An Introduction to Parallel-in-time methods

Ausra Pogozelskyte

University of Geneva

Paris, 16th of May 2023

Motivation

What is a Parallel-in-Time method?

Methods that compute solutions at a further time step before that the solution at a closer time step has been computed.

- Usually iterative methods
- Usually comes at the expense of additional work

Why are Parallel-in-Time methods interesting?

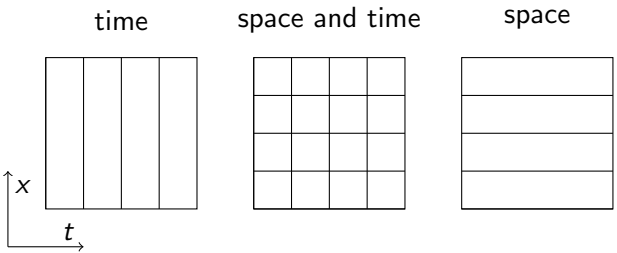
- Problems that need a solution by a certain deadline
- Problems that are very long in time

Outline

Traditionally, Parallel-in-Time methods are classified in the following categories:

1. Shooting type methods
   - Parareal
2. Multigrid methods
   - Multigrid Reduction-in-Time
   - Space-Time Multigrid
3. Domain Decomposition methods
   - Schwarz Waveform Relaxation
4. Direct methods
   - ParaExp

Space-time decomposition in ...



References

- 50 Years of Time Parallel Time Integration (2015)
  M. J. Gander
- Applications of time parallelization (2020)
  B. W. Ong and J. B. Schroder
- Multigrid methods with space-time concurrency (2017)
  R. D. Falgout, S. Friedhoff, Tz. V. Kolev, S. P. MacLachlan, J. B. Schroder, S. Vandewalle

Setting

We will solve the 1d heat equation ($c = 1$)

$$\begin{cases} u_t(x,t) = u_{xx}(x,t)\,, & (x,t) \in (0,L) \times (0,T)\,, \\ u(0,t) = u(L,t) = 0\,, & t \in (0,T)\,, \\ u(x,0) = u_0(x)\,, & x \in (0,L)\,. \end{cases}$$

We will discretize it using finite differences in space,

$$\begin{cases} v_t(t) = \frac{1}{\Delta x^2} L\, v(t)\,, & t \in (0,T) \\ v(0) = u_0\,. \end{cases}$$

Setting

We will solve the 1d heat equation ($c = 1$)

$$\begin{cases} u_t(x,t) = u_{xx}(x,t)\,, & (x,t) \in (0,L) \times (0,T)\,, \\ u(0,t) = u(L,t) = 0\,, & t \in (0,T)\,, \\ u(x,0) = u_0(x)\,, & x \in (0,L)\,. \end{cases}$$

We will discretize it using finite differences in space,

$$\begin{cases} v_t(t) = \frac{1}{\Delta x^2} L\, v(t)\,, & t \in (0,T) \\ v(0) = u_0\,. \end{cases}$$

$$L = \begin{pmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix}$$

Setting

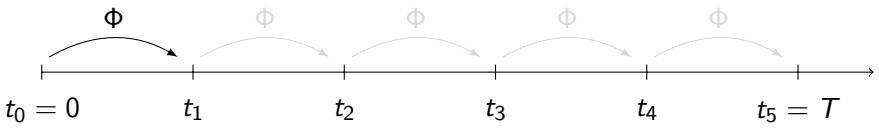
Given the problem

$$\begin{cases} v_t(t) = \frac{1}{\Delta x^2} L\, v(t)\,, & t \in (0, T) \\ v(0) = u_0 \in \mathbb{R}^{n_x}\,. \end{cases}$$

We discretize in time using a Runge-Kutta scheme

$$u_{n+1} = \Phi u_n\,, \quad u_0 = v(0) \in \mathbb{R}^{n_x}\,, \quad n = 0, 1, \ldots, N_t$$

Example: If we use Backward Euler $\Phi = (I - \frac{\Delta t}{\Delta x^2} L)^{-1}$.



Setting

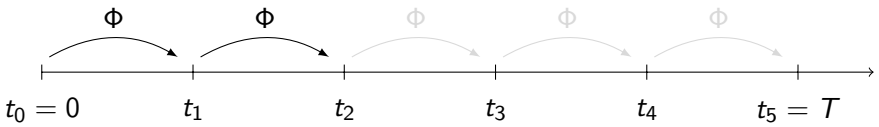
Given the problem

$$\begin{cases} v_t(t) = \frac{1}{\Delta x^2} L\, v(t)\ , & t \in (0, T) \\ v(0) = u_0 \in \mathbb{R}^{n_x}\ . \end{cases}$$

We discretize in time using a Runge-Kutta scheme

$$u_{n+1} = \Phi u_n\ , \quad u_0 = v(0) \in \mathbb{R}^{n_x}\ , \quad n = 0, 1, \ldots, N_t$$

Example: If we use Backward Euler $\Phi = (I - \frac{\Delta t}{\Delta x^2} L)^{-1}$.



Setting

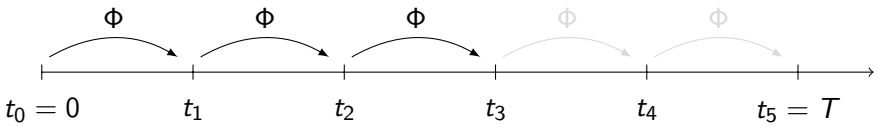
Given the problem

$$\begin{cases} v_t(t) = \frac{1}{\Delta x^2} L\, v(t)\ , & t \in (0, T) \\ v(0) = u_0 \in \mathbb{R}^{n_x}\ . \end{cases}$$

We discretize in time using a Runge-Kutta scheme

$$u_{n+1} = \Phi u_n\ , \quad u_0 = v(0) \in \mathbb{R}^{n_x}\ , \quad n = 0, 1, \ldots, N_t$$

Example: If we use Backward Euler $\Phi = (I - \frac{\Delta t}{\Delta x^2} L)^{-1}$.



Setting

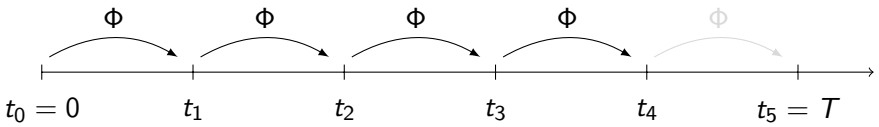
Given the problem

$$\begin{cases} v_t(t) = \frac{1}{\Delta x^2} L\, v(t)\ , & t \in (0, T) \\ v(0) = u_0 \in \mathbb{R}^{n_x}\ . \end{cases}$$

We discretize in time using a Runge-Kutta scheme

$$u_{n+1} = \Phi u_n\ , \quad u_0 = v(0) \in \mathbb{R}^{n_x}\ , \quad n = 0, 1, \ldots, N_t$$

Example: If we use Backward Euler $\Phi = (I - \frac{\Delta t}{\Delta x^2} L)^{-1}$.



Setting

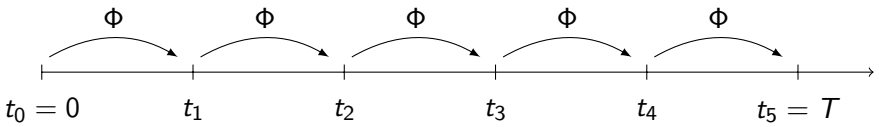
Given the problem

$$\begin{cases} v_t(t) = \frac{1}{\Delta x^2} L\, v(t)\ , & t \in (0, T) \\ v(0) = u_0 \in \mathbb{R}^{n_x}\ . \end{cases}$$

We discretize in time using a Runge-Kutta scheme

$$u_{n+1} = \Phi u_n\ , \quad u_0 = v(0) \in \mathbb{R}^{n_x}\ , \quad n = 0, 1, \ldots, N_t$$

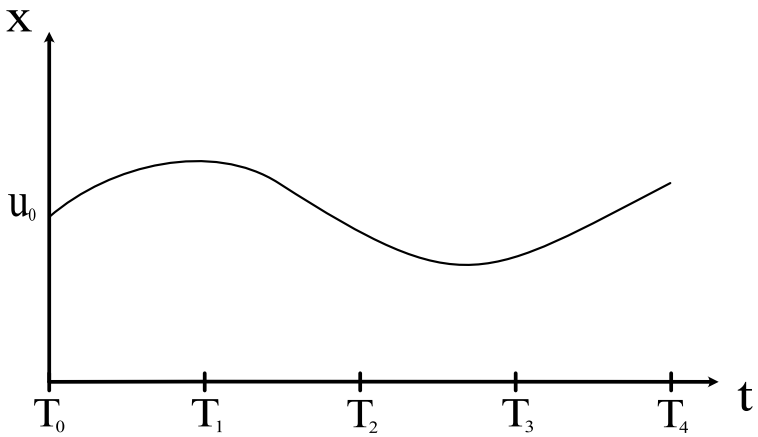
Example: If we use Backward Euler $\Phi = (I - \frac{\Delta t}{\Delta x^2} L)^{-1}$.



Parareal

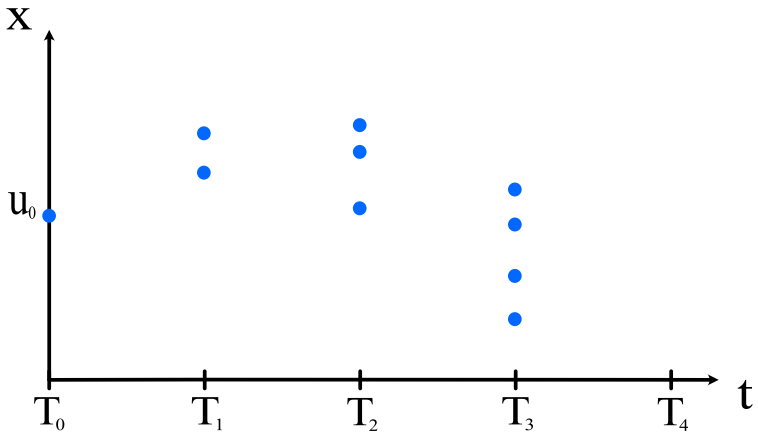
First step: Nievergelt's method (1964)

Method introduced in a 3-page paper in 1964 by J. Nievergelt.



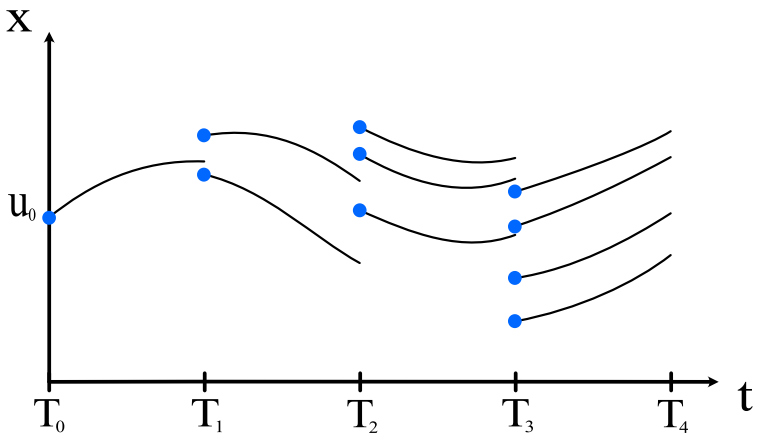
First step: Nievergelt's method (1964)

Method introduced in a 3-page paper in 1964 by J. Nievergelt.



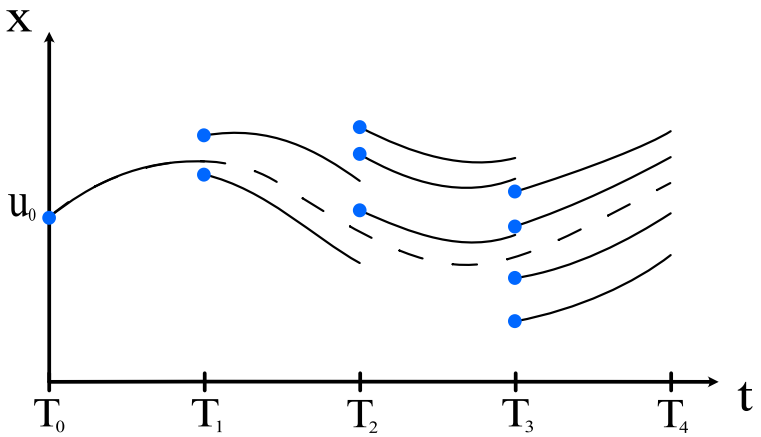
First step: Nievergelt's method (1964)

Method introduced in a 3-page paper in 1964 by J. Nievergelt.



First step: Nievergelt's method (1964)

Method introduced in a 3-page paper in 1964 by J. Nievergelt.



Developments of the method

1989 Bellen and Zenaro: solve $u_{n+1} = \Phi\, u_n$ using (a variant of) Newton's method.

$$\boldsymbol{u}^{k+1} = \varphi(\boldsymbol{u}^k) + \Delta\varphi(\boldsymbol{u}^k)(\boldsymbol{u}^{k+1} - \boldsymbol{u}^k)\,,$$

$$\text{where } \varphi(\boldsymbol{u}) = \begin{pmatrix} u_0 - v(0) \\ u_1 - \Phi\, u_0 \\ \vdots \\ u_N - \Phi\, u_{N-1} \end{pmatrix} \text{ and } \boldsymbol{u} = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{pmatrix}.$$

- Quadratic convergence
- Finite-time convergence

1993 Chartier and Philippe: Noticed that it isn't effective for all kinds of problems.

Parareal[1]

Consider a coarsening factor of m (fine/coarse grid). Let $F = \Phi^m$ and G a cheap approximation of F.

Idea: Approximate $\Delta\varphi$ by the finite difference $(G\,U_n^{k+1} - G\,U_n^k)/(U_n^{k+1} - U_n^k)$.

Initialization:

$$\begin{cases} U_0^0 = u_0 \\ U_{n+1}^0 = G\,U_n^0\,, \quad n = 0, \dots, N_t - 1\,. \end{cases}$$

Parareal iteration: for $k = 0, \dots, K-1$

$$\begin{cases} U_0^{k+1} = u_0\,, \\ U_{n+1}^{k+1} = F\,U_n^k + G\,U_n^{k+1} - G\,U_n^k\,, \quad n = 0, \dots, N_t - 1\,. \end{cases}$$

[1] Lions, Maday, Turinici (2001)

Illustration of Parareal

Initialization:

$$\begin{cases} U_0^0 = u_0 \\ U_{n+1}^0 = G\,U_n^0\,, \quad n = 0, \ldots, N_t - 1\,. \end{cases}$$

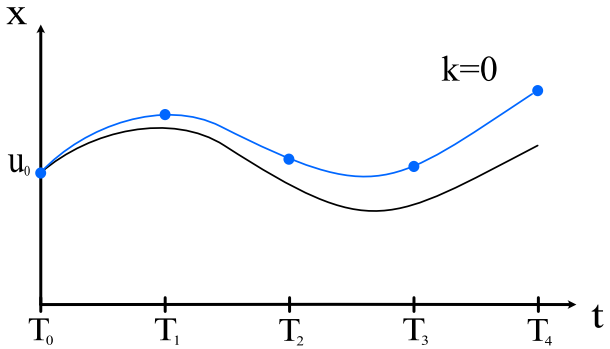


Illustration of Parareal

Parareal iteration: for $k = 0, \ldots, K-1$

$$\begin{cases} U_0^{k+1} = u_0 \ , \\ U_{n+1}^{k+1} = F\, U_n^k + G\, U_n^{k+1} - G\, U_n^k \ , \quad n = 0, \ldots, N_t - 1 \ . \end{cases}$$

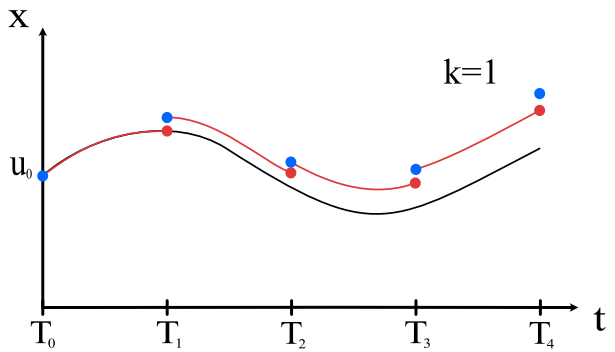
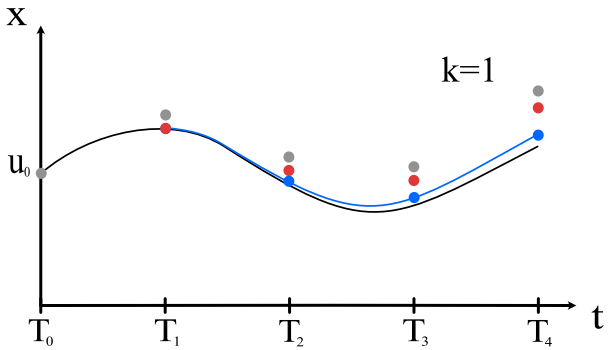


Illustration of Parareal

Parareal iteration: for $k = 0, \ldots, K-1$

$$\begin{cases} U_0^{k+1} = u_0 \ , \\ U_{n+1}^{k+1} = F\, U_n^k + G\, U_n^{k+1} - G\, U_n^k \ , \quad n = 0, \ldots, N_t - 1 \ . \end{cases}$$



Summary

- Nievergelt's method is too expensive for large problems
- Parallelism in Parareal is achieved though additional work of a coarse operator
- For Parareal to be effective, the coarse operator needs to:
  - be a good approximation of the fine
  - be much cheaper to compute

References

- Résolution d'EDP par un schéma en temps "pararéel" (2001)
  J.-L. Lions, Y. Maday, G. Turinici
- A Micro-Macro Parareal Algorithm: Application to Singularly Perturbed Ordinary Differential Equations (2013)
  F. Legoll, T. Lelièvre, G. Samaey
- PARAOPT: A Parareal Algorithm for Optimality Systems (2020)
  M. J. Gander, F. Kwok, J. Salomon
- Low-rank Parareal: a low-rank parallel-in-time integrator (2023)
  B. Carrel, M. J. Gander, B. Vandereycken

Multigrid Reduction-in-Time

All-at-once system

We are interested in solving the time stepping scheme

$$u_{n+1} = \Phi u_n \,, \quad u_0 = u_0, \quad n = 0, 1, \ldots, N_t - 1 \,.$$

This can be written as the following system to solve

$$\underbrace{\begin{pmatrix} I & & & \\ -\Phi & I & & \\ & \ddots & \ddots & \\ & & -\Phi & I \end{pmatrix}}_{=:A} \underbrace{\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_t} \end{pmatrix}}_{=:\boldsymbol{u}} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{=:\boldsymbol{b}}$$

$\rightarrow$ It can be solved using Multigrid!

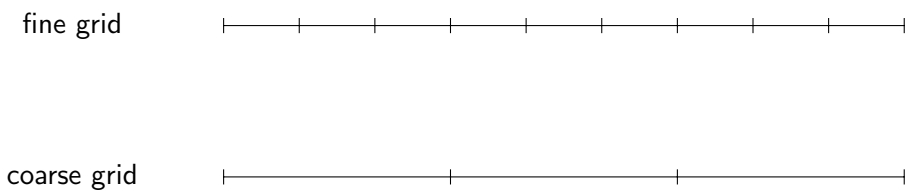
Multigrid Reduction-in-Time

The Multigrid Reduction-in-Time algorithm[2] (MGRIT) follows the same steps as a traditional Multigrid algorithm:

1. Pre-smoothing: FCF-relaxation
2. Computation of the residual and restriction: Injection
3. Coarse grid solve
4. Prolongation and correction: Ideal prolongation
5. Post-smoothing: None

[2] Falgout, Friedhoff, Kolev, MacLachlan, Schroder (2014)

Smoothing: What is FCF-relaxation?



C-points points that belong both to the coarse and fine grids

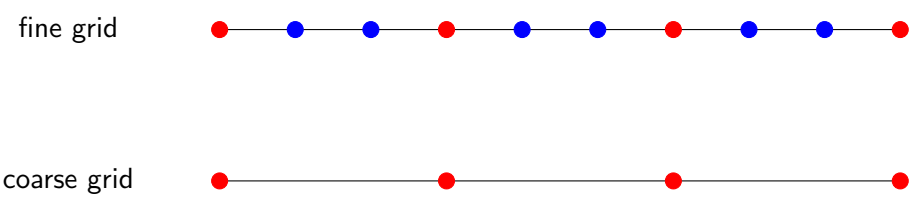
F-points the others

C-relaxation update C-points

F-relaxation update F-points

Link to Parareal: $F = \Phi^m$

Smoothing: What is FCF-relaxation?



C-points points that belong both to the coarse and fine grids

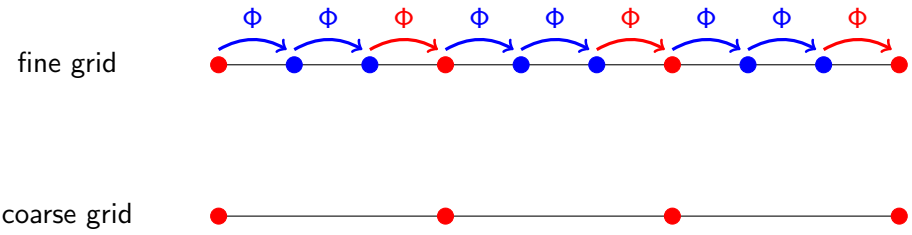
F-points the others

C-relaxation update C-points

F-relaxation update F-points

Link to Parareal: $F = \Phi^m$

Smoothing: What is FCF-relaxation?



C-points points that belong both to the coarse and fine grids

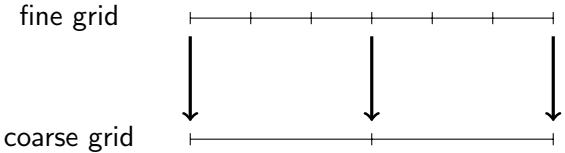
F-points the others

C-relaxation update C-points

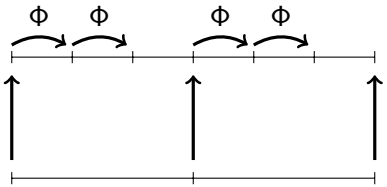
F-relaxation update F-points

Link to Parareal: $F = \Phi^m$

Transfer operators in MGRIT



Restriction by injection



Ideal prolongation

Parareal and MGRIT are related

Recall (Parareal)

$$\begin{cases} U_0^{k+1} = u_0 & k = 0, \dots, K \ , \\ U_{n+1}^{k+1} = F\, U_n^k + G\, U_n^{k+1} - G\, U_n^k & n = 0, \dots, N_t - 1, \quad k = 0, \dots, K \ . \end{cases}$$

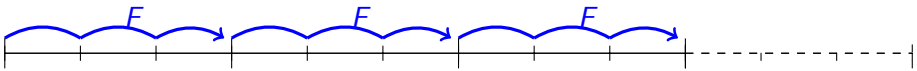
Theorem (Gander, Kwok, Zhang, 2018)

The two-level MGRIT algorithm with FCF-relaxation computes the same iterations as the Parareal algorithm using generous overlap of one coarse time interval

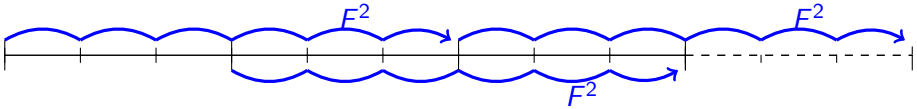
$$\begin{cases} U_0^{k+1} = u_0 & k = 0, \dots, K \ , \\ U_1^{k+1} = F\, u_0 & k = 0, \dots, K \ , \\ U_{n+1}^{k+1} = F\, F\, U_{n-1}^k + G\, U_n^{k+1} - G\, F\, U_{n-1}^k & n = 0, \dots, N_t - 1, \quad k = 0, \dots, K \ . \end{cases}$$

Parareal and MGRIT are related

Parareal (MGRIT with F-relaxation)



MGRIT with FCF-relaxation



How do Parareal and MGRIT converge?

The convergence of Parareal: from the iteration

$$U_{n+1}^{k+1} = F\, U_n^k + G\, U_n^{k+1} - G\, U_n^k, \quad n = 1, \ldots, n_t\ ,$$

the error, $\epsilon_n^k := u_n - U_n^k$, can be computed as

$$\epsilon_{n+1}^{k+1} = F\, \epsilon_n^k + G\, \epsilon_n^{k+1} - G\, \epsilon_n^k\ .$$

In turn, it can be bounded as

$$\|\epsilon_{n+1}^{k+1}\| =: e_{n+1}^{k+1} \leq \underbrace{\|F - G\|}_{=\alpha}\, e_n^k + \underbrace{\|G\|}_{=\beta}\, e_n^{k+1}\ .$$

We thus only need to solve the iteration

$$e_{n+1}^{k+1} = \alpha\, e_n^k + \beta\, e_n^{k+1}\ .$$

How do Parareal and MGRIT converge?

The convergence of Parareal: from the iteration

$$U_{n+1}^{k+1} = F\, U_n^k + G\, U_n^{k+1} - G\, U_n^k, \quad n = 1, \ldots, n_t\ ,$$

the error, $\epsilon_n^k := u_n - U_n^k$, can be computed as

$$\epsilon_{n+1}^{k+1} = F\, \epsilon_n^k + G\, \epsilon_n^{k+1} - G\, \epsilon_n^k\ .$$

In turn, it can be bounded as

$$\|\epsilon_{n+1}^{k+1}\| =: e_{n+1}^{k+1} \leq \underbrace{\|F - G\|}_{=\alpha}\, e_n^k + \underbrace{\|G\|}_{=\beta}\, e_n^{k+1}\ .$$

We thus only need to solve the iteration

$$e_{n+1}^{k+1} = \alpha\, e_n^k + \beta\, e_n^{k+1}\ .$$

How do Parareal and MGRIT converge?

The iteration $e_{n+1}^k = \alpha\, e_n^{k-1} + \beta\, e_n^k$ can be written in matrix form as

$$\begin{pmatrix} I & & & \\ -\beta & I & & \\ & \ddots & \ddots & \\ & & -\beta & I \end{pmatrix} \begin{pmatrix} e_0^k \\ e_1^k \\ \vdots \\ e_{n_t}^k \end{pmatrix} = \begin{pmatrix} 0 & & & \\ \alpha & 0 & & \\ & \ddots & \ddots & \\ & & \alpha & 0 \end{pmatrix} \begin{pmatrix} e_0^{k-1} \\ e_1^{k-1} \\ \vdots \\ e_{n_t}^{k-1} \end{pmatrix}.$$

Lemma (Recurrence solving)

Assuming that α and β are scalars, the error at step k is given by

$$\boldsymbol{e}^k = M(\beta)\,(I_{n_t} \otimes \alpha)\,\boldsymbol{e}^{k-1} = \ldots = M(\beta)^k\,(I_{n_t} \otimes \alpha^k)\,\boldsymbol{e}^0\,,$$

How do Parareal and MGRIT converge?

We then want to bound

$$\boldsymbol{e}^k = M(\beta)^k \left(I_{n_t} \otimes \alpha^k\right) \boldsymbol{e}^0 \, .$$

Linear Bound [Lemma 4.4, Gander, Vandewalle, 2007]

$$\|M(\beta)^k\|_\infty \leq \|M(\beta)\|_\infty^k = \left(\frac{1-|\beta|^{n_t}}{1-|\beta|}\right)^k .$$

Superlinear Bound [Lemma 4.3, Gander, Vandewalle, 2007]

$$\|M(\beta)^k\|_\infty = \sum_{i=0}^{n_t-k} \binom{i+k-1}{k-1} |\beta|^i = \frac{1}{(k-1)!} \sum_{i=0}^{n_t-k} \left[\prod_{l=1}^{k-1} (i+l)\right] |\beta|^i$$

M. J. Gander, S. Vandewalle. "Analysis of the Parareal Time-Parallel Time-Integration Method", 2007.

How do Parareal and MGRIT converge?

We then want to bound

$$\boldsymbol{e}^k = M(\beta)^k \left(I_{n_t} \otimes \alpha^k\right) \boldsymbol{e}^0 \, .$$

Linear Bound [Lemma 4.4, Gander, Vandewalle, 2007]

$$\|M(\beta)^k\|_\infty \leq \|M(\beta)\|_\infty^k = \left(\frac{1-|\beta|^{n_t}}{1-|\beta|}\right)^k \, .$$

Explicit Superlinear Bound [Lemma 4.4, Gander, Vandewalle, 2007]

If $|\beta| < 1$, then

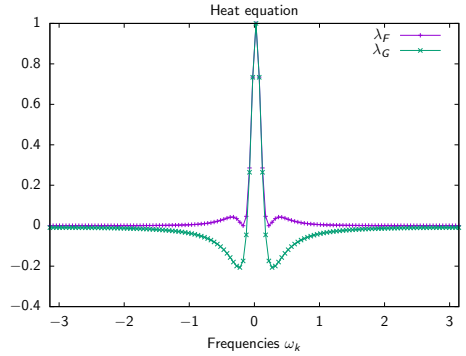
$$\|M(\beta)^k\|_\infty \leq \binom{n_t}{k} = \frac{1}{k!} \prod_{l=0}^{k-1} (n_t - l) \, .$$

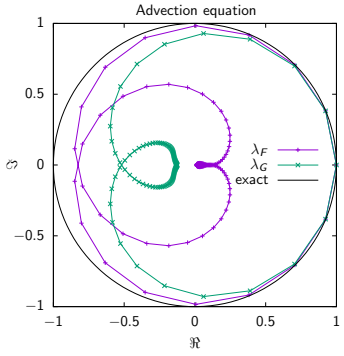
M. J. Gander, S. Vandewalle. "Analysis of the Parareal Time-Parallel Time-Integration Method", 2007.

Bound on the eigenvalues[3]

Assume λ_F and λ_G are the eigenvalues of F and G.
Let $\alpha = \lambda_F - \lambda_G$ and $\beta = \lambda_G$, then

$$\|M(\beta)(I_{n_t} \otimes \alpha)\|_\infty \leq |\lambda_F - \lambda_G| \frac{1 - |\lambda_G|^{n_t}}{1 - |\lambda_G|}$$



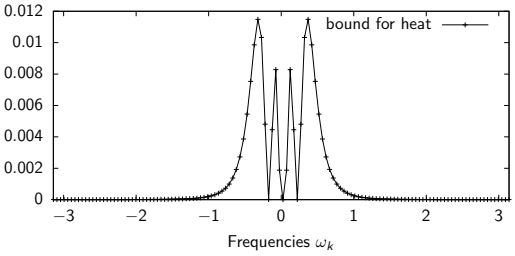


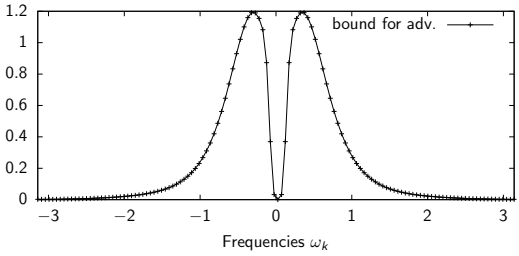
[3] Dobrev, Kolev, Petersson, Schroder (2017)

Bound on the eigenvalues[3]

Assume λ_F and λ_G are the eigenvalues of F and G.
Let $\alpha = \lambda_F - \lambda_G$ and $\beta = \lambda_G$, then

$$\|M(\beta)(I_{n_t} \otimes \alpha)\|_\infty \leq |\lambda_F - \lambda_G| \frac{1 - |\lambda_G|^{n_t}}{1 - |\lambda_G|}$$

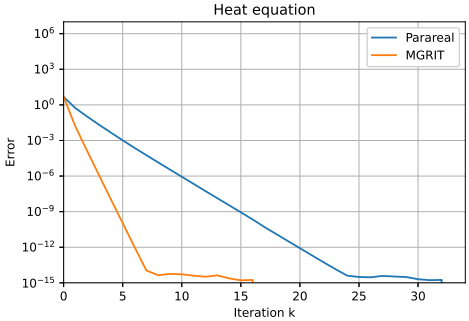


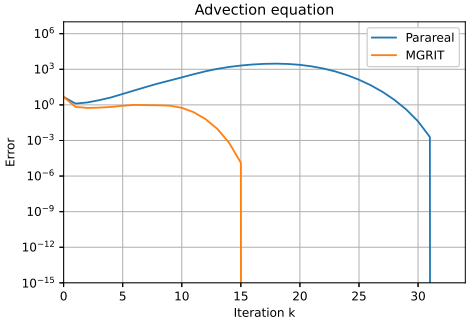


[3] Dobrev, Kolev, Petersson, Schroder (2017)

Convergence of Parareal and MGRIT

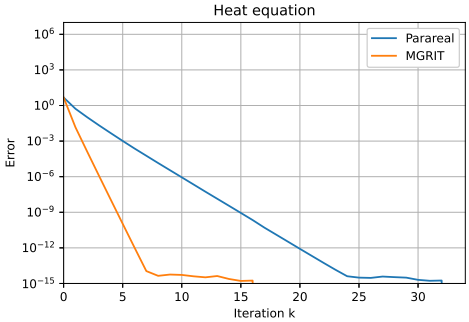
Initial guess: $u_0(x) = \sin(\pi x)$, $x \in [0, 1]$. $n_x = 100$, $n_t = 256$, $T = 0.02$ for heat and $T = 1$ for advection. SDIRK2 in time and second order centered for heat and upwind for advection.

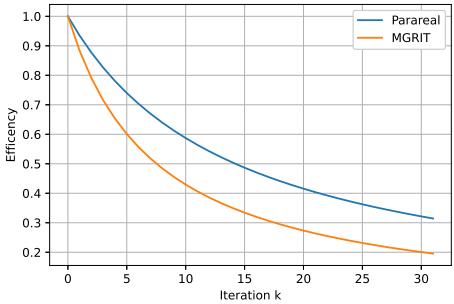




Convergence of Parareal and MGRIT

$$\text{Efficiency} = \frac{\text{Speedup}}{\#\text{ processors}} = \frac{T_{seq}}{T_{para} \cdot \#\text{ processors}}$$





Summary

- MGRIT is a multilevel version of Parareal
- MGRIT usually uses FCF-relaxation, whereas Parareal uses F-relaxation
- From superlinear bound: superlinear + finite time convergence
- From linear bound: eigenvalues that are close to unity need to be well approximated.

References

- Analysis of the Parareal Time-Parallel Time-Integration Method (2007)
  M. J. Gander, S. Vandewalle
- Parallel Time Integration with Multigrid (2014)
  R. D. Falgout, S. Friedhoff, Tz. V. Kolev, S. P. MacLachlan, J. B. Schroder
- Two-Level Convergence Theory for Multigrid Reduction in Time (MGRIT) (2017)
  V. A. Dobrev, Tz. V. Kolev, N. A. Petersson, J. B. Schroder
- Scheduling of tasks in the Parareal algorithm (2011)
  E. Aubanel
- A Unified Analysis Framework for Iterative Parallel-in-Time Algorithms (2022)
  M. J. Gander, T. Lunet, D. Ruprecht, R. Speck

Space-Time Multigrid

All-at-once system

The time-stepping procedure

$$u_{n+1} = \Phi u_n \,, \quad u_0 = u_0 \,, \quad n = 0, 1, \ldots, N_t - 1 \,,$$

was formulated as a all-at-once system for MGRIT. For the Space-Time algorithm, we consider a different formulation.

Separate the operator $\Phi = Q^{-1}P$ where Q and P are the implicit and explicit parts of Φ. In turn, it can be written as the all-at-once system,

$$\underbrace{\begin{pmatrix} Q & & & \\ -P & Q & & \\ & \ddots & \ddots & \\ & & -P & Q \end{pmatrix}}_{=:A} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N_t} \end{pmatrix}}_{=:\boldsymbol{u}} = \underbrace{\begin{pmatrix} P\, u_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{=:\boldsymbol{b}} .$$

Space-Time Multigrid with point-wise smoother[4]

Idea: Apply Multigrid on the space-time problem.

1. Pre-smoothing: ν_1 damped Jacobi iterations
2. Computation of the residual and restriction: Full-weighting
3. Coarse grid solve
4. Prolongation and correction: Linear interpolation
5. Post-smoothing: ν_2 damped Jacobi iterations

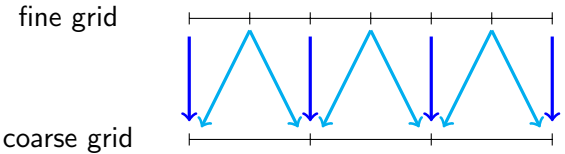
Reminder: The point-wise recall the damped Jacobi smoother is given by

$$\boldsymbol{v}^{k+1} = \boldsymbol{v}^k + \omega D^{-1}[\boldsymbol{b} - A\boldsymbol{v}^k]$$

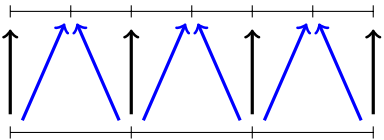
where D is the diagonal matrix such that $D_{ii} = A_{ii}$.

[4] Horton, Vandewalle (1995)

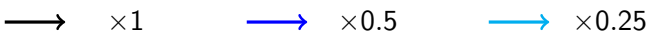
Transfer operators in STMG



Restriction by full-weighting



Linear interpolation



Space-Time Multigrid with point-wise smoother

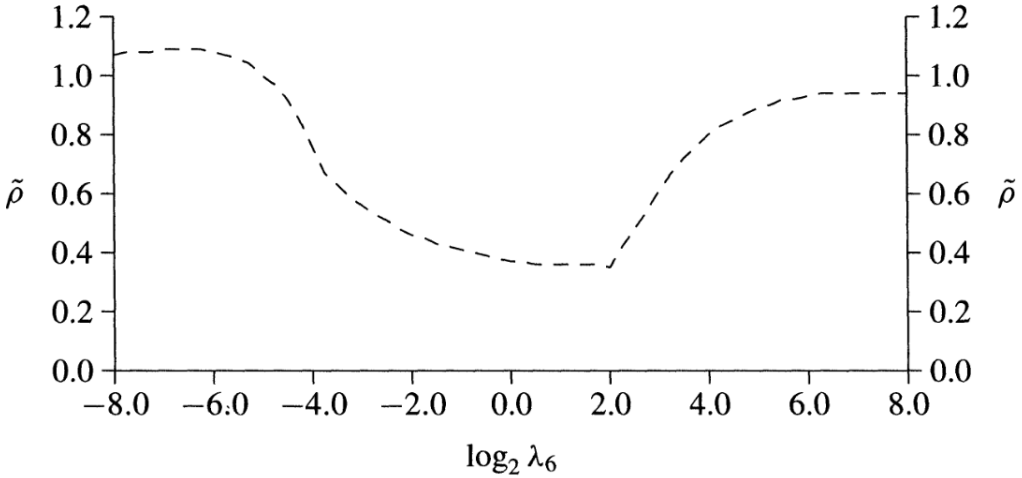


Figure: Convergence factor of STMG with point-wise smoother from Horton, Vandewalle (1995)

STMG with block Jacobi smoother[5]

Goal: Be able to always smooth in the time dimension.

Consider the damped block-Jacobi smoother

$$\boldsymbol{v}^{k+1} = \boldsymbol{v}^k + \omega D^{-1}[\boldsymbol{b} - A\boldsymbol{v}^k] \,,$$

with D a block diagonal matrix with blocks Q and $\omega \in [0, 2]$ a damping parameter.

$\rightarrow$ Always allows for coarsening in time

[5] Gander, Neumüller (2016)

STMG has good scaling properties

Weak and Strong scaling results: 3D Heat Equation

	Weak scaling					Strong scaling			
cores	n_t	dof	iter	time	fwd. subs.	n_t	dof	iter	time
1	2	59,768	7	28.8	19.0	512	15,300,608	7	7,635.2
2	4	119,536	7	29.8	37.9	512	15,300,608	7	3,821.7
4	8	239,072	7	29.8	75.9	512	15,300,608	7	1,909.9
8	16	478,144	7	29.9	152.2	512	15,300,608	7	954.2
16	32	956,288	7	29.9	305.4	512	15,300,608	7	477.2
32	64	1,912,576	7	29.9	613.6	512	15,300,608	7	238.9
64	128	3,825,152	7	29.9	1,220.7	512	15,300,608	7	119.5
128	256	7,650,304	7	29.9	2,448.4	512	15,300,608	7	59.7
256	512	15,300,608	7	30.0	4,882.4	512	15,300,608	7	30.0

Table: Vulcan BlueGene /Q Supercomputer in Livermore (Martin Neumüller)

STMG has good scaling properties

Weak and Strong scaling results: 3D Heat Equation

	Weak scaling					Strong scaling			
cores	n_t	dof	iter	time	fwd. subs.	n_t	dof	iter	time
1	2	59,768	7	28.8	19.0	512	15,300,608	7	7,635.2
2	4	119,536	7	29.8	37.9	512	15,300,608	7	3,821.7
4	8	239,072	7	29.8	75.9	512	15,300,608	7	1,909.9
8	16	478,144	7	29.9	152.2	512	15,300,608	7	954.2
16	32	956,288	7	29.9	305.4	512	15,300,608	7	477.2
32	64	1,912,576	7	29.9	613.6	512	15,300,608	7	238.9
64	128	3,825,152	7	29.9	1,220.7	512	15,300,608	7	119.5
128	256	7,650,304	7	29.9	2,448.4	512	15,300,608	7	59.7
256	512	15,300,608	7	30.0	4,882.4	512	15,300,608	7	30.0

Table: Vulcan BlueGene /Q Supercomputer in Livermore (Martin Neumüller)

STMG has good scaling properties

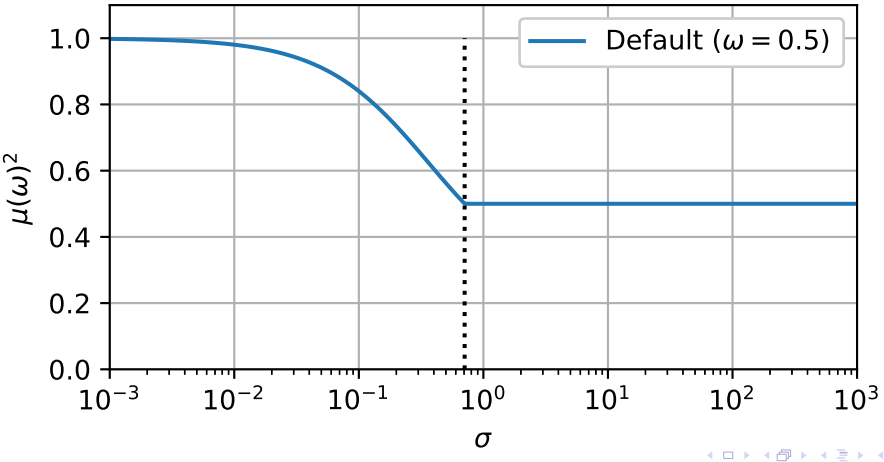
Weak and Strong scaling results: 3D Heat Equation

	Weak scaling					Strong scaling			
cores	n_t	dof	iter	time	fwd. subs.	n_t	dof	iter	time
1	2	59,768	7	28.8	19.0	512	15,300,608	7	7,635.2
2	4	119,536	7	29.8	37.9	512	15,300,608	7	3,821.7
4	8	239,072	7	29.8	75.9	512	15,300,608	7	1,909.9
8	16	478,144	7	29.9	152.2	512	15,300,608	7	954.2
16	32	956,288	7	29.9	305.4	512	15,300,608	7	477.2
32	64	1,912,576	7	29.9	613.6	512	15,300,608	7	238.9
64	128	3,825,152	7	29.9	1,220.7	512	15,300,608	7	119.5
128	256	7,650,304	7	29.9	2,448.4	512	15,300,608	7	59.7
256	512	15,300,608	7	30.0	4,882.4	512	15,300,608	7	30.0

Table: Vulcan BlueGene /Q Supercomputer in Livermore (Martin Neumüller)

The convergence of STMG depends on the ratio $\sigma = \Delta t/\Delta x^2$

Let μ be the smoothing factor associated to the block-Jacobi smoother obtained by Local Fourier Analysis (LFA),



Summary

- STMG has very good scaling properties
- The convergence of STMG depends on the ratio $\Delta t/\Delta x^2$
- STMG with block-smoothing allows to always coarsen in time

References

- A Space-Time Multigrid for Parabolic Partial Differential Equations (1995)
  G. Horton, S. Vandewalle
- Analysis of a New Space-Time Parallel Multigrid Algorithm for Parabolic Problems (2016)
  M. J. Gander, M. Neumüller
- An Optimized Space-Time Multigrid Algorithm for Parabolic PDEs (2023, in review)
  B. Chaudet-Dumas, M. J. Gander, A. P.

Schwarz Waveform Relaxation

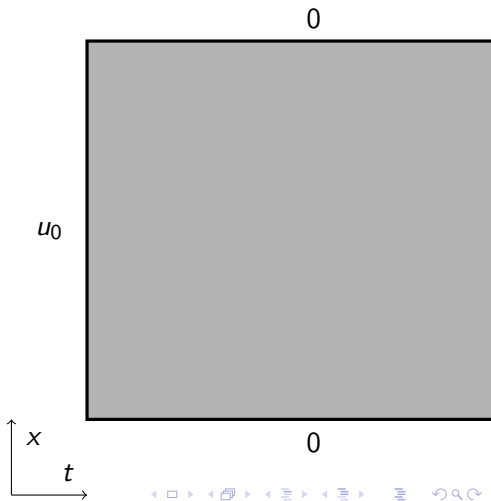
Schwarz Waveform Relaxation

We will solve the 1D heat equation

$$\begin{cases} u_t(x,t) = u_{xx}(x,t)\,, & (x,t) \in \Omega \times (0,T)\,, \\ u(x,t) = 0\,, & x \in \partial\Omega,\ t \in (0,T)\,, \\ u(x,0) = u_0(x)\,, & x \in \Omega\,. \end{cases}$$

Partition the domain $\Omega = \Omega_1 \cup \Omega_2$.
Define for $i \in \{0,1\}$,

$$\begin{cases} \partial_t u_i^{k+1}(x,t) = \partial_{xx} u_i^{k+1}(x,t) & (x,t) \in \Omega_1 \times (0,T] \\ u_i^{k+1}(x,0) = u_0(x) & x \in \Omega_i \\ u_i^{k+1}(x,t) = 0 & x \in \partial\Omega,\ t \in (0,T] \\ u_i^{k+1}(\Gamma_i,t) = u_{1-i}^{k}(\Gamma_i,t) & t \in (0,T] \end{cases}$$



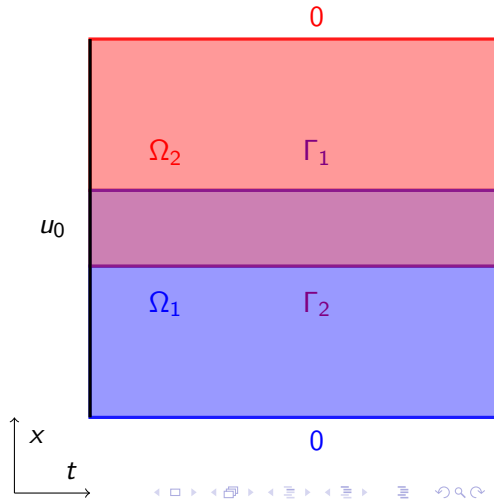
Schwarz Waveform Relaxation

We will solve the 1D heat equation

$$\begin{cases} u_t(x,t) = u_{xx}(x,t)\,, & (x,t) \in \Omega \times (0,T)\,, \\ u(x,t) = 0\,, & x \in \partial\Omega,\ t \in (0,T)\,, \\ u(x,0) = u_0(x)\,, & x \in \Omega\,. \end{cases}$$

Partition the domain $\Omega = \Omega_1 \cup \Omega_2$.
Define for $i \in \{0,1\}$,

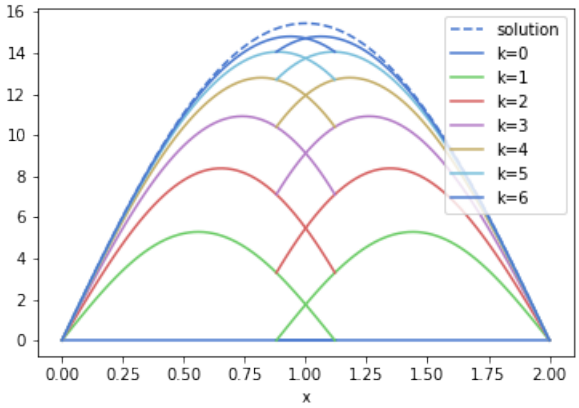
$$\begin{cases} \partial_t u_i^{k+1}(x,t) = \partial_{xx} u_i^{k+1}(x,t) & (x,t) \in \Omega_1 \times (0,T] \\ u_i^{k+1}(x,0) = u_0(x) & x \in \Omega_i \\ u_i^{k+1}(x,t) = 0 & x \in \partial\Omega,\ t \in (0,T] \\ u_i^{k+1}(\Gamma_i,t) = u_{1-i}^{k}(\Gamma_i,t) & t \in (0,T] \end{cases}$$



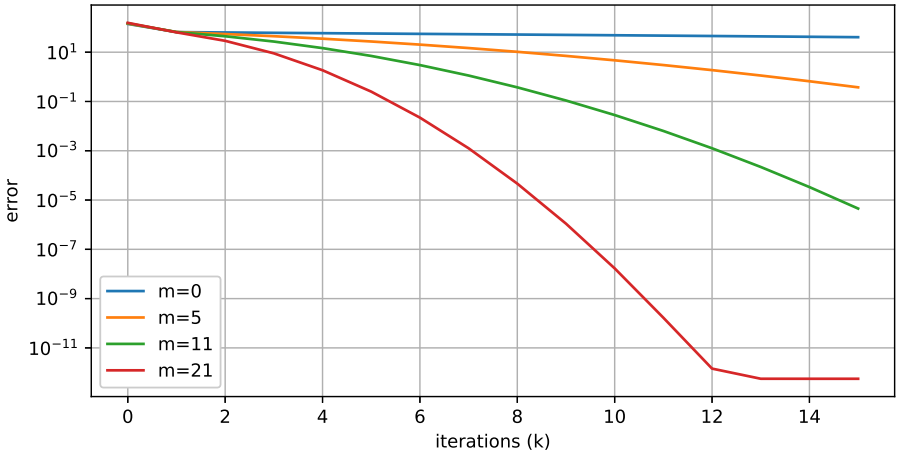
Convergence of SWR

Goal: Compute the solution for the 1D heat equation with Dirichlet boundary conditions with $u_0 = 20$ and initial guess $u_0 = 0$.

On the right, first 6 iterations of SWR at time $T = 0.5$.



Convergence is superlinear[6]



[6] M. J. Gander (1997, PhD thesis)

Optimized Schwarz Waveform Relaxation

Recall: Schwarz wave relaxation is given for $i \in \{0, 1\}$ by

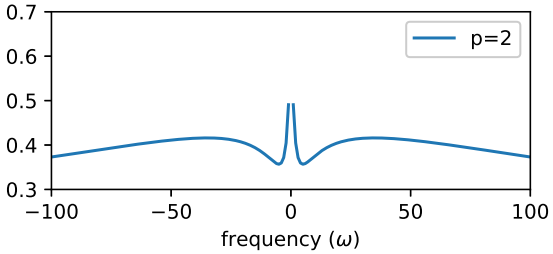
$$\begin{cases} \partial_t u_i^{k+1}(x,t) = \partial_{xx} u_i^{k+1}(x,t) & (x,t) \in \Omega_1 \times (0,T] \\ u_i^{k+1}(x,0) = u_0(x) & x \in \Omega_i \\ u_i^{k+1}(x,t) = 0 & x \in \partial\Omega,\ t \in (0,T] \\ u_i^{k+1}(\Gamma_i,t) = u_{1-i}^{k}(\Gamma_i,t) & t \in (0,T] \end{cases}$$

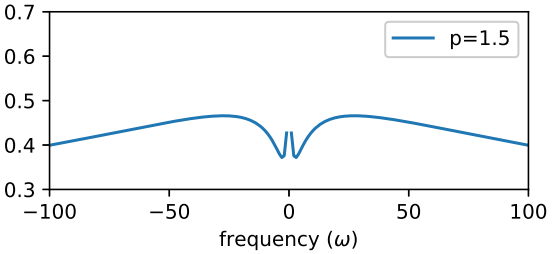
Goal: Do SWR without overlap.

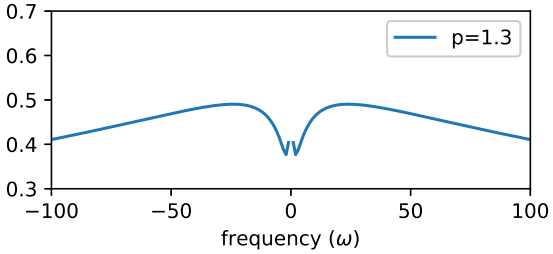
$$\begin{cases} \partial_t u_i^{k+1}(x,t) = \partial_{xx} u_i^{k+1}(x,t) & (x,t) \in \Omega_1 \times (0,T] \\ u_i^{k+1}(x,0) = u_0(x) & x \in \Omega_i \\ u_i^{k+1}(x,t) = 0 & x \in \partial\Omega,\ t \in (0,T] \\ (\partial_x + p) u_i^{k+1}(\Gamma_i,t) = (\partial_x + p) u_{1-i}^{k}(\Gamma_i,t) & t \in (0,T] \end{cases}$$

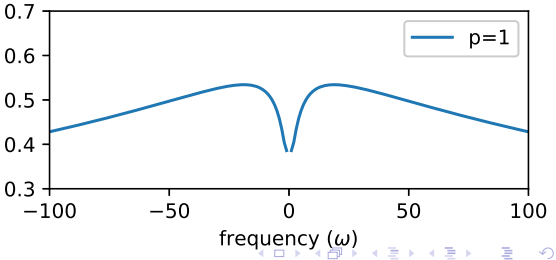
Solution: Change transmission conditions to Robin type and optimize for p.

Optimizing Optimized Schwarz Waveform Relaxation









Summary

- Schwarz Wave Relaxation allows parallelism in time by cutting the spatial domain in two (or more) subdomains
- Schwarz Wave Relaxation has superlinear convergence
- Optimized Schwarz Waveform Relaxation allows to have no overlap between domains.

References

- The waveform relaxation method for time-domain analysis of large scale integrated circuits (1983)
  E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli.
- Overlapping Schwarz of linear and non-linear Parabolic problems (1996)
  M. J. Gander
- Dirichlet-Neumann and Neumann-Neumann waveform relaxation algorithms for parabolic problems (2013)
  M. J. Gander, F. Kwok, B. C. Mandal
- Work by M. Gander, L. Halpern, V. Martin, F. Nataf.

ParaExp

Introduced by Gander and Güttel in 2013. Let $A \in \mathbb{R}^{n \times n}$ and f a non-linear function.

$$u'(t) = A\,u(t) + f(t), \quad t \in (0, T), \quad u(0) = u_0\ .$$

Observation

A homogenous problem can be integrated much faster than an inhomogenous problem.

Separate $u = v + w$, where

$$v'(t) = A\,v(t), \quad v(0) = u_0\ ,$$

$$w'(t) = A\,w(t) + f(t), \quad w(0) = 0\ .$$

ParaExp — illustration

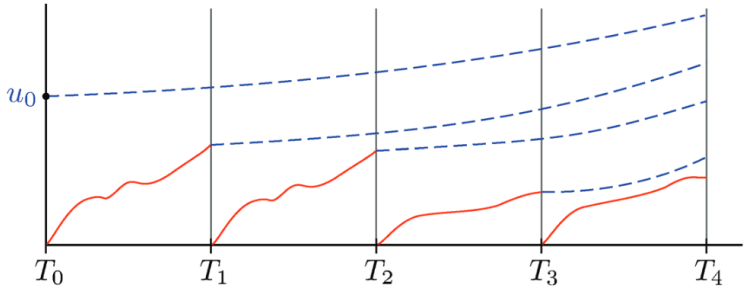


Figure: Illustration of ParaExp from the original article (Gander, Güttel, 2013)

1. On $[T_n, T_{n+1}]$: Solve with serial integrator
2. On $[T_n, T]$: Solve with near optimal exponential integrator
3. Get the final solution by superposition of solutions

Fast computation of matrix exponentials

Question: If A is very big, how to compute efficiently $\exp(t\,A)$?

Idea: Compute $\exp(t\,A)\,v$ directly for some initial condition v using Krylov methods.

1. Generate the Krylov space $\mathcal{K}_m = \text{span}\{v, Av, \ldots, A^{m-1}v\}$.
   - Use the Arnoldi method.
2. Get an orthonormal basis V_m of $\mathcal{K}_m$.
   - Use a modified Gram-Schmidt process.
3. Compute H_m the projection of A on the space $\mathcal{K}_m$ with respect to the basis V_m.
   - The matrix H_m is upper Hessenberg and verifies $H_m = V_m^\top A V_m$.
4. Get the approximation $e^{tA} \approx \beta V_m e^{tH_m} e_1$ (good even for small m)
   - Apply traditional techniques to compute $e^{tH_m} e_1$.

Fast computation of matrix exponentials

Question: If A is very big, how to compute efficiently $\exp(t\,A)$?

Idea: Compute $\exp(t\,A)\,v$ directly for some initial condition v using Krylov methods.

1. Generate the Krylov space $\mathcal{K}_m = \text{span}\{v, Av, \ldots, A^{m-1}v\}$.
   - Use the Arnoldi method.
2. Get an orthonormal basis V_m of $\mathcal{K}_m$.
   - Use a modified Gram-Schmidt process.
3. Compute H_m the projection of A on the space $\mathcal{K}_m$ with respect to the basis V_m.
   - The matrix H_m is upper Hessenberg and verifies $H_m = V_m^\top A V_m$.
4. Get the approximation $e^{tA} \approx \beta V_m e^{tH_m} e_1$ (good even for small m)
   - Apply traditional techniques to compute $e^{tH_m} e_1$.

What about non-linear problems?

Consider the non-linear problem, with $B : \mathbb{R}^n \to \mathbb{R}^n$ a non-linear application,

$$u'(t) = A\, u(t) + B(u(t)) + f(t),\ t \in [0, T], \qquad u(0) = u_0\ .$$

Problem: Applying the splitting $u = v + w$ does not verify the original equation.

$$\begin{aligned} v'(t) &= A\, v(t), & v(0) &= u_0\ , \\ w'(t) &= A\, w(t) + B(w(t)) + f(t), & w(0) &= 0\ . \end{aligned}$$

Non-linear ParaExp

Consider the following iterative method: for $k = 0$ initialize

$$u_n^0 = w_n^0 = 0, \qquad n = 0, \dots, N_t - 1 .$$

For $k = 1, 2, \dots$ solve the homogenous problems

$$(w_n^k)'(t) = A\, w_n^k(t) \qquad t \in [T_{n-1}, T_n]$$

$$w_1^k(T_0) = u_0, \;\; w_n^k(0) = u_{n-1}^{k-1}(0) - \sum_{j=1}^{n-1} w_j^{k-1}(0)$$

and then solve the non-homogenous problem

$$(u_n^k)'(t) = A\, u_n^k(t) + B(u_n^k(t)) + f(t), \qquad t \in [T_{n-1}, T_n]$$

$$u_n^k(0) = \sum_{j=1}^{n} w_j^k(T_{n-1})$$

Non-linear ParaExp equivalence with Parareal

Theorem [Theorem 2, Gander, Güttel, Petcu, 2018]

Let the coarse propagator $G(U_n)$ solve the linear problem

$$u'(t) = A\,u(t), \quad t \in (T_n, T_{n+1}), \quad u(0) = U_n$$

and the fine propagator $F(U_n)$ solve the non-linear problem

$$u'(t) = A\,u(t) + B(u(t)) + f(t), \quad t \in (T_n, T_{n+1}), \quad u(0) = U_n$$

Then the Parareal iteration computed with those operators is equivalent to the non-linear ParaExp algorithm.

Summary

- Direct method for linear problems
- Acceleration of computation of the exponential using Krylov methods
- Iterative method for non-linear problems
- Equivalence of non-linear ParaExp with Parareal

References

- ParaExp: a Parallel Integrator for Linear Initial-Value Problems (2013)
  M. J. Gander, S. Güttel
- A Nonlinear ParaExp Algorithm (2018)
  M. J. Gander, S. Güttel, M. Petcu
- Analysis of Some Krylov Subspace Approximations to the Matrix Exponential (1992)
  Y. Saad