TUHH
Hamburg University of Technology

UNIFIED ANALYSIS FRAMEWORK FOR ITERATIVE PARALLEL-IN-TIME ALGORITHMS

Thibaut Lunet, & Martin J. Gander (University of Geneva) & Daniel Ruprecht (TUHH) & Robert Speck (Forschungszentrum Jülich GmbH) & Jens Hahne (University of Wuppertal)

Chair Computational Mathematics
Institute of Mathematics (E-10)
Hamburg University of Technology

IMPDE Research School, LJLL Paris Sorbonne

May 16, 2023

Main problematic

$$\text{Solving ODE/PDE} \quad \frac{dU}{dt} = f(U,t), \quad \text{for} \quad t \in [0,T]$$

To use a PinT algorithm, you should ...

1. Define the number of time sub-intervals N (processors ...)

Main problematic

$$\text{Solving ODE/PDE} \quad \frac{dU}{dt} = f(U,t), \quad \text{for} \quad t \in [0,T]$$

To use a PinT algorithm, you should ...

1. Define the number of time sub-intervals N (processors ...)
2. Define some inner discretization of time sub-intervals

Main problematic

$$\text{Solving ODE/PDE} \quad \frac{dU}{dt} = f(U,t), \quad \text{for} \quad t \in [0,T]$$

To use a PinT algorithm, you should ...

1. Define the number of time sub-intervals N (processors ...)
2. Define some inner discretization of time sub-intervals
3. Choose a time-parallel algorithm (Parareal, MGRIT, PFASST, ...)

Main problematic

$$\text{Solving ODE/PDE} \quad \frac{dU}{dt} = f(U,t), \quad \text{for} \quad t \in [0,T]$$

To use a PinT algorithm, you should ...

1. Define the number of time sub-intervals N (processors ...)
2. Define some inner discretization of time sub-intervals
3. Choose a time-parallel algorithm (Parareal, MGRIT, PFASST, ...)
4. Choose time-integrators for each time sub-interval

Main problematic

$$\text{Solving ODE/PDE} \quad \frac{dU}{dt} = f(U,t), \quad \text{for} \quad t \in [0,T]$$

To use a PinT algorithm, you should ...

1. Define the number of time sub-intervals N (processors ...)
2. Define some inner discretization of time sub-intervals
3. Choose a time-parallel algorithm (Parareal, MGRIT, PFASST, ...)
4. Choose time-integrators for each time sub-interval
5. Select time-integration settings (number of inner time steps, ...)

Main problematic

$$\text{Solving ODE/PDE} \quad \frac{dU}{dt} = f(U,t), \quad \text{for} \quad t \in [0,T]$$

To use a PinT algorithm, you should ...

1. Define the number of time sub-intervals N (processors ...)
2. Define some inner discretization of time sub-intervals
3. Choose a time-parallel algorithm (Parareal, MGRIT, PFASST, ...)
4. Choose time-integrators for each time sub-interval
5. Select time-integration settings (number of inner time steps, ...)
6. Select PinT algorithm parameters (number of iterations, ...)

Main problematic

$$\text{Solving ODE/PDE} \quad \frac{dU}{dt} = f(U,t), \quad \text{for} \quad t \in [0,T]$$

To use a PinT algorithm, you should ...

1. Define the number of time sub-intervals N (processors ...)
2. Define some inner discretization of time sub-intervals
3. Choose a time-parallel algorithm (Parareal, MGRIT, PFASST, ...)
4. Choose time-integrators for each time sub-interval
5. Select time-integration settings (number of inner time steps, ...)
6. Select PinT algorithm parameters (number of iterations, ...)
7. ...

Perspective from a Starting PhD Student

TUHH
Hamburg University of Technology

Supervisor idea :

"Let's use PinT to speedup simulation for this problem, it looks cool !"

Perspective from a Starting PhD Student

TUHH
Hamburg University of Technology

Supervisor idea :

"Let's use PinT to speedup simulation for this problem, it looks cool !"

PhD student reaction :

... and don't get me started on the problem's influence (parabolic, hyperbolic, ...)

Known Issues in PinT Community

- Numerous publications, usually complex and dense ...
  $\rightarrow$ see `http://parallel-in-time.org/references/index.html`

- Many algorithm defined with specific notations / scheme
  $\rightarrow$ need a lot of time to understand and select the best solution

- No standardized performance comparison of each algorithms
  $\rightarrow$ very hard to find the most efficient one for the problem of interest

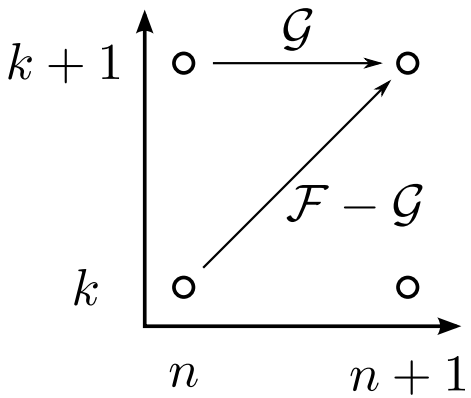
Motivation to describe and analyze everything from the same perspective

Chronological Pathway

1. First (relatively easy) contact with Parareal

$$u_{n+1}^{k+1} = (F - G)u_n^k + Gu_n^{k+1}$$

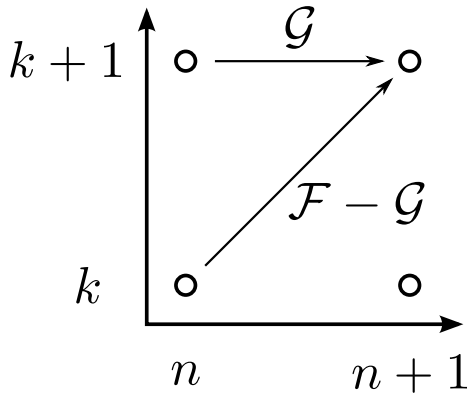
$\rightarrow$ use a "block"-like formulation



1. First (relatively easy) contact with Parareal

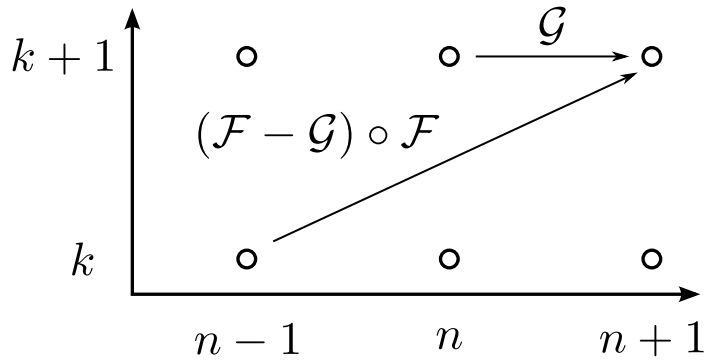
$$u_{n+1}^{k+1} = (F - G)u_n^k + Gu_n^{k+1}$$

$\rightarrow$ use a "block"-like formulation



2. Discovering MGRIT with FCF-relaxation
   $\rightarrow$ **block formulation** found by Gander, Kwok & Zhang

$$u_{n+1}^{k+1} = (F - G)Fu_{n-1}^k + Gu_n^{k+1}$$

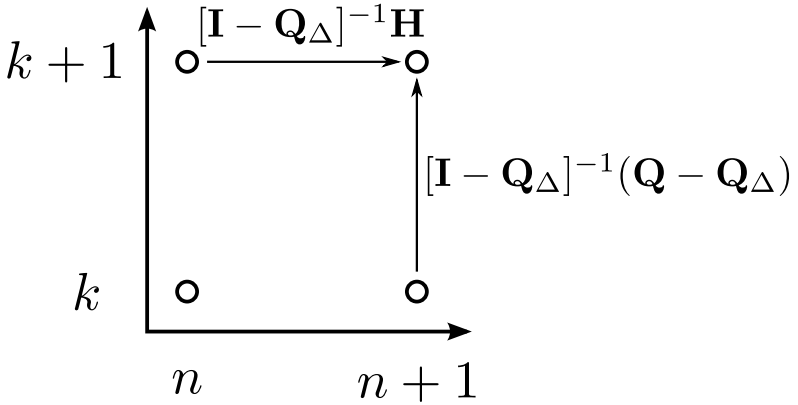


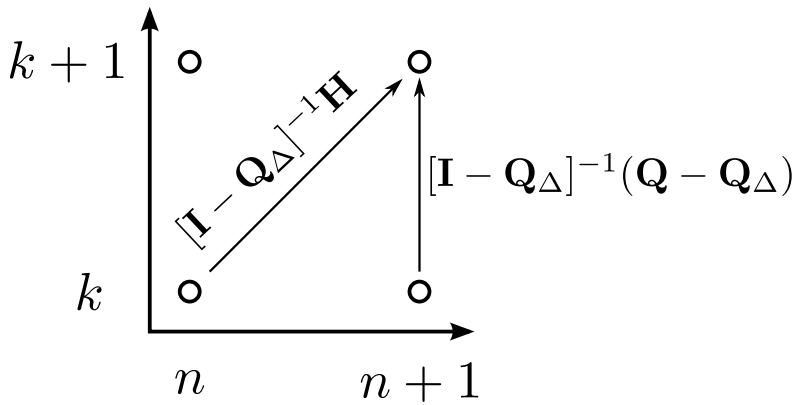
$\Rightarrow$ Idea to use **generating functions** to obtain generic error bounds (Gander & Hairer)

3. Trying to understand PFASST components for simple linear ODE

$$\text{BGS-SDC}: \quad \boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^{k} + [\mathbf{I} - \mathbf{Q}_\Delta]^{-1} \left(\mathbf{H}\boldsymbol{u}_{n}^{k+1} - (\mathbf{I} - \mathbf{Q})\boldsymbol{u}_{n+1}^{k} \right)$$

$$\text{BJ-SDC}: \quad \boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^{k} + [\mathbf{I} - \mathbf{Q}_\Delta]^{-1} \left(\mathbf{H}\boldsymbol{u}_{n}^{k} - (\mathbf{I} - \mathbf{Q})\boldsymbol{u}_{n+1}^{k} \right)$$

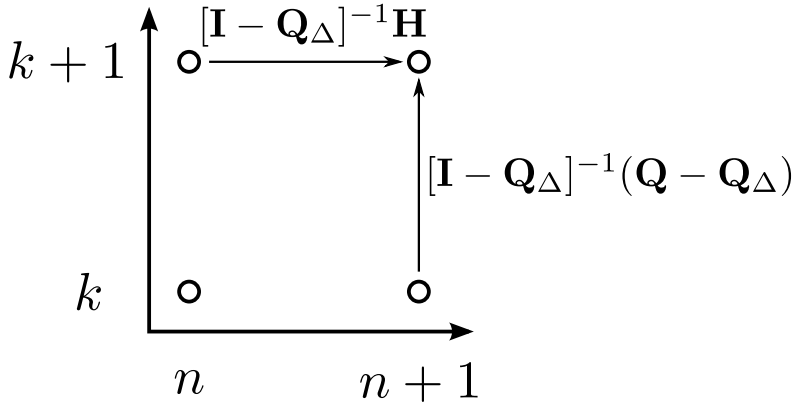


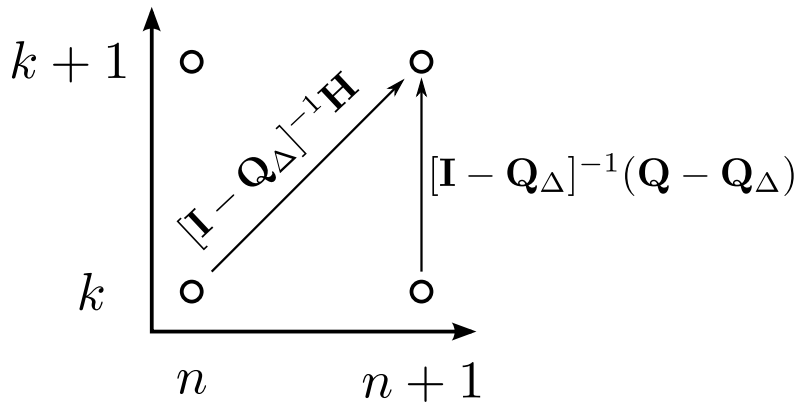


3. Trying to understand PFASST components for simple linear ODE

$$\text{BGS-SDC}: \quad \boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^{k} + [\mathbf{I} - \mathbf{Q}_\Delta]^{-1} \left(\mathbf{H}\boldsymbol{u}_{n}^{k+1} - (\mathbf{I} - \mathbf{Q})\boldsymbol{u}_{n+1}^{k} \right)$$

$$\text{BJ-SDC}: \quad \boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^{k} + [\mathbf{I} - \mathbf{Q}_\Delta]^{-1} \left(\mathbf{H}\boldsymbol{u}_{n}^{k} - (\mathbf{I} - \mathbf{Q})\boldsymbol{u}_{n+1}^{k} \right)$$





$\Rightarrow$ Also a **block formulation**, can use generating functions too !

$\Rightarrow$ Let's try to write every PinT algorithm with the same formalism ...

$\Rightarrow$ **Generating Function Method (GFM) Framework**

Let us focus on an elementary time-dependent ODE :

$$\frac{du}{dt} = \lambda u, \quad \lambda \in \mathbb{C}, \quad t \in [0, T]$$

$\Rightarrow$ iterative PinT algorithms applied on it solve the linear system :

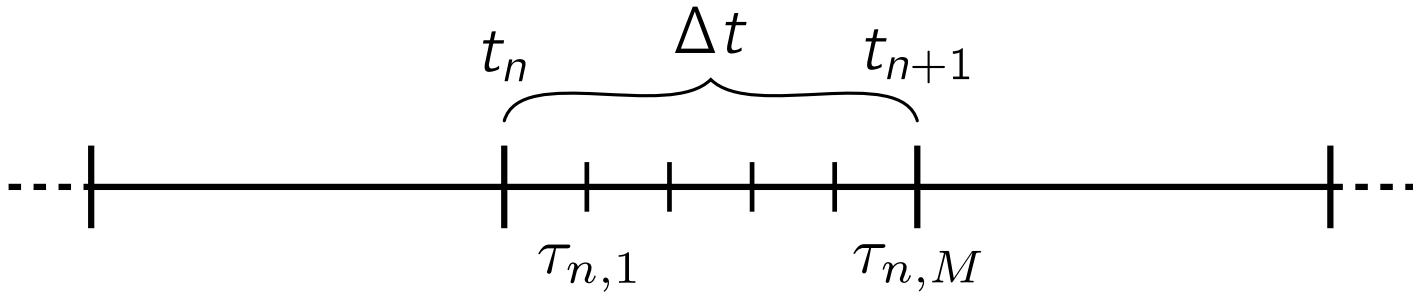
$$\begin{pmatrix} \phi & & & \\ -\chi & \phi & & \\ & \ddots & \ddots & \\ & & -\chi & \phi \end{pmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_N \end{bmatrix} = \begin{bmatrix} \chi(u_0 \mathbb{1}) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \Leftrightarrow \quad \mathbf{A}\boldsymbol{u} = \boldsymbol{f}$$

using a given **iterative method**.

Specific lower-diagonal block formulation due to the time-dependency

First step : decomposition into **time blocks**

- $[0, T]$ decomposed into N sub-intervals $[t_n, t_{n+1}]$ of size Δt
- *Block* discretization: $\tau_{n,m} = t_n + \Delta t \tau_m, \quad m \in \{1, 2, ..., M\}$



$\Rightarrow$ time steps of a time discretization, nodes of collocation method, ...

- Numerical approximation of the solution on one block

$$\boldsymbol{u}_n = [u_{n,1}, u_{n,2}, ..., u_{n,M}]^T$$

$\rightarrow$ can be vector or scalar (with $M = 1$)

- In practice : one block $\rightarrow$ one computation process for PinT

Second step : define some **block operators**

Linear operators mapping two consecutive block variables

$$\phi(\boldsymbol{u}_{n+1}) = \chi(\boldsymbol{u}_n) \quad \Leftrightarrow \quad \boldsymbol{u}_{n+1} = \phi^{-1}\chi\boldsymbol{u}_n := \psi(\boldsymbol{u}_n)$$

- ϕ is a bijective operator (time-integration scheme)
  - Runge-Kutta type
  - Multistep
  - Collocation based method
  - ...
- χ is used to build the initial solution for the next block
  $\rightarrow$ depends on the chosen time-integration scheme

Example with Runge-Kutta time integration

Stability function $R(z) \approx e^z$, ℓ equidistant time steps per block.

- *interface formulation* (natural approach): $M := 1$

$$\phi := R(\lambda \Delta t/\ell)^{-\ell}, \quad \chi := 1$$

- *volume formulation*: $M := \ell$, $\tau_m := m/\ell$, $r := R(\lambda \Delta t/\ell)^{-1}$

$$\phi := \begin{pmatrix} r & & \\ -1 & r & \\ & \ddots & \ddots \end{pmatrix}, \quad \chi := \begin{pmatrix} 0 & \dots & 0 & 1 \\ \vdots & & \vdots & 0 \\ \vdots & & \vdots & \vdots \end{pmatrix}$$

- other volume formulations are possible
- similar idea for other time discretizations (multistep, collocation, ...)

Back to the generic global problem

$$\begin{pmatrix} \phi & & & \\ -\chi & \phi & & \\ & \ddots & \ddots & \\ & & -\chi & \phi \end{pmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_N \end{bmatrix} = \begin{bmatrix} \chi(u_0 \mathbb{1}) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \Leftrightarrow \quad \mathbf{A}\boldsymbol{u} = \boldsymbol{f}$$

First iterative approach to solve such system :

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \mathbf{P}^{-1}(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k)$$

$\Rightarrow$ For "standard" preconditionners $\mathbf{P}$, write the associated **Block Iteration Formula**

1) Damped Block Jacobi (BJ)

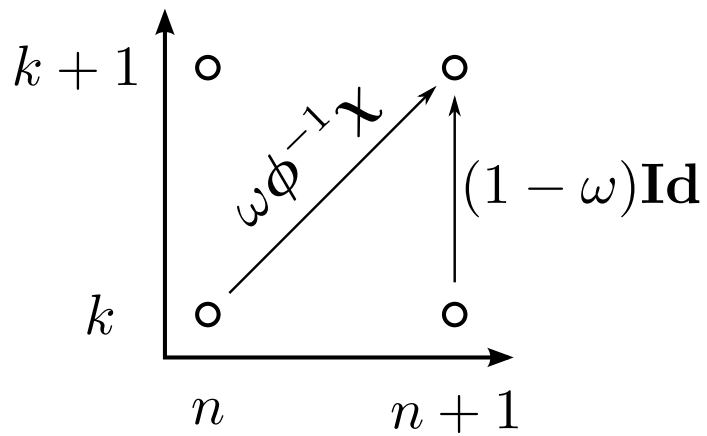
- Full matrix representation :

$$\mathbf{P}_{BJ} = \frac{1}{\omega} \begin{pmatrix} \phi & & \\ & \ddots & \\ & & \phi \end{pmatrix}, \quad \omega > 0$$

$\rightarrow$ completely parallel within each block once $\boldsymbol{u}^k$ is known
$\rightarrow$ in practice, used as a smoother (slow convergence)

- Block Iteration formula :

$$\boldsymbol{u}_{n+1}^{k+1} = (1-\omega)\boldsymbol{u}_{n+1}^{k} + \omega\phi^{-1}\boldsymbol{\chi}\boldsymbol{u}_{n}^{k}$$



2) Approximate Block Gauss-Seidel (ABGS)

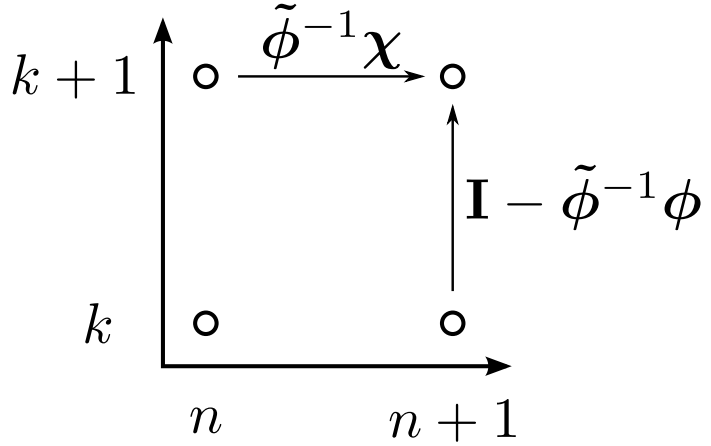
- Full matrix representation :

$$\mathbf{P}_{ABGS} = \begin{pmatrix} \tilde{\phi} & & & \\ -\chi & \tilde{\phi} & & \\ & \ddots & \ddots & \\ & & -\chi & \tilde{\phi} \end{pmatrix}$$

$\rightarrow \tilde{\phi}$ is a cheaper approximation of ϕ (*e.g.* easier to invert)
$\rightarrow$ not fully parallel, but sequential part is supposed to be cheap

- Block Iteration formula :

$$\boldsymbol{u}_{n+1}^{k+1} = [\mathbf{I} - \tilde{\phi}^{-1}\phi]\boldsymbol{u}_{n+1}^{k} + \tilde{\phi}^{-1}\chi\boldsymbol{u}_{n}^{k+1}$$



First application : description of Parareal

1. One BJ iteration ($\omega = 1$) : $\boldsymbol{u}_{n+1}^{k+1/2} = \phi^{-1}\chi \boldsymbol{u}_n^k$
2. One ABGS iteration : $\boldsymbol{u}_{n+1}^{k+1} = [\mathbf{I} - \tilde{\phi}^{-1}\phi]\boldsymbol{u}_{n+1}^{k+1/2} + \tilde{\phi}^{-1}\chi \boldsymbol{u}_n^{k+1}$

Second application : description of Parareal with overlap

1. Two BJ iterations :

$$\boldsymbol{u}_{n+1}^{k+1/2} = \phi^{-1}\chi \boldsymbol{u}_n^k$$

$$\boldsymbol{u}_{n+1}^{k+1/3} = \phi^{-1}\chi \boldsymbol{u}_n^{k+1/2}$$

2. One ABGS iteration : $\boldsymbol{u}_{n+1}^{k+1} = [\mathbf{I} - \tilde{\phi}^{-1}\phi]\boldsymbol{u}_{n+1}^{k+1/3} + \tilde{\phi}^{-1}\chi \boldsymbol{u}_n^{k+1}$

$\rightarrow$ also equivalent (to some extend) to MGRIT with FCF-Relaxation

Coarse Grid Correction (CGC) for Time Multi-Grid (TMG)

1) Coarse problem for each block of size $M^C < M$:

$$\begin{pmatrix} \phi_C & & & \\ -\chi_C & \phi_C & & \\ & \ddots & \ddots & \\ & & -\chi_C & \phi_C \end{pmatrix} \begin{bmatrix} \boldsymbol{u}_1^C \\ \boldsymbol{u}_2^C \\ \vdots \\ \boldsymbol{u}_N^C \end{bmatrix} = \begin{bmatrix} \mathbf{T}_F^C \chi(u_0 \mathbb{1}) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \Leftrightarrow \quad \mathbf{A}_C \boldsymbol{u}^C = \boldsymbol{f}^C$$

$\rightarrow$ same time-stepping method, but more points (h-coarsening)
$\rightarrow$ lower order Collocation based time-stepping (p-coarsening)

2) Restriction and prolongation operators between coarse and fine block :

- $\mathbf{T}_F^C$ or shape (M^C, M), from fine to coarse
- $\mathbf{T}_C^F$ or shape (M, M^C), from coarse to fine

3) CGC in global form :

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^{k} + \bar{\mathbf{T}}_C^F \mathbf{A}_C^{-1} \bar{\mathbf{T}}_F^C (\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^{k})$$

4) Additional simplifying assumptions :

1. $\mathbf{T}_F^C \mathbf{T}_C^F = \mathbf{I}$: invariance of interpolation + restriction on a coarse field
2. $\Delta_\chi = \mathbf{T}_F^C \boldsymbol{\chi} - \boldsymbol{\chi}_C \mathbf{T}_F^C = 0$: usually verified in TMG-based algorithms

$\Rightarrow$ Block Iteration of the CGC :

$$\boldsymbol{u}_{n+1}^{k+1} = (\mathbf{I} - \mathbf{T}_C^F \boldsymbol{\phi}_C^{-1} \mathbf{T}_F^C \boldsymbol{\phi}) \boldsymbol{u}_{n+1}^{k} + \mathbf{T}_C^F \boldsymbol{\phi}_C^{-1} \mathbf{T}_F^C \boldsymbol{\chi} \boldsymbol{u}_{n}^{k+1}$$

$\rightarrow$ *similarities with ABGS :*

$$\boldsymbol{u}_{n+1}^{k+1} = [\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1} \boldsymbol{\chi}] \boldsymbol{u}_{n+1}^{k} + \tilde{\boldsymbol{\phi}}^{-1} \boldsymbol{\chi} \boldsymbol{u}_{n}^{k+1}$$

Time Multi-Grid on two levels

1. One BJ iteration (smoother) : $\boldsymbol{u}_{n+1}^{k+1/2} = (1-\omega)\boldsymbol{u}_{n+1}^{k} + \omega\phi^{-1}\boldsymbol{\chi}\boldsymbol{u}_{n}^{k}$
2. One CGC : $\boldsymbol{u}_{n+1}^{k+1} = (\mathbf{I} - \mathbf{T}_C^F\phi_C^{-1}\mathbf{T}_F^C\phi)\boldsymbol{u}_{n+1}^{k+1/2} + \mathbf{T}_C^F\phi_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}\boldsymbol{u}_{n}^{k+1}$

PFASST with two levels

1. One Approximate BJ iteration :
   $\boldsymbol{u}_{n+1}^{k+1/2} = \left[\mathbf{I} - \tilde{\phi}^{-1}\phi\right]\boldsymbol{u}_{n+1}^{k} + \tilde{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_{n}^{k}$
2. One ABGS iteration on coarse level to solve the CGC
   $\rightarrow$ details not shown ...

Block Iterations for PinT

$\Rightarrow$ represent any iterative PinT algorithm like this :

$$\boldsymbol{u}_{n+1}^{k+1} = \mathbf{B}_0^0 \boldsymbol{u}_n^k + \mathbf{B}_1^0 \boldsymbol{u}_{n+1}^k + \mathbf{B}_0^1 \boldsymbol{u}_n^{k+1} + \dots$$

Algorithm	$\mathbf{B}_1^0$ $(\boldsymbol{u}_{n+1}^k)$	$\mathbf{B}_0^0$ $(\boldsymbol{u}_n^k)$	$\mathbf{B}_0^1$ $(\boldsymbol{u}_n^{k+1})$
damped Block Jacobi	$\mathbf{I} - \omega\mathbf{I}$	$\omega\boldsymbol{\phi}^{-1}\boldsymbol{\chi}$	–
ABJ	$\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi}$	$\tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}$	–
ABGS	$\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi}$	–	$\tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}$
PARAREAL	–	$(\boldsymbol{\phi}^{-1} - \tilde{\boldsymbol{\phi}}^{-1})\boldsymbol{\chi}$	$\tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}$
TMG	$(1-\omega)(\mathbf{I} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi})$	$\omega(\boldsymbol{\phi}^{-1} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C)\boldsymbol{\chi}$	$\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$
TMG_c	–	$(\boldsymbol{\phi}^{-1} - \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C)\boldsymbol{\chi}$	$\mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$
TMG_f	$(\mathbf{I} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi})(\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi})$	$(\tilde{\boldsymbol{\phi}}^{-1} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi}\tilde{\boldsymbol{\phi}}^{-1})\boldsymbol{\chi}$	$\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$
PFASST	$(\mathbf{I} - \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi})(\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi})$	$(\tilde{\boldsymbol{\phi}}^{-1} - \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi}\tilde{\boldsymbol{\phi}}^{-1})\boldsymbol{\chi}$	$\mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$

$\rightarrow$ write generic error bound for each iterative PinT methods

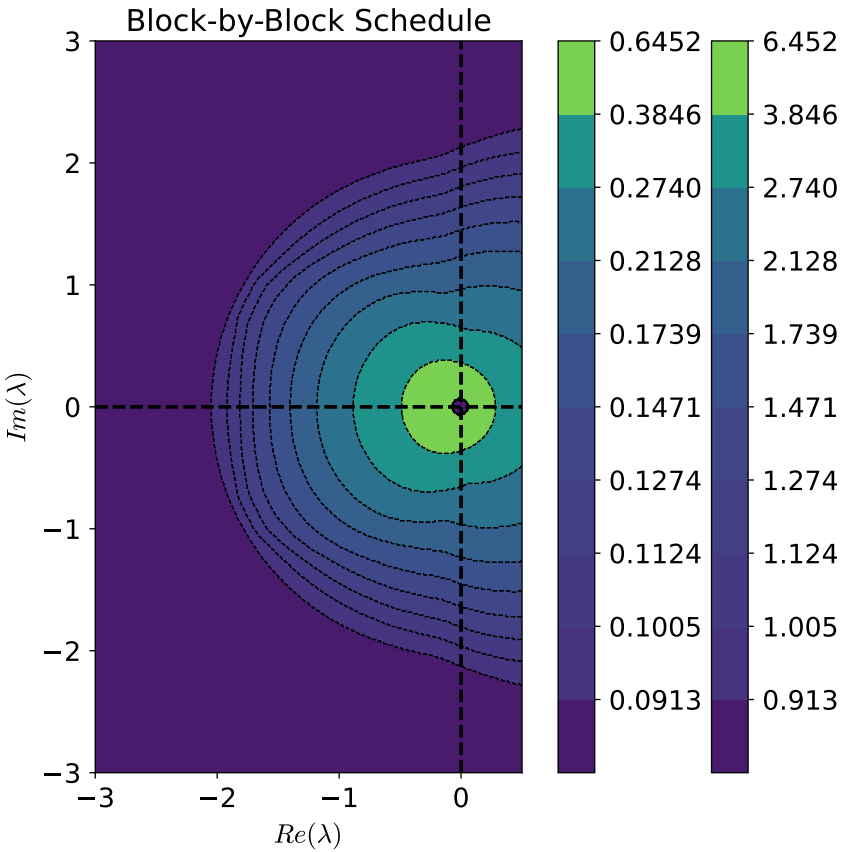
$\rightarrow$ compare convergence/speedup of each algorithm on equivalent basis

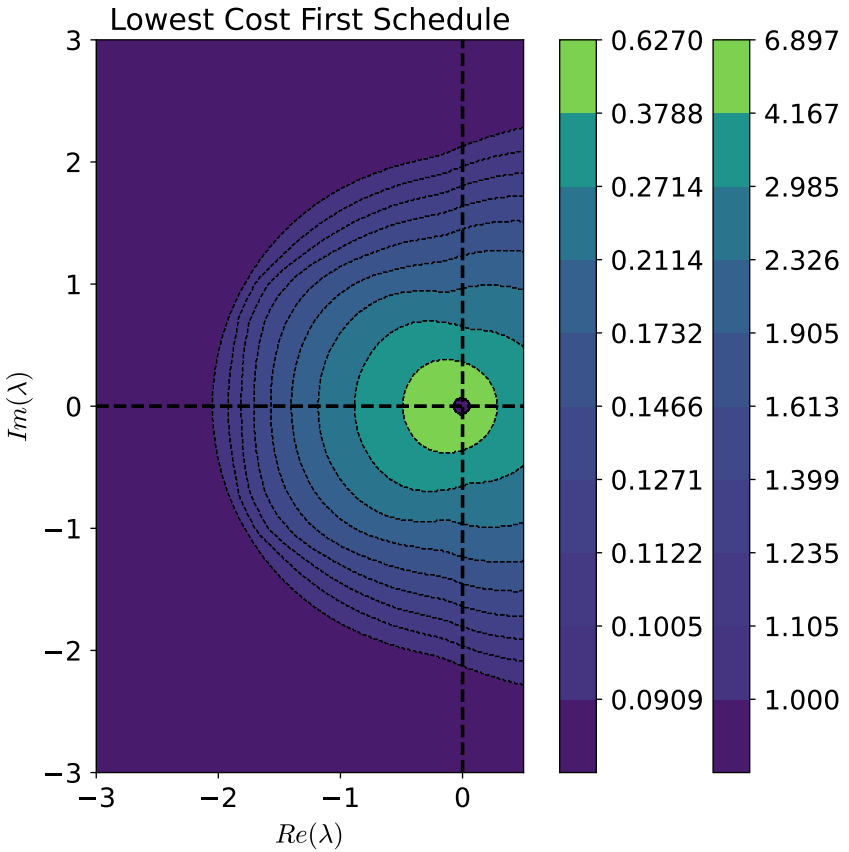
First Application

Hamburg University of Technology

Modeled maximum parallel speedup in complex plane

→ example : Parareal on 10 blocks, RK4, coarse/fine time-step ratio of 20





⇒ can be used for a first selection of optimum algorithms / parameters

Conclusion

TUHH
Hamburg University of Technology

Current progress

- Main paper accepted : [Gander, L., Ruprecht & Speck, 2022]
- Python Library (in development with J. Hahne) :
  `https://github.com/Parallel-in-Time/time4apint`
  → many algorithms implemented : Parareal, PFASST, ...
  → many time-stepping method available : Runge-Kutta, SDC, ...

Perspectives

- Generic open-source code for first accuracy & performance analysis
  → generic/robust implementations for many PinT methods,
  → documentation and tutorials, continuous testing, ...
- Implementation of a Website for demonstrations and first tests
- Extend analysis tool to more complex algorithms :
  ⇒ MGRIT-FCF, multi-level, ...

Acknowledgments

TUHH
Hamburg University of Technology

EuroHPC
Joint Undertaking

SPONSORED BY THE

Federal Ministry
of Education
and Research

The Time-X project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701.
The JU receives support from the European Union's Horizon 2020 research and innovation programme and Belgium, France, Germany, and Switzerland.
This project also received funding from the German Federal Ministry of Education and Research (BMBF) grant 16HPC048.

Partners

- Bergische Universität Wuppertal (BUW)
- Ecole des Ponts ParisTech (ENPC)
- Forschungszentrum Jülich (FZJ)
- Katholieke Universiteit Leuven (KUL)
- Sorbonne Université (SU)
- Technische Universität Darmstadt (TUD)
- Technische Universität Hamburg (TUHH)
- Technische Universität München (TUM)
- Università della Svizzera italiana (USI)
- Université de Genève (UNIGE)