Efficient solution of large scale Lyapunov and Riccati equations arising in model order reduction problems

Peter Benner[1] Hermann Mena [2] **Jens Saak**[1]

[1]Professur Mathematik in Industrie und Technik (MiIT)
Fakultät für Mathematik
Technische Universität Chemnitz

[2]Departamento de Matemática
Escuela Politécnica Nacional
Quito Ecuador

Department of Mathematics

CHEMNITZ UNIVERSITY OF TECHNOLOGY

Young Researchers Minisymposium (MY4)
Model reduction and its applications

GAMM 2008 Bremen

Introduction

...and now for something completely different...

[Monty Python's Flying Circus]

Introduction

What is this talk all about?

Introduction

What is this talk all about?

Advertise the mess that we created!

Introduction

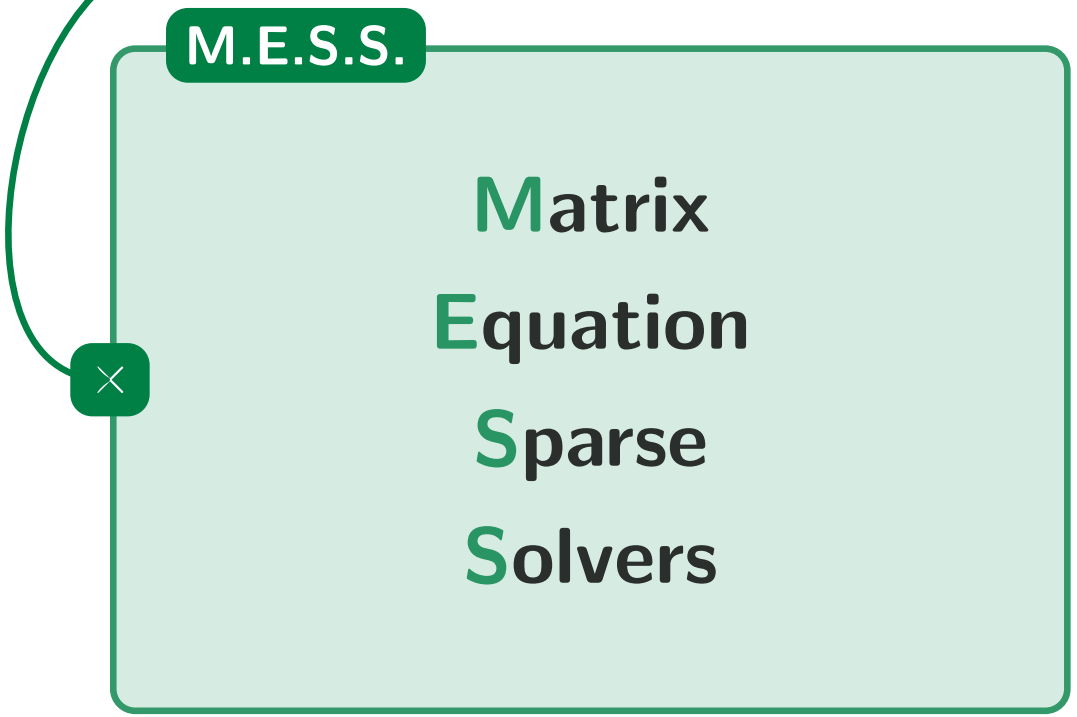
What is this talk all about?

Advertise the MESS that we created!

Introduction

What is this talk all about?

Advertise the MESS that we created!



Outline

Introduction

What is MESS?

- **Successor of the well known LyaPack**
- **MATLAB toolbox for large scale sparse (or sparse + low rank) matrix equations, i.e.,**
  - **Riccati equations**

$$0 = \mathfrak{R}(\mathbf{X}) \quad \text{or} \quad -\dot{\mathbf{X}} = \mathfrak{R}(\mathbf{X})$$

where

$$\mathfrak{R}(\mathbf{X}) = \mathbf{F} + \mathbf{A}^T\mathbf{X} + \mathbf{X}\mathbf{A} - \mathbf{X}\mathbf{G}\mathbf{X}$$

  - **Lyapunov equations**

$$\mathbf{F}^T\mathbf{X} + \mathbf{X}\mathbf{F} = \mathbf{G}^T\mathbf{G}$$

Introduction

What is MESS?

- **Successor of the well known LyaPack**
- **MATLAB toolbox for large scale sparse (or sparse + low rank) matrix equations, i.e.,**
  - **Riccati equations**

$$0 = \mathfrak{R}(\mathbf{X}) \quad \text{or} \quad -\dot{\mathbf{X}} = \mathfrak{R}(\mathbf{X})$$

where

$$\mathfrak{R}(\mathbf{X}) = \mathbf{F} + \mathbf{A}^T\mathbf{X} + \mathbf{X}\mathbf{A} - \mathbf{X}\mathbf{G}\mathbf{X}$$

  - **Lyapunov equations**

$$\mathbf{F}^T\mathbf{X} + \mathbf{X}\mathbf{F} = \mathbf{G}^T\mathbf{G}$$

Introduction

What's new?

New features

- different reordering strategies have been added
- new ADI shift parameter choices are available
- column compression for the low rank factors was introduced
- additional demos explain LQR design for systems with mass matrix and second order MOR problems
- Solvers for **Differential Riccati equations** are included
- easy to use user interface functions have been added
  `care_nk_lradi`, `lyap_lradi`

Introduction

What's new?

New features

- different reordering strategies have been added
- new ADI shift parameter choices are available
- column compression for the low rank factors was introduced
- additional demos explain LQR design for systems with mass matrix and second order MOR problems
- Solvers for **Differential Riccati equations** are included
- easy to use user interface functions have been added
  `care_nk_lradi`, `lyap_lradi`

Introduction

What's new?

New features `care_nk_lradi`

- different reordering strategies have been added
- new ADI shift parameter choices are available
- column compression for the low rank factors was introduced
- additional demos explain LQR design for systems with mass matrix and second order MOR problems
- Solvers for **Differential Riccati equations** are included
- easy to use user interface functions have been added

$$0 = \mathbf{C}^T\mathbf{Q}\mathbf{C} + \mathbf{A}^T\mathbf{X} + \mathbf{X}\mathbf{A} - \mathbf{X}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{X}, \quad \mathbf{X} = \mathbf{Z}\mathbf{Z}^T$$

```
Z=care_nk_lradi(A,B,C,Q,R)
```

`care_nk_lradi`, `lyap_lradi`

Introduction

What's new?

New features

- different reordering strategies have been added
- new ADI shift parameter choices are available
- column compression for the low rank factors was introduced
- additional demos explain LQR design for systems with mass matrix and second order MOR problems
- Solvers for **Differential Riccati equations** are included
- easy to use user interface functions have been added

  `care_nk_lradi`, `lyap_lradi`

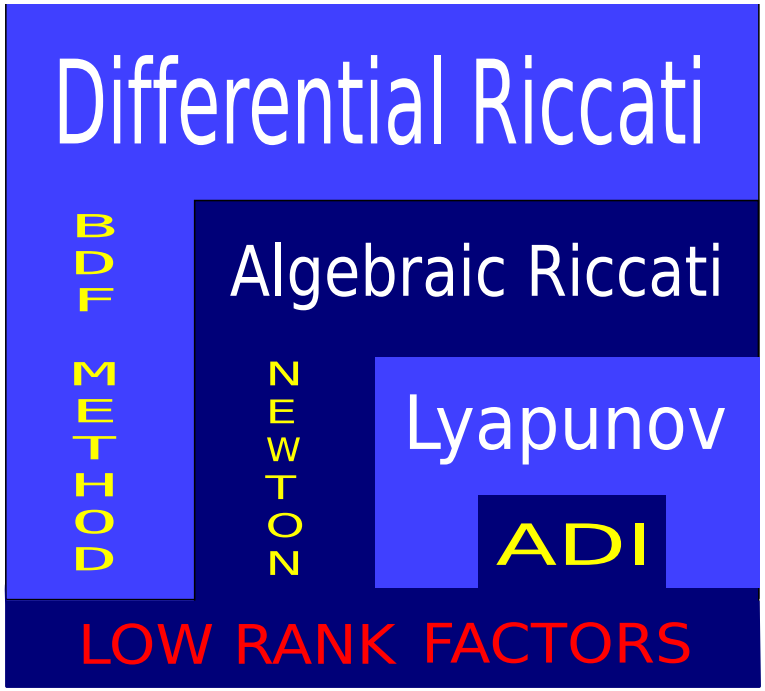
`lyap_lradi`

$$\mathbf{AX} + \mathbf{XA}^T + \mathbf{BB}^T = 0, \quad \mathbf{X} = \mathbf{ZZ}^T$$

```
Z = lyap_lradi(A,B)
```

Algorithms
Solving the DRE

[Benner/Mena 2004], [Benner/Mena 2007],[Mena 2007] showed that ODE solvers of BDF and Rosenbrock type can efficiently be applied to matrix valued problems.



Algorithms

Structure of the equations of interest

We are interested in solving

algebraic Riccati equations

$$0 = A^T X + XA - XBB^T X + C^T C =: \mathfrak{R}(X), \qquad \text{(ARE)}$$

where

- $A \in \mathbb{R}^{n\times n}$ sparse, $n \in \mathbb{N}$ "large"
- $B \in \mathbb{R}^{n\times m}$ and $m \in \mathbb{N}$ with $m \ll n$
- $C \in \mathbb{R}^{p\times n}$ and $p \in \mathbb{N}$ with $p \ll n$

and

Lyapunov equations

$$F^T X + XF = -GG^T, \qquad \text{(LE)}$$

with

- $F \in \mathbb{R}^{n\times n}$ sparse or sparse + low rank update, $n \in \mathbb{N}$ "large"
- $G \in \mathbb{R}^{n\times m}$ and $m \in \mathbb{N}$ with $m \ll n$

Algorithms
LRCF-NM for AREs

Newton's iteration for the ARE

$$\mathfrak{R}'|_X(N_\ell) = -\mathfrak{R}(X_\ell), \qquad X_{\ell+1} = X_\ell + N_\ell,$$

where the Frechét derivative of $\mathfrak{R}$ at X is the Lyapunov operator

$$\mathfrak{R}'|_X : \quad Q \mapsto (A - BB^T X)^T Q + Q(A - BB^T X),$$

can be rewritten as the

one iteration step

$$(A - BB^T X_\ell)^T X_{\ell+1} + X_{\ell+1}(A - BB^T X_\ell) = -C^T C - X_\ell BB^T X_\ell,$$

i.e., in every Newton step we have to solve a Lyapunov equation of the form (LE).

Algorithms

ADI applied to the Lyapunov equations

Recall **Peaceman Rachford ADI**[2]**:**
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 commuting $H, V \in \mathbb{R}^{n\times n}$, such that

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

can be solved easily/efficiently.

ADI Iteration

If H, V spd $\Rightarrow \exists p_j$, $j = 1, 2, \ldots J$, such that

$$\begin{aligned} u_0 &= 0 \\ (H + p_j I)u_{j-\frac{1}{2}} &= (p_j I - V)u_{j-1} + s \\ (V + p_j I)u_j &= (p_j I - H)u_{j-\frac{1}{2}} + s \end{aligned} \qquad \text{(PR-ADI)}$$

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

[2] [Peaceman & Rachford 1954], see also [Wachspress 1966]

Algorithms

ADI applied to the Lyapunov equations

The Lyapunov operator

$$\mathcal{L}: \quad X \quad \mapsto \quad F^T X + XF$$

can be decomposed into the linear operators

$$\mathcal{L}_H : X \mapsto F^T X, \qquad \mathcal{L}_V : X \mapsto XF,$$

such that in analogy to (PR-ADI) we find the

ADI iteration for the Lyapunov equation (LE)

$$\begin{aligned} X_0 &= 0, \\ (F^T + p_j I) X_{j-\frac{1}{2}} &= -GG^T - X_{j-1}(F - p_j I), \\ (F^T + p_j I) X_j^T &= -GG^T - X_{j-\frac{1}{2}}^T (F - p_j I). \end{aligned} \qquad \text{(LE-ADI)}$$

Algorithms

Remarks on efficiency and implementation

Remarks:

- If F is sparse or sparse + low rank update, i.e., $F = A + VU^T$, then $F^T + p_j I$ can be written as $\tilde{A} + UV^T$, where $\tilde{A} = A^T + p_j I$ and its inverse can be expressed as

$$(F^T + p_j I)^{-1} = (\tilde{A} + UV^T)^{-1} = \tilde{A}^{-1} - \tilde{A}^{-1}U(I + V^T\tilde{A}^{-1}U)^{-1}V^T\tilde{A}^{-1}$$

  by the Sherman-Morrison-Woodbury formula.

- (LE-ADI) can be rewritten to iterate on the low rank Cholesky factors Z_j of X_j to exploit $\operatorname{rk}(X_j) \ll n$. [Li & White 2002; Penzl 1999; Benner, Li, Penzl 2000]
- While solving (ARE) to compute the feedback in an LQR-problem for a semidiscretized PDE, the LRCF-NM can directly iterate on the feedback matrix $K \in \mathbb{R}^{n \times p}$ to save even more memory. [Penzl 1999; Benner, Li, Penzl 2000]

Algorithms

Remarks on efficiency and implementation

Remarks:

- If F is sparse or sparse + low rank update, i.e., $F = A + VU^T$, then $F^T + p_j I$ can be written as $\tilde{A} + UV^T$, where $\tilde{A} = A^T + p_j I$ and its inverse can be expressed as

$$(F^T + p_j I)^{-1} = (\tilde{A} + UV^T)^{-1} = \tilde{A}^{-1} - \tilde{A}^{-1}U(I + V^T\tilde{A}^{-1}U)^{-1}V^T\tilde{A}^{-1}$$

  by the Sherman-Morrison-Woodbury formula.

- (LE-ADI) can be rewritten to iterate on the low rank Cholesky factors Z_j of X_j to exploit $\operatorname{rk}(X_j) \ll n$. [Li & White 2002; Penzl 1999; Benner, Li, Penzl 2000]
- While solving (ARE) to compute the feedback in an LQR-problem for a semidiscretized PDE, the LRCF-NM can directly iterate on the feedback matrix $K \in \mathbb{R}^{n \times p}$ to save even more memory. [Penzl 1999; Benner, Li, Penzl 2000]

Algorithms

Remarks on efficiency and implementation

Remarks:

- If F is sparse or sparse + low rank update, i.e., $F = A + VU^T$, then $F^T + p_j I$ can be written as $\tilde{A} + UV^T$, where $\tilde{A} = A^T + p_j I$ and its inverse can be expressed as

$$(F^T + p_j I)^{-1} = (\tilde{A} + UV^T)^{-1} = \tilde{A}^{-1} - \tilde{A}^{-1}U(I + V^T\tilde{A}^{-1}U)^{-1}V^T\tilde{A}^{-1}$$

  by the Sherman-Morrison-Woodbury formula.

- (LE-ADI) can be rewritten to iterate on the low rank Cholesky factors Z_j of X_j to exploit $\operatorname{rk}(X_j) \ll n$. [Li & White 2002; Penzl 1999; Benner, Li, Penzl 2000]

- While solving (ARE) to compute the feedback in an LQR-problem for a semidiscretized PDE, the LRCF-NM can directly iterate on the feedback matrix $K \in \mathbb{R}^{n \times p}$ to save even more memory. [Penzl 1999; Benner, Li, Penzl 2000]

Algorithms

Remarks on efficiency and implementation

Further remarks:

- reordering strategies (e.g., RCM, AMD) for the indices can reduce memory requirements by far where matrix decompositions are involved [Benner/Mena/S. 2008]
- new shift parameter selection allows easy improvements in ADI performance [Benner/Mena/S. 2006]
- column compression via RRQR drastically reduces storage requirements. Especially helpful in differential Riccati equation solvers where 1 ARE (BDF), or 1 Lyapunov (Rosenbrock) solution needs to be stored in every step. [Benner/Mena/S. 2008]

Algorithms

Remarks on efficiency and implementation

Further remarks:

- reordering strategies (e.g., RCM, AMD) for the indices can reduce memory requirements by far where matrix decompositions are involved [Benner/Mena/S. 2008]
- new shift parameter selection allows easy improvements in ADI performance [Benner/Mena/S. 2006]
- column compression via RRQR drastically reduces storage requirements. Especially helpful in differential Riccati equation solvers where 1 ARE (BDF), or 1 Lyapunov (Rosenbrock) solution needs to be stored in every step. [Benner/Mena/S. 2008]

Algorithms

Remarks on efficiency and implementation

Further remarks:

- reordering strategies (e.g., RCM, AMD) for the indices can reduce memory requirements by far where matrix decompositions are involved [Benner/Mena/S. 2008]
- new shift parameter selection allows easy improvements in ADI performance [Benner/Mena/S. 2006]
- column compression via RRQR drastically reduces storage requirements. Especially helpful in differential Riccati equation solvers where 1 ARE (BDF), or 1 Lyapunov (Rosenbrock) solution needs to be stored in every step. [Benner/Mena/S. 2008]

Algorithms

Remarks on efficiency and implementation

Further remarks:

- reordering strategies (e.g., RCM, AMD) for the indices can reduce memory requirements by far where matrix decompositions are involved [Benner/Mena/S. 2008]
- new shift parameter selection allows easy improvements in ADI performance [Benner/Mena/S. 2006]
- column compression via RRQR drastically reduces storage requirements. Especially helpful in differential Riccati equation solvers where 1 ARE (BDF), or 1 Lyapunov (Rosenbrock) solution needs to be stored in every step. [Benner/Mena/S. 2008]

Outlook

Theoretical outlook

- Improve stopping criteria for the ADI process,
  e.g., inside the LRCF-Newton method by interpretation as inexact Newton method following the ideas of Sachs et al.
- Optimize truncation tolerances for the RRQR
  Investigate dependency of residual errors in X on the truncation tolerance
- Stabilizing initial feedback computation
  Investigate whether the method in [Gallivan, Rao, Van Dooren 2006] can be implemented exploiting sparse matrix arithmetics.

Outlook

Implementational issues

- Include saddle-point-formulation for projected systems
  Based on [Heinkenschloss,Sorenson,Sun 2007] treat projected LQOCP for Oseen and Navier-Stokes equation.
- Optimize treatment of generalized systems
  Following [Benner 2004] rewrite kernel routines to work with the matrix pencil $(A - \lambda M)$ instead of decomposing the matrix M
- Improve second order MOR problem handling
  Avoid assembly of equivalent first order system of order $2n$

Outlook

Implementational issues

- Include saddle-point-formulation for projected systems
  Based on [Heinkenschloss,Sorensen,Sun 2007] treat projected LQOCP for Oseen and Navier-Stokes equation.
- Optimize treatment of generalized systems
  Following [Benner 2004] rewrite kernel routines to work with the matrix pencil $(A - \lambda M)$ instead of decomposing the matrix M
- Improve second order MOR problem handling
  Avoid assembly of equivalent first order system of order $2n$

Thank you for your attention

Watch

`http://www.tu-chemnitz.de/~saak/Software/mess.php`

for updates !