MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

CSC
COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

Solving Differential Matrix Equations using Parareal

Martin Köhler

joint work with

Jens Saak and Norman Lang

GAMM Annual Meeting
March 7th – 11th 2016

Partners:

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Differential Riccati Equations

Consider the linear quadratic optimal control problem:

$$\min_u \mathcal{J}(y,u) = \frac{1}{2}\left(\int_0^{t_f} y^T y + u^T u \,\mathrm{d}t \; + y_{t_f}^T Q y_{t_f}\right),$$

$$\text{subject to} \quad E\dot{x}(t) = Ax(t) + Bu(t),$$
$$y(t) = Cx(t)$$

where A, E, B, and C may depend on t as well and the states $x(t) \in \mathbb{R}^n$, inputs $u(t) \in \mathbb{R}^m$, and outputs $y(t) \in \mathbb{R}^q$.

Differential Riccati Equations

Consider the linear quadratic optimal control problem:

$$\min_{u} \mathcal{J}(y, u) = \frac{1}{2}\left(\int_0^{t_f} y^T y + u^T u \,\mathrm{d}t \; + y_{t_f}^T Q y_{t_f}\right),$$

$$\text{subject to} \quad E\dot{x}(t) = Ax(t) + Bu(t),$$
$$y(t) = Cx(t)$$

where A, E, B, and C may depend on t as well.

Feedback law **e.g.** [Locatelli '01]

$$u(t) = -B^T X(t) E x(t),$$

Differential Riccati Equations

Consider the linear quadratic optimal control problem:

$$\min_u \mathcal{J}(y,u) = \frac{1}{2}\left(\int_0^{t_f} y^T y + u^T u \,\mathrm{d}t \; + y_{t_f}^T Q y_{t_f}\right),$$

$$\text{subject to} \quad E\dot{x}(t) = Ax(t) + Bu(t),$$
$$y(t) = Cx(t)$$

where A, E, B, and C may depend on t as well.

Feedback law **e.g.** [Locatelli '01]

$$u(t) = -B^T X(t) E x(t),$$

where $X(t)$ is the solution of the Differential Riccati Equation (DRE)

$$E^T \dot{X}(t) E = C^T C + A^T X(t) E + E^T X(t) A - E^T X(t) B B^T X(t) E := \mathcal{R}(X(t)),$$
$$X(t = t_f) := Q.$$

Differential Lyapunov Equation

Simplification of the DRE

By setting $B = 0$ in the DRE we get the Differential Lyapunov Equation (DLE):

$$E^T \dot{X}(t) E = C^T C + A^T X(t) E + E^T X(t) A,$$
$$X(t = t_f) := X_f.$$

Differential Lyapunov Equation

Simplification of the DRE

By setting $B = 0$ in the DRE we get the Differential Lyapunov Equation (DLE):

$$E^T \dot{X}(t) E = C^T C + A^T X(t) E + E^T X(t) A,$$
$$X(t = t_f) := X_f.$$

Application in Model Order Reduction:

$\rightarrow$ used for *Linear Time-Variant (LTV)* Balanced Truncation.

Time integration methods

- The **DLE** is a matrix-valued ordinary differential equation.
- The **DRE** is a non-linear, matrix-valued, and highly stiff ordinary differential equation.

Time integration methods

- The **DLE** is a matrix-valued ordinary differential equation.
- The **DRE** is a non-linear, matrix-valued, and highly stiff ordinary differential equation.

Implicit time integrators [Mena '07, Benner/Mena '12]

- Backward differentiation formula (BDF)
- Linear implicit Runge-Kutta (Rosenbrock) methods
- Midpoint and Trapezoidal rule

Time integration methods

- The **DLE** is a matrix-valued ordinary differential equation.
- The **DRE** is a non-linear, matrix-valued, and highly stiff ordinary differential equation.

Implicit time integrators [Mena '07, Benner/Mena '12]

- Backward differentiation formula (BDF)
- Linear implicit Runge-Kutta (Rosenbrock) methods
- Midpoint and Trapezoidal rule

Time integration methods

- The **DLE** is a matrix-valued ordinary differential equation.
- The **DRE** is a non-linear, matrix-valued, and highly stiff ordinary differential equation.

Implicit time integrators [Mena '07, Benner/Mena '12]

- Backward differentiation formula (BDF)
- Linear implicit Runge-Kutta (Rosenbrock) methods
- Midpoint and Trapezoidal rule

Numerical issues

- Methods are fairly time and storage consuming for large-scale problems.
- High accuracy requires small time steps or high order methods.
- At every time step a number of algebraic matrix equations needs to be solved.

Rosenbrock Methods

Restriction by Parareal:

Only single-step integrators are well suited.

Rosenbrock Methods

Restriction by Parareal:

Only single-step integrators are well suited.

General Rosenbrock Scheme

The s-stage Rosenbrock method applied to a matrix differential equation of the form $\dot{X} = F(X)$ is given as

$$X_{k+1} = X_k + \tau_k \sum_{\ell=1}^{s} b_\ell K_\ell^{(k)},$$

$$K_i^{(k)} = F(X_k + \tau_k \sum_{\ell=1}^{i-1} \alpha_{i,\ell} K_\ell^{(k)}) + \tau_k \mathcal{J}_k \sum_{\ell=1}^{i} \gamma_{i,\ell} K_\ell^{(k)}, \quad \forall i = 1, \ldots, s.$$

- s : order of the method
- τ_k : time step
- $\mathcal{J}_k$: Fréchet derivative of F at X_k
- $\alpha_{i,\ell}, \gamma_{i,\ell}, \mu_\ell$: determining coefficients

Rosenbrock Methods

Restriction by Parareal:

Only single-step integrators are well suited.

General Rosenbrock Scheme

The s-stage Rosenbrock method applied to a matrix differential equation of the form $\dot{X} = F(X)$ is given as

$$X_{k+1} = X_k + \tau_k \sum_{\ell=1}^{s} b_\ell K_\ell^{(k)},$$

$$K_i^{(k)} = F(X_k + \tau_k \sum_{\ell=1}^{i-1} \alpha_{i,\ell} K_\ell^{(k)}) + \tau_k \mathcal{J}_k \sum_{\ell=1}^{i} \gamma_{i,\ell} K_\ell^{(k)}, \quad \forall i = 1, \ldots, s.$$

- s : order of the method
- τ_k : time step
- $\mathcal{J}_k$: Fréchet derivative of F at X_k
- $\alpha_{i,\ell}, \gamma_{i,\ell}, \mu_\ell$: determining coefficients

Fréchet derivative of $\mathcal{R}(X)$ is a Lyapunov operator.

Rosenbrock Methods

[Mena '07, Benner/Mena '12]

We only consider the DRE, where $E^T \dot{X} E = \mathcal{R}(X)$:

1st order Rosenbrock scheme (Ros1)

$$X_{k+1} = X_k + \tau_k K_1^{(k)}$$

Rosenbrock Methods

[Mena '07, Benner/Mena '12]

We only consider the DRE, where $E^T \dot{X} E = \mathcal{R}(X)$:

1st order Rosenbrock scheme (Ros1)

$$X_{k+1} = X_k + \tau_k K_1^{(k)}$$

$$E^T K_1^{(k)} E - \tau_k \mathcal{R}'|_{X_k}(K_1^{(k)}) = \mathcal{R}(X)$$

Rosenbrock Methods

[Mena '07, Benner/Mena '12]

We only consider the DRE, where $E^T \dot{X} E = \mathcal{R}(X)$:

1st order Rosenbrock scheme (Ros1)

$$X_{k+1} = X_k + \tau_k K_1^{(k)}$$

$$E^T K_1^{(k)} E - \tau_k (A - BB^T X_k E)^T K_1^{(k)} E - \tau_k E^T K_1^{(k)} (A - BB^T X_k E) = \mathcal{R}(X)$$

Rosenbrock Methods

[Mena '07, Benner/Mena '12]

We only consider the DRE, where $E^T \dot{X} E = \mathcal{R}(X)$:

1st order Rosenbrock scheme (Ros1)

$$X_{k+1} = X_k + \tau_k K_1^{(k)}$$

$$(\tau_k(A - BB^T X_k E) - \frac{1}{2}E)^T K_1^{(k)} E + E^T K_1^{(k)} (\tau_k(A - BB^T X_k E) - \frac{1}{2}E) = -\mathcal{R}(X)$$

Rosenbrock Methods [Mena '07, Benner/Mena '12]

We only consider the DRE, where $E^T \dot{X} E = \mathcal{R}(X)$:

1st order Rosenbrock scheme (Ros1)

$$X_{k+1} = X_k + \tau_k K_1^{(k)}$$

$$\tilde{A}^T K_1^{(k)} E + E^T K_1^{(k)} \tilde{A} = -\mathcal{R}(X_k)$$

$$\tilde{A} := \tau_k (A - BB^T X_k E) - \frac{1}{2} E$$

Solve one Algebraic Lyapunov Equation (ALE) inside the 1-stage Rosenbrock method.

Rosenbrock Methods [Mena '07, Benner/Mena '12]

We only consider the DRE, where $E^T \dot{X} E = \mathcal{R}(X)$:

1st order Rosenbrock scheme (Ros1)

$$X_{k+1} = X_k + \tau_k K_1^{(k)}$$

$$\tilde{A}^T K_1^{(k)} E + E^T K_1^{(k)} \tilde{A} = -\mathcal{R}(X_k)$$

2nd order Rosenbrock scheme (Ros2) [Dekker/Verwer '84]

$$X_{k+1} = X_k + \frac{3}{2}\tau_k K_1^{(k)} + \frac{1}{2}\tau_k K_2^{(k)}$$

$$\tilde{A}^T K_1^{(k)} E + E^T K_1^{(k)} \tilde{A} = -\mathcal{R}(X_k)$$

$$\tilde{A}^T K_2^{(k)} E + E^T K_2^{(k)} \tilde{A} = -\mathcal{R}(X_k + \tau_k K_1^{(k)}) + 2E^T K_1^{(k)} E$$

$$\tilde{A} := \gamma \tau_k (A - BB^T X_k E) - \frac{1}{2} E$$

Rosenbrock Methods [Mena '07, Benner/Mena '12]

We only consider the DRE, where $E^T \dot{X} E = \mathcal{R}(X)$:

1st order Rosenbrock scheme (Ros1)

$$X_{k+1} = X_k + \tau_k K_1^{(k)}$$

$$\tilde{A}^T K_1^{(k)} E + E^T K_1^{(k)} \tilde{A} = -\mathcal{R}(X_k)$$

2nd order Rosenbrock scheme (Ros2) [Dekker/Verwer '84]

$$X_{k+1} = X_k + \frac{3}{2}\tau_k K_1^{(k)} + \frac{1}{2}\tau_k K_2^{(k)}$$

$$\tilde{A}^T K_1^{(k)} E + E^T K_1^{(k)} \tilde{A} = -\mathcal{R}(X_k)$$

$$\tilde{A}^T K_2^{(k)} E + E^T K_2^{(k)} \tilde{A} = -\mathcal{R}(X_k + \tau_k K_1^{(k)}) + 2E^T K_1^{(k)} E$$

Solve two ALEs inside the 2-stage Rosenbrock method.

Rosenbrock Methods [Mena '07, Benner/Mena '12]

We only consider the DRE, where $E^T \dot{X} E = \mathcal{R}(X)$:

1st order Rosenbrock scheme (Ros1)

$$X_{k+1} = X_k + \tau_k K_1^{(k)}$$

$$\tilde{A}^T K_1^{(k)} E + E^T K_1^{(k)} \tilde{A} = -\mathcal{R}(X_k)$$

2nd order Rosenbrock scheme (Ros2) [Dekker/Verwer '84]

$$X_{k+1} = X_k + \frac{3}{2}\tau_k K_1^{(k)} + \frac{1}{2}\tau_k K_2^{(k)}$$

$$\tilde{A}^T K_1^{(k)} E + E^T K_1^{(k)} \tilde{A} = -\mathcal{R}(X_k)$$

$$\tilde{A}^T K_2^{(k)} E + E^T K_2^{(k)} \tilde{A} = -\mathcal{R}(X_k + \tau_k K_1^{(k)}) + 2E^T K_1^{(k)} E$$

Higher Order Schemes:

- 3rd order scheme based on `ode23s` (numerically unstable)
- 3rd order scheme based on `Ros3P` [Lang, Verwer '01]
- 4th order scheme [Shampine '82]

Optimize before Parallelize

High Computational Cost

- Need the solution of s algebraic Lyapunov equations per time step.
  $\rightarrow$ Bartels-Stewart algorithm requires a QZ decomposition.
- Mostly matrix-matrix products.

Optimize before Parallelize

High Computational Cost

- Need the solution of s algebraic Lyapunov equations per time step.
  $\rightarrow$ Bartels-Stewart algorithm requires a QZ decomposition.
- Mostly matrix-matrix products.

Redundant Information

- Each right hand side of the Lyapunov equation includes $\mathcal{R}(X_k)$.
- Redundant information in the linear part of $\mathcal{R}(X_k + \tau_k K_j + \ldots)$.
- Solutions of the Lyapunov equations K_j are symmetric.

Optimize before Parallelize

High Computational Cost

- Need the solution of s algebraic Lyapunov equations per time step.
  $\rightarrow$ Bartels-Stewart algorithm requires a QZ decomposition.
- Mostly matrix-matrix products.

Redundant Information

- Each right hand side of the Lyapunov equation includes $\mathcal{R}(X_k)$.
- Redundant information in the linear part of $\mathcal{R}(X_k + \tau_k K_j + \ldots)$.
- Solutions of the Lyapunov equations K_j are symmetric.

Strategies

- Computational Cost: Use level-3 BLAS enabled algorithms.
- Redundant Information: Reformulation of the right hand sides.

Level-3 BLAS enabled Algorithms

Matrix-Matrix Products

Use Intel® MKL, AMD ACML, BLIS, OpenBLAS or ATLAS.

Level-3 BLAS enabled Algorithms

Matrix-Matrix Products

Use Intel® MKL, AMD ACML, BLIS, OpenBLAS or ATLAS.

Lyapunov Equations

Generalized Bartels-Stewart algorithm available in SLICOT: [Penzl '97]

- All stages have the same coefficient matrices.
- Only one QZ decomposition per time step and reuse it.

Level-3 BLAS enabled Algorithms

Matrix-Matrix Products

Use Intel® MKL, AMD ACML, BLIS, OpenBLAS or ATLAS.

Lyapunov Equations

Generalized Bartels-Stewart algorithm available in SLICOT: [Penzl '97]

- All stages have the same coefficient matrices.
- Only one QZ decomposition per time step and reuse it.
- **But QZ is mostly a level-2 BLAS algorithm.**

Level-3 BLAS enabled Algorithms

Matrix-Matrix Products

Use Intel® MKL, AMD ACML, BLIS, OpenBLAS or ATLAS.

Lyapunov Equations

Generalized Bartels-Stewart algorithm available in SLICOT: [Penzl '97]

- All stages have the same coefficient matrices.
- Only one QZ decomposition per time step and reuse it.
- **But QZ is mostly a level-2 BLAS algorithm.**
- **Bartels-Stewart algorithm is level-2 BLAS as well.**

Level-3 BLAS enabled Algorithms

Matrix-Matrix Products

Use Intel® MKL, AMD ACML, BLIS, OpenBLAS or ATLAS.

Lyapunov Equations

Generalized Bartels-Stewart algorithm available in SLICOT: [Penzl '97]

- All stages have the same coefficient matrices.
- Only one QZ decomposition per time step and reuse it.
- **But QZ is mostly a level-2 BLAS algorithm.**
- **Bartels-Stewart algorithm is level-2 BLAS as well.**

Need for an efficient Lyapunov solver

Level-3 BLAS enabled Algorithms

Matrix-Matrix Products

Use Intel® MKL, AMD ACML, BLIS, OpenBLAS or ATLAS.

Lyapunov Equations

Generalized Bartels-Stewart algorithm available in SLICOT: [Penzl '97]

- All stages have the same coefficient matrices.
- Only one QZ decomposition per time step and reuse it.
- **But QZ is mostly a level-2 BLAS algorithm.**
- **Bartels-Stewart algorithm is level-2 BLAS as well.**

Need for an efficient Lyapunov solver

- Matrix Sign Function Iteration [Quintana-Ortí/ Benner '99]
  $\rightarrow$ Without QZ decomposition, but no advantage out of operator repetition.

Level-3 BLAS enabled Algorithms

Matrix-Matrix Products

Use Intel® MKL, AMD ACML, BLIS, OpenBLAS or ATLAS.

Lyapunov Equations

Generalized Bartels-Stewart algorithm available in SLICOT: [Penzl '97]

- All stages have the same coefficient matrices.
- Only one QZ decomposition per time step and reuse it.
- **But QZ is mostly a level-2 BLAS algorithm.**
- **Bartels-Stewart algorithm is level-2 BLAS as well.**

Need for an efficient Lyapunov solver

- Matrix Sign Function Iteration [Quintana-Ortí/ Benner '99]
  $\rightarrow$ Without QZ decomposition, but no advantage out of operator repetition.
- Reuse of the QZ decomposition and level-3 BLAS block generalized Bartels-Stewart algorithm. [GLYAP3: K./Saak '14 '15]

Right-Hand-Side Rearrangement

Consider the stages of the 2nd order Rosenbrock scheme:

$$\tilde{A}_k^T K_1 E + E^T K_1 \tilde{A}_k = -\mathcal{R}(X_k)$$

and

$$\tilde{A}_k^T K_2 E + E^T K_2 \tilde{A}_k = -\mathcal{R}(X_k + \tau_k K_1) + 2E^T K_1 E$$

with $\tilde{A} := \gamma\tau_k(A - BB^T X_k E) - \frac{1}{2}E$.

Right-Hand-Side Rearrangement

Consider the stages of the 2nd order Rosenbrock scheme:

$$\tilde{A}_k^T K_1 E + E^T K_1 \tilde{A}_k = -\mathcal{R}(X_k)$$

and

$$\begin{aligned}\tilde{A}_k^T K_2 E + E^T K_2 \tilde{A}_k = & -\mathcal{R}(X_k + \tau_k K_1) + 2E^T K_1 E \\ = & -\mathcal{R}(X_k) - \tau_k \left((A^T - BB^T X_k E)^T K_1 E + E^T K_1 (A^T - BB^T X_k E) \right) \\ & + \tau_k^2 E^T K_1 BB^T K_1 E + 2E^T K_1 E\end{aligned}$$

Right-Hand-Side Rearrangement

Consider the stages of the 2nd order Rosenbrock scheme:

$$\tilde{A}_k^T K_1 E + E^T K_1 \tilde{A}_k = -\mathcal{R}(X_k)$$

and

$$\begin{aligned}
\tilde{A}_k^T K_2 E + E^T K_2 \tilde{A}_k &= -\mathcal{R}(X_k + \tau_k K_1) + 2E^T K_1 E \\
&= -\mathcal{R}(X_k) - \tau_k \left((A^T - BB^T X_k E)^T K_1 E + E^T K_1 (A^T - BB^T X_k E) \right) \\
&\quad + \tau_k^2 E^T K_1 BB^T K_1 E + 2E^T K_1 E \\
&= -\mathcal{R}(X_k) - \frac{1}{\gamma} \left(\tilde{A}_k^T K_1 E + E^T K_1 \tilde{A}_k \right) - \frac{1}{\gamma} E^T K_1 E \\
&\quad + \tau_k^2 E^T K_1 BB^T K_1 E + 2E^T K_1 E
\end{aligned}$$

Right-Hand-Side Rearrangement

Consider the stages of the 2nd order Rosenbrock scheme:

$$\tilde{A}_k^T K_1 E + E^T K_1 \tilde{A}_k = -\mathcal{R}(X_k)$$

and

$$\begin{aligned}
\tilde{A}_k^T K_2 E + E^T K_2 \tilde{A}_k &= -\mathcal{R}(X_k + \tau_k K_1) + 2E^T K_1 E \\
&= -\mathcal{R}(X_k) - \tau_k \left((A^T - BB^T X_k E)^T K_1 E + E^T K_1 (A^T - BB^T X_k E) \right) \\
&\quad + \tau_k^2 E^T K_1 BB^T K_1 E + 2E^T K_1 E \\
&= -\mathcal{R}(X_k) - \frac{1}{\gamma} \left(\tilde{A}_k^T K_1 E + E^T K_1 \tilde{A}_k \right) - \frac{1}{\gamma} E^T K_1 E \\
&\quad + \tau_k^2 E^T K_1 BB^T K_1 E + 2E^T K_1 E \\
&= -(1 - \frac{1}{\gamma})\mathcal{R}(X_k) + \tau_k^2 E^T K_1 BB^T K_1 E + (2 - \frac{1}{\gamma}) E^T K_1 E.
\end{aligned}$$

Right-Hand-Side Rearrangement

Using the linearity of the Lyapunov-Equation we reformulate the 2nd order Rosenbrock scheme as:

$$
\begin{aligned}
X_{k+1} &= X_k + \frac{3}{2}\tau_k K_1 + \frac{1}{2}\tau_k K_2, \\
\tilde{A}_k^T K_1 E + E^T K_1 \tilde{A}_k &= -\mathcal{R}(X_k), \\
\tilde{A}_k^T \tilde{K}_2 E + E^T \tilde{K}_2 \tilde{A}_k &= \tau_k^2 E^T K_1 B B^T K_1 E + (2 - \frac{1}{\gamma}) E^T K_1 E, \\
K_2 &= \tilde{K}_2 + (1 - \frac{1}{\gamma}) K_1.
\end{aligned}
$$

Right-Hand-Side Rearrangement

Using the linearity of the Lyapunov-Equation we reformulate the 2nd order Rosenbrock scheme as:

$$
\begin{aligned}
X_{k+1} &= X_k + \frac{3}{2}\tau_k K_1 + \frac{1}{2}\tau_k K_2, \\
\tilde{A}_k^T K_1 E + E^T K_1 \tilde{A}_k &= -\mathcal{R}(X_k), \\
\tilde{A}_k^T \tilde{K}_2 E + E^T \tilde{K}_2 \tilde{A}_k &= \tau_k^2 E^T K_1 B B^T K_1 E + (2 - \frac{1}{\gamma}) E^T K_1 E, \\
K_2 &= \tilde{K}_2 + (1 - \frac{1}{\gamma}) K_1.
\end{aligned}
$$

Right-Hand-Side Rearrangement

Using the linearity of the Lyapunov-Equation we reformulate the 2nd order Rosenbrock scheme as:

$$
\begin{aligned}
X_{k+1} &= X_k + \frac{3}{2}\tau_k K_1 + \frac{1}{2}\tau_k K_2, \\
\tilde{A}_k^T K_1 E + E^T K_1 \tilde{A}_k &= -\mathcal{R}(X_k), \\
\tilde{A}_k^T \tilde{K}_2 E + E^T \tilde{K}_2 \tilde{A}_k &= \tau_k^2 E^T K_1 B B^T K_1 E + (2 - \frac{1}{\gamma}) E^T K_1 E, \\
K_2 &= \tilde{K}_2 + (1 - \frac{1}{\gamma}) K_1.
\end{aligned}
$$

- The 3rd and 4th order scheme can be rearranged in the same way.

Right-Hand-Side Rearrangement

Using the linearity of the Lyapunov-Equation we reformulate the 2nd order Rosenbrock scheme as:

$$
\begin{aligned}
X_{k+1} &= X_k + \frac{3}{2}\tau_k K_1 + \frac{1}{2}\tau_k K_2, \\
\tilde{A}_k^T K_1 E + E^T K_1 \tilde{A}_k &= -\mathcal{R}(X_k), \\
\tilde{A}_k^T \tilde{K}_2 E + E^T \tilde{K}_2 \tilde{A}_k &= \tau_k^2 E^T K_1 B B^T K_1 E + (2 - \frac{1}{\gamma}) E^T K_1 E, \\
K_2 &= \tilde{K}_2 + (1 - \frac{1}{\gamma}) K_1.
\end{aligned}
$$

- The 3rd and 4th order scheme can be rearranged in the same way.
- Symmetric terms are computed like

$$
E^T K_j B B^T K_j E = K_E^{(j)^T} K_E^{(j)} \quad \text{with} \quad K_E^{(j)} := B^T K_j E.
$$

Parareal Implementation

Following [Maday/Lions/Turinici '01] we use

$$X_0^{(k+1)} := X(t = t_f),$$
$$X_p^{(k+1)} := F(t_{p-1}, t_p, X_{p-1}^{(k)}) + G(t_{p-1}, t_p, X_{p-1}^{(k+1)}) - G(t_{p-1}, t_p, X_{p-1}^{(k)})$$

as parareal-scheme, where $F(t_p, t_{p+1}, X_s)$ and $G(t_p, t_{p+1}, X_s)$ integrate the DRE from t_p to t_{p+1} with the initial value X_s.

Parareal Implementation

Following [Maday/Lions/Turinici '01] we use

$$X_0^{(k+1)} := X(t = t_f),$$
$$X_p^{(k+1)} := F(t_{p-1}, t_p, X_{p-1}^{(k)}) + G(t_{p-1}, t_p, X_{p-1}^{(k+1)}) - G(t_{p-1}, t_p, X_{p-1}^{(k)})$$

as parareal-scheme, where $F(t_p, t_{p+1}, X_s)$ and $G(t_p, t_{p+1}, X_s)$ integrate the DRE from t_p to t_{p+1} with the initial value X_s.

Coarse and Fine Solvers

Use our four Rosenbrock methods as coarse and the fine solvers:

- The coarse solver G performs one time step from t_p to t_{p+1}.
- The fine solver F performs f time steps from t_p to t_{p+1}.

Parareal Implementation

Following [Maday/Lions/Turinici '01] we use

$$X_0^{(k+1)} := X(t = t_f),$$
$$X_p^{(k+1)} := F(t_{p-1}, t_p, X_{p-1}^{(k)}) + G(t_{p-1}, t_p, X_{p-1}^{(k+1)}) - G(t_{p-1}, t_p, X_{p-1}^{(k)})$$

as parareal-scheme, where $F(t_p, t_{p+1}, X_s)$ and $G(t_p, t_{p+1}, X_s)$ integrate the DRE from t_p to t_{p+1} with the initial value X_s.

Coarse and Fine Solvers

Use our four Rosenbrock methods as coarse and the fine solvers:

- The coarse solver G performs one time step from t_p to t_{p+1}.
- The fine solver F performs f time steps from t_p to t_{p+1}.

We obtain 16 combinations for Parareal setup.

Parareal Implementation

Classical Pipeline Implementation: Stage-Code

```
for it:=1 to maxit do
    Receive X_s^(it) from ProcessID-1
    X_G^(it) = G(X_s^(it))
    if it = 1 then
        Send X_G^(it) to ProcessID+1
    else
        X^(it) := X_G^(it) + X_F^(it-1) - X_G^(it-1)
    end if
    X_F^(it) = F(X_s^(it-1))
    if ||X^(it) - X^(it-1)|| < δ||X^(it)|| then
        Stop.
    end if
end for
```

Parareal Implementation

Classical Pipeline Implementation: Stage-Code

```
for it:=1 to maxit do
    Receive X_s^(it) from ProcessID-1
    X_G^(it) = G(X_s^(it))
    if it = 1 then
        Send X_G^(it) to ProcessID+1
    else
        X^(it) := X_G^(it) + X_F^(it-1) - X_G^(it-1)
    end if
    X_F^(it) = F(X_s^(it-1))
    if ||X^(it) - X^(it-1)|| < δ||X^(it)|| then
        Stop.
    end if
end for
```

Can be easily implemented on

- distributed systems using MPI,
- shared memory systems using OpenMP or PThreads.

Large Scale Problems

For large scale problems $n \geq 1\,000$ we approximate $X(t) \approx L(t)D(t)L(t)^T$, where $L(t) \in \mathbb{R}^{n \times p}$ and $D \in \mathbb{R}^{p \times p}$ diagonal with $p \lll n$.

Large Scale Problems

For large scale problems $n \geq 1\,000$ we approximate $X(t) \approx L(t)D(t)L(t)^T$, where $L(t) \in \mathbb{R}^{n \times p}$ and $D \in \mathbb{R}^{p \times p}$ diagonal with $p \lll n$.

- Use the LDL^T-ADI for solving the Lyapunov equations. [Lang/Mena/Saak '15]

Large Scale Problems

For large scale problems $n \geq 1\,000$ we approximate $X(t) \approx L(t)D(t)L(t)^T$, where $L(t) \in \mathbb{R}^{n \times p}$ and $D \in \mathbb{R}^{p \times p}$ diagonal with $p \lll n$.

- Use the LDL^T-ADI for solving the Lyapunov equations. [Lang/Mena/Saak '15]
- Replace the Parareal update

$$X^{(i)} := X_G^{(i)} + X_F^{(i-1)} - X_G^{(i-1)}$$

by

$$L^{(i)} D^{(i)} {L^{(i)}}^T = L_G^{(i)} D_G^{(i)} {L_G^{(i)}}^T + L_F^{(i-1)} D_F^{(i-1)} {L_F^{(i-1)}}^T - L_G^{(i-1)} D_G^{(i-1)} {L_G^{(i-1)}}^T$$

Large Scale Problems

For large scale problems $n \geq 1\,000$ we approximate $X(t) \approx L(t)D(t)L(t)^T$, where $L(t) \in \mathbb{R}^{n\times p}$ and $D \in \mathbb{R}^{p\times p}$ diagonal with $p \lll n$.

- Use the LDL^T-ADI for solving the Lyapunov equations. [Lang/Mena/Saak '15]
- Replace the Parareal update

$$X^{(i)} := X_G^{(i)} + X_F^{(i-1)} - X_G^{(i-1)}$$

by

$$\begin{aligned} L^{(i)}D^{(i)}L^{(i)^T} &= L_G^{(i)}D_G^{(i)}L_G^{(i)^T} + L_F^{(i-1)}D_F^{(i-1)}L_F^{(i-1)^T} - L_G^{(i-1)}D_G^{(i-1)}L_G^{(i-1)^T} \\ &= \underbrace{\begin{bmatrix} L_G^{(i)} & L_F^{(i-1)} & L_G^{(i-1)} \end{bmatrix}}_{L^{(i)}} \underbrace{\begin{bmatrix} D_G^{(i)} & & \\ & D_F^{(i-1)} & \\ & & -D_G^{(i-1)} \end{bmatrix}}_{D^{(i)}} \underbrace{\begin{bmatrix} L_G^{(i)} \\ L_F^{(i-1)} \\ L_G^{(i-1)} \end{bmatrix}^T}_{L^{(i)^T}}. \end{aligned}$$

Large Scale Problems

- The size of the factor $L^{(i)}$ increases in every iteration.

Large Scale Problems

- The size of the factor $L^{(i)}$ increases in every iteration.
- Redundant information in $L_G^{(i)}$, $L_G^{(i-1)}$, and $L_F^{(i-1)}$ on convergence.

Large Scale Problems

- The size of the factor $L^{(i)}$ increases in every iteration.
- Redundant information in $L_G^{(i)}$, $L_G^{(i-1)}$, and $L_F^{(i-1)}$ on convergence.

Truncate $L^{(i)}$ and $D^{(i)}$:

$$L^{(i)} D^{(i)} {L^{(i)}}^T = \overbrace{U \Sigma V^T}^{\text{SVD of } L^{(i)}} D^{(i)} V \Sigma U^T$$

Large Scale Problems

- The size of the factor $L^{(i)}$ increases in every iteration.
- Redundant information in $L_G^{(i)}$, $L_G^{(i-1)}$, and $L_F^{(i-1)}$ on convergence.

Truncate $L^{(i)}$ and $D^{(i)}$:

$$
\begin{aligned}
L^{(i)} D^{(i)} {L^{(i)}}^T &= \overbrace{U \Sigma V^T}^{\text{SVD of } L^{(i)}} \; D^{(i)} V \Sigma U^T \\
&\approx U_k \Sigma_k V_k^T D^{(i)} V_k \Sigma_k U_k^T \qquad\qquad k = \operatorname{rank}\left(L^{(i)}, \delta\right)
\end{aligned}
$$

Large Scale Problems

- The size of the factor $L^{(i)}$ increases in every iteration.
- Redundant information in $L_G^{(i)}$, $L_G^{(i-1)}$, and $L_F^{(i-1)}$ on convergence.

Truncate $L^{(i)}$ and $D^{(i)}$:

$$
\begin{aligned}
L^{(i)} D^{(i)} {L^{(i)}}^T &= \overbrace{U \Sigma V^T}^{\text{SVD of } L^{(i)}} \; D^{(i)} V \Sigma U^T \\
&\approx U_k \Sigma_k V_k^T D^{(i)} V_k \Sigma_k U_k^T \qquad\qquad k = \operatorname{rank}\left(L^{(i)}, \delta\right) \\
&= \underbrace{U_k \Sigma_k \tilde{V}_k^T}_{\tilde{L}^{(i)}} \overbrace{\tilde{D}^{(i)} \tilde{V}_k}^{\text{EVD of } V_k^T D^{(i)} V_k} \Sigma_k U_k^T
\end{aligned}
$$

Large Scale Problems

- The size of the factor $L^{(i)}$ increases in every iteration.
- Redundant information in $L_G^{(i)}$, $L_G^{(i-1)}$, and $L_F^{(i-1)}$ on convergence.

Truncate $L^{(i)}$ and $D^{(i)}$:

$$
\begin{aligned}
L^{(i)} D^{(i)} {L^{(i)}}^T &= \overbrace{U \Sigma V^T}^{\text{SVD of } L^{(i)}} \; D^{(i)} V \Sigma U^T \\
&\approx U_k \Sigma_k V_k^T D^{(i)} V_k \Sigma_k U_k^T \qquad\qquad k = \operatorname{rank}\left(L^{(i)}, \delta\right) \\
&= \underbrace{U_k \Sigma_k \overbrace{\tilde{V}_k^T \tilde{D}^{(i)} \tilde{V}_k}^{\text{EVD of } V_k^T D^{(i)} V_k}}_{\tilde{L}^{(i)}} \Sigma_k U_k^T \\
&= \tilde{L}^{(i)} \tilde{D}^{(i)} {\tilde{L}^{(i)}}^T
\end{aligned}
$$

Large Scale Problems

- The size of the factor $L^{(i)}$ increases in every iteration.
- Redundant information in $L_G^{(i)}$, $L_G^{(i-1)}$, and $L_F^{(i-1)}$ on convergence.

Truncate $L^{(i)}$ and $D^{(i)}$:

$$
\begin{aligned}
L^{(i)} D^{(i)} {L^{(i)}}^T &= \overbrace{U \Sigma V^T}^{\text{SVD of } L^{(i)}} \; D^{(i)} V \Sigma U^T \\
&\approx U_k \Sigma_k V_k^T D^{(i)} V_k \Sigma_k U_k^T \qquad\qquad k = \operatorname{rank}\left(L^{(i)}, \delta\right) \\
&= \underbrace{U_k \Sigma_k \overbrace{\tilde{V}_k^T \tilde{D}^{(i)} \tilde{V}_k}^{\text{EVD of } V_k^T D^{(i)} V_k} \Sigma_k}_{\tilde{L}^{(i)}} U_k^T \\
&= \tilde{L}^{(i)} \tilde{D}^{(i)} {\tilde{L}^{(i)}}^T
\end{aligned}
$$

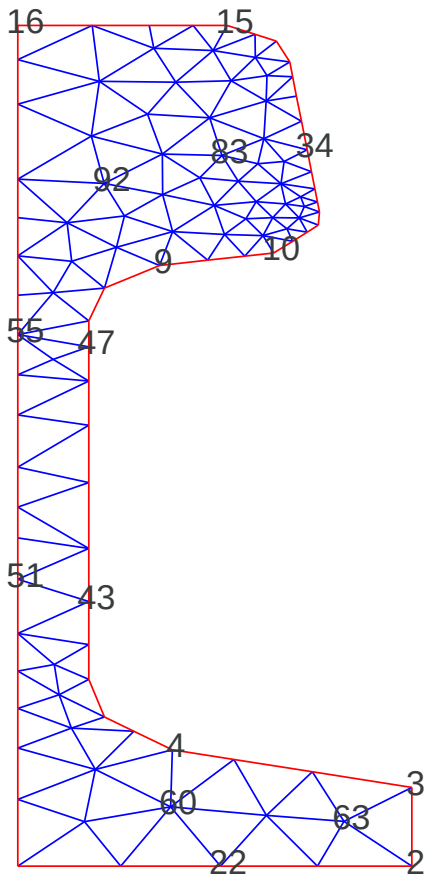
$\rightarrow$ Only proof of concept due to bad load balancing in the pipeline parallelism.

Model Problem

[Oberwolfach Rail]

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

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



- FEM discretization with $n = 371$ states, $m = 7$ inputs, and $q = 6$ outputs
- computations with $\tau = 0.1ms$ on $[0, 45]s$
- evaluations for one component of the feedback $K(t) = -B^T X(t) E$

Experimental Results

Hardware Environment

HPC-Cluster `otto`

- 450 Intel® Xeon® Westmere-EP cores @ 2.66GHz with 4 GB RAM each
- QDR-Infiniband interconnect

Software

- Intel® Parallel Studio 2015 XE
- OpenMPI 1.8.1 with threading support
- Intel® MKL 11.2.1

Experimental Results

Hardware Environment

HPC-Cluster `otto`

- 450 Intel® Xeon® Westmere-EP cores @ 2.66GHz with 4 GB RAM each
- QDR-Infiniband interconnect

Software

- Intel® Parallel Studio 2015 XE
- OpenMPI 1.8.1 with threading support
- Intel® MKL 11.2.1

Reference Result

- 450 000 steps with $\tau = 0.1$ms with Ros4 in **3.46 days**
- Memory requirements for the trajectory: $X(t)$ – **1.4 TB**, $K(t)$ – **9 GB**

Experimental Results

Sequential Runtimes

Rosenbrock Order	SLICOT	GLYAP 3	ratio
1	4.40 d	2.87 d	1.53
2	6.15 d	3.06 d	2.01
3	7.84 d	3.27 d	2.40
4	9.55 d	3.46 d	2.75

Table: Sequential Runtime of the Rosenbrock methods with different Lyapunov solvers.

Experimental Results

Sequential Runtimes

Rosenbrock Order	SLICOT	GLYAP 3	ratio
1	4.40 d	2.87 d	1.53
2	6.15 d	3.06 d	2.01
3	7.84 d	3.27 d	2.40
4	9.55 d	3.46 d	2.75

Table: Sequential Runtime of the Rosenbrock methods with different Lyapunov solvers.

First observations

- A fast solver for the Lyapunov equations is a key ingredient,
- Choosing the correct Lyapunov solver already gains a speed up of 2.75,
- But 3.46 days are still too long for a accurate solution of such a small problem.

Experimental Results

Distributed Parallel Execution

Parareal Setup

- 450 coarse steps, $\tau_{coarse} = 100\text{ms}$
- 1000 fine steps per coarse step, $\tau = 0.1\text{ms}$
- Maximum number of iterations: 10
- Stopping criteria: $\delta = 10^{-6}$

Experimental Results

Distributed Parallel Execution

Parareal Setup

- 450 coarse steps, $\tau_{coarse} = 100$ms
- 1000 fine steps per coarse step, $\tau = 0.1$ms
- Maximum number of iterations: 10
- Stopping criteria: $\delta = 10^{-6}$

Fine \ Coarse	Ros1		Ros2		Ros3		Ros4	
	Time	Iter	Time	Iter	Time	Iter	Time	Iter
Ros1	3.30 h	9	2.97 h	8	3.29 h	9	1.74 h	4
Ros2	3.59 h	9	3.27 h	8	3.61 h	9	1.89 h	4
Ros3	3.87 h	9	3.51 h	8	3.88 h	9	2.02 h	4
Ros4	4.17 h	9	3.78 h	8	4.18 h	9	2.19 h	4

Table: Runtime and maximum iteration number.

Experimental Results

Distributed Parallel Execution

Parareal Setup

- 450 coarse steps, $\tau_{coarse} = 100$ms
- 1000 fine steps per coarse step, $\tau = 0.1$ms
- Maximum number of iterations: 10
- Stopping criteria: $\delta = 10^{-6}$

Fine \ Coarse	Ros1	Ros2	Ros3	Ros4
Ros1	2.01e-05	2.01e-05	2.01e-05	2.01e-05
Ros2	2.07e-05	2.07e-05	2.07e-05	2.07e-05
Ros3	2.07e-05	2.07e-05	2.07e-05	2.07e-05
Ros4	**1.27e-09**	**3.00e-10**	**9.71e-08**	**6.10e-14**

Table: Relative 1-norm error between the Parareal solution and the reference.

Experimental Results

Distributed Parallel Execution

Parareal Setup

- 450 coarse steps, $\tau_{coarse} = 100$ms
- 1000 fine steps per coarse step, $\tau = 0.1$ms
- Maximum number of iterations: 10
- Stopping criteria: $\delta = 10^{-6}$

Fine \ Coarse	Ros1	Ros2	Ros3	Ros4
Ros1	*	7.69e-05	7.68e-05	7.68e-05
Ros2	*	7.81e-05	7.80e-05	7.80e-05
Ros3	*	7.81e-05	7.80e-05	7.80e-05
Ros4	*	**3.14e-11**	**5.76e-09**	**1.14e-14**

Table: Relative 1-norm error between the Parareal solution and the reference for $K(t)_{1,77}$.

Conclusions and Open Problems

Conclusions

We have seen that:

- "optimize before parallelize" already gains nearly a factor of up to 3,
- Parareal shrinks the runtime down to 2.19h with accurate results.

Conclusions and Open Problems

Conclusions

We have seen that:

- "optimize before parallelize" already gains nearly a factor of up to 3,
- Parareal shrinks the runtime down to 2.19h with accurate results.

We observed that:

- (our) Parareal implementation requires the same computational complexity for each evaluation of the coarse or the fine solver,
- we have relatively long startup phase until all processors work in parallel,
- we are restricted to one-step methods on the coarse level.

Conclusions and Open Problems

Conclusions

We have seen that:

- "optimize before parallelize" already gains nearly a factor of up to 3,
- Parareal shrinks the runtime down to 2.19h with accurate results.

We observed that:

- (our) Parareal implementation requires the same computational complexity for each evaluation of the coarse or the fine solver,
- we have relatively long startup phase until all processors work in parallel,
- we are restricted to one-step methods on the coarse level.

Thank you for your attention! Questions?