = W^T \in \mathbb{R}^{n\times n}$ and unknown $X \in \mathbb{R}^{n\times n}$.

Here: focus on (Sylvester and) Lyapunov equations; analogous results and methods for discrete versions exist.

Linear Matrix Equations

Solvability

Using the Kronecker (tensor) product, $AX + XB = W$ is equivalent to

$$\left((I_m \otimes A) + \left(B^T \otimes I_n\right)\right) \operatorname{vec}(X) = \operatorname{vec}(W).$$

Hence,

Sylvester equation has a unique solution

$$\Longleftrightarrow$$

$M := (I_m \otimes A) + \left(B^T \otimes I_n\right)$ is invertible.

$$\Longleftrightarrow$$

$$0 \notin \Lambda(M) = \Lambda\left((I_m \otimes A) + (B^T \otimes I_n)\right) = \{\lambda_j + \mu_k, \mid \lambda_j \in \Lambda(A),\ \mu_k \in \Lambda(B)\}.$$

$$\Longleftrightarrow$$

$$\Lambda(A) \cap \Lambda(-B) = \emptyset$$

Corollary

A, B Hurwitz $\Longrightarrow$ Sylvester equation has unique solution.

Linear Matrix Equations

Solvability

Using the Kronecker (tensor) product, $AX + XB = W$ is equivalent to

$$\left((I_m \otimes A) + \left(B^T \otimes I_n\right)\right) \operatorname{vec}(X) = \operatorname{vec}(W).$$

Hence,

Sylvester equation has a unique solution

$$\Longleftrightarrow$$

$M := (I_m \otimes A) + \left(B^T \otimes I_n\right)$ is invertible.

$$\Longleftrightarrow$$

$$0 \notin \Lambda(M) = \Lambda\left((I_m \otimes A) + (B^T \otimes I_n)\right) = \{\lambda_j + \mu_k, \mid \lambda_j \in \Lambda(A),\ \mu_k \in \Lambda(B)\}.$$

$$\Longleftrightarrow$$

$$\Lambda(A) \cap \Lambda(-B) = \emptyset$$

Corollary

A, B Hurwitz $\Longrightarrow$ Sylvester equation has unique solution.

Linear Matrix Equations
Solvability

Using the Kronecker (tensor) product, $AX + XB = W$ is equivalent to

$$\left((I_m \otimes A) + \left(B^T \otimes I_n\right)\right) \operatorname{vec}(X) = \operatorname{vec}(W).$$

Hence,

Sylvester equation has a unique solution

$$\Longleftrightarrow$$

$M := (I_m \otimes A) + \left(B^T \otimes I_n\right)$ is invertible.

$$\Longleftrightarrow$$

$$0 \notin \Lambda(M) = \Lambda((I_m \otimes A) + (B^T \otimes I_n)) = \{\lambda_j + \mu_k, \mid \lambda_j \in \Lambda(A),\ \mu_k \in \Lambda(B)\}.$$

$$\Longleftrightarrow$$

$$\Lambda(A) \cap \Lambda(-B) = \emptyset$$

Corollary

A, B Hurwitz $\Longrightarrow$ Sylvester equation has unique solution.

Linear Matrix Equations
Solvability

Using the Kronecker (tensor) product, $AX + XB = W$ is equivalent to

$$\left((I_m \otimes A) + \left(B^T \otimes I_n\right)\right) \operatorname{vec}(X) = \operatorname{vec}(W).$$

Hence,

Sylvester equation has a unique solution

$$\Longleftrightarrow$$

$M := (I_m \otimes A) + \left(B^T \otimes I_n\right)$ is invertible.

$$\Longleftrightarrow$$

$$0 \notin \Lambda(M) = \Lambda((I_m \otimes A) + (B^T \otimes I_n)) = \{\lambda_j + \mu_k, \mid \lambda_j \in \Lambda(A), \ \mu_k \in \Lambda(B)\}.$$

$$\Longleftrightarrow$$

$$\Lambda(A) \cap \Lambda(-B) = \emptyset$$

Corollary

A, B Hurwitz $\Longrightarrow$ Sylvester equation has unique solution.

Linear Matrix Equations

Solvability

Using the Kronecker (tensor) product, $AX + XB = W$ is equivalent to

$$\left((I_m \otimes A) + \left(B^T \otimes I_n\right)\right) \operatorname{vec}(X) = \operatorname{vec}(W).$$

Hence,

Sylvester equation has a unique solution

$$\Longleftrightarrow$$

$$M := (I_m \otimes A) + \left(B^T \otimes I_n\right) \text{ is invertible.}$$

$$\Longleftrightarrow$$

$$0 \notin \Lambda(M) = \Lambda((I_m \otimes A) + (B^T \otimes I_n)) = \{\lambda_j + \mu_k, \mid \lambda_j \in \Lambda(A), \ \mu_k \in \Lambda(B)\}.$$

$$\Longleftrightarrow$$

$$\Lambda(A) \cap \Lambda(-B) = \emptyset$$

Corollary

A, B Hurwitz $\Longrightarrow$ Sylvester equation has unique solution.

Linear Matrix Equations

Complexity Issues

Solving the Sylvester equation

$$AX + XB = W$$

via the equivalent linear system of equations

$$\left((I_m \otimes A) + \left(B^T \otimes I_n\right)\right) \operatorname{vec}(X) = \operatorname{vec}(W)$$

requires

- LU factorization of $nm \times nm$ matrix; for $n \approx m$, complexity is $\frac{2}{3}n^6$;
- storing $n \cdot m$ unknowns: for $n \approx m$ we have n^4 data for X.

Example

$n = m = 1,000 \Rightarrow$ Gaussian elimination on an Intel core i7 (Westmere, 6 cores, 3.46 GHz $\rightsquigarrow$ 83.2 GFLOP peak) would take $>$ 94 DAYS and 7.3 TB of memory!

Linear Matrix Equations

Complexity Issues

Solving the Sylvester equation

$$AX + XB = W$$

via the equivalent linear system of equations

$$\left((I_m \otimes A) + \left(B^T \otimes I_n\right)\right) \operatorname{vec}(X) = \operatorname{vec}(W)$$

requires

- LU factorization of $nm \times nm$ matrix; for $n \approx m$, complexity is $\frac{2}{3}n^6$;
- storing $n \cdot m$ unknowns: for $n \approx m$ we have n^4 data for X.

Example

$n = m = 1,000 \Rightarrow$ Gaussian elimination on an Intel core i7 (Westmere, 6 cores, 3.46 GHz $\rightsquigarrow$ 83.2 GFLOP peak) would take $>$ 94 DAYS and 7.3 TB of memory!

Numerical Methods for Solving Lyapunov Equations
Traditional Methods

Bartels-Stewart method for Sylvester and Lyapunov equation (`lyap`);
Hessenberg-Schur method for Sylvester equations (`lyap`);
Hammarling's method for Lyapunov equations $AX + XA^T + GG^T = 0$ with A Hurwitz (`lyapchol`).

All based on the fact that if A, B^T are in Schur form, then

$$M = (I_m \otimes A) + \left(B^T \otimes I_n\right)$$

is block-upper triangular. Hence, solve $Mx = b$ by back-substitution.

- Clever implementation of back-substitution process requires $nm(n+m)$ flops.
- For Sylvester eqns., B in Hessenberg form is enough ($\rightsquigarrow$ Hessenberg-Schur method).
- Hammarling's method computes Cholesky factor Y of X directly.
- All methods require Schur decomposition of A and Schur or Hessenberg decomposition of B $\Rightarrow$ need QR algorithm which requires $25n^3$ flops for Schur decomposition.

Not feasible for large-scale problems ($n > 10,000$).

Numerical Methods for Solving Lyapunov Equations
Traditional Methods

Bartels-Stewart method for Sylvester and Lyapunov equation (`lyap`);
Hessenberg-Schur method for Sylvester equations (`lyap`);
Hammarling's method for Lyapunov equations $AX + XA^T + GG^T = 0$ with A Hurwitz (`lyapchol`).

All based on the fact that if A, B^T are in Schur form, then

$$M = (I_m \otimes A) + \left(B^T \otimes I_n\right)$$

is block-upper triangular. Hence, solve $Mx = b$ by back-substitution.

- Clever implementation of back-substitution process requires $nm(n+m)$ flops.
- For Sylvester eqns., B in Hessenberg form is enough ($\rightsquigarrow$ Hessenberg-Schur method).
- Hammarling's method computes Cholesky factor Y of X directly.
- All methods require Schur decomposition of A and Schur or Hessenberg decomposition of B $\Rightarrow$ need QR algorithm which requires $25n^3$ flops for Schur decomposition.

Not feasible for large-scale problems ($n > 10,000$).

Numerical Methods for Solving Lyapunov Equations
Traditional Methods

Bartels-Stewart method for Sylvester and Lyapunov equation (`lyap`);
Hessenberg-Schur method for Sylvester equations (`lyap`);
Hammarling's method for Lyapunov equations $AX + XA^T + GG^T = 0$ with A Hurwitz (`lyapchol`).

All based on the fact that if A, B^T are in Schur form, then

$$M = (I_m \otimes A) + \left(B^T \otimes I_n\right)$$

is block-upper triangular. Hence, solve $Mx = b$ by back-substitution.

- Clever implementation of back-substitution process requires $nm(n+m)$ flops.
- For Sylvester eqns., B in Hessenberg form is enough ($\rightsquigarrow$ Hessenberg-Schur method).
- Hammarling's method computes Cholesky factor Y of X directly.
- All methods require Schur decomposition of A and Schur or Hessenberg decomposition of B $\Rightarrow$ need QR algorithm which requires $25n^3$ flops for Schur decomposition.

Not feasible for large-scale problems ($n > 10,000$).

Numerical Methods for Solving Lyapunov Equations
The Sign Function Method

Definition

For $Z \in \mathbb{R}^{n\times n}$ with $\Lambda(Z) \cap \imath\mathbb{R} = \emptyset$ and Jordan canonical form

$$Z = S \begin{bmatrix} J^+ & 0 \\ 0 & J^- \end{bmatrix} S^{-1}$$

the matrix sign function is

$$\operatorname{sign}(Z) := S \begin{bmatrix} I_k & 0 \\ 0 & -I_{n-k} \end{bmatrix} S^{-1}.$$

Numerical Methods for Solving Lyapunov Equations
The Sign Function Method

Definition

For $Z \in \mathbb{R}^{n \times n}$ with $\Lambda(Z) \cap \imath\mathbb{R} = \emptyset$ and Jordan canonical form

$$Z = S \begin{bmatrix} J^+ & 0 \\ 0 & J^- \end{bmatrix} S^{-1}$$

the matrix sign function is

$$\operatorname{sign}(Z) := S \begin{bmatrix} I_k & 0 \\ 0 & -I_{n-k} \end{bmatrix} S^{-1}.$$

Lemma

Let $T \in \mathbb{R}^{n \times n}$ be nonsingular and Z as before, then

$$\operatorname{sign}\left(TZT^{-1}\right) = T \operatorname{sign}(Z)\, T^{-1}$$

Numerical Methods for Solving Lyapunov Equations
The Sign Function Method

Computation of $\operatorname{sign}(Z)$

$\operatorname{sign}(Z)$ is root of $I_n \Longrightarrow$ use Newton's method to compute it:

$$Z_0 \leftarrow Z, \qquad Z_{j+1} \leftarrow \frac{1}{2}\left(c_j Z_j + \frac{1}{c_j} Z_j^{-1}\right), \qquad j = 1, 2, \ldots$$

$\Longrightarrow \quad \operatorname{sign}(Z) = \lim_{j\to\infty} Z_j.$

$c_j > 0$ is scaling parameter for convergence acceleration and rounding error minimization, e.g.

$$c_j = \sqrt{\frac{\|Z_j^{-1}\|_F}{\|Z_j\|_F}},$$

based on "equilibrating" the norms of the two summands [Higham '86].

Solving Lyapunov Equations with the Matrix Sign Function Method

Key observation:
If $X \in \mathbb{R}^{n\times n}$ is a solution of $AX + XA^T + W = 0$, then

$$\underbrace{\begin{bmatrix} I_n & -X \\ 0 & I_n \end{bmatrix}}_{=T^{-1}} \underbrace{\begin{bmatrix} A & W \\ 0 & -A^T \end{bmatrix}}_{=:H} \underbrace{\begin{bmatrix} I_n & X \\ 0 & I_n \end{bmatrix}}_{=:T} = \begin{bmatrix} A & 0 \\ 0 & -A^T \end{bmatrix}.$$

Hence, if A is Hurwitz (i.e., asymptotically stable), then

$$\begin{aligned} \operatorname{sign}(H) &= \operatorname{sign}\left(T \begin{bmatrix} A & 0 \\ 0 & -A^T \end{bmatrix} T^{-1}\right) = T \operatorname{sign}\left(\begin{bmatrix} A & 0 \\ 0 & -A^T \end{bmatrix}\right) T^{-1} \\ &= \begin{bmatrix} -I_n & 2X \\ 0 & I_n \end{bmatrix}. \end{aligned}$$

Solving Lyapunov Equations with the Matrix Sign Function Method

Key observation:
If $X \in \mathbb{R}^{n\times n}$ is a solution of $AX + XA^T + W = 0$, then

$$\underbrace{\begin{bmatrix} I_n & -X \\ 0 & I_n \end{bmatrix}}_{=T^{-1}} \underbrace{\begin{bmatrix} A & W \\ 0 & -A^T \end{bmatrix}}_{=:H} \underbrace{\begin{bmatrix} I_n & X \\ 0 & I_n \end{bmatrix}}_{=:T} = \begin{bmatrix} A & 0 \\ 0 & -A^T \end{bmatrix}.$$

Hence, if A is Hurwitz (i.e., asymptotically stable), then

$$\begin{aligned} \operatorname{sign}(H) &= \operatorname{sign}\left(T \begin{bmatrix} A & 0 \\ 0 & -A^T \end{bmatrix} T^{-1}\right) = T \operatorname{sign}\left(\begin{bmatrix} A & 0 \\ 0 & -A^T \end{bmatrix}\right) T^{-1} \\ &= \begin{bmatrix} -I_n & 2X \\ 0 & I_n \end{bmatrix}. \end{aligned}$$

Solving Lyapunov Equations with the Matrix Sign Function Method

Apply sign function iteration $Z \leftarrow \frac{1}{2}(Z + Z^{-1})$ to $H = \begin{bmatrix} A & W \\ 0 & -A^T \end{bmatrix}$:

$$H + H^{-1} = \begin{bmatrix} A & W \\ 0 & -A^T \end{bmatrix} + \begin{bmatrix} A^{-1} & A^{-1}WA^{-T} \\ 0 & -A^{-T} \end{bmatrix}$$

$\Longrightarrow$ Sign function iteration for Lyapunov equation:

$$\begin{aligned} A_0 &\leftarrow A, \quad A_{j+1} \leftarrow \frac{1}{2}\left(A_j + A_j^{-1}\right), \\ W_0 &\leftarrow G, \quad W_{j+1} \leftarrow \frac{1}{2}\left(W_j + A_j^{-1} W_j A_j^{-T}\right), \end{aligned} \qquad j = 0, 1, 2, \ldots.$$

Define $A_\infty := \lim_{j\to\infty} A_j$, $W_\infty := \lim_{j\to\infty} W_j$.

Theorem

If A is Hurwitz, then

$$A_\infty = -I_n \qquad \text{and} \qquad X = \frac{1}{2} W_\infty.$$

Solving Lyapunov Equations with the Matrix Sign Function Method
Factored form

Recall sign function iteration for $AX + XA^T + W = 0$:

$$\begin{aligned} A_0 &\leftarrow A, \quad A_{j+1} \leftarrow \tfrac{1}{2}\left(A_j + A_j^{-1}\right), \\ W_0 &\leftarrow G, \quad W_{j+1} \leftarrow \tfrac{1}{2}\left(W_j + A_j^{-1} W_j A_j^{-T}\right), \end{aligned} \qquad j = 0, 1, 2, \ldots.$$

Now consider the second iteration for $W = BB^T$, starting with $W_0 = BB^T =: B_0 B_0^T$:

$$\begin{aligned} \frac{1}{2}\left(W_j + A_j^{-1} W_j A_j^{-T}\right) &= \frac{1}{2}\left(B_j B_j^T + A_j^{-1} B_j B_j^T A_j^{-T}\right) \\ &= \frac{1}{2}\left[B_j \quad A_j^{-1} B_j\right]\left[B_j \quad A_j^{-1} B_j\right]^T. \end{aligned}$$

Hence, obtain factored iteration

$$B_{j+1} \leftarrow \frac{1}{\sqrt{2}}\left[B_j \quad A_j^{-1} B_j\right]$$

with $S := \frac{1}{\sqrt{2}} \lim_{j\to\infty} B_j$ and $X = SS^T$.

Solving Lyapunov Equations with the Matrix Sign Function Method
Factored form

Recall sign function iteration for $AX + XA^T + W = 0$:

$$\begin{aligned} A_0 &\leftarrow A, \quad A_{j+1} \leftarrow \tfrac{1}{2}\left(A_j + A_j^{-1}\right), \\ W_0 &\leftarrow G, \quad W_{j+1} \leftarrow \tfrac{1}{2}\left(W_j + A_j^{-1} W_j A_j^{-T}\right), \end{aligned} \qquad j = 0, 1, 2, \ldots .$$

Now consider the second iteration for $W = BB^T$, starting with $W_0 = BB^T =: B_0 B_0^T$:

$$\begin{aligned} \frac{1}{2}\left(W_j + A_j^{-1} W_j A_j^{-T}\right) &= \frac{1}{2}\left(B_j B_j^T + A_j^{-1} B_j B_j^T A_j^{-T}\right) \\ &= \frac{1}{2}\begin{bmatrix} B_j & A_j^{-1} B_j \end{bmatrix} \begin{bmatrix} B_j & A_j^{-1} B_j \end{bmatrix}^T . \end{aligned}$$

Hence, obtain factored iteration

$$B_{j+1} \leftarrow \frac{1}{\sqrt{2}} \begin{bmatrix} B_j & A_j^{-1} B_j \end{bmatrix}$$

with $S := \frac{1}{\sqrt{2}} \lim_{j\to\infty} B_j$ and $X = SS^T$.

Solving Lyapunov Equations with the Matrix Sign Function Method

Factored form

Recall sign function iteration for $AX + XA^T + W = 0$:

$$\begin{array}{ll} A_0 \leftarrow A, & A_{j+1} \leftarrow \frac{1}{2}\left(A_j + A_j^{-1}\right), \\ W_0 \leftarrow G, & W_{j+1} \leftarrow \frac{1}{2}\left(W_j + A_j^{-1} W_j A_j^{-T}\right), \end{array} \qquad j = 0, 1, 2, \ldots.$$

Now consider the second iteration for $W = BB^T$, starting with $W_0 = BB^T =: B_0 B_0^T$:

$$\begin{aligned} \frac{1}{2}\left(W_j + A_j^{-1} W_j A_j^{-T}\right) &= \frac{1}{2}\left(B_j B_j^T + A_j^{-1} B_j B_j^T A_j^{-T}\right) \\ &= \frac{1}{2}\begin{bmatrix} B_j & A_j^{-1} B_j \end{bmatrix}\begin{bmatrix} B_j & A_j^{-1} B_j \end{bmatrix}^T. \end{aligned}$$

Hence, obtain factored iteration

$$B_{j+1} \leftarrow \frac{1}{\sqrt{2}}\begin{bmatrix} B_j & A_j^{-1} B_j \end{bmatrix}$$

with $S := \frac{1}{\sqrt{2}} \lim_{j\to\infty} B_j$ and $X = SS^T$.

Solving Lyapunov Equations with the Matrix Sign Function Method
Factored form

Recall sign function iteration for $AX + XA^T + W = 0$:

$$\begin{array}{ll} A_0 \leftarrow A, \quad A_{j+1} \leftarrow \frac{1}{2}\left(A_j + A_j^{-1}\right), & \\ W_0 \leftarrow G, \quad W_{j+1} \leftarrow \frac{1}{2}\left(W_j + A_j^{-1} W_j A_j^{-T}\right), & j = 0, 1, 2, \ldots. \end{array}$$

Now consider the second iteration for $W = BB^T$, starting with $W_0 = BB^T =: B_0 B_0^T$:

$$\begin{aligned} \frac{1}{2}\left(W_j + A_j^{-1} W_j A_j^{-T}\right) &= \frac{1}{2}\left(B_j B_j^T + A_j^{-1} B_j B_j^T A_j^{-T}\right) \\ &= \frac{1}{2}\left[B_j \quad A_j^{-1} B_j\right]\left[B_j \quad A_j^{-1} B_j\right]^T. \end{aligned}$$

Hence, obtain factored iteration

$$B_{j+1} \leftarrow \frac{1}{\sqrt{2}}\left[B_j \quad A_j^{-1} B_j\right]$$

with $S := \frac{1}{\sqrt{2}} \lim_{j\to\infty} B_j$ and $X = SS^T$.

Solving Lyapunov Equations with the Matrix Sign Function Method

Factored form [B./Quintana-Ortí '97]

Factored sign function iteration for $A(SS^T) + (SS^T)A^T + BB^T = 0$

$$\begin{array}{ll} A_0 \leftarrow A, \quad A_{j+1} \leftarrow \frac{1}{2}\left(A_j + A_j^{-1}\right), & \\ B_0 \leftarrow B, \quad B_{j+1} \leftarrow \frac{1}{\sqrt{2}}\left[B_j \quad A_j^{-1}B_j\right], & \end{array} \qquad j = 0, 1, 2, \ldots.$$

Remarks:

- To get both Gramians, run in parallel

$$C_{j+1} \leftarrow \frac{1}{\sqrt{2}} \begin{bmatrix} C_j \\ C_j A_j^{-1} \end{bmatrix}.$$

- To avoid growth in numbers of columns of B_j (or rows of C_j): column compression by RRLQ or truncated SVD.
- Several options to incorporate scaling, e.g., scale "A"-iteration only.
- Simple stopping cirterion: $\|A_j + I_n\|_F \leq tol$.

Numerical Methods for Solving Lyapunov Equations
The ADI Method

Recall Peaceman Rachford ADI:
Consider $Au = s$ where $A \in \mathbb{R}^{n \times n}$ spd, $s \in \mathbb{R}^n$. ADI Iteration Idea:
Decompose $A = H + V$ with $H, V \in \mathbb{R}^{n \times n}$ such that

$$(H + pI)v = r$$
$$(V + pI)w = t$$

can be solved easily/efficiently.

ADI Iteration

If H, V spd $\Rightarrow \exists p_k,\ k = 1, 2, \ldots$ such that

$$\begin{aligned} u_0 &= 0 \\ (H + p_k I)u_{k-\frac{1}{2}} &= (p_k I - V)u_{k-1} + s \\ (V + p_k I)u_k &= (p_k I - H)u_{k-\frac{1}{2}} + s \end{aligned}$$

converges to $u \in \mathbb{R}^n$ solving $Au = s$.

Numerical Methods for Solving Lyapunov Equations
The ADI Method

Recall Peaceman Rachford ADI:
Consider $Au = s$ where $A \in \mathbb{R}^{n \times n}$ spd, $s \in \mathbb{R}^n$. ADI Iteration Idea:
Decompose $A = H + V$ with $H, V \in \mathbb{R}^{n \times n}$ such that

$$\begin{array}{c}(H + pI)v = r\\(V + pI)w = t\end{array}$$

can be solved easily/efficiently.

ADI Iteration

If H, V spd $\Rightarrow \exists p_k,\ k = 1, 2, \ldots$ such that

$$\begin{array}{rcl} u_0 & = & 0 \\ (H + p_k I)u_{k-\frac{1}{2}} & = & (p_k I - V)u_{k-1} + s \\ (V + p_k I)u_k & = & (p_k I - H)u_{k-\frac{1}{2}} + s \end{array}$$

converges to $u \in \mathbb{R}^n$ solving $Au = s$.

Numerical Methods for Solving Lyapunov Equations

The Lyapunov operator

$$\mathcal{L}: \quad P \quad \mapsto \quad AX + XA^T$$

can be decomposed into the linear operators

$$\mathcal{L}_H : X \mapsto AX, \qquad \mathcal{L}_V : X \mapsto XA^T.$$

In analogy to the standard ADI method we find the

ADI iteration for the Lyapunov equation [WACHSPRESS '88]

$$\begin{aligned} X_0 &= 0 \\ (A + p_k I)X_{k-\frac{1}{2}} &= -W - X_{k-1}(A^T - p_k I) \\ (A + p_k I)X_k^T &= -W - X_{k-\frac{1}{2}}^T (A^T - p_k I). \end{aligned}$$

Numerical Methods for Solving Lyapunov Equations

Low-Rank ADI

Consider $AX + XA^T = -BB^T$ for stable A; $B \in \mathbb{R}^{n\times m}$ with $m \ll n$.

ADI iteration for the Lyapunov equation [Wachspress '95]

For $k = 1, \ldots, k_{\max}$

$$\begin{aligned} X_0 &= 0 \\ (A + p_k I)X_{k-\frac{1}{2}} &= -BB^T - X_{k-1}(A^T - p_k I) \\ (A + p_k I)X_k^T &= -BB^T - X_{k-\frac{1}{2}}^T(A^T - p_k I) \end{aligned}$$

Rewrite as one step iteration and factorize $X_k = Z_k Z_k^T$, $k = 0, \ldots, k_{\max}$

$$\begin{aligned} Z_0 Z_0^T &= 0 \\ Z_k Z_k^T &= -2p_k(A + p_k I)^{-1}BB^T(A + p_k I)^{-T} \\ &\quad +(A + p_k I)^{-1}(A - p_k I)Z_{k-1}Z_{k-1}^T(A - p_k I)^T(A + p_k I)^{-T} \end{aligned}$$

... $\rightsquigarrow$ low-rank Cholesky factor ADI

[Penzl '97/'00, Li/White '99/'02, B./Li/Penzl '99/'08, Gugercin/Sorensen/Antoulas '03]

Numerical Methods for Solving Lyapunov Equations
Low-Rank ADI

Consider $AX + XA^T = -BB^T$ for stable A; $B \in \mathbb{R}^{n \times m}$ with $m \ll n$.

ADI iteration for the Lyapunov equation [Wachspress '95]

For $k = 1, \ldots, k_{\max}$

$$\begin{aligned} X_0 &= 0 \\ (A + p_k I)X_{k-\frac{1}{2}} &= -BB^T - X_{k-1}(A^T - p_k I) \\ (A + p_k I)X_k^T &= -BB^T - X_{k-\frac{1}{2}}^T(A^T - p_k I) \end{aligned}$$

Rewrite as one step iteration and factorize $X_k = Z_k Z_k^T$, $k = 0, \ldots, k_{\max}$

$$\begin{aligned} Z_0 Z_0^T &= 0 \\ Z_k Z_k^T &= -2p_k(A + p_k I)^{-1}BB^T(A + p_k I)^{-T} \\ &\quad +(A + p_k I)^{-1}(A - p_k I)Z_{k-1}Z_{k-1}^T(A - p_k I)^T(A + p_k I)^{-T} \end{aligned}$$

$\ldots \rightsquigarrow$ low-rank Cholesky factor ADI

[Penzl '97/'00, Li/White '99/'02, B./Li/Penzl '99/'08, Gugercin/Sorensen/Antoulas '03]

Numerical Methods for Solving Lyapunov Equations

Low-Rank ADI

Consider $AX + XA^T = -BB^T$ for stable A; $B \in \mathbb{R}^{n \times m}$ with $m \ll n$.

ADI iteration for the Lyapunov equation [Wachspress '95]

For $k = 1, \ldots, k_{\max}$

$$\begin{aligned} X_0 &= 0 \\ (A + p_k I)X_{k-\frac{1}{2}} &= -BB^T - X_{k-1}(A^T - p_k I) \\ (A + p_k I)X_k^T &= -BB^T - X_{k-\frac{1}{2}}^T(A^T - p_k I) \end{aligned}$$

Rewrite as one step iteration and factorize $X_k = Z_k Z_k^T$, $k = 0, \ldots, k_{\max}$

$$\begin{aligned} Z_0 Z_0^T &= 0 \\ Z_k Z_k^T &= -2p_k(A + p_k I)^{-1} BB^T (A + p_k I)^{-T} \\ &\quad + (A + p_k I)^{-1}(A - p_k I) Z_{k-1} Z_{k-1}^T (A - p_k I)^T (A + p_k I)^{-T} \end{aligned}$$

$\ldots \rightsquigarrow$ low-rank Cholesky factor ADI

[Penzl '97/'00, Li/White '99/'02, B./Li/Penzl '99/'08, Gugercin/Sorensen/Antoulas '03]

Solving Large-Scale Matrix Equations

Numerical Methods for Solving Lyapunov Equations

$$Z_k = [\sqrt{-2p_k}(A + p_k I)^{-1}B,\ (A + p_k I)^{-1}(A - p_k I)Z_{k-1}]$$

[Penzl '00]

Observing that $(A - p_i I)$, $(A + p_k I)^{-1}$ commute, we rewrite $Z_{k_{\max}}$ as

$$Z_{k_{\max}} = [z_{k_{\max}},\ P_{k_{\max}-1}z_{k_{\max}},\ P_{k_{\max}-2}(P_{k_{\max}-1}z_{k_{\max}}),\ \ldots,\ P_1(P_2 \cdots P_{k_{\max}-1}z_{k_{\max}})],$$

[Li/White '02]

where

$$z_{k_{\max}} = \sqrt{-2p_{k_{\max}}}(A + p_{k_{\max}} I)^{-1}B$$

and

$$P_i := \frac{\sqrt{-2p_i}}{\sqrt{-2p_{i+1}}}\left[I - (p_i + p_{i+1})(A + p_i I)^{-1}\right].$$

Solving Large-Scale Matrix Equations
Numerical Methods for Solving Lyapunov Equations

$$Z_k = [\sqrt{-2p_k}(A + p_k I)^{-1}B,\ (A + p_k I)^{-1}(A - p_k I)Z_{k-1}]$$

[PENZL '00]

Observing that $(A - p_i I)$, $(A + p_k I)^{-1}$ commute, we rewrite $Z_{k_{\max}}$ as

$$Z_{k_{\max}} = [z_{k_{\max}},\ P_{k_{\max}-1}z_{k_{\max}},\ P_{k_{\max}-2}(P_{k_{\max}-1}z_{k_{\max}}),\ \ldots,\ P_1(P_2 \cdots P_{k_{\max}-1}z_{k_{\max}})],$$

[LI/WHITE '02]

where

$$z_{k_{\max}} = \sqrt{-2p_{k_{\max}}}(A + p_{k_{\max}} I)^{-1}B$$

and

$$P_i := \frac{\sqrt{-2p_i}}{\sqrt{-2p_{i+1}}}\left[I - (p_i + p_{i+1})(A + p_i I)^{-1}\right].$$

Numerical Methods for Solving Lyapunov Equations

Lyapunov equation $0 = AX + XA^T + BB^T$.

Algorithm [Penzl '97/'00, Li/White '99/'02, B. 04, B./Li/Penzl '99/'08]

$$V_1 \leftarrow \sqrt{-2\,\mathrm{re}\,p_1}(A + p_1 I)^{-1}B, \qquad Z_1 \leftarrow V_1$$

FOR $k = 2, 3, \ldots$

$$V_k \leftarrow \sqrt{\frac{\mathrm{re}\,p_k}{\mathrm{re}\,p_{k-1}}}\left(V_{k-1} - (p_k + \overline{p_{k-1}})(A + p_k I)^{-1}V_{k-1}\right)$$

$$Z_k \leftarrow \begin{bmatrix} Z_{k-1} & V_k \end{bmatrix}$$

$$Z_k \leftarrow \mathrm{rrlq}(Z_k, \tau) \qquad \text{column compression}$$

At convergence, $Z_{k_{\max}} Z_{k_{\max}}^T \approx X$, where (without column compression)

$$Z_{k_{\max}} = \begin{bmatrix} V_1 & \ldots & V_{k_{\max}} \end{bmatrix}, \quad V_k = \begin{bmatrix} \, \end{bmatrix} \in \mathbb{C}^{n \times m}.$$

Note: Implementation in real arithmetic possible by combining two steps [B./Li/Penzl '99/'08] or using new idea employing the relation of 2 consecutive complex factors [B./Kürschner/Saak '11].

Numerical Methods for Solving Lyapunov Equations

Lyapunov equation $0 = AX + XA^T + BB^T$.

Algorithm [Penzl '97/'00, Li/White '99/'02, B. 04, B./Li/Penzl '99/'08]

$$V_1 \leftarrow \sqrt{-2\,\mathrm{re}\,p_1}(A + p_1 I)^{-1}B, \qquad Z_1 \leftarrow V_1$$

FOR $k = 2, 3, \ldots$

$$V_k \leftarrow \sqrt{\frac{\mathrm{re}\,p_k}{\mathrm{re}\,p_{k-1}}}\left(V_{k-1} - (p_k + \overline{p_{k-1}})(A + p_k I)^{-1} V_{k-1}\right)$$

$$Z_k \leftarrow \begin{bmatrix} Z_{k-1} & V_k \end{bmatrix}$$

$$Z_k \leftarrow \mathrm{rrlq}(Z_k, \tau) \qquad \text{column compression}$$

At convergence, $Z_{k_{\max}} Z_{k_{\max}}^T \approx X$, where (without column compression)

$$Z_{k_{\max}} = \begin{bmatrix} V_1 & \ldots & V_{k_{\max}} \end{bmatrix}, \quad V_k = \Box \in \mathbb{C}^{n \times m}.$$

Note: Implementation in real arithmetic possible by combining two steps [B./Li/Penzl '99/'08] or using new idea employing the relation of 2 consecutive complex factors [B./Kürschner/Saak '11].

Numerical Results for ADI

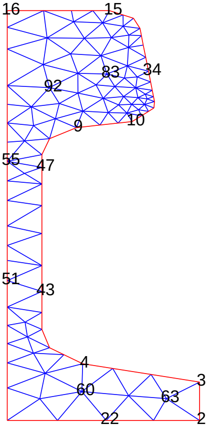
Optimal Cooling of Steel Profiles

- Mathematical model: boundary control for linearized 2D heat equation.

$$
\begin{aligned}
c \cdot \rho \frac{\partial}{\partial t} x &= \lambda \Delta x, & \xi \in \Omega \\
\lambda \frac{\partial}{\partial n} x &= \kappa(u_k - x), & \xi \in \Gamma_k,\ 1 \leq k \leq 7, \\
\frac{\partial}{\partial n} x &= 0, & \xi \in \Gamma_7.
\end{aligned}
$$

$\Longrightarrow m = 7, q = 6.$

- FEM Discretization, different models for initial mesh ($n = 371$), 1, 2, 3, 4 steps of mesh refinement $\Rightarrow$ $n = 1357, 5177, 20209, 79841$.



Source: Physical model: courtesy of Mannesmann/Demag.

Math. model: Tröltzsch/Unger 1999/2001, Penzl 1999, Saak 2003.

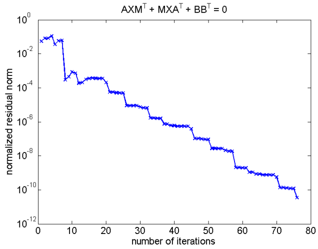
Numerical Results for ADI

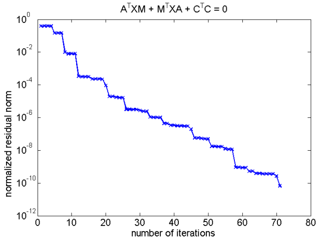
Optimal Cooling of Steel Profiles

- Solve dual Lyapunov equations needed for balanced truncation, i.e.,

$$APM^T + MPA^T + BB^T = 0, \quad A^TQM + M^TQA + C^TC = 0,$$

  for $n = 79,841$.
- 25 shifts chosen by Penzl heuristic from 50/25 Ritz values of A of largest/smallest magnitude, no column compression performed.
- No factorization of mass matrix required.
- Computations done on Core2Duo at 2.8GHz with 3GB RAM and 32Bit-MATLAB.





CPU times: 626 / 356 sec.

Numerical Results for ADI

Scaling / Mesh Independence

Computations by Martin Köhler '10

- $A \in \mathbb{R}^{n \times n} \equiv$ FDM matrix for 2D heat equation on $[0,1]^2$ (LYAPACK benchmark `demo_l1`, $m = 1$).
- 16 shifts chosen by Penzl heuristic from 50/25 Ritz values of A of largest/smallest magnitude.
- Computations on 2 dual core Intel Xeon 5160 with 16 GB RAM using M.E.S.S. (http://svncsc.mpi-magdeburg.mpg.de/trac/messtrac/).

Numerical Results for ADI

Scaling / Mesh Independence

Computations by Martin Köhler '10

- $A \in \mathbb{R}^{n \times n} \equiv$ FDM matrix for 2D heat equation on $[0,1]^2$ (LYAPACK benchmark `demo_l1`, $m = 1$).
- 16 shifts chosen by Penzl heuristic from 50/25 Ritz values of A of largest/smallest magnitude.
- Computations on 2 dual core Intel Xeon 5160 with 16 GB RAM using M.E.S.S. (http://svncsc.mpi-magdeburg.mpg.de/trac/messtrac/).

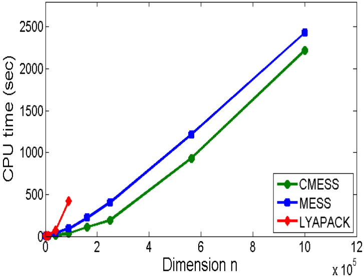
CPU Times

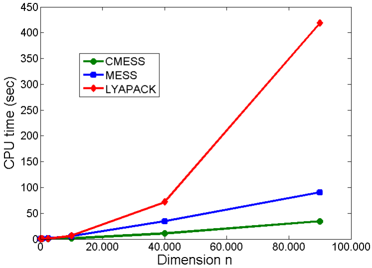
n	M.E.S.S.[1] (C)	LyaPack	M.E.S.S. (MATLAB)
100	0.023	0.124	0.158
625	0.042	0.104	0.227
2,500	0.159	0.702	0.989
10,000	0.965	6.22	5.644
40,000	11.09	71.48	34.55
90,000	34.67	418.5	90.49
160,000	109.3	out of memory	219.9
250,000	193.7	out of memory	403.8
562,500	930.1	out of memory	1216.7
1,000,000	2220.0	out of memory	2428.6

Numerical Results for ADI

Scaling / Mesh Independence — Computations by Martin Köhler '10

- $A \in \mathbb{R}^{n \times n} \equiv$ FDM matrix for 2D heat equation on $[0,1]^2$ (LYAPACK benchmark demo_l1, $m = 1$).
- 16 shifts chosen by Penzl heuristic from 50/25 Ritz values of A of largest/smallest magnitude.
- Computations on 2 dual core Intel Xeon 5160 with 16 GB RAM using M.E.S.S. (http://svncsc.mpi-magdeburg.mpg.de/trac/messtrac/).





Note: for $n = 1,000,000$, first sparse LU needs $\sim 1,100$ sec., using UMFPACK this reduces to 30 sec.

Factored Galerkin-ADI Iteration

Lyapunov equation $0 = AX + XA^T + BB^T$

Projection-based methods for Lyapunov equations with $A + A^T < 0$:

1. Compute orthonormal basis $\mathrm{range}(Z)$, $Z \in \mathbb{R}^{n \times r}$, for subspace $\mathcal{Z} \subset \mathbb{R}^n$, $\dim \mathcal{Z} = r$.
2. Set $\hat{A} := Z^T A Z$, $\hat{B} := Z^T B$.
3. Solve small-size Lyapunov equation $\hat{A}\hat{X} + \hat{X}\hat{A}^T + \hat{B}\hat{B}^T = 0$.
4. Use $X \approx Z\hat{X}Z^T$.

Examples:

- Krylov subspace methods, i.e., for $m = 1$:

$$\mathcal{Z} = \mathcal{K}(A, B, r) = \mathrm{span}\{B, AB, A^2B, \ldots, A^{r-1}B\}$$

[Saad '90, Jaimoukha/Kasenally '94, Jbilou '02–'08].

- K-PIK [Simoncini '07],

$$\mathcal{Z} = \mathcal{K}(A, B, r) \cup \mathcal{K}(A^{-1}, B, r).$$

- Rational Krylov [Druskin/Simoncini '11] (⇝ exercises).

Factored Galerkin-ADI Iteration

Lyapunov equation $0 = AX + XA^T + BB^T$

Projection-based methods for Lyapunov equations with $A + A^T < 0$:

1. Compute orthonormal basis $\mathrm{range}(Z)$, $Z \in \mathbb{R}^{n \times r}$, for subspace $\mathcal{Z} \subset \mathbb{R}^n$, $\dim \mathcal{Z} = r$.
2. Set $\hat{A} := Z^T A Z$, $\hat{B} := Z^T B$.
3. Solve small-size Lyapunov equation $\hat{A}\hat{X} + \hat{X}\hat{A}^T + \hat{B}\hat{B}^T = 0$.
4. Use $X \approx Z\hat{X}Z^T$.

Examples:

- Krylov subspace methods, i.e., for $m = 1$:

$$\mathcal{Z} = \mathcal{K}(A, B, r) = \mathrm{span}\{B, AB, A^2B, \ldots, A^{r-1}B\}$$

  [Saad '90, Jaimoukha/Kasenally '94, Jbilou '02–'08].

- K-PIK [Simoncini '07],

$$\mathcal{Z} = \mathcal{K}(A, B, r) \cup \mathcal{K}(A^{-1}, B, r).$$

- Rational Krylov [Druskin/Simoncini '11] ($\rightsquigarrow$ exercises).

Factored Galerkin-ADI Iteration

Lyapunov equation $0 = AX + XA^T + BB^T$

Projection-based methods for Lyapunov equations with $A + A^T < 0$:

1. Compute orthonormal basis $\mathrm{range}(Z)$, $Z \in \mathbb{R}^{n\times r}$, for subspace $\mathcal{Z} \subset \mathbb{R}^n$, $\dim \mathcal{Z} = r$.
2. Set $\hat{A} := Z^T AZ$, $\hat{B} := Z^T B$.
3. Solve small-size Lyapunov equation $\hat{A}\hat{X} + \hat{X}\hat{A}^T + \hat{B}\hat{B}^T = 0$.
4. Use $X \approx Z\hat{X}Z^T$.

Examples:

- Krylov subspace methods, i.e., for $m = 1$:

$$\mathcal{Z} = \mathcal{K}(A, B, r) = \mathrm{span}\{B, AB, A^2B, \ldots, A^{r-1}B\}$$

  [Saad '90, Jaimoukha/Kasenally '94, Jbilou '02–'08].

- K-PIK [Simoncini '07],

$$\mathcal{Z} = \mathcal{K}(A, B, r) \cup \mathcal{K}(A^{-1}, B, r).$$

- Rational Krylov [Druskin/Simoncini '11] ($\rightsquigarrow$ exercises).

Factored Galerkin-ADI Iteration

Lyapunov equation $0 = AX + XA^T + BB^T$

Projection-based methods for Lyapunov equations with $A + A^T < 0$:

1. Compute orthonormal basis $\mathrm{range}(Z)$, $Z \in \mathbb{R}^{n \times r}$, for subspace $\mathcal{Z} \subset \mathbb{R}^n$, $\dim \mathcal{Z} = r$.
2. Set $\hat{A} := Z^T AZ$, $\hat{B} := Z^T B$.
3. Solve small-size Lyapunov equation $\hat{A}\hat{X} + \hat{X}\hat{A}^T + \hat{B}\hat{B}^T = 0$.
4. Use $X \approx Z\hat{X}Z^T$.

Examples:

- ADI subspace [B./R.-C. Li/Truhar '08]:

$$\mathcal{Z} = \mathrm{colspan}\left[\; V_1, \quad \ldots, \quad V_r \;\right].$$

Note:

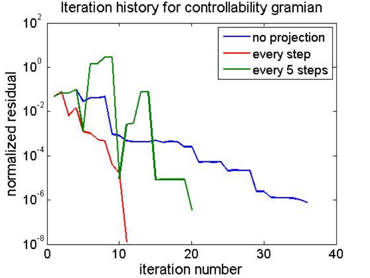
1. ADI subspace is rational Krylov subspace [J.-R. Li/White '02].
2. Similar approach: ADI-preconditioned global Arnoldi method [Jbilou '08].

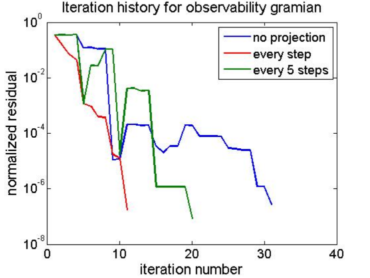
Numerical Methods for Solving Lyapunov Equations
Numerical examples for Galerkin-ADI

FEM semi-discretized control problem for parabolic PDE:

- optimal cooling of rail profiles,
- $n = 20,209,\ m = 7,\ q = 6.$

Good ADI shifts





CPU times: 80s (projection every 5th ADI step) vs. 94s (no projection).

Computations by Jens Saak '10.

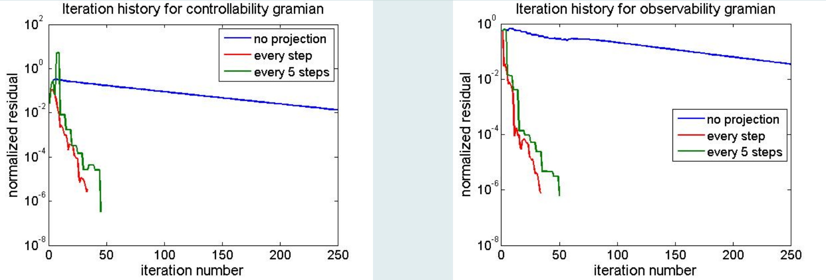
Numerical Methods for Solving Lyapunov Equations

Numerical examples for Galerkin-ADI

FEM semi-discretized control problem for parabolic PDE:

- optimal cooling of rail profiles,
- $n = 20,209,\ m = 7,\ q = 6.$

Bad ADI shifts



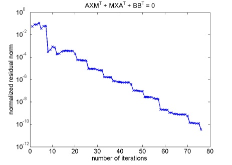
CPU times: 368s (projection every 5th ADI step) vs. 1207s (no projection).

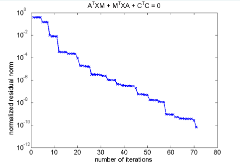
Computations by Jens Saak '10.

Numerical Methods for Solving Lyapunov Equations

Numerical examples for Galerkin-ADI: optimal cooling of rail profiles, $n = 79,841$.

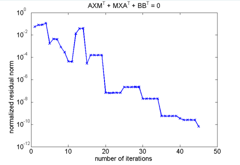
M.E.S.S. w/o Galerkin projection and column compression

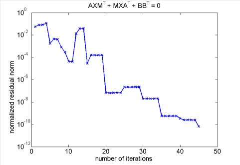




Rank of solution factors: 532 / 426

M.E.S.S. with Galerkin projection and column compression



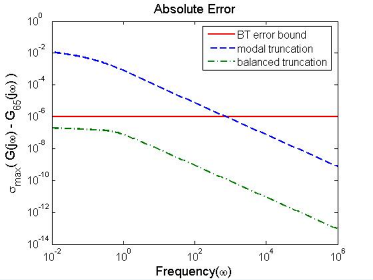


Rank of solution factors: 269 / 205

Solving Large-Scale Matrix Equations

Numerical example for BT: Optimal Cooling of Steel Profiles

$n = 1,357$, Absolute Error

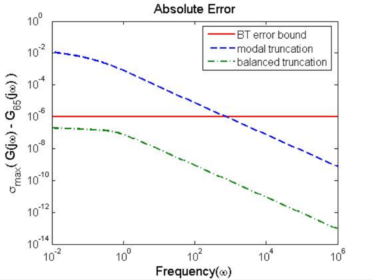


- BT model computed with sign function method,
- MT w/o static condensation, same order as BT model.

Solving Large-Scale Matrix Equations

Numerical example for BT: Optimal Cooling of Steel Profiles

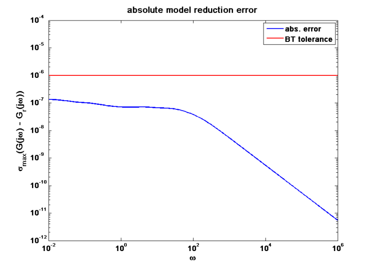
$n = 1,357$, Absolute Error



– BT model computed with sign function method,

– MT w/o static condensation, same order as BT model.

$n = 79,841$, Absolute Error



– BT model computed using M.E.S.S. in MATLAB,

– dualcore, computation time: <10 min.

Solving Large-Scale Matrix Equations

Numerical example for BT: Microgyroscope (Butterfly Gyro)

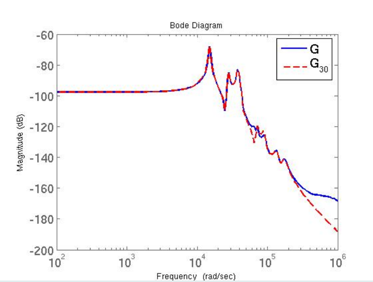
- FEM discretization of structure dynamical model using quadratic tetrahedral elements (ANSYS-SOLID187)
  $\rightsquigarrow$ $n = 34,722$, $m = 1$, $q = 12$.
- Reduced model computed using SpaRed, $r = 30$.

Solving Large-Scale Matrix Equations

Numerical example for BT: Microgyroscope (Butterfly Gyro)

- FEM discretization of structure dynamical model using quadratic tetrahedral elements (ANSYS-SOLID187)
  $\rightsquigarrow n = 34,722,\ m = 1,\ q = 12.$
- Reduced model computed using SPARED, $r = 30$.

Frequency Repsonse Analysis

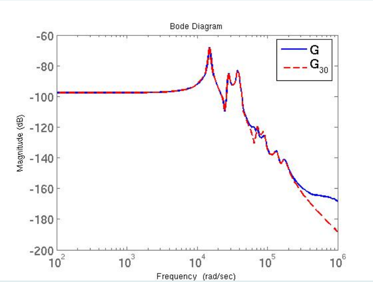


Solving Large-Scale Matrix Equations

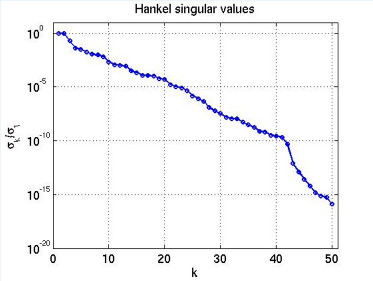
Numerical example for BT: Microgyroscope (Butterfly Gyro)

- FEM discretization of structure dynamical model using quadratic tetrahedral elements (ANSYS-SOLID187)
  $\rightsquigarrow$ $n = 34,722$, $m = 1$, $q = 12$.
- Reduced model computed using SPARED, $r = 30$.

Frequency Repsonse Analysis



Hankel Singular Values



Solving Large-Scale Algebraic Riccati Equations

Theory **[Lancaster/Rodman '95]**

Theorem

Consider the (continuous-time) algebraic Riccati equation (ARE)

$$0 = \mathcal{R}(X) = C^T C + A^T X + XA - XBB^T X,$$

with $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{q \times n}$, (A, B) stabilizable, (A, C) detectable.
Then:

(a) There exists a unique stabilizing $X_* \in \{X \in \mathbb{R}^{n \times n} \,|\, \mathcal{R}(X) = 0\}$, i.e., $\Lambda(A - BB^T X_*) \in \mathbb{C}^-$.

(b) $X_* = X_*^T \geq 0$ and $X_* \geq X$ for all $X \in \{X \in \mathbb{R}^{n \times n} \,|\, \mathcal{R}(X) = 0\}$.

(c) If (A, C) observable, then $X_* > 0$.

(d) $\operatorname{span}\left\{\begin{bmatrix} I_n \\ -X_* \end{bmatrix}\right\}$ is the unique invariant subspace of the Hamiltonian matrix

$$H = \begin{bmatrix} A & BB^T \\ C^T C & -A^T \end{bmatrix}$$

corresponding to $\Lambda(H) \cap \mathbb{C}^-$.

Solving Large-Scale Algebraic Riccati Equations

Numerical Methods **[Bini/Iannazzo/Meini '12]**

Numerical Methods (incomplete list)

- Invariant subspace methods ($\rightsquigarrow$ eigenproblem for Hamiltonian matrix):
  - Schur vector method (`care`) [Laub '79]
  - Hamiltonian SR algorithm [Bunse-Gerstner/Mehrmann '86]
  - Symplectic URV-based method [B./Mehrmann/Xu '97/'98, Chu/Liu/Mehrmann '07]
- Spectral projection methods
  - Sign function method [Roberts '71, Byers '87]
  - Disk function method [Bai/Demmel/Gu '94, B. '97]
- (rational, global) Krylov subspace techniques [Jaimoukha/Kasenally '94, Jbilou '03/'06, Heyouni/Jbilou '09]
- Newton's method
  - Kleinman iteration [Kleinman '68]
  - Line search acceleration [B./Byers '98]
  - Newton-ADI [B./J.-R. Li/Penzl '99/'08]
  - Inexact Newton [Feitzinger/Hylla/Sachs '09]

Solving Large-Scale Algebraic Riccati Equations

Newton's Method for AREs

[Kleinman '68, Mehrmann '91, Lancaster/Rodman '95, B./Byers '94/'98, B. '97, Guo/Laub '99]

- Consider $0 = \mathcal{R}(X) = C^T C + A^T X + XA - XBB^T X.$
- Frechét derivative of $\mathcal{R}(X)$ at X:

$$\mathcal{R}'_X : Z \to (A - BB^T X)^T Z + Z(A - BB^T X).$$

- Newton-Kantorovich method:

$$X_{j+1} = X_j - \left(\mathcal{R}'_{X_j}\right)^{-1} \mathcal{R}(X_j), \quad j = 0, 1, 2, \ldots$$

Newton's method (with line search) for AREs

FOR $j = 0, 1, \ldots$

1. $A_j \leftarrow A - BB^T X_j =: A - BK_j.$
2. Solve the Lyapunov equation $A_j^T N_j + N_j A_j = -\mathcal{R}(X_j).$
3. $X_{j+1} \leftarrow X_j + t_j N_j.$

END FOR j

Solving Large-Scale Algebraic Riccati Equations

Newton's Method for AREs

[Kleinman '68, Mehrmann '91, Lancaster/Rodman '95, B./Byers '94/'98, B. '97, Guo/Laub '99]

- Consider $0 = \mathcal{R}(X) = C^T C + A^T X + XA - XBB^T X.$
- Frechét derivative of $\mathcal{R}(X)$ at X:

$$\mathcal{R}'_X : Z \to (A - BB^T X)^T Z + Z(A - BB^T X).$$

- Newton-Kantorovich method:

$$X_{j+1} = X_j - \left(\mathcal{R}'_{X_j}\right)^{-1} \mathcal{R}(X_j), \quad j = 0, 1, 2, \ldots$$

Newton's method (with line search) for AREs

FOR $j = 0, 1, \ldots$

1. $A_j \leftarrow A - BB^T X_j =: A - BK_j.$
2. Solve the Lyapunov equation $A_j^T N_j + N_j A_j = -\mathcal{R}(X_j).$
3. $X_{j+1} \leftarrow X_j + t_j N_j.$

END FOR j

Solving Large-Scale Algebraic Riccati Equations

Newton's Method for AREs

[Kleinman '68, Mehrmann '91, Lancaster/Rodman '95, B./Byers '94/'98, B. '97, Guo/Laub '99]

- Consider $\quad 0 = \mathcal{R}(X) = C^TC + A^TX + XA - XBB^TX.$
- Frechét derivative of $\mathcal{R}(X)$ at X:

$$\mathcal{R}'_X : Z \to (A - BB^TX)^TZ + Z(A - BB^TX).$$

- Newton-Kantorovich method:

$$X_{j+1} = X_j - \left(\mathcal{R}'_{X_j}\right)^{-1}\mathcal{R}(X_j), \quad j = 0, 1, 2, \ldots$$

Newton's method (with line search) for AREs

FOR $j = 0, 1, \ldots$

1. $A_j \leftarrow A - BB^TX_j =: A - BK_j.$
2. Solve the Lyapunov equation $\quad A_j^TN_j + N_jA_j = -\mathcal{R}(X_j).$
3. $X_{j+1} \leftarrow X_j + t_jN_j.$

END FOR j

Solving Large-Scale Algebraic Riccati Equations

Newton's Method for AREs

[Kleinman '68, Mehrmann '91, Lancaster/Rodman '95, B./Byers '94/'98, B. '97, Guo/Laub '99]

- Consider $\quad 0 = \mathcal{R}(X) = C^T C + A^T X + XA - XBB^T X.$
- Frechét derivative of $\mathcal{R}(X)$ at X:

$$\mathcal{R}'_X : Z \to (A - BB^T X)^T Z + Z(A - BB^T X).$$

- Newton-Kantorovich method:

$$X_{j+1} = X_j - \left(\mathcal{R}'_{X_j}\right)^{-1} \mathcal{R}(X_j), \quad j = 0, 1, 2, \ldots$$

Newton's method (with line search) for AREs

FOR $j = 0, 1, \ldots$

1. $A_j \leftarrow A - BB^T X_j =: A - BK_j.$
2. Solve the Lyapunov equation $\quad A_j^T N_j + N_j A_j = -\mathcal{R}(X_j).$
3. $X_{j+1} \leftarrow X_j + t_j N_j.$

END FOR j

Newton's Method for AREs

Properties and Implementation

- Convergence for K_0 stabilizing:
  - $A_j = A - BK_j = A - BB^T X_j$ is stable $\forall\, j \geq 0$.
  - $\lim_{j\to\infty} \|\mathcal{R}(X_j)\|_F = 0$ (monotonically).
  - $\lim_{j\to\infty} X_j = X_* \geq 0$ (locally quadratic).
- Need large-scale Lyapunov solver; here, ADI iteration:
  linear systems with dense, but "sparse+low rank" coefficient matrix A_j:

$$A_j = A - B \cdot K_j$$

= sparse − m · [ ]

- $m \ll n \implies$ efficient "inversion" using Sherman-Morrison-Woodbury formula:

$$(A - BK_j + p_k^{(j)} I)^{-1} = (I_n + (A + p_k^{(j)} I)^{-1} B (I_m - K_j (A + p_k^{(j)} I)^{-1} B)^{-1} K_j)(A + p_k^{(j)} I)^{-1}.$$

- BUT: $X = X^T \in \mathbb{R}^{n\times n} \implies n(n+1)/2$ unknowns!

Newton's Method for AREs

Properties and Implementation

- Convergence for K_0 stabilizing:
  - $A_j = A - BK_j = A - BB^T X_j$ is stable $\forall\, j \geq 0$.
  - $\lim_{j\to\infty} \|\mathcal{R}(X_j)\|_F = 0$ (monotonically).
  - $\lim_{j\to\infty} X_j = X_* \geq 0$ (locally quadratic).
- Need large-scale Lyapunov solver; here, ADI iteration:
  linear systems with dense, but "sparse+low rank" coefficient matrix A_j:

$$A_j = A - B \cdot K_j$$

= sparse − m · [ ]

- $m \ll n \Longrightarrow$ efficient "inversion" using Sherman-Morrison-Woodbury formula:

$$(A - BK_j + p_k^{(j)} I)^{-1} = (I_n + (A + p_k^{(j)} I)^{-1} B (I_m - K_j (A + p_k^{(j)} I)^{-1} B)^{-1} K_j)(A + p_k^{(j)} I)^{-1}.$$

- BUT: $X = X^T \in \mathbb{R}^{n\times n} \Longrightarrow n(n+1)/2$ unknowns!

Newton's Method for AREs

Properties and Implementation

- Convergence for K_0 stabilizing:
  - $A_j = A - BK_j = A - BB^T X_j$ is stable $\forall\ j \geq 0$.
  - $\lim_{j\to\infty} \|\mathcal{R}(X_j)\|_F = 0$ (monotonically).
  - $\lim_{j\to\infty} X_j = X_* \geq 0$ (locally quadratic).
- Need large-scale Lyapunov solver; here, ADI iteration: linear systems with dense, but "sparse+low rank" coefficient matrix A_j:

$$A_j \;=\; A \;-\; B \;\cdot\; K_j$$

= sparse − m ·

- $m \ll n \implies$ efficient "inversion" using Sherman-Morrison-Woodbury formula:

$$(A - BK_j + p_k^{(j)} I)^{-1} = (I_n + (A + p_k^{(j)} I)^{-1} B (I_m - K_j (A + p_k^{(j)} I)^{-1} B)^{-1} K_j)(A + p_k^{(j)} I)^{-1}.$$

- BUT: $X = X^T \in \mathbb{R}^{n\times n} \implies n(n+1)/2$ unknowns!

Newton's Method for AREs

Properties and Implementation

- Convergence for K_0 stabilizing:
  - $A_j = A - BK_j = A - BB^T X_j$ is stable $\forall\, j \geq 0$.
  - $\lim_{j\to\infty} \|\mathcal{R}(X_j)\|_F = 0$ (monotonically).
  - $\lim_{j\to\infty} X_j = X_* \geq 0$ (locally quadratic).
- Need large-scale Lyapunov solver; here, ADI iteration:
  linear systems with dense, but "sparse+low rank" coefficient matrix A_j:

$$A_j \quad = \quad A \quad - \quad B \quad \cdot \quad K_j$$

- $m \ll n \implies$ efficient "inversion" using Sherman-Morrison-Woodbury formula:

$$(A - BK_j + p_k^{(j)} I)^{-1} = (I_n + (A + p_k^{(j)} I)^{-1} B(I_m - K_j(A + p_k^{(j)} I)^{-1} B)^{-1} K_j)(A + p_k^{(j)} I)^{-1}.$$

- BUT: $X = X^T \in \mathbb{R}^{n\times n} \implies n(n+1)/2$ unknowns!

Low-Rank Newton-ADI for AREs

Re-write Newton's method for AREs

$$A_j^T N_j + N_j A_j = -\mathcal{R}(X_j)$$
$$\Longleftrightarrow$$
$$A_j^T \underbrace{(X_j + N_j)}_{=X_{j+1}} + \underbrace{(X_j + N_j)}_{=X_{j+1}} A_j = \underbrace{-C^T C - X_j BB^T X_j}_{=:-W_j W_j^T}$$

Set $X_j = Z_j Z_j^T$ for $\operatorname{rank}(Z_j) \ll n \Longrightarrow$

$$A_j^T \left(Z_{j+1} Z_{j+1}^T\right) + \left(Z_{j+1} Z_{j+1}^T\right) A_j = -W_j W_j^T$$

Factored Newton Iteration [B./Li/Penzl 1999/2008]

Solve Lyapunov equations for Z_{j+1} directly by factored ADI iteration and use 'sparse + low-rank' structure of A_j.

Low-Rank Newton-ADI for AREs

Re-write Newton's method for AREs

$$A_j^T N_j + N_j A_j = -\mathcal{R}(X_j)$$

$$\Longleftrightarrow$$

$$A_j^T \underbrace{(X_j + N_j)}_{=X_{j+1}} + \underbrace{(X_j + N_j)}_{=X_{j+1}} A_j = \underbrace{-C^T C - X_j B B^T X_j}_{=:-W_j W_j^T}$$

Set $X_j = Z_j Z_j^T$ for $\operatorname{rank}(Z_j) \ll n \Longrightarrow$

$$A_j^T \left(Z_{j+1} Z_{j+1}^T\right) + \left(Z_{j+1} Z_{j+1}^T\right) A_j = -W_j W_j^T$$

Factored Newton Iteration [B./Li/Penzl 1999/2008]

Solve Lyapunov equations for Z_{j+1} directly by factored ADI iteration and use 'sparse + low-rank' structure of A_j.

Low-Rank Newton-ADI for AREs

Feedback Iteration

Optimal feedback

$$K_* = B^T X_* = B^T Z_* Z_*^T$$

can be computed by direct feedback iteration:

- jth Newton iteration:

$$K_j = B^T Z_j Z_j^T = \sum_{k=1}^{k_{\max}} (B^T V_{j,k}) V_{j,k}^T \xrightarrow{j \to \infty} K_* = B^T Z_* Z_*^T$$

- K_j can be updated in ADI iteration, no need to even form Z_j, need only fixed workspace for $K_j \in \mathbb{R}^{m \times n}$!

Related to earlier work by [Banks/Ito 1991].

Solving Large-Scale Matrix Equations

Galerkin-Newton-ADI

Basic ideas

- Hybrid method of Galerkin projection methods for AREs [Jaimoukha/Kasenally '94, Jbilou '06, Heyouni/Jbilou '09] and Newton-ADI, i.e., use column space of current Newton iterate for projection, solve projected ARE, and prolongate.
- Independence of good parameters observed for Galerkin-ADI applied to Lyapunov equations $\rightsquigarrow$ fix ADI parameters for all Newton iterations.

Solving Large-Scale Matrix Equations

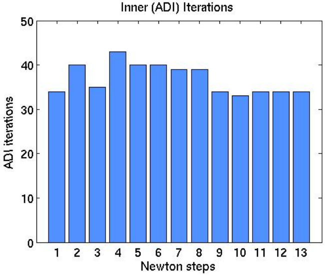
Galerkin-Newton-ADI

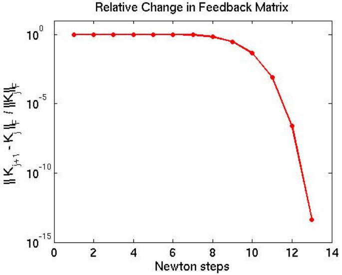
Basic ideas

- Hybrid method of Galerkin projection methods for AREs [Jaimoukha/Kasenally '94, Jbilou '06, Heyouni/Jbilou '09] and Newton-ADI, i.e., use column space of current Newton iterate for projection, solve projected ARE, and prolongate.
- Independence of good parameters observed for Galerkin-ADI applied to Lyapunov equations $\rightsquigarrow$ fix ADI parameters for all Newton iterations.

Numerical Results
LQR Problem for 2D Geometry

- Linear 2D heat equation with homogeneous Dirichlet boundary and point control/observation.
- FD discretization on uniform 150×150 grid.
- $n = 22.500$, $m = p = 1$, 10 shifts for ADI iterations.
- Convergence of large-scale matrix equation solvers:





Numerical Results

Newton-ADI vs. Newton-ADI-Gelerkin

- FDM for 2D heat/convection-diffusion equations on $[0,1]^2$ (LYAPACK benchmarks, $m = p = 1$) $\rightsquigarrow$ symmetric/nonsymmetric $A \in \mathbb{R}^{n\times n}$, $n = 10,000$.
- 15 shifts chosen by Penzl's heuristic from 50/25 Ritz/harmonic Ritz values of A.
- Computations using Intel Core 2 Quad CPU of type Q9400 at 2.66GHz with 4 GB RAM and 64Bit-MATLAB.

Numerical Results

Newton-ADI vs. Newton-ADI-Gelerkin

- FDM for 2D heat/convection-diffusion equations on $[0,1]^2$ (LYAPACK benchmarks, $m = p = 1$) $\rightsquigarrow$ symmetric/nonsymmetric $A \in \mathbb{R}^{n\times n}$, $n = 10,000$.
- 15 shifts chosen by Penzl's heuristic from 50/25 Ritz/harmonic Ritz values of A.
- Computations using Intel Core 2 Quad CPU of type Q9400 at 2.66GHz with 4 GB RAM and 64Bit-MATLAB.

Newton-ADI

step	rel. change	rel. residual	ADI
1	1	9.99e–01	200
2	9.99e–01	3.41e+01	23
3	5.25e–01	6.37e+00	20
4	5.37e–01	1.52e+00	20
5	7.03e–01	2.64e–01	23
6	5.57e–01	1.56e–02	23
7	6.59e–02	6.30e–05	23
8	4.02e–04	9.68e–10	23
9	8.45e–09	1.09e–11	23
10	1.52e–14	1.09e–11	23

CPU time: 76.9 sec.

Numerical Results

Newton-ADI vs. Newton-ADI-Gelerkin

- FDM for 2D heat/convection-diffusion equations on $[0,1]^2$ (LYAPACK benchmarks, $m = p = 1$) $\rightsquigarrow$ symmetric/nonsymmetric $A \in \mathbb{R}^{n \times n}$, $n = 10,000$.
- 15 shifts chosen by Penzl's heuristic from 50/25 Ritz/harmonic Ritz values of A.
- Computations using Intel Core 2 Quad CPU of type Q9400 at 2.66GHz with 4 GB RAM and 64Bit-MATLAB.

Newton-ADI

step	rel. change	rel. residual	ADI
1	1	9.99e–01	200
2	9.99e–01	3.41e+01	23
3	5.25e–01	6.37e+00	20
4	5.37e–01	1.52e+00	20
5	7.03e–01	2.64e–01	23
6	5.57e–01	1.56e–02	23
7	6.59e–02	6.30e–05	23
8	4.02e–04	9.68e–10	23
9	8.45e–09	1.09e–11	23
10	1.52e–14	1.09e–11	23

CPU time: 76.9 sec.

Newton-Galerkin-ADI

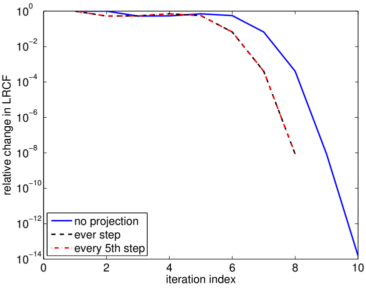
step	rel. change	rel. residual	ADI
1	1	3.56e–04	20
2	5.25e–01	6.37e+00	10
3	5.37e–01	1.52e+00	6
4	7.03e–01	2.64e–01	10
5	5.57e–01	1.57e–02	10
6	6.59e–02	6.30e–05	10
7	4.03e–04	9.79e–10	10
8	8.45e–09	1.43e–15	10

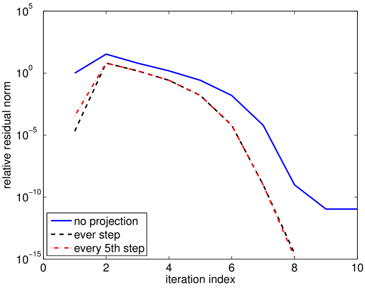
CPU time: 38.0 sec.

Numerical Results

Newton-ADI vs. Newton-ADI-Gelerkin

- FDM for 2D heat/convection-diffusion equations on $[0,1]^2$ (LYAPACK benchmarks, $m = p = 1$) $\rightsquigarrow$ symmetric/nonsymmetric $A \in \mathbb{R}^{n \times n}$, $n = 10,000$.
- 15 shifts chosen by Penzl's heuristic from 50/25 Ritz/harmonic Ritz values of A.
- Computations using Intel Core 2 Quad CPU of type Q9400 at 2.66GHz with 4 GB RAM and 64Bit-MATLAB.





Numerical Results

Newton-ADI vs. Newton-ADI-Gelerkin

- FDM for 2D heat/convection-diffusion equations on $[0,1]^2$ (LYAPACK benchmarks, $m = p = 1$) $\rightsquigarrow$ symmetric/nonsymmetric $A \in \mathbb{R}^{n \times n}$, $n = 10,000$.
- 15 shifts chosen by Penzl's heuristic from 50/25 Ritz/harmonic Ritz values of A.
- Computations using Intel Core 2 Quad CPU of type Q9400 at 2.66GHz with 4 GB RAM and 64Bit-MATLAB.

Newton-ADI

step	rel. change	rel. residual	ADI
1	1	9.99e–01	200
2	9.99e–01	3.56e+01	60
3	3.11e–01	3.72e+00	39
4	2.88e–01	9.62e–01	40
5	3.41e–01	1.68e–01	45
6	1.22e–01	5.25e–03	42
7	3.88e–03	2.96e–06	47
8	2.30e–06	6.09e–13	47

CPU time: 185.9 sec.

Numerical Results

Newton-ADI vs. Newton-ADI-Gelerkin

- FDM for 2D heat/convection-diffusion equations on $[0,1]^2$ (LYAPACK benchmarks, $m = p = 1$) $\rightsquigarrow$ symmetric/nonsymmetric $A \in \mathbb{R}^{n \times n}$, $n = 10,000$.
- 15 shifts chosen by Penzl's heuristic from 50/25 Ritz/harmonic Ritz values of A.
- Computations using Intel Core 2 Quad CPU of type Q9400 at 2.66GHz with 4 GB RAM and 64Bit-MATLAB.

Newton-ADI

step	rel. change	rel. residual	ADI
1	1	9.99e–01	200
2	9.99e–01	3.56e+01	60
3	3.11e–01	3.72e+00	39
4	2.88e–01	9.62e–01	40
5	3.41e–01	1.68e–01	45
6	1.22e–01	5.25e–03	42
7	3.88e–03	2.96e–06	47
8	2.30e–06	6.09e–13	47

CPU time: 185.9 sec.

Newton-Galerkin-ADI

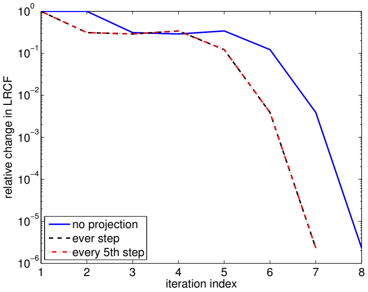
step	rel. change	rel. residual	ADI it.
1	1	1.78e–02	35
2	3.11e–01	3.72e+00	15
3	2.88e–01	9.62e–01	20
4	3.41e–01	1.68e–01	15
5	1.22e–01	5.25e–03	20
6	3.89e–03	2.96e–06	15
7	2.30e–06	6.14e–13	20

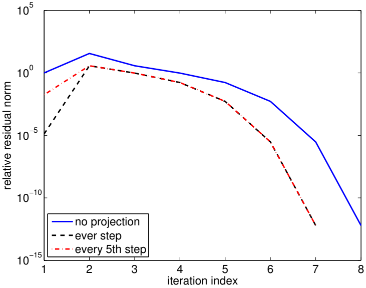
CPU time: 75.7 sec.

Numerical Results

Newton-ADI vs. Newton-ADI-Gelerkin

- FDM for 2D heat/convection-diffusion equations on $[0,1]^2$ (LYAPACK benchmarks, $m = p = 1$) $\rightsquigarrow$ symmetric/nonsymmetric $A \in \mathbb{R}^{n\times n}$, $n = 10,000$.
- 15 shifts chosen by Penzl's heuristic from 50/25 Ritz/harmonic Ritz values of A.
- Computations using Intel Core 2 Quad CPU of type Q9400 at 2.66GHz with 4 GB RAM and 64Bit-MATLAB.





Numerical Results

Example: LQR Problem for 3D Geometry

Control problem for 3d Convection-Diffusion Equation

- FDM for 3D convection-diffusion equation on $[0,1]^3$
- proposed in [SIMONCINI '07], $q = p = 1$
- non-symmetric $A \in \mathbb{R}^{n \times n}$, $n = 10\,648$

Test system:

INTEL Xeon 5160 3.00GHz ; 16 GB RAM; 64Bit-MATLAB (R2010a) using threaded BLAS; stopping tolerance: 10^{-10}

Numerical Results

Example: LQR Problem for 3D Geometry

Newton-ADI

NWT	rel. change	rel. residual	ADI
1	$1.0 \cdot 10^{0}$	$9.3 \cdot 10^{-01}$	100
2	$3.7 \cdot 10^{-02}$	$9.6 \cdot 10^{-02}$	94
3	$1.4 \cdot 10^{-02}$	$1.1 \cdot 10^{-03}$	98
4	$3.5 \cdot 10^{-04}$	$1.0 \cdot 10^{-07}$	97
5	$6.4 \cdot 10^{-08}$	$1.3 \cdot 10^{-10}$	97
6	$7.5 \cdot 10^{-16}$	$1.3 \cdot 10^{-10}$	97

CPU time: 4 805.8 sec.

NG-ADI inner= 5, outer= 1

NWT	rel. change	rel. residual	ADI
1	$1.0 \cdot 10^{0}$	$5.0 \cdot 10^{-11}$	80

CPU time: 497.6 sec.

NG-ADI inner= 1, outer= 1

NWT	rel. change	rel. residual	ADI
1	$1.0 \cdot 10^{0}$	$7.4 \cdot 10^{-11}$	71

CPU time: 856.6 sec.

NG-ADI inner= 0, outer= 1

NWT	rel. change	rel. residual	ADI
1	$1.0 \cdot 10^{0}$	$6.5 \cdot 10^{-13}$	100

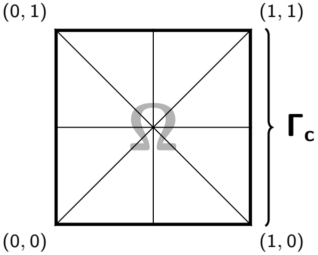
CPU time: 506.6 sec.

Test system:

INTEL Xeon 5160 3.00GHz ; 16 GB RAM; 64Bit-MATLAB (R2010a) using threaded BLAS; stopping tolerance: 10^{-10}

Numerical Results

Scaling of CPU times / Mesh Independence



$$
\begin{aligned}
\partial_t x(\xi,t) &= \Delta x(\xi,t) && \text{in } \Omega\\
\partial_\nu x &= b(\xi)\cdot u(t) - x && \text{on } \Gamma_c\\
\partial_\nu x &= -x && \text{on } \partial\Omega\setminus\Gamma_c\\
x(\xi,0) &= 1
\end{aligned}
$$

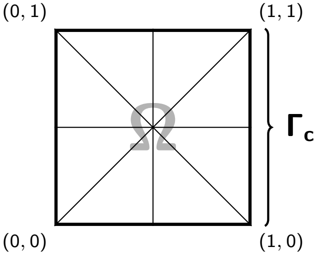
Note:

Here $b(\xi) = 4\,(1-\xi_2)\,\xi_2$ for $\xi \in \Gamma_c$ and 0 otherwise, thus $\forall t \in \mathbb{R}_{>0}$, we have $u(t) \in \mathbb{R}$.

$$\Rightarrow B_h = M_{\Gamma,h} \cdot b.$$

Numerical Results

Scaling of CPU times / Mesh Independence



$$
\begin{aligned}
\partial_t x(\xi, t) &= \Delta x(\xi, t) && \text{in } \Omega \\
\partial_\nu x &= b(\xi) \cdot u(t) - x && \text{on } \Gamma_c \\
\partial_\nu x &= -x && \text{on } \partial\Omega \setminus \Gamma_c
\end{aligned}
$$

$$x(\xi, 0) = 1$$

Consider: output equation $y = Cx$, where

$$
\begin{aligned}
C : \mathcal{L}^2(\Omega) &\to \mathbb{R} \\
x(\xi, t) &\mapsto y(t) = \int_\Omega x(\xi, t)\, d\xi
\end{aligned}
\quad \Rightarrow C_h = \underline{1} \cdot M_h.
$$

Numerical Results

Scaling of CPU times / Mesh Independence

Simplified Low Rank Newton-Galerkin ADI

- generalized state space form implementation
- Penzl shifts (16/50/25) with respect to initial matrices
- projection acceleration in every outer iteration step
- projection acceleration in every 5-th inner iteration step

Test system:

INTEL Xeon 5160 @ 3.00 GHz; 16 GB RAM; 64Bit-MATLAB (R2010a) using threaded BLAS,
stopping criterion tolerances: 10^{-10}

Numerical Results

Scaling of CPU times / Mesh Independence

Computation Times

discretization level	problem size	time in seconds
3	81	$4.87 \cdot 10^{-2}$
4	289	$2.81 \cdot 10^{-1}$
5	1 089	$5.87 \cdot 10^{-1}$
6	4 225	2.63
7	16 641	$2.03 \cdot 10^{+1}$
8	66 049	$1.22 \cdot 10^{+2}$
9	263 169	$1.05 \cdot 10^{+3}$
10	1 050 625	$1.65 \cdot 10^{+4}$
11	4 198 401	$1.35 \cdot 10^{+5}$

Test system:

INTEL Xeon 5160 @ 3.00 GHz; 16 GB RAM; 64Bit-MATLAB (R2010a) using threaded BLAS,
stopping criterion tolerances: 10^{-10}

Solving Large-Scale Matrix Equations
Software

Lyapack [Penzl 2000]

MATLAB toolbox for solving

- Lyapunov equations and algebraic Riccati equations,
- model reduction and LQR problems.

Main work horse: Low-rank ADI and Newton-ADI iterations.

Solving Large-Scale Matrix Equations

Software

Lyapack [Penzl 2000]

MATLAB toolbox for solving

- Lyapunov equations and algebraic Riccati equations,
- model reduction and LQR problems.

Main work horse: Low-rank ADI and Newton-ADI iterations.

M.E.S.S. – Matrix Equations Sparse Solvers [B./Köhler/Saak '08–]

- Extended and revised version of LYAPACK.
- Includes solvers for large-scale differential Riccati equations (based on Rosenbrock and BDF methods).
- Many algorithmic improvements:
  - new ADI parameter selection,
  - column compression based on RRQR,
  - more efficient use of direct solvers,
  - treatment of generalized systems without factorization of the mass matrix,
  - new ADI versions avoiding complex arithmetic etc.
- C and MATLAB versions.

Solving Large-Scale Matrix Equations
Software

Lyapack [Penzl 2000]

MATLAB toolbox for solving

- Lyapunov equations and algebraic Riccati equations,
- model reduction and LQR problems.

Main work horse: Low-rank ADI and Newton-ADI iterations.

M.E.S.S. – Matrix Equations Sparse Solvers [B./Köhler/Saak '08–]

- Extended and revised version of LYAPACK.
- Includes solvers for large-scale differential Riccati equations (based on Rosenbrock and BDF methods).
- Many algorithmic improvements:
  - new ADI parameter selection,
  - column compression based on RRQR,
  - more efficient use of direct solvers,
  - treatment of generalized systems without factorization of the mass matrix,
  - new ADI versions avoiding complex arithmetic etc.
- C and MATLAB versions.

Solving Large-Scale Matrix Equations
Software

Lyapack [Penzl 2000]

MATLAB toolbox for solving

- Lyapunov equations and algebraic Riccati equations,
- model reduction and LQR problems.

Main work horse: Low-rank ADI and Newton-ADI iterations.

M.E.S.S. – Matrix Equations Sparse Solvers [B./Köhler/Saak '08–]

- Extended and revised version of LYAPACK.
- Includes solvers for large-scale differential Riccati equations (based on Rosenbrock and BDF methods).
- Many algorithmic improvements:
  - new ADI parameter selection,
  - column compression based on RRQR,
  - more efficient use of direct solvers,
  - treatment of generalized systems without factorization of the mass matrix,
  - new ADI versions avoiding complex arithmetic etc.
- C and MATLAB versions.