The Matrix Factorization Paradigm in Solving Matrix Equations

Peter Benner

Professur Mathematik in Industrie und Technik
Fakultät für Mathematik
Technische Universität Chemnitz
benner@mathematik.tu-chemnitz.de

Householder Symposium XVI
Seven Springs, May 22 – 27, 2005

Thanks to: Enrique Quintana-Ortí, Gregorio Quintana-Ortí, Ulrike Baur

Outline

Problem

Want solution of

$$F(X) = 0, \qquad F : \mathbb{R}^m \to \mathbb{R}^n.$$

Hence, $X \in \mathbb{R}^{n \times m}$, i.e., we have a nonlinear system of equations with $n \cdot m$ unknowns!

Example: *$n = m = 1,000$, then the solution needs $1,000,000$ words of memory, i.e., 8MB in double precision.*

In control design or model reduction for partial differential equations, this task has to be solved for $n = 10,000$ (800 MB), $n = 100,000$ (80 GB), $n = 1,000,000$ (8 TB).

Problem

Want solution of

$$F(X) = 0, \qquad F : \mathbb{R}^m \to \mathbb{R}^n.$$

Hence, $X \in \mathbb{R}^{n \times m}$, *i.e., we have a nonlinear system of equations with* $n \cdot m$ *unknowns!*

Example: $n = m = 1,000$, *then the solution needs* $1,000,000$ *words of memory, i.e., 8MB in double precision.*

In control design or model reduction for partial differential equations, this task has to be solved for $n = 10,000$ *(800 MB),* $n = 100,000$ *(80 GB),* $n = 1,000,000$ *(8 TB).*

Introduction: The Matrix Factorization Paradigm

Problem

Want solution of

$$F(X) = 0, \qquad F : \mathbb{R}^m \to \mathbb{R}^n.$$

Hence, $X \in \mathbb{R}^{n \times m}$*, i.e., we have a nonlinear system of equations with* $n \cdot m$ *unknowns!*

Example: $n = m = 1,000$*, then the solution needs* $1,000,000$ *words of memory, i.e., 8MB in double precision.*

In control design or model reduction for partial differential equations, this task has to be solved for $n = 10,000$ *(800 MB),* $n = 100,000$ *(80 GB),* $n = 1,000,000$ *(8 TB).*

$$0 = \mathcal{L}(X) := A^T X + XA + C^T C, \qquad A \in \mathbb{R}^{n\times n}, \quad C \in \mathbb{R}^{p\times n}$$

$\Lambda(A) \subset \mathbb{C}^- \Rightarrow$ there exists unique solution $X = -\mathcal{L}^{-1}(C^T C) \geq 0.$

$$0 = \mathcal{L}(X) := A^T X + XA + C^T C, \qquad A \in \mathbb{R}^{n \times n}, \quad C \in \mathbb{R}^{p \times n}$$

$\Lambda(A) \subset \mathbb{C}^- \Rightarrow$ there exists unique solution $X = -\mathcal{L}^{-1}(C^T C) \geq 0$.

In many situations, $\operatorname{rank}(X, \varepsilon) = n_X \ll n$!

[Penzl '00, Sorensen/Zhou '02, Grasedyck '04]

$$0 = \mathcal{L}(X) := A^T X + XA + C^T C, \qquad A \in \mathbb{R}^{n \times n}, \quad C \in \mathbb{R}^{p \times n}$$

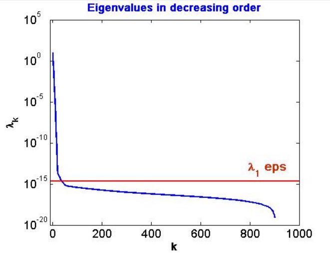
$\Lambda(A) \subset \mathbb{C}^- \Rightarrow$ there exists unique solution $X = -\mathcal{L}^{-1}(C^T C) \geq 0$.

In many situations, $\operatorname{rank}(X, \varepsilon) = n_X \ll n$!

[Penzl '00, Sorensen/Zhou '02, Grasedyck '04]

Example

- Control of 2D heat equation in unit square; heat source on the left boundary.
- FDM discretization, $h = 1/30$.
- $n = 900, \ m = 1$.



$$0 = \mathcal{R}(X) := C^T C + A^T X + XA - XBB^T X,$$
$$A \in \mathbb{R}^{n\times n}, \quad B \in \mathbb{R}^{n\times m}, \quad C \in \mathbb{R}^{p\times n}.$$

In most applications (optimal control, model reduction) need stabilizing solution X_s, i.e., $\Lambda(A - BB^T X_s) \subset \mathbb{C}^-$.

$$0 = \mathcal{R}(X) := C^T C + A^T X + XA - XBB^T X,$$
$$A \in \mathbb{R}^{n\times n}, \quad B \in \mathbb{R}^{n\times m}, \quad C \in \mathbb{R}^{p\times n}.$$

In most applications (optimal control, model reduction) need stabilizing solution X_s, i.e., $\Lambda\left(A - BB^T X_s\right) \subset \mathbb{C}^-$.

Usually, $X_s \geq 0$ and often, $\mathrm{rank}\left(X_s, \varepsilon\right) = n_X \ll n$!

Matrix Equations
Algebraic Riccati Equations

$$0 = \mathcal{R}(X) := C^T C + A^T X + XA - XBB^T X,$$
$$A \in \mathbb{R}^{n\times n}, \quad B \in \mathbb{R}^{n\times m}, \quad C \in \mathbb{R}^{p\times n}.$$

In most applications (optimal control, model reduction) need stabilizing solution X_s, i.e., $\Lambda(A - BB^T X_s) \subset \mathbb{C}^-$.

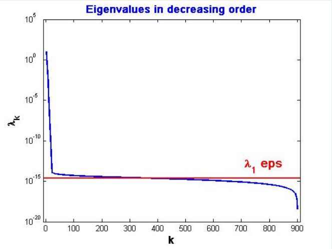
Usually, $X_s \geq 0$ and often, $\operatorname{rank}(X_s, \varepsilon) = n_X \ll n$!

Example

– Control of 2D heat equation in unit square; heat source on the left boundary, observations on right boundary.

– FDM discretization, $h = 1/30$.

– $n = 900$, $m = 1$, $p = 1$.



$$0 = \mathcal{S}(X) := AX + XB + W, \quad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{m\times m},\ W \in \mathbb{R}^{n\times m}$$

$\Lambda(A), \Lambda(B) \subset \mathbb{C}^- \Rightarrow$ exists unique solution $X = -\mathcal{S}^{-1}(W)$.

$$0 = \mathcal{S}(X) := AX + XB + W, \quad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{m\times m},\ W \in \mathbb{R}^{n\times m}$$

$\Lambda(A), \Lambda(B) \subset \mathbb{C}^- \Rightarrow$ exists unique solution $X = -\mathcal{S}^{-1}(W)$.

In many situations, $\operatorname{rank}(X, \varepsilon) = n_S \ll n$! [Grasedyck '04]

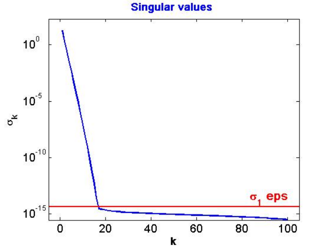
$$0 = \mathcal{S}(X) := AX + XB + W, \quad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{m\times m},\ W \in \mathbb{R}^{n\times m}$$

$\Lambda(A), \Lambda(B) \subset \mathbb{C}^- \Rightarrow$ exists unique solution $X = -\mathcal{S}^{-1}(W)$.

In many situations, $\operatorname{rank}(X, \varepsilon) = n_S \ll n$! [Grasedyck '04]

Example

– Observer-like equation for 2D heat equation in unit square; heat source on the left boundary.
– FDM discretization, $h = 1/30$.
– $n = 900,\ m = 100$.



Matrix Factorization

In Numerical Linear Algebra, solving problems often means to compute a matrix factorization (decomposition).

Alston S. Householder [*Principles of Numerical Analysis, 1954*] was probably the first one to understand this as a paradigm: it is a fundamental algorithmic pattern combined with the idea of re-using factors.

"This [the decompositional] approach,. . . ,has revolutionized matrix computation" [G.W. Stewart in The Top 10 of Algorithms, 2000]

One step further:

Never form a product once you are given factors, use these factors as much as possible!

[Parlett 1996, plenary talk at ILAS Conference Chemnitz]

Of course, this idea was driving earlier efforts to construct the QZ algorithm, quotient/product SVDs, . . .

Matrix Factorization

In Numerical Linear Algebra, solving problems often means to compute a matrix factorization (decomposition).

Alston S. Householder [*Principles of Numerical Analysis, 1954*] was probably the first one to understand this as a paradigm: it is a fundamental algorithmic pattern combined with the idea of re-using factors.

"This [the decompositional] approach,. . . ,has revolutionized matrix computation" [G.W. Stewart in The Top 10 of Algorithms, 2000]

One step further:

Never form a product once you are given factors, use these factors as much as possible!

[Parlett 1996, plenary talk at ILAS Conference Chemnitz]

Of course, this idea was driving earlier efforts to construct the QZ algorithm, quotient/product SVDs, . . .

Matrix Factorization

In Numerical Linear Algebra, solving problems often means to compute a matrix factorization (decomposition).

Alston S. Householder [*Principles of Numerical Analysis, 1954*] was probably the first one to understand this as a paradigm: it is a fundamental algorithmic pattern combined with the idea of re-using factors.

"This [the decompositional] approach,. . . ,has revolutionized matrix computation" [G.W. Stewart in The Top 10 of Algorithms, 2000]

One step further:

Never form a product once you are given factors, use these factors as much as possible!

[Parlett 1996, plenary talk at ILAS Conference Chemnitz]

Of course, this idea was driving earlier efforts to construct the QZ algorithm, quotient/product SVDs, . . .

Factorized Solution of Matrix Equations

Here, we will only consider algorithms which directly compute factors of the solution to $F(X) = 0$, e.g.,

- Cholesky factors,
- full-rank factors, i.e., $X = YZ, \quad Y, Z^T \in \mathbb{R}^{n \times \mathrm{rank}(X)}$.

Not in this class: projection-type methods, i.e.,

1. *project* equation to $R^\ell \rightsquigarrow V^T F(V\hat{X}V^T)V = 0,\ V \in \mathbb{R}^{n\times\ell}$,
2. *solve* small-dimension problem $\rightsquigarrow \hat{X}$,
3. *lift* to $\mathbb{R}^n \rightsquigarrow X \approx V\hat{X}V^T$,

like in

- [SAAD '90, JAIMOUKHA/KASENALLY '94, HOCHBRUCK/STARKE '95, JBILOU/RIQUET '03] for Lyapunov equations;
- [HU/REICHEL '92, GRASEDYCK/HACKBUSCH 04] for Sylvester equations;
- [FREUND/MEHRMANN '92, B./FASSBENDER 97, JBILOU '03, AMODEI '03] for Riccati equations.

Factorized Solution of Matrix Equations

Here, we will only consider algorithms which directly compute factors of the solution to $F(X) = 0$, e.g.,

- Cholesky factors,
- full-rank factors, i.e., $X = YZ, \quad Y, Z^T \in \mathbb{R}^{n \times \mathrm{rank}(X)}$.

Not in this class: projection-type methods, i.e.,

1. *project* equation to $R^\ell \rightsquigarrow V^T F(V\hat{X}V^T)V = 0, \; V \in \mathbb{R}^{n\times\ell}$,
2. *solve* small-dimension problem $\rightsquigarrow \; \hat{X}$,
3. *lift* to $\mathbb{R}^n \rightsquigarrow X \approx V\hat{X}V^T$,

like in

- [Saad '90, Jaimoukha/Kasenally '94, Hochbruck/Starke '95, Jbilou/Riquet '03] for Lyapunov equations;
- [Hu/Reichel '92, Grasedyck/Hackbusch 04] for Sylvester equations;
- [Freund/Mehrmann '92, B./Fassbender 97, Jbilou '03, Amodei '03] for Riccati equations.

Lyapunov Equations

$$X \geq 0 \Rightarrow X = YY^T,\ Y = \blacksquare \in \mathbb{R}^{n \times n_X}.$$

- $n_X = n$, $Y =$ **Cholesky factor**
  - *Hammarling's method* (variant of Bartels-Stewart, based on Schur decomposition) [Hammarling '82]
  - *Sign function method* [Larin/Aliev '93, B./Quintana-Ortí '97]
- $n_X < n$, $Y =$ **(approximate) full-rank factor** $(X \approx YY^T)$
  - *Sign function method*, $\mathcal{O}(n^3)$ [B./Quintana-Ortí '97]
  - *H-Matrix based sign function method*, $\mathcal{O}(n \log^2(n))$ [Baur/B. '03–'05]
  - Low-rank ADI/Smith methods [Penzl '99, Li/white '00/'02, Sorensen et al '03]

Factorized Solution of Matrix Equations

Lyapunov Equations

$$X \geq 0 \Rightarrow X = YY^T, \; Y = \blacksquare \in \mathbb{R}^{n \times n_X}.$$

- $n_X = n$, $Y =$ **Cholesky factor**
  - *Hammarling's method* (variant of Bartels-Stewart, based on Schur decomposition) [Hammarling '82]
  - *Sign function method* [Larin/Aliev '93, B./Quintana-Ortí '97]
- $n_X < n$, $Y =$ **(approximate) full-rank factor** ($X \approx YY^T$)
  - *Sign function method*, $\mathcal{O}(n^3)$ [B./Quintana-Ortí '97]
  - *H-Matrix based sign function method*, $\mathcal{O}(n \log^2(n))$ [Baur/B. '03–'05]
  - Low-rank ADI/Smith methods [Penzl '99, Li/white '00/'02, Sorensen et al '03]

Factorized Solution of Matrix Equations

Lyapunov Equations

$$X \geq 0 \Rightarrow X = YY^T, \; Y = \blacksquare \in \mathbb{R}^{n \times n_X}.$$

- $n_X = n$, $Y =$ **Cholesky factor**
  - *Hammarling's method* (variant of Bartels-Stewart, based on Schur decomposition) [Hammarling '82]
  - *Sign function method* [Larin/Aliev '93, B./Quintana-Ortí '97]
- $n_X < n$, $Y =$ **(approximate) full-rank factor** $(X \approx YY^T)$
  - *Sign function method*, $\mathcal{O}(n^3)$ [B./Quintana-Ortí '97]
  - *H-Matrix based sign function method*, $\mathcal{O}(n \log^2(n))$ [Baur/B. '03–'05]
  - Low-rank ADI/Smith methods [Penzl '99, Li/white '00/'02, Sorensen et al '03]

Factorized Solution of Matrix Equations

Algebraic Riccati Equations

$X \geq 0 \Rightarrow X = YY^T,\ Y =$  $\in \mathbb{R}^{n \times n_X}.$

- $n_X = n$, $Y =$ **Cholesky factor**
  Newton's method combined with
  - *Hammarling's method* [Hammarling '82]
  - *sign function method* [B./Byers/Quintana-Ortí/Quintana-Ortí '98]
- $n_X < n$, $Y =$ **(approximate) full-rank factor** $(X \approx YY^T)$
  Newton's method combined with
  - *sign function method*, $\mathcal{O}(n^3)$ [B./Quintana-Ortí '97]
  - *low-rank Smith* [Banks/Ito '91, Rosen/Wang '95]
  - *low-rank ADI* [Penzl '98, B./Li/Penzl '05]

Factorized Solution of Matrix Equations

Algebraic Riccati Equations

$X \geq 0 \Rightarrow X = YY^T,\ Y =$  $\in \mathbb{R}^{n \times n_X}$.

- $n_X = n$, $Y =$ **Cholesky factor**
  Newton's method combined with
  - *Hammarling's method* [Hammarling '82]
  - *sign function method* [B./Byers/Quintana-Ortí/Quintana-Ortí '98]
- $n_X < n$, $Y =$ **(approximate) full-rank factor** ($X \approx YY^T$)
  Newton's method combined with
  - *sign function method*, $\mathcal{O}(n^3)$ [B./Quintana-Ortí '97]
  - *low-rank Smith* [Banks/Ito '91, Rosen/Wang '95]
  - *low-rank ADI* [Penzl '98, B./Li/Penzl '05]

Factorized Solution of Matrix Equations

Algebraic Riccati Equations

$X \geq 0 \Rightarrow X = YY^T,\ Y =$  $\in \mathbb{R}^{n \times n_X}$.

- $n_X = n$, $Y =$ **Cholesky factor**
  Newton's method combined with
  - *Hammarling's method* [Hammarling '82]
  - *sign function method* [B./Byers/Quintana-Ortí/Quintana-Ortí '98]
- $n_X < n$, $Y =$ **(approximate) full-rank factor** ($X \approx YY^T$)
  Newton's method combined with
  - *sign function method*, $\mathcal{O}(n^3)$ [B./Quintana-Ortí '97]
  - *low-rank Smith* [Banks/Ito '91, Rosen/Wang '95]
  - *low-rank ADI* [Penzl '98, B./Li/Penzl '05]

Factorized Solution of Matrix Equations

Sylvester Equations

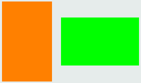
$$X = YZ = \quad , \; Y \in \mathbb{R}^{n \times n_X}, \; Z \in \mathbb{R}^{n_X \times m}.$$

Here, only $n_X < m$ leads to algorithmic advantages:

- *sign function method*, $\mathcal{O}(n^3)$ [B. MTNS'04]
- *H-Matrix based sign function method*, $\mathcal{O}(n \log^2(n))$ [Baur '05]
- *low-rank ADI* ⇝ later

Factorized Solution of Matrix Equations

Sylvester Equations

$X = YZ =$  $, \quad Y \in \mathbb{R}^{n \times n_X}, \quad Z \in \mathbb{R}^{n_X \times m}.$

Here, only $n_X < m$ leads to algorithmic advantages:

- *sign function method*, $\mathcal{O}(n^3)$ [B. MTNS'04]
- *H-Matrix based sign function method*, $\mathcal{O}(n \log^2(n))$ [Baur '05]
- *low-rank ADI* $\rightsquigarrow$ later

Cross-Gramian Based Model Reduction

[Fernando/Nicholson '83/'84, Aldhaheri '91, Sorensen/Antoulas '02]

Approximate $\begin{array}{rcl} \dot{x} & = & Ax + Bu \\ y & = & Cx \\ & \text{order } n & \end{array}$ by $\begin{array}{rcl} \dot{\hat{x}} & = & \hat{A}\hat{x} + \hat{B}u \\ \hat{y} & = & \hat{C}\hat{x} \\ & \text{order } r \ll n & \end{array}$.

Idea: instead of balanced truncation (based on controllability and observability Gramians P, Q), project onto dominant subspace of cross-Gramian, defined as solution of

$$AX + XA + BC = 0$$

Note: for SISO and symmetric systems, this is equivalent to balanced truncation as $X = PQ$; hence error bounds etc. apply.

Other applications:

- Wiener filters/image restoration [Calvetti/Reichel '96]
- Luenberger observer

Cross-Gramian Based Model Reduction

[Fernando/Nicholson '83/'84, Aldhaheri '91, Sorensen/Antoulas '02]

Approximate $\begin{array}{rcl} \dot{x} & = & Ax + Bu \\ y & = & Cx \\ & \text{order } n & \end{array}$ by $\begin{array}{rcl} \dot{\hat{x}} & = & \hat{A}\hat{x} + \hat{B}u \\ \hat{y} & = & \hat{C}\hat{x} \\ & \text{order } r \ll n & \end{array}$.

Idea: instead of balanced truncation (based on controllability and observability Gramians P, Q), project onto dominant subspace of cross-Gramian, defined as solution of

$$AX + XA + BC = 0$$

Note: for SISO and symmetric systems, this is equivalent to balanced truncation as $X = PQ$; hence error bounds etc. apply.

Other applications:

- Wiener filters/image restoration [Calvetti/Reichel '96]
- Luenberger observer

The Sign Function Method

For $Z \in \mathbb{R}^{n\times n}$, $\Lambda(Z) \cap \imath\mathbb{R} = \emptyset$ and JCF $Z = S^{-1}\begin{bmatrix} J^+ & 0 \\ 0 & J^- \end{bmatrix} S$:
($J^\pm$ = Jordan blocks corresponding to $\Lambda(Z) \cap \mathbb{C}^\pm$)

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

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

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

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

The Sign Function Method

For $Z \in \mathbb{R}^{n\times n}$, $\Lambda(Z) \cap \imath\mathbb{R} = \emptyset$ and JCF $Z = S^{-1}\begin{bmatrix} J^+ & 0 \\ 0 & J^- \end{bmatrix} S$:
($J^\pm$ = Jordan blocks corresponding to $\Lambda(Z) \cap \mathbb{C}^\pm$)

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

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

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

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

The Sign Function Method

For $Z \in \mathbb{R}^{n\times n}$, $\Lambda(Z) \cap \imath\mathbb{R} = \emptyset$ and JCF $Z = S^{-1} \begin{bmatrix} J^+ & 0 \\ 0 & J^- \end{bmatrix} S$:
($J^\pm$ = Jordan blocks corresponding to $\Lambda(Z) \cap \mathbb{C}^\pm$)

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

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

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

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

Solving Sylvester Equations with the Sign Function Method

Consider Sylvester equation

$$AX + XB + W = 0, \qquad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{m\times m} \ \text{stable.}$$

$\Longrightarrow \begin{bmatrix} I_n \\ -X_* \end{bmatrix}$ is stable invariant subspace of $Z := \begin{bmatrix} A & W \\ 0 & -B \end{bmatrix}$.

Apply sign function iteration to $Z \Longrightarrow$

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

Sign Function Iteration for Sylvester Equations

$AX + XB + FG = 0, \qquad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{m\times m},\ F \in \mathbb{R}^{n\times p},\ G \in \mathbb{R}^{p\times m}.$

Solving Sylvester Equations with the Sign Function Method

Consider Sylvester equation

$$AX + XB + W = 0, \qquad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{m\times m} \quad \text{stable.}$$

$$\Longrightarrow \begin{bmatrix} I_n \\ -X_* \end{bmatrix} \text{ is stable invariant subspace of } Z := \begin{bmatrix} A & W \\ 0 & -B \end{bmatrix}.$$

Apply sign function iteration to $Z \Longrightarrow$

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

Block-Triangular Iteration

Newton iteration (with scaling) is equivalent to

$$A_0 \leftarrow A, \quad B_0 \leftarrow B, \quad W_0 \leftarrow W$$

for $j = 0, 1, 2, \ldots$

$$A_{j+1} \leftarrow \frac{1}{2c_j}\left(A_j + c_j^2 A_j^{-1}\right),$$

$$B_{j+1} \leftarrow \frac{1}{2c_j}\left(B_j + c_j^2 B_j^{-1}\right),$$

$$W_{j+1} \leftarrow \frac{1}{2c_j}\left(W_j + c_j^2 A_j^{-1} W_j B_j^{-1}\right).$$

end for

$\Longrightarrow \quad X = \frac{1}{2}\lim_{j\to\infty} W_j$

Block-Triangular Iteration

Newton iteration (with scaling) is equivalent to

$$A_0 \leftarrow A, \quad B_0 \leftarrow B, \quad W_0 \leftarrow W$$

`for` $j = 0, 1, 2, \ldots$

$$A_{j+1} \leftarrow \frac{1}{2c_j}\left(A_j + c_j^2 A_j^{-1}\right),$$

$$B_{j+1} \leftarrow \frac{1}{2c_j}\left(B_j + c_j^2 B_j^{-1}\right),$$

$$W_{j+1} \leftarrow \frac{1}{2c_j}\left(W_j + c_j^2 A_j^{-1} W_j B_j^{-1}\right).$$

`end for`

$$\Longrightarrow \qquad X = \tfrac{1}{2}\lim_{j\to\infty} W_j$$

Sign Function Iteration for Sylvester Equations

$AX + XB + FG = 0, \qquad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{m\times m},\ F \in \mathbb{R}^{n\times p},\ G \in \mathbb{R}^{p\times m}.$

Computing Full-Rank Factors of Solution

$\Longrightarrow$ re-write (unscaled) iteration $W_{j+1} \leftarrow \frac{1}{2}\left(W_j + A_j^{-1} W_j B_j^{-1}\right)$ as

$$F_{j+1} G_{j+1} \leftarrow \frac{1}{2}\left(F_j G_j + (A_j^{-1} F_j)(G_j B_j^{-1})\right),$$

yielding

$$\begin{array}{lllll} F_0 & \leftarrow & F, & F_{j+1} & \leftarrow \frac{1}{\sqrt{2}}\left[F_j,\ A_j^{-1} F_j\right], \\ G_0 & \leftarrow & G, & G_{j+1} & \leftarrow \frac{1}{\sqrt{2}}\left[\begin{array}{c} G_j \\ G_j B_j^{-1} \end{array}\right]. \end{array}$$

Problem: workspace doubles/iteration, $F_j \in \mathbb{R}^{n\times p_j} \Rightarrow F_{j+1} \in \mathbb{R}^{n\times 2p_j}$.

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

Computing Full-Rank Factors of Solution

$\Longrightarrow$ re-write (unscaled) iteration $W_{j+1} \leftarrow \frac{1}{2}\left(W_j + A_j^{-1} W_j B_j^{-1}\right)$ as

$$F_{j+1} G_{j+1} \leftarrow \frac{1}{2}\left(F_j G_j + (A_j^{-1} F_j)(G_j B_j^{-1})\right),$$

yielding

$$\begin{array}{lllll} F_0 & \leftarrow & F, & F_{j+1} & \leftarrow \frac{1}{\sqrt{2}}\left[F_j,\ A_j^{-1} F_j\right], \\ G_0 & \leftarrow & G, & G_{j+1} & \leftarrow \frac{1}{\sqrt{2}}\left[\begin{array}{c} G_j \\ G_j B_j^{-1} \end{array}\right]. \end{array}$$

Problem: workspace doubles/iteration, $F_j \in \mathbb{R}^{n\times p_j} \;\Rightarrow\; F_{j+1} \in \mathbb{R}^{n\times 2p_j}$.

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

Computing Full-Rank Factors of Solution

$\Longrightarrow$ re-write (unscaled) iteration $W_{j+1} \leftarrow \frac{1}{2}\left(W_j + A_j^{-1} W_j B_j^{-1}\right)$ as

$$F_{j+1}G_{j+1} \leftarrow \frac{1}{2}\left(F_j G_j + (A_j^{-1}F_j)(G_j B_j^{-1})\right),$$

yielding

$$\begin{array}{llll} F_0 \leftarrow F, & F_{j+1} \leftarrow \frac{1}{\sqrt{2}}\left[F_j,\ A_j^{-1}F_j\right], \\ G_0 \leftarrow G, & G_{j+1} \leftarrow \frac{1}{\sqrt{2}}\begin{bmatrix} G_j \\ G_j B_j^{-1} \end{bmatrix}. \end{array}$$

Problem: workspace doubles/iteration, $F_j \in \mathbb{R}^{n\times p_j} \Rightarrow F_{j+1} \in \mathbb{R}^{n\times 2p_j}$.

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

Full-Rank Iteration

$$
\begin{aligned}
F_0 &\leftarrow F, & G_0 &\leftarrow G,\\
\tilde{F}_{j+1} &\leftarrow \frac{1}{\sqrt{2}}\left[\begin{array}{cc} F_j & A_j^{-1}F_j \end{array}\right], & \tilde{G}_{j+1} &\leftarrow \frac{1}{\sqrt{2}}\left[\begin{array}{c} G_j \\ G_jB_j^{-1} \end{array}\right],\\
[F_{j+1}, G_{j+1}] &\leftarrow \mathtt{compress}(\tilde{F}_{j+1}, \tilde{G}_{j+1}).
\end{aligned}
$$

Full-Rank Iteration

$$\begin{aligned} F_0 &\leftarrow F, & G_0 &\leftarrow G, \\ \tilde{F}_{j+1} &\leftarrow \tfrac{1}{\sqrt{2}}\left[\begin{array}{cc} F_j & A_j^{-1}F_j \end{array}\right], & \tilde{G}_{j+1} &\leftarrow \tfrac{1}{\sqrt{2}}\left[\begin{array}{c} G_j \\ G_j B_j^{-1} \end{array}\right], \\ [F_{j+1}, G_{j+1}] &\leftarrow \texttt{compress}(\tilde{F}_{j+1}, \tilde{G}_{j+1}). \end{aligned}$$

$[Y, Z] = \texttt{compress}(\tilde{Y}, \tilde{Z}, \tau)$

1. Compute RRQR
   $\widetilde{Z} =: \left[\begin{array}{cc} U_1 & U_2 \end{array}\right] \left[\begin{array}{c} R_1 \\ R_2 \end{array}\right] \Pi_G,\ R_1 \in \mathbb{R}^{r\times n},\ r = \operatorname{rank}\left(\widetilde{Z}, \tau\right),$
   $(\|R_2\|_2 < \tau \|R_1\|_2).$
2. Compute RRQR
   $\widetilde{Y} U_1 = \left[\begin{array}{cc} V_1 & V_2 \end{array}\right] \left[\begin{array}{c} T_1 \\ T_2 \end{array}\right] \Pi_F,\ T_1 \in \mathbb{R}^{t\times p},\ t = \operatorname{rank}\left(\widetilde{Y} U, \tau\right),$
   $(\|T_2\|_2 < \tau \|T_1\|_2).$
3. $\left[\begin{array}{cc} T_{11} & T_{12} \end{array}\right] := T_1 \Pi_F$, where $T_{11} \in \mathbb{R}^{t\times r}$.
4. Set $Y := V_1 T_{11}, \qquad Z := R_1 \Pi_G.$

Sign Function Iteration for Sylvester Equations

$AX + XB + FG = 0, \qquad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{m\times m},\ F \in \mathbb{R}^{n\times p},\ G \in \mathbb{R}^{p\times m}.$

Full-Rank Iteration

$$\begin{aligned} F_0 &\leftarrow F, & G_0 &\leftarrow G, \\ \tilde{F}_{j+1} &\leftarrow \frac{1}{\sqrt{2}} \begin{bmatrix} F_j & A_j^{-1}F_j \end{bmatrix}, & \tilde{G}_{j+1} &\leftarrow \frac{1}{\sqrt{2}} \begin{bmatrix} G_j \\ G_j B_j^{-1} \end{bmatrix}, \\ [F_{j+1}, G_{j+1}] &\leftarrow \texttt{compress}(\tilde{F}_{j+1}, \tilde{G}_{j+1}). \end{aligned}$$

What do we loose with $[Y, Z] = \texttt{compress}(\tilde{Y}, \tilde{Z}, \tau)$?

In exact arithmetic,

$$\|YZ - \widetilde{Y}\widetilde{Z}\|_2 < 2\tau\|\widetilde{Y}\|_2\|\widetilde{Z}\|_2 + \mathcal{O}(\tau^2).$$

In symmetric (Lyapunov) case, we have

$$\|YY^T - \widetilde{Y}\widetilde{Y}^T\|_2 < \tau^2\|\widetilde{Y}\|_2^2.$$

$\Rightarrow$ Need higher accuracy ($\equiv$ rank) of factors for the same approximation quality.

ADI Iteration for Sylvester Equations

$AX + XB + FG = 0, \qquad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{m\times m},\ F, G^T \in \mathbb{R}^{n\times p}.$

ADI Iteration

For $X_0 = 0$,

$$\begin{aligned} (A+\mu_j I)X_{j+\frac{1}{2}} &= -X_j(B-\mu_j I) - FG, \\ X_{j+1}(B+\nu_j I) &= -(A-\nu_j I)X_{j+\frac{1}{2}} - FG, \end{aligned} \qquad j = 0, 1, \ldots$$

with parameter sets μ_j, ν_j guaranteeing (accelerating) convergence [Starke '91, Levenberg/Reichel '93, Wachspress '00].

ADI Iteration

For $X_0 = 0$,

$$\begin{aligned} X_{j+1} &= (A - \nu_j I)(A + \mu_j I)^{-1} X_j (B - \mu_j I)(B + \nu_j I)^{-1} \\ &\quad -(\mu_j + \nu_j)(A + \mu_j I)^{-1} FG (B + \nu_j I)^{-1}, \quad j = 0, 1, \ldots \end{aligned}$$

with parameter sets μ_j, ν_j guaranteeing (accelerating) convergence [Starke '91, Levenberg/Reichel '93, Wachspress '00].

Factorized ADI Iteration

For $X_j =: Y_j Z_j$ with $Y_0 = 0 = Z_0^T$, we obtain for $j = 0, 1, \ldots,$

$$
\begin{aligned}
Y_{j+1}Z_{j+1} &= (A-\nu_j I)(A+\mu_j I)^{-1} Y_j Z_j (B-\mu_j I)(B+\nu_j I)^{-1} \\
&\quad -(\mu_j+\nu_j)(A+\mu_j I)^{-1} FG (B+\nu_j I)^{-1}, \\
&= \left[\ S_j^A F,\quad T_j^A S_j^A Y_j\ \right] \begin{bmatrix} -(\mu_j+\nu_j) G S_j^B \\ Z_j T_j^B S_j^B \end{bmatrix},
\end{aligned}
$$

where

$$
\begin{aligned}
S_j^A &:= (A+\mu_j I)^{-1}, \qquad & T_j^A &:= (A-\nu_j I), \\
S_j^B &:= (B+\nu_j I)^{-1}, \qquad & T_j^B &:= (B-\mu_j I).
\end{aligned}
$$

Factorized ADI Iteration

LR-ADI method: for $Y_0 = [\,] = Z_0^T$, iterate for $j = 0, 1, \ldots$

$$
\begin{aligned}
Y_{j+1} &:= \left[\ (A+\mu_j I)^{-1}F, \quad (A-\nu_j I)(A+\mu_j I)^{-1}Y_j\ \right], \\
Z_{j+1} &:= \left[\begin{array}{c} -(\mu_j+\nu_j)G(B+\nu_j I)^{-1} \\ Z_j(B-\mu_j I)(B+\nu_j I)^{-1} \end{array}\right].
\end{aligned}
$$

$\Rightarrow Y_j \in \mathbb{R}^{n\times jp},\ Z_j \in \mathbb{R}^{jp\times m}.$

ADI Iteration for Sylvester Equations

$AX + XB + FG = 0, \qquad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{m\times m},\ F, G^T \in \mathbb{R}^{n\times p}.$

Factorized ADI Iteration

LR-ADI with compression: for $Y_0 = [\,] = Z_0^T$, iterate for $j = 0, 1, \ldots$

$$
\begin{aligned}
\tilde{Y}_{j+1} &:= \left[\ (A+\mu_j I)^{-1}F, \quad (A-\nu_j I)(A+\mu_j I)^{-1} Y_j\ \right], \\
\tilde{Z}_{j+1} &:= \left[\begin{array}{c} -(\mu_j+\nu_j)G(B+\nu_j I)^{-1} \\ Z_j(B-\mu_j I)(B+\nu_j I)^{-1} \end{array}\right], \\
\{Y_{j+1}, Z_{j+1}\} &\leftarrow \texttt{compress}\,(\tilde{Y}_{j+1}, \tilde{Z}_{j+1}, \tau).
\end{aligned}
$$

$\Rightarrow Y_j \in \mathbb{R}^{n\times p_j},\ Z_j \in \mathbb{R}^{p_j\times n},\ p_j \le jp, \qquad X \approx Z_{j_{\max}} Y_{j_{\max}}.$

Factorized ADI Iteration

faster LR-ADI with compression $\equiv$ factorized ADI:
exploit commuting properties of S_j^A, T_j^A, S_j^B, T_j^B and renumber $\rightsquigarrow$

$$Y_1 := F_1 := (A + \mu_0 I)^{-1} F, \quad Z_1 := G_1 := (\mu_0 + \nu_0) G (B + \nu_0 I)^{-1}$$

$\texttt{for } j = 1, 2, \ldots$

$$\begin{aligned}
F_{j+1} &:= \left(I - (\mu_j + \nu_{j-1})(A + \mu_j I)^{-1}\right) F_j, \\
G_{j+1} &:= G_j \left(I - (\mu_{j-1} + \nu_j)(B + \nu_j I)^{-1}\right), \\
\tilde{Y}_{j+1} &:= \left[\ Y_j, \quad F_{j+1}\ \right], \\
\tilde{Z}_{j+1} &:= \begin{bmatrix} Z_j \\ G_{j+1} \end{bmatrix}, \\
\{Y_{j+1}, Z_{j+1}\} &\leftarrow \texttt{compress}\,(\tilde{Y}_{j+1}, \tilde{Z}_{j+1}, \tau).
\end{aligned}$$

$\Rightarrow Y_j \in \mathbb{R}^{n\times p_j},\ Z_j \in \mathbb{R}^{p_j \times n},\ p_j \le jp.$

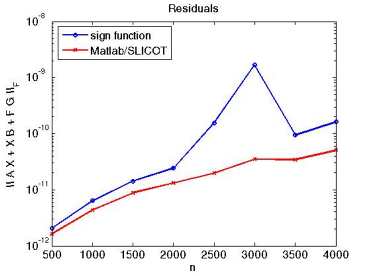
Numerical Results

Factorized Sign Function Method for Sylvester Equations

- $n = 500 : 500 : 4000$, $m = 1$, $\mathrm{sep}(A, B) = \frac{2}{n}$.
- Compare sign function based method and MATLAB's 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 $n = 4000$, $Y, Z^T \in \mathbb{R}^{4000\times 47}$, i.e., 1.4MB instead of 122MB storage required for solution.

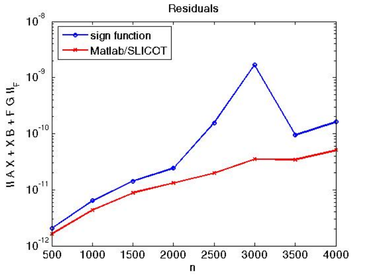
- $n = 500 : 500 : 4000$, $m = 1$, $\text{sep}(A, B) = \frac{2}{n}$.
- Compare sign function based method and MATLAB's SLICOT-based implementation of the Hessenberg-Schur method.
- Use MATLAB Release 14 on a Xeon 3GHz workstation with 8GB Ram.
- 8–14 iterations for sign function iteration.
- For $n = 4000$, $Y, Z^T \in \mathbb{R}^{4000 \times 47}$, i.e., 1.4MB instead of 122MB storage required for solution.

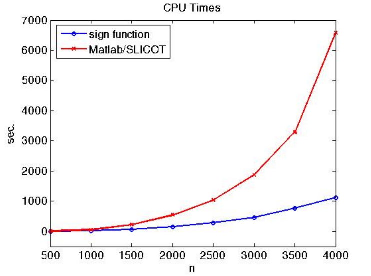
Results



- $n = 500 : 500 : 4000$, $m = 1$, $\text{sep}(A, B) = \frac{2}{n}$.
- Compare sign function based method and MATLAB's 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 $n = 4000$, $Y, Z^T \in \mathbb{R}^{4000\times 47}$, i.e., 1.4MB instead of 122MB storage required for solution.

Results





- FEM discretized 2D heat equation with boundary control.
- Accuracy, rank of factors:

n	r	$\frac{\|AX+XA^T+BB^T\|_2}{2\|A\|_2\|X\|_2+\|BB^T\|_2}$
256	11	$8 \cdot 10^{-8}$
1024	13	$4 \cdot 10^{-6}$
4096	14	$5 \cdot 10^{-6}$
16384	15	$5 \cdot 10^{-6}$

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

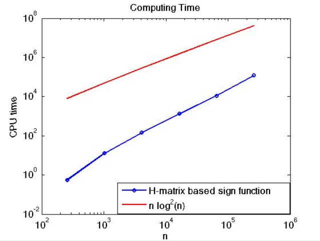
- FEM discretized 2D heat equation with boundary control.

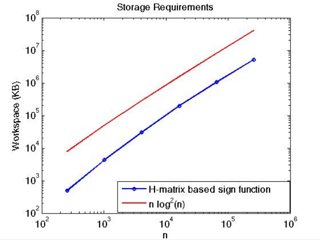
- Accuracy, rank of factors:

n	r	$\frac{\Vert AX+XA^T+BB^T\Vert_2}{2\Vert A\Vert_2\Vert X\Vert_2+\Vert BB^T\Vert_2}$
256	11	$8\cdot 10^{-8}$
1024	13	$4\cdot 10^{-6}$
4096	14	$5\cdot 10^{-6}$
16384	15	$5\cdot 10^{-6}$

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

Results





Conclusions

- Using the matrix factorization paradigm allows to solve really large-scale matrix equations.
- Other types of matrix equations can be treated analogously.
- Easiest case: matrix equations associated to discrete-time control problems (Stein equations, discrete-time Riccati equations) lead to fix-point iterations — can be re-written in factorized form immediately.
- Further work: application to large-scale Riccati differential equations.