Ansätze zur numerischen Lösung hochdimensionaler Lyapunov- und Sylvestergleichungen

Peter Benner

Mathematik in Industrie und Technik
Fakultät für Mathematik
TU Chemnitz

www.tu-chemnitz.de/~benner

benner@mathematik.tu-chemnitz.de

Hagen, 28. Oktober 2004

Outline

- Linear matrix equations
- Applications
- Direct Methods: Hessenberg-Schur, Bartels-Stewart, Hammarling
- Sign function method
- Low-rank approximation
- Large, sparse Lyapunov equations
- ADI and Smith iterations
- $\mathcal{H}$-matrix sign function method
- Conclusions

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.

Theoretical Background

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

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

Theoretical Background

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

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

Hence,

Sylvester equation has a unique solution

$$\Longleftrightarrow$$

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

Theoretical Background

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

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

Hence,

Sylvester equation has a unique solution

$$\Longleftrightarrow$$

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

$$\Longleftrightarrow$$

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

Theoretical Background

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

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

Hence,

Sylvester equation has a unique solution

$$\Longleftrightarrow$$

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

$$\Longleftrightarrow$$

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

$$\Longleftrightarrow$$

$$\sigma\,(A) \cap \sigma\,(-B) = \emptyset$$

Complexity Issues

Solving the Sylvester equation

$$AX + XB = W$$

via the linear system of equations

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

requires

- LU factorization of $n \cdot m \times n \cdot m$ matrix; for $n \approx m$, the complexity is $\mathcal{O}(n^6)$;
- storing $n \cdot m$ unknowns; i.e., for $n \approx m$ we have n^2 storage requirement for X.

Example:
$n = m = 1000 \Rightarrow$ Gaussian elimination on a Pentium IV (2 GHz) would take $10\frac{1}{2}$ YEARS.

Applications

Stability analysis of linear dynamical systems (Lyapunov's first method):

$\dot{x} = Ax$ is asymptotically (exponentially, Lyapunov) stable
$\iff$ $AP + PA^T + I_n = 0$ has solution $X > 0$.
$\iff$ $V(x) := x^T P x$ is a Lyapunov function for $\dot{x} = Ax$.

Feedback stabilization for $\dot{x} = Ax + Bu$.

Model reduction based on balancing and truncation

Block-diagonalization of matrices $\rightsquigarrow$ decoupling techniques for ordinary and partial differential equations

Image restoration and registration

Linear vibration analysis and damper optimization

.....

Model Reduction

- Given linear (control) system

$$\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t), \quad x(t) \in \mathbb{R}^n,$$

with A stable ($\sigma(A) \subset \mathbb{C}^-$), find reduced-order model

$$\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{B}u(t), \quad \tilde{y}(t) = \tilde{C}\tilde{x}(t), \quad \tilde{x}(t) \in \mathbb{R}^\ell$$

of degree $\ell \ll n$ such that $\|y - \tilde{y}\|$ "small".

- Balanced truncation: compute

$$(\tilde{A}, \tilde{B}, \tilde{C}) = (ZAT, ZB, CT)$$

where $Z := \Sigma_1^{-1/2} V_1^T R$, $T := S^T U_1 \Sigma_1^{-1/2}$ are defined via the SVD

$$SR^T = [U_1\ U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}$$

and S, R satisfy the controllability and observability Lyapunov equations

$$AS^TS + S^TSA^T + BB^T = 0, \qquad A^TR^TR + R^TRA + C^TC = 0.$$

Direct Methods

Bartels-Stewart method for Sylvester and Lyapunov equation;

Hessenberg-Schur method for Sylvester equations;

Hammarling's method for stable Lyapunov equations $AX + XA^T + GG^T = 0$.

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

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

is block-upper triangular. Hence, $Mx = b$ can be solved by back-substitution.

- Clever (recursive) implementation of back-substitution process requires $nm(n+m)$ flops.
- For Sylvester equations, B in Hessenberg form is enough ($\rightsquigarrow$ Hessenberg-Schur method).
- Hammarling's method modifies recursive process to compute 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 ($n > 1000$).

The Sign Function Method

[*Roberts '71*]

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

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

($J^{\pm}$ = Jordan blocks corresponding to $\sigma(Z) \cap \mathbb{C}^{\pm}$)

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

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

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

($c_j > 0$ is scaling parameter for convergence acceleration and rounding error minimization.)

Solving Sylvester Equations with the Sign Function Method

Consider Lyapunov equation $AX + XB + W = 0$, A, B stable.

As

$$\text{sign}\left(S^{-1}ZS\right) = S^{-1}\text{sign}\left(Z\right)S$$

and

$$\begin{bmatrix} I_n & 0 \\ X & I_m \end{bmatrix} \begin{bmatrix} A & 0 \\ W & -B \end{bmatrix} \begin{bmatrix} I_n & 0 \\ -X & I_m \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & -B \end{bmatrix}$$

it follows that

$$\text{sign}\left(\begin{bmatrix} A & 0 \\ W & -B \end{bmatrix}\right) = \begin{bmatrix} -I_n & 0 \\ 2X & I_m \end{bmatrix}.$$

Solution of Sylvester (Lyapunov) equation can be read off $\text{sign}\left(\begin{bmatrix} A & 0 \\ W & -B \end{bmatrix}\right)$.

The Sign Function Method for Sylvester Equations

Apply sign function Newton iteration $Z_{j+1} \leftarrow (Z_j + Z_j^{-1})/2$ to $Z = \begin{bmatrix} A & 0 \\ W & -B \end{bmatrix}$.

$$\Longrightarrow \quad \boxed{\operatorname{sign}(Z) = \lim_{j\to\infty} Z_j = \begin{bmatrix} -I_n & 0 \\ 2X_* & I_n \end{bmatrix}.}$$

Newton iteration (with scaling) is equivalent to

$$\begin{aligned} &A_0 \leftarrow A, \quad B_0 \leftarrow B, \quad W_0 \leftarrow W \\ &\texttt{for } j = 0, 1, 2, \ldots \\ &\qquad A_{j+1} \leftarrow \frac{1}{2c_j}\left(A_j + c_j^2 A_j^{-1}\right), \\ &\qquad B_{j+1} \leftarrow \frac{1}{2c_j}\left(B_j + c_j^2 B_j^{-1}\right), \\ &\qquad W_{j+1} \leftarrow \frac{1}{2c_j}\left(W_j + c_j^2 A_j^{-1} W_j B_j^{-1}\right). \end{aligned}$$

$$\Longrightarrow \quad \boxed{X_* = \tfrac{1}{2}\lim_{j\to\infty} W_j}$$

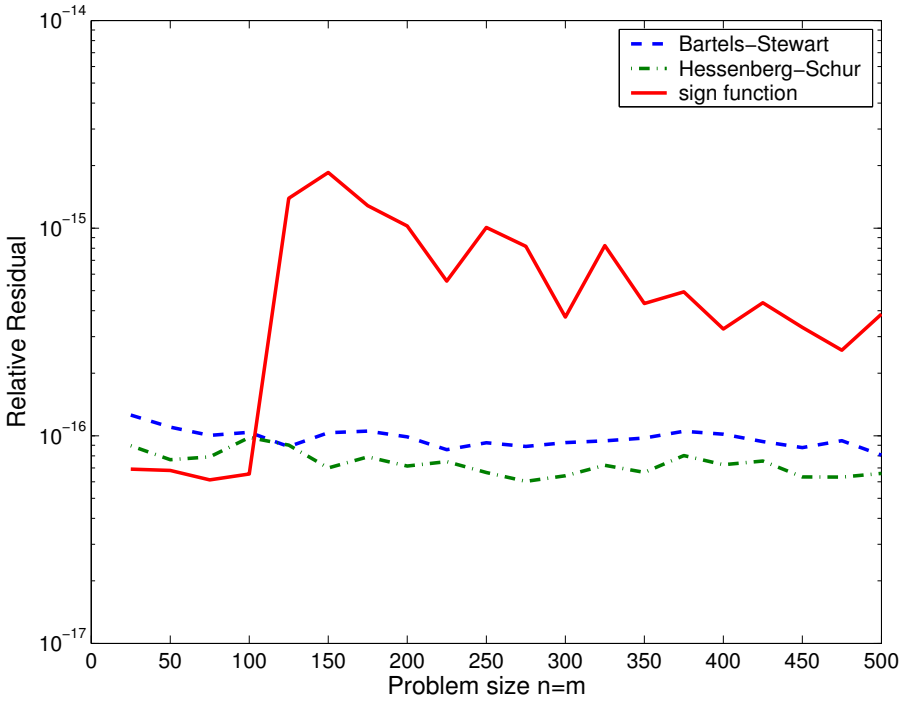
Requires explicit inverses $\Rightarrow$ not feasible for large-scale problems ($n > 1000$).

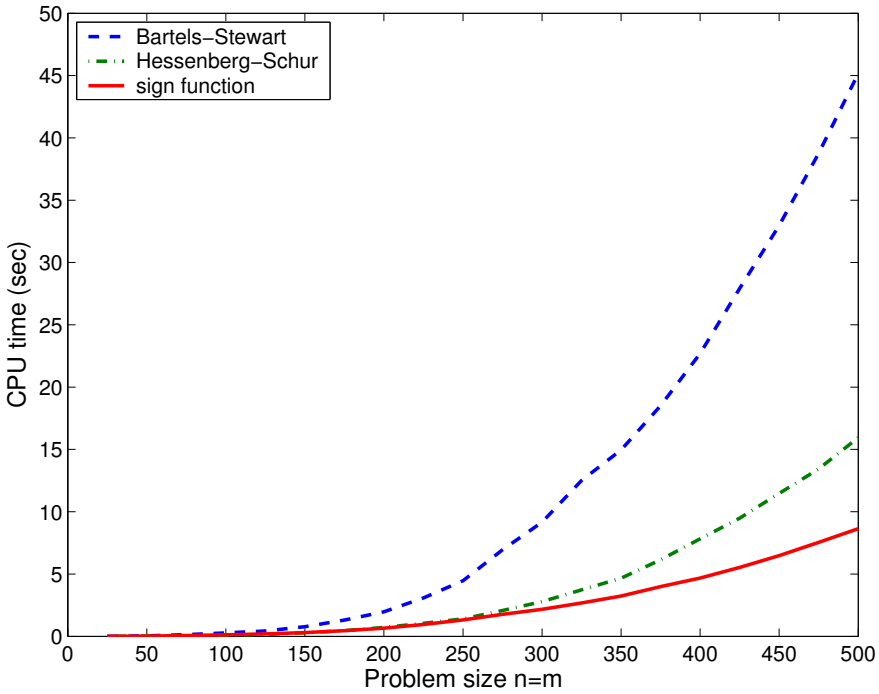
Comparison

Compare

- Bartels-Stewart method as implemented in the MATLAB Control Toolbox function `lyap`,
- Hessenberg-Schur method as implemented in the SLICOT-based MATLAB function `slsylv`
- sign function method

for random Sylvester equations.





Low-Rank Approximation

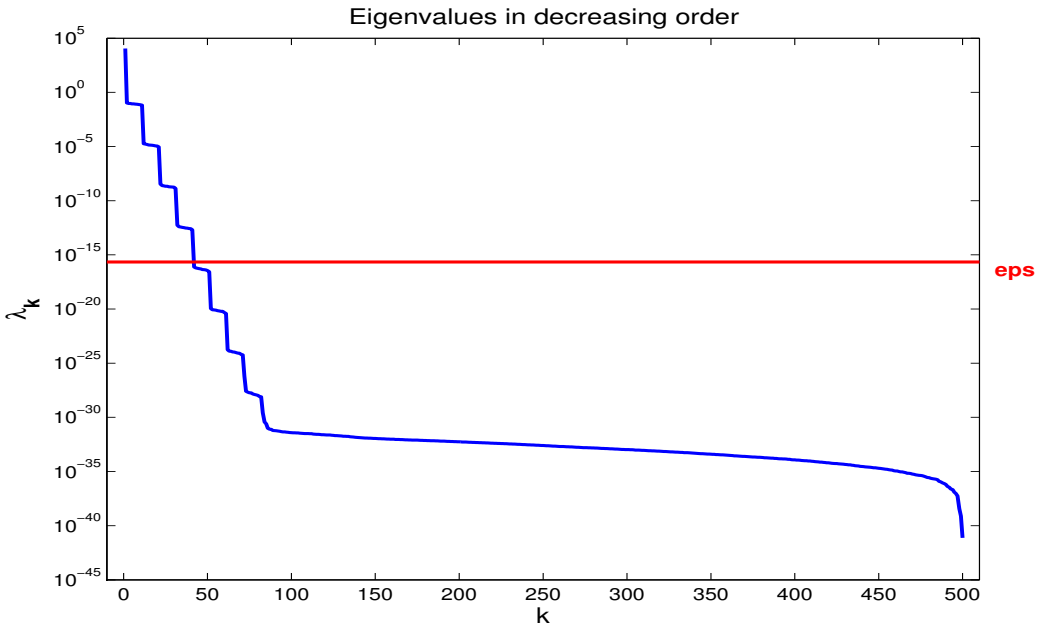
Consider the stable Lyapunov equation

$$A^T X + X A + C^T C = 0, \qquad A \text{ stable}, \; C \in \mathbb{R}^{p \times n}.$$

- $\exists$ unique solution $X \geq 0 \Longrightarrow X = Y^T Y$ for $Y \in \mathbb{R}^{n_Y \times n}$, $n_Y \leq n$.
- In almost all applications, $p \ll n$.
- Often, X has low (numerical) rank.

Example:

- "random" matrix A,
- stability margin ≈ 0.055,
- $n = 500$, $m = 10$,
- numerical rank $= 31$.



Computation of Y

Standard approach:

$Y = \left[\begin{array}{c} \diagdown \\ \end{array}\right] \in \mathbb{R}^{n \times n}$, Cholesky factor of X, $n_Y = n$.

Can be computed by Hammarling's method.

Approach here:

$Y \in \mathbb{R}^{\mathrm{rank}(X) \times n}$, full rank factor of X, $n_Y \ll n$.

Advantages:

- more reliable if Cholesky factor is numerically singular;
- more efficient if $\mathrm{rank}\,(X) \ll n$ (in particular for subsequent computations as in balanced truncation model reduction);

Sign Function Method Again

Want factor Y of solution of $A^T X + X A + C^T C = 0$.

For $W_0 = C_0^T C_0 := C^T C$, $C \in \mathbb{R}^{p \times n}$ obtain

$$W_{j+1} = \frac{1}{2c_j}\left(W_j + c_j^2 A_j^{-T} W_j A_j^{-1}\right) = \frac{1}{2c_j}\begin{bmatrix} C_j \\ c_j C_j A_j^{-1} \end{bmatrix}^T \begin{bmatrix} C_j \\ c_j C_j A_j^{-1} \end{bmatrix}.$$

$\Longrightarrow$ re-write W_j–iteration:

$$C_0 := C, \qquad C_{j+1} := \frac{1}{\sqrt{2c_j}}\begin{bmatrix} C_j \\ c_j C_j A_j^{-1} \end{bmatrix}.$$

Problem: $C_j \in \mathbb{R}^{p_j \times n} \quad \Longrightarrow \quad C_{j+1} \in \mathbb{R}^{2p_j \times n}$,
i.e., the necessary workspace doubles in each iteration.

Two approaches in order to limit work space...

Compute Cholesky factor Y_c of X

Require $p_j \leq n$: for $j > \log_2(n/p)$ compute QR factorization

$$\frac{1}{\sqrt{2c_j}} \begin{bmatrix} C_j \\ c_j C_j A_j^{-1} \end{bmatrix} = U_j \begin{bmatrix} \hat{C}_j \\ 0 \end{bmatrix}, \qquad \hat{C}_j = \begin{bmatrix} \diagdown \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

$$\Longrightarrow \quad W_j = \hat{C}_j^T \hat{C}_j, \qquad \boxed{Y_c = \tfrac{1}{\sqrt{2}} \lim_{j \to \infty} \hat{C}_j}$$

Compute full-rank factor Y_f of X

In every step compute rank-revealing QR factorization:

$$\frac{1}{\sqrt{2c_j}} \begin{bmatrix} C_j \\ c_j C_j A_j^{-1} \end{bmatrix} = U_{j+1} \begin{bmatrix} R_{j+1} & T_{j+1} \\ 0 & S_{j+1} \end{bmatrix} \Pi_{j+1},$$

where $R_{j+1} \in \mathbb{R}^{p_{j+1} \times p_{j+1}}$, $p_{j+1} = \operatorname{rank}\left(\begin{bmatrix} C_j \\ c_j C_j A_j^{-1} \end{bmatrix} \right)$. Then

$$C_{j+1} := [\, R_{j+1} \; T_{j+1} \,] \Pi_{j+1}, \quad W_{j+1} = C_{j+1}^T C_{j+1}, \qquad \boxed{Y_f = \tfrac{1}{\sqrt{2}} \lim_{j \to \infty} C_j}$$

Parallelization

- Newton iteration for sign function easy to parallelize – need basic linear algebra (systems of linear equations, matrix inverse, matrix addition, matrix product).
- Use MPI, BLACS for communication, PBLAS and ScaLAPACK for numerical linear algebra $\longrightarrow$ portable code.
- Development of software library PLiC.
- Testing on PC Cluster (Linux) with 32 Intel Xeon-2.4GHz processors with
  - workspace/processor: 1 GByte,
  - Myrinet Switch, bandwidth $\approx$ 1.4 Gbit/sec,

  shows good efficiency and high scalability.

Factorized Solution of Sylvester Equations

Let $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{m\times m}$ stable, $F \in \mathbb{R}^{n\times p}$, $G \in \mathbb{R}^{p\times m}$ and consider

$$AX + XB + FG = 0.$$

Idea: often, $\operatorname{rank}(X) \ll \min\{n, m\}$ or $X \approx \tilde{X}$ with $\operatorname{rank}\left(\tilde{X}\right) \ll n$.

$\operatorname{rank}(X) = r \implies \exists Y, Z^T \in \mathbb{R}^{n\times r}$ with $X = YZ$ (full-rank factorization, FRF).

Computing FRF with standard methods,

- Bartels-Stewart method
- Hessenberg-Schur method

would require to compute $X \in \mathbb{R}^{n\times m}$ and then use SVD or SVD-like factorization.

Here: compute FRF directly from F, G using sign function iteration.

Computing Full-Rank Factors of Solution

$\Longrightarrow$ re-write (unscaled) W_j iteration converging to $2X_*$ as

$$F_{j+1}G_{j+1} = W_{j+1} \leftarrow \frac{1}{2}\left(W_j + A_j^{-1}W_jB_j^{-1}\right) = \frac{1}{2}\left(F_jG_j + (A_j^{-1}F_j)(G_jB_j^{-1})\right),$$

yielding

$$F_0 := F, \quad F_{j+1} = \frac{1}{\sqrt{2}}\left[\, F_j,\ A_j^{-1}F_j \,\right],$$

$$G_0 := G, \quad G_{j+1} = \frac{1}{\sqrt{2}}\begin{bmatrix} G_j \\ G_jB_j^{-1} \end{bmatrix}.$$

Again: $F_j \in \mathbb{R}^{n\times p_j} \implies F_{j+1} \in \mathbb{R}^{n\times 2p_j}$, i.e., workspace doubles per iteration.

In order to limit workspace during iteration, compute full-rank factorization in each iteration step directly from F_{j+1}, G_{j+1}.

Keeping the Rank Small

ALGORITHM

1. Compute rank-revealing QR factorization (RRQR) $\begin{bmatrix} G_j \\ G_j B_j^{-1} \end{bmatrix} =: U \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \Pi_G,$
   with U orthogonal, Π_G a permutation matrix, $R_1 \in \mathbb{R}^{r \times m}$ upper triangular of full row-rank.
2. Compute RRQR $\begin{bmatrix} F_j, & A_j^{-1} F_j \end{bmatrix} U = V \begin{bmatrix} T_1 \\ 0 \end{bmatrix} \Pi_F,$
   with V orthogonal, Π_F a permutation matrix, $T_1 \in \mathbb{R}^{t \times 2p_j}$ upper triangular of full row-rank.
3. Partition $V = [\, V_1, \; V_2 \,]$, $V_1 \in \mathbb{R}^{n \times t}$, and compute $[\, T_{11}, \; T_{12} \,] := T_1 \Pi_F$, where $T_{11} \in \mathbb{R}^{t \times r}$.
4. Set $F_{j+1} := V_1 T_{11}, \qquad G_{j+1} := R_1 \Pi_G.$

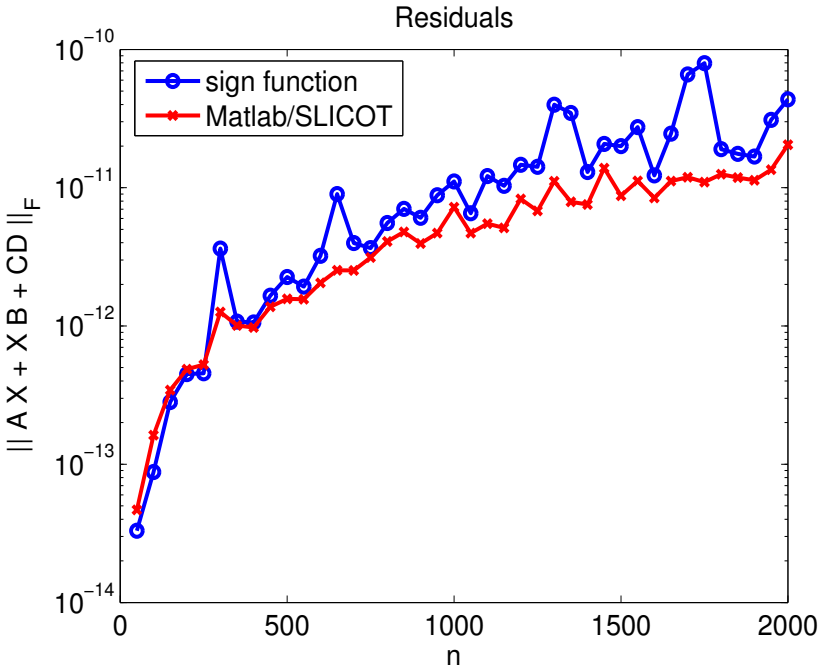
THEOREM

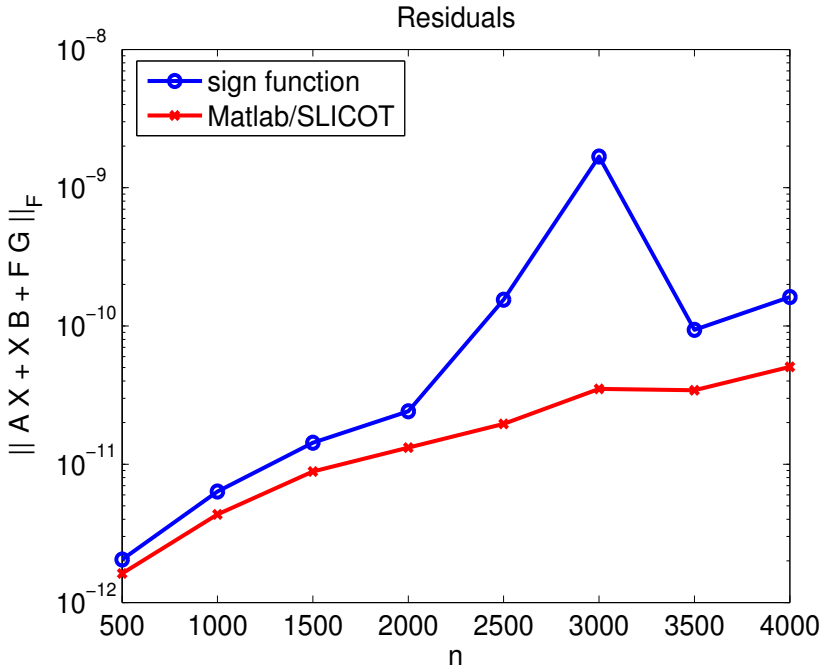
$$W_{j+1} = F_{j+1} G_{j+1}, \qquad Y_* := \frac{1}{\sqrt{2}} \lim_{j \to \infty} F_j \quad \text{and} \quad Z_* := \frac{1}{\sqrt{2}} \lim_{j \to \infty} G_j \quad \text{exist,}$$

and $X_* = Y_* Z_*$ is FRF.

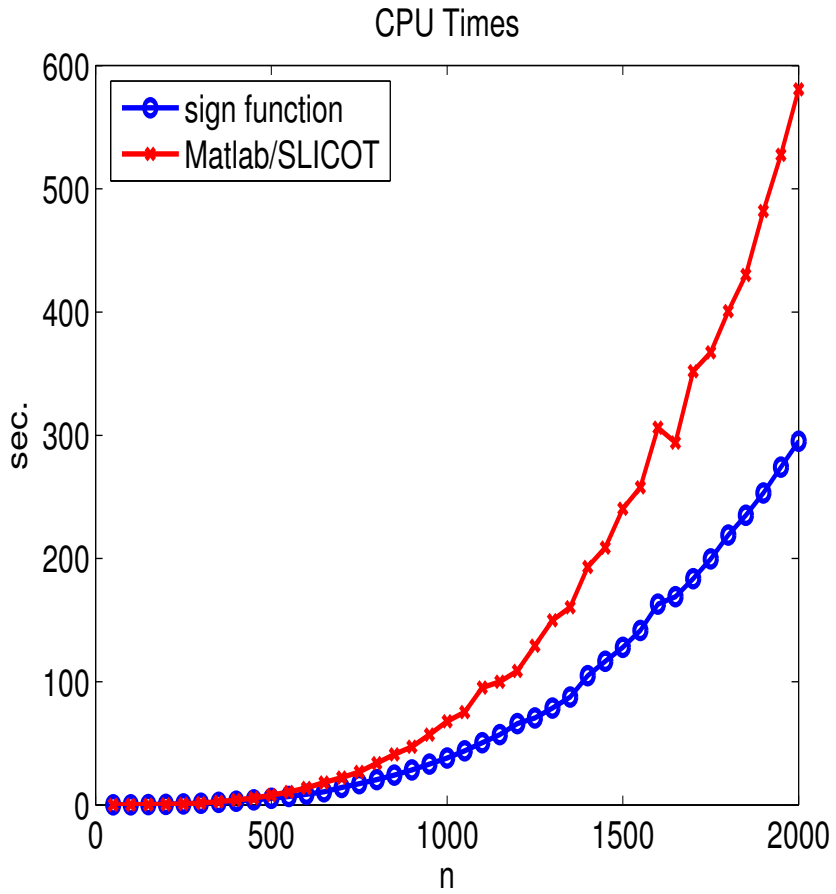
Numerical Example

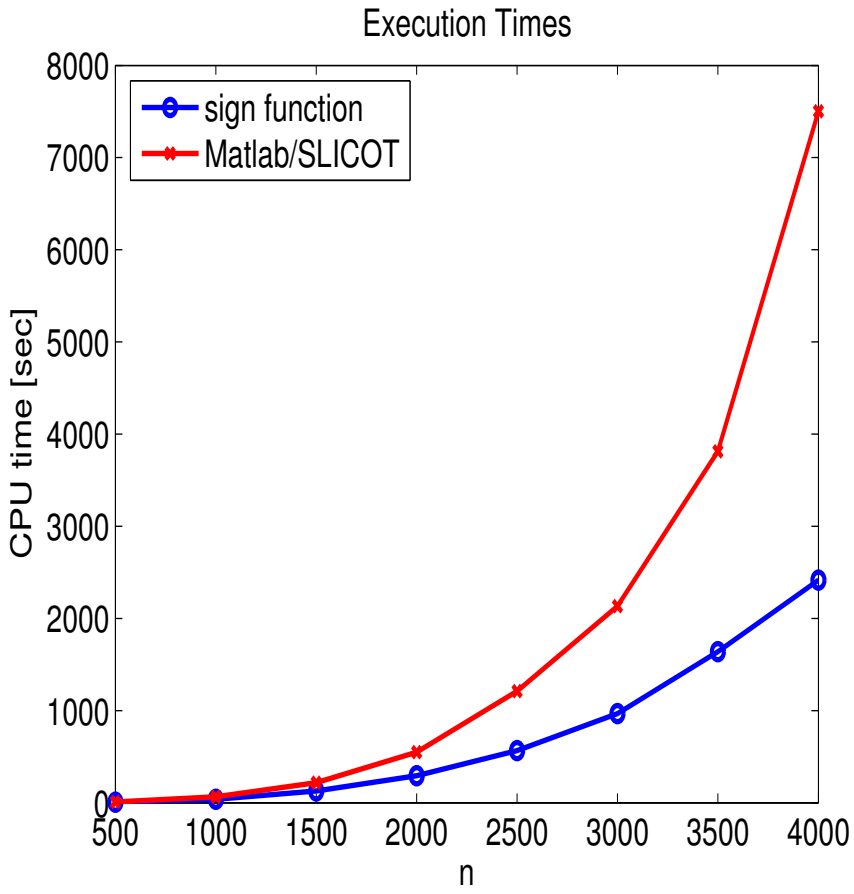
- $n = 50 : 50 : 2000$, $n = 500 : 500 : 4000$, $m = 1$, $\mathrm{sep}(A, B) = \frac{2}{n}$.
- Compare sign function based method and SLICOT-based implementation of the Hessenberg-Schur method.
- Use MATLAB Release 14 on a Xeon 3GHz workstation with 8GB Ram.
- 8–14 iterations for sign function iteration.
- For $n = 4000$, $Y, Z^T \in \mathbb{R}^{2000\times 47}$, i.e., 1.4MB instead of 122MB storage required for solution.





Timings


CPU Times
sign function
Matlab/SLICOT
sec.
n
0
100
200
300
400
500
600
0
500
1000
1500
2000


Execution Times
sign function
Matlab/SLICOT
CPU time [sec]
n
0
1000
2000
3000
4000
5000
6000
7000
8000
500
1000
1500
2000
2500
3000
3500
4000

Methods for Large, Sparse Lyapunov Equations

- Apply adapted splitting methods (Gauß-Seidel, SSOR, etc.)
- Apply adapted iterative solvers (GMRES, TFQMR, etc.)
- $X \approx V\tilde{X}V^T$ for solution $\tilde{X}$ of projected Lyapunov equation

$$(V^TAV)\tilde{X} + \tilde{X}(V^TAV)^T = V^TWV, \qquad V \in \mathbb{R}^{n\times\ell}.$$

Can be computed via Krylov subspace methods.

None of these approaches is efficient in general.

Idea: try to compute low-rank approximation to X via (approximation to) full-rank factor Y.

ADI Method for Lyapunov Equations

- For $A \in \mathbb{R}^{n\times n}$ stable, $W \in \mathbb{R}^{n\times w}$ $(w \ll n)$, consider Lyapunov equation

$$A^T X + XA = -BB^T.$$

- ADI Iteration: [*Wachspress '88*]

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

  with parameters $p_k \in \mathbb{C}^-$ and $p_{k+1} = \overline{p_k}$ if $p_k \notin \mathbb{R}$.

- For $X_0 = 0$ and proper choice of p_k: $\lim_{k\to\infty} X_k = X$ superlinear.

- Re-formulation using $X_k = Y_k Y_k^T$ yields iteration for Y_k...

Factored ADI Iteration

[*Penzl '97, Li/Wang/White '99, B./Li/Penzl*]

Set $X_k = Y_k Y_k^T$, some algebraic manipulations $\Longrightarrow$

$$V_1 \leftarrow \sqrt{-2\mathrm{Re}\,(p_1)}(A^T + p_1 I)^{-1} B, \qquad Y_1 \leftarrow V_1$$

FOR $j = 2, 3, \ldots$

$$V_k \leftarrow \sqrt{\frac{\mathrm{Re}\,(p_k)}{\mathrm{Re}\,(p_{k-1})}}\left(I - (p_k + \overline{p_{k-1}})(A^T + p_k I)^{-1}\right) V_{k-1}, \qquad Y_k \leftarrow \begin{bmatrix} Y_{k-1} & V_k \end{bmatrix}$$

$$\Downarrow$$

$$Y_{k_{\max}} = \begin{bmatrix} V_1 & \ldots & V_{k_{\max}} \end{bmatrix}$$

where

$$V_k = \boxed{} \in \mathbb{C}^{n \times w}$$

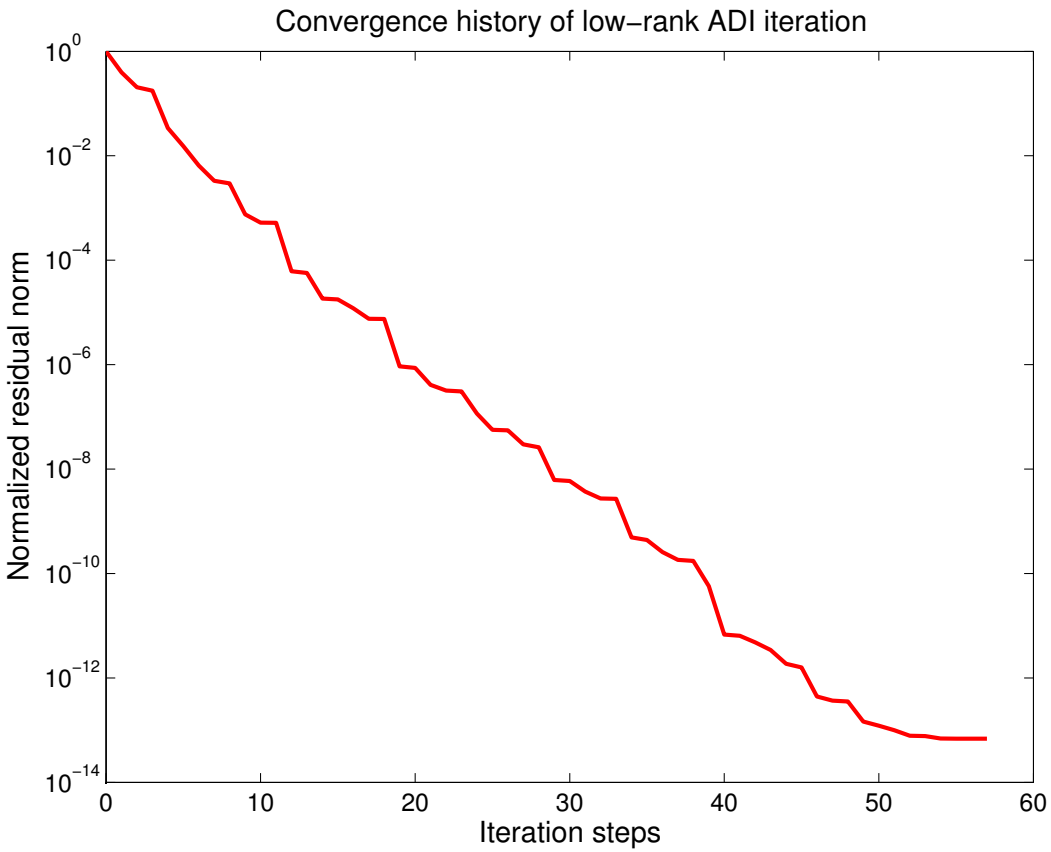
and

$$Y_{k_{\max}} Y_{k_{\max}}^T \approx X$$

Note: Implementation in real arithmetic possible by combining two steps.

Low-rank ADI in Action

- Solve $AX + XA^T + BB^T = 0$.
- Finite differences method for 2D heat equation with boundary control.
- $n = 10,000$, $m = 1$.
- Use LYAPACK implementation [*Penzl '99*].
- Need $k_{\max} = 57$ iterations, i.e., $Y_{k_{\max}} \in \mathbb{R}^{10,000 \times 57}$.
- ca. 25 sec. on Centrino 1.4 GHz notebook



Smith Iteration

For Stein equations

$$FXF^H - X = G \tag{1}$$

with $\|F\| < 1$, the fix point iteration

$$X_{k+1} = FX_kF^H - G, \quad k = 0, 1, \ldots \quad (X_0 = 0)$$

converges to the unique solution X_*.

Smith iteration for Lyapunov equations: if X_* is a solution of

$$AX + XA^T = W,$$

then it is a solution of (1) with

$$F = (I_n + \mu A)^{-1}(I_n - \bar{\mu}A), \quad G = 2(\mu + \bar{\mu})(I_n + \mu A)^{-1}W(I_n + \mu A)^{-H}.$$

Choosing μ such that $\sigma(F) \subset \{|z| < 1\}$, we can compute X_* as the limit of

$$X_{k+1} = (I_n + \mu A)^{-1}\left((I_n - \bar{\mu}A)X_k(I_n - \mu A)^H + 2(\mu + \bar{\mu})W\right)(I_n + \mu A)^{-H}.$$

Details and Extensions

- If A is stable, any $\mu \in \mathbb{C}^-$ can be used.
  $\rightsquigarrow$ Select μ in order to optimize convergence rate ($\rho(F)^2$).

  For $\sigma(A) \subset \mathbb{R}^-$, $\mu = -\sqrt{\lambda_{\max}\lambda_{\min}}$ is optimal. [*Varga '62, Rosen/Wang '95*]

- Convergence can be accelerated by changing the shift μ in each iteration; in order to limit work needed for factorizations, use finite number ℓ of different shifts and apply them cyclically, store and re-use the ℓ different factorizations if workspace permits.

  $\rightsquigarrow$ (cyclic) Smith(ℓ) method. [*Penzl '00*].

- Low-rank version possible if $W = BB^T$, mathematically equivalent to low rank ADI with parameters applied cyclically. [*Penzl '00*].

  Number of columns in Y_k ($X_k = Y_k Y_k^H$) can be reduced by column compression technique. [*Antoulas/Sorensen/Gugercin '03*].

$\mathcal{H}$-Matrix Implementation of the Sign Function Method

Recall: solution of the Lyapunov equation

$$AX + XA^T + W = 0$$

with the sign function method:

$$\begin{aligned} A_0 &= A, \quad W_0 = BB^T \\ A_{j+1} &= \frac{1}{2}(A_j + A_j^{-1}) \\ W_{j+1} &= \frac{1}{2}(W_j + A_j^{-T} W_j A_j^{-1}) \end{aligned}$$

involves the inversion, addition and multiplication of $n \times n$ matrices
$\hookrightarrow$ complexity: $\mathcal{O}(n^3)$

Approximation of A in $\mathcal{H}$-matrix format, use of the formatted $\mathcal{H}$-matrix arithmetic
$\hookrightarrow$ complexity: $\mathcal{O}(n \log^2 n)$.

Hierarchical Matrices: Short Introduction

Hierarchical ($\mathcal{H}$-)matrices are a data-sparse approximation of large, dense matrices arising

- from the discretization of non-local integral operators occurring in BEM,
- as inverses of FEM discretized elliptic differential operators,

but can also be used to represent FEM matrices directly.

Properties of $\mathcal{H}$-matrices:

- only few data are needed for the representation of the matrix,
- matrix-vector multiplication can be performed in almost linear complexity ($\mathcal{O}(n \log n)$),
- building sums, products, inverses is of "almost" linear complexity.

Hierarchical Matrices: Construction

Consider matrices over a product index set $I \times I$.

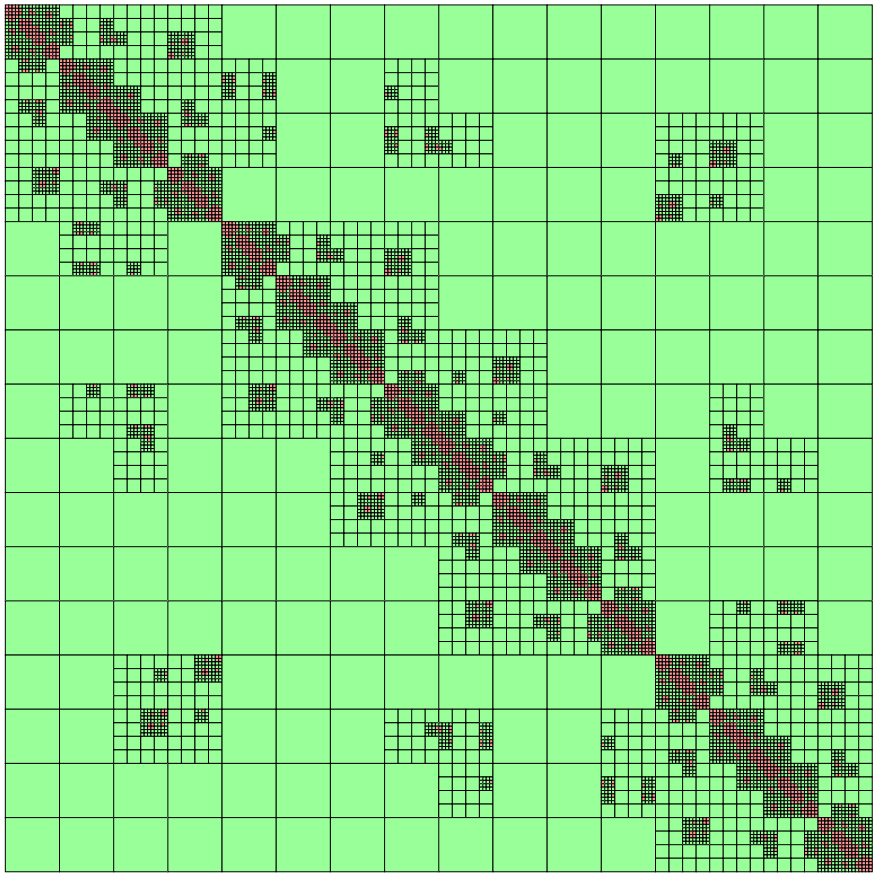
Partition $I \times I$ by the $\mathcal{H}$-tree $T_{I\times I}$, where a problem dependent admissibility condition is used to decide whether a block $t \times s \subset I \times I$ allows for a low rank approximation.

Definition: Hierarchical matrices ($\mathcal{H}$-matrices)
The set of the hierarchical matrices is defined by

$$\mathcal{H}(T_{I\times I}, k) := \{M \in \mathbb{R}^{I\times I} |\ \operatorname{rank}(M|_{t\times s}) \leq k \,\forall \text{ admissible leaves } t\times s \text{ of } T_{I\times I}\}$$

Submatrices of $M \in \mathcal{H}(T_{I\times I}, k)$ corresponding to inadmissible leaves are stored as dense blocks whereas those corresponding to admissible leaves are stored in factorized form as rank-k matrices, called R_k-format.

Example



Stiffness matrix of 2D heat equation with distributed control and isolation BC
$n = 1024$ and $k = 4$

Hierarchical Matrices: Formatted Arithmetic

$\mathcal{H}(T_{I\times I}, k)$ is not a linear subspace of $\mathbb{R}^{I\times I}$ $\rightsquigarrow$ formatted arithmetics
$\rightsquigarrow$ projection of the sum, product and inverse into $\mathcal{H}(T_{I\times I}, k)$

1. Formatted Addition ($\oplus$)
   with complexity $\mathcal{N}_{\mathcal{H}\oplus\mathcal{H}} = \mathcal{O}(nk^2 \log n))$ (for sparse $T_{I\times I}$)
   Corresponds to best approximation (in the Frobenius-norm).

2. Formatted Multiplication ($\odot$)
   $\mathcal{N}_{\mathcal{H}\odot\mathcal{H}} = \mathcal{O}(nk^2 \log^2 n)$ (under some technical assumptions on $T_{I\times I}$)

3. Formatted inversion ($\widetilde{Inv}$)
   $\mathcal{N}_{\mathcal{H},\widetilde{Inv}} = \mathcal{O}(nk^2 \log^2 n)$ (under some technical assumptions on $T_{I\times I}$)

$\mathcal{H}$-Matrix Sign Function Method

Sign function iteration with formatted $\mathcal{H}$-matrix arithmetic:

$$\tilde{A}_0 = (A)_{\mathcal{H}}, \qquad \tilde{A}_{j+1} = \frac{1}{2}(\tilde{A}_j \oplus \widetilde{Inv}(A_j))$$

Accuracy control for iterates $\rightsquigarrow k = \mathcal{O}(\log \frac{1}{\delta} + \log \frac{1}{\rho})$, where

$$\begin{aligned} \|(\tilde{A}_j^{-1} - \widetilde{\mathrm{Inv}}(\tilde{A}_j))\|_2 &\leq \delta \\ \|(\tilde{A}_j^{-1} + \widetilde{\mathrm{Inv}}(\tilde{A}_j)) - (\tilde{A}_j^{-1} \oplus \widetilde{\mathrm{Inv}}(\tilde{A}_j))\|_2 &\leq \rho \end{aligned}$$

$\Longrightarrow$ forward error bound (assuming $c_j(\delta+\rho)\|A_j^{-1}\|_2 < 1 \ \forall j$):

$$\|\tilde{A}_j - A_j\|_2 \leq c_j(\delta+\rho),$$

where

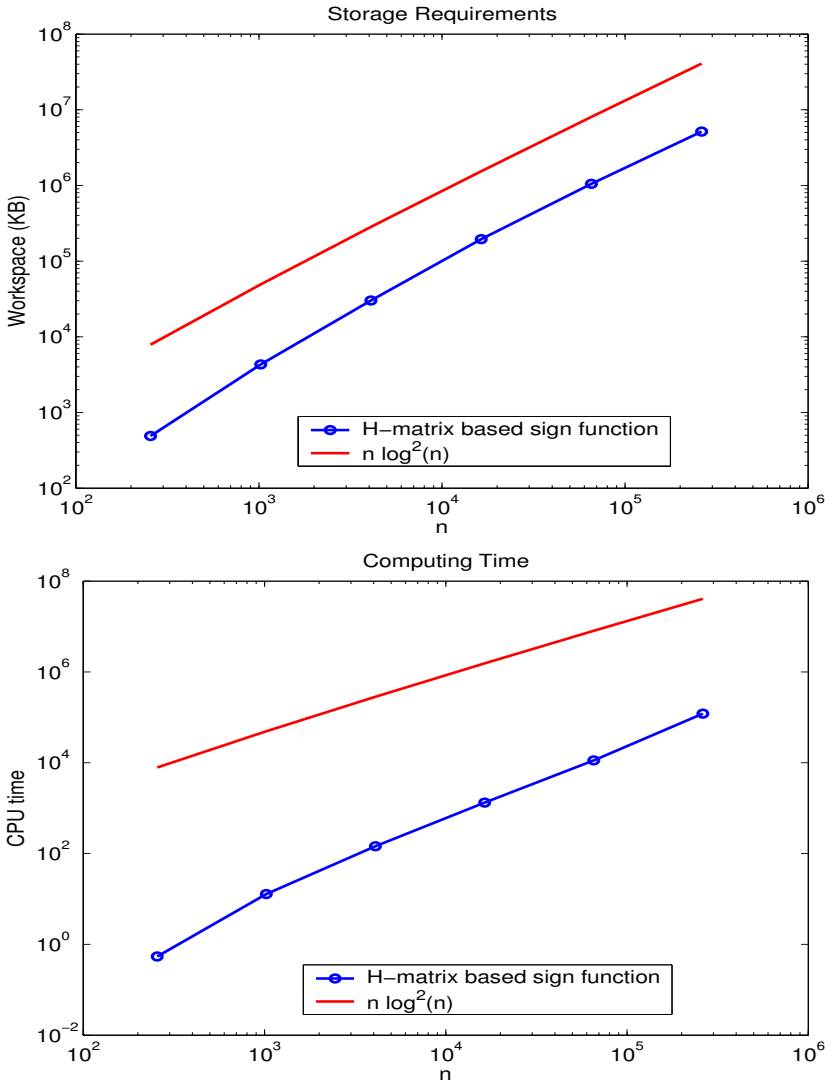
$$c_0 = \|\tilde{A}_0 - A\|_2(\delta+\rho)^{-1}, \quad c_{j+1} = \frac{1}{2}\left(1 + c_j + c_j \frac{\|A_j^{-1}\|_2^2}{1 - c_j(\delta+\rho)\|A_j^{-1}\|_2}\right).$$

Numerical Performance

- Solve $AX + XA^T + BB^T = 0$.
- FEM discretization of 2D heat equation with boundary control.
- Accuracy and rank of computed factor

n	r	$\frac{\|AX+XA^T+BB^T\|_2}{2\|A\|_2\|X\|_2+\|BB^T\|_2}$
256	11	$8.4 \cdot 10^{-8}$
1024	13	$4.4 \cdot 10^{-6}$
4096	14	$5.3 \cdot 10^{-6}$
16384	15	$4.8 \cdot 10^{-6}$

- For $n = 262,144$ (that is, 34 billion unknowns in X) we get $r = 21$
  $\Rightarrow$ 5MB for solution instead of 64GB!



results by U. Baur

Conclusions

- Large-scale dense problems can be efficiently solved using parallel implementation of the sign function method.
- Large-scale dense, but data-sparse problems can be efficiently solved using $\mathcal{H}$-matrix implementation of the sign function method.
- The most promising methods for large-scale sparse problems are low-rank Smith and ADI methods; selection of acceleration shifts remains a tricky issue.
- Krylov subspace methods and splitting methods are not competitive in general.
- To-Do:
  - $\mathcal{H}$-matrix sign function for Sylvester equations
  - Low-rank ADI method for Sylvester equations
  - Low-rank Smith method for Sylvester equations