Why Parallel in Time (PinT) methods are different for Parabolic and Hyperbolic Problems

Martin J. Gander
martin.gander@unige.ch

University of Geneva

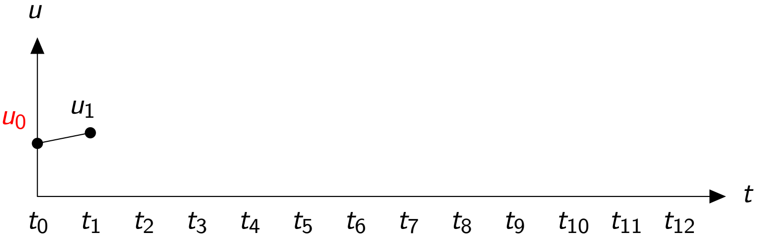
Pisa, Italy, April 3rd, 2024

Time Parallelization and the Causality Principle

The time direction is special for parallelization, because of the *causality principle*: the solution later in time is determined by the solution earlier in time, and never the other way round.

Example: $\frac{du}{dt} = f(u)$, $u(t_0) = u_0$, Euler: $\frac{du}{dt} \approx \frac{u(t_{n+1}) - u(t_n)}{\Delta t}$

$$u_1 = u_0 + \Delta t f(u_0)$$

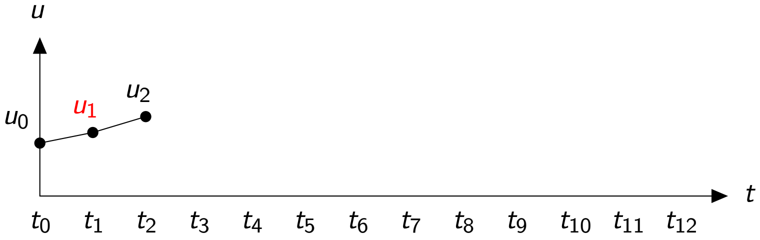


Time Parallelization and the Causality Principle

The time direction is special for parallelization, because of the *causality principle*: the solution later in time is determined by the solution earlier in time, and never the other way round.

Example: $\frac{du}{dt} = f(u)$, $u(t_0) = u_0$, Euler: $\frac{du}{dt} \approx \frac{u(t_{n+1}) - u(t_n)}{\Delta t}$

$$u_2 = u_1 + \Delta t f(u_1)$$

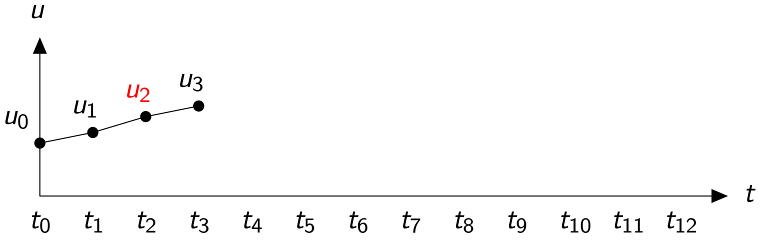


Time Parallelization and the Causality Principle

The time direction is special for parallelization, because of the *causality principle*: the solution later in time is determined by the solution earlier in time, and never the other way round.

Example: $\frac{du}{dt} = f(u)$, $u(t_0) = u_0$, Euler: $\frac{du}{dt} \approx \frac{u(t_{n+1}) - u(t_n)}{\Delta t}$

$$u_3 = u_2 + \Delta t f(u_2)$$

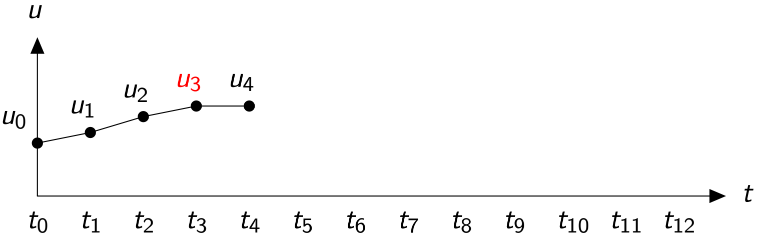


Time Parallelization and the Causality Principle

The time direction is special for parallelization, because of the *causality principle*: the solution later in time is determined by the solution earlier in time, and never the other way round.

Example: $\frac{du}{dt} = f(u)$, $u(t_0) = u_0$, Euler: $\frac{du}{dt} \approx \frac{u(t_{n+1}) - u(t_n)}{\Delta t}$

$$u_4 = u_3 + \Delta t f(u_3)$$

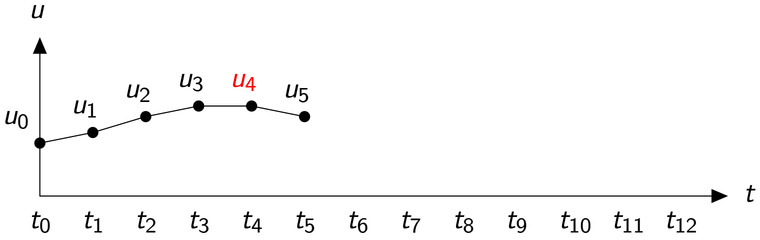


Time Parallelization and the Causality Principle

The time direction is special for parallelization, because of the *causality principle*: the solution later in time is determined by the solution earlier in time, and never the other way round.

Example: $\frac{du}{dt} = f(u)$, $u(t_0) = u_0$, Euler: $\frac{du}{dt} \approx \frac{u(t_{n+1})-u(t_n)}{\Delta t}$

$$u_5 = u_4 + \Delta t f(u_4)$$

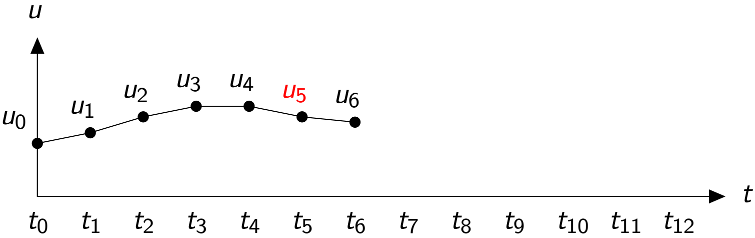


Time Parallelization and the Causality Principle

The time direction is special for parallelization, because of the *causality principle*: the solution later in time is determined by the solution earlier in time, and never the other way round.

Example: $\frac{du}{dt} = f(u)$, $u(t_0) = u_0$, Euler: $\frac{du}{dt} \approx \frac{u(t_{n+1}) - u(t_n)}{\Delta t}$

$$u_6 = u_5 + \Delta t f(u_5)$$

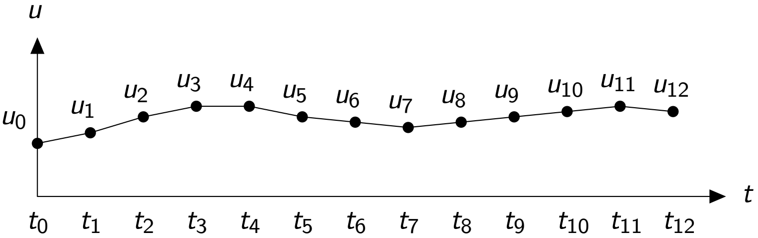


Time Parallelization and the Causality Principle

The time direction is special for parallelization, because of the *causality principle*: the solution later in time is determined by the solution earlier in time, and never the other way round.

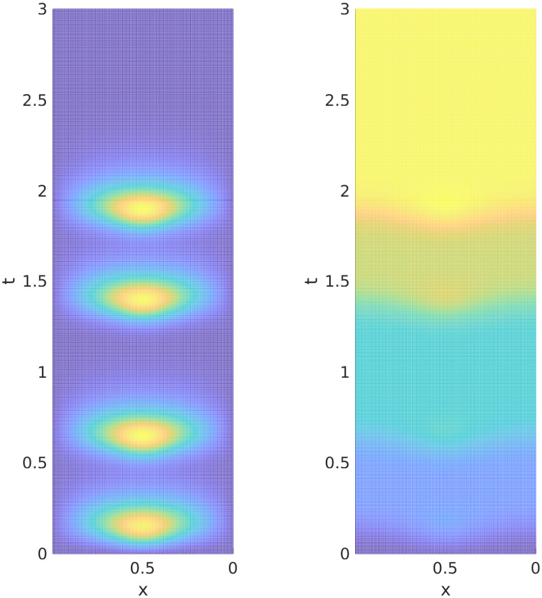
Example: $\frac{du}{dt} = f(u)$, $u(t_0) = u_0$, Euler: $\frac{du}{dt} \approx \frac{u(t_{n+1}) - u(t_n)}{\Delta t}$

$$u_{n+1} = u_n + \Delta t f(u_n)$$



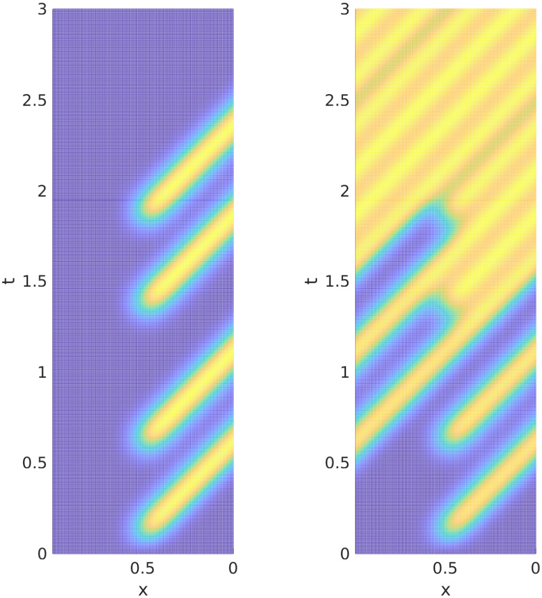
Heat Equation: Dirichlet and Neumann Conditions

$$u_t = u_{xx} + f,\ u(0,t) = u(1,t) = 0 \text{ and } u_x(0,t) = u_x(1,t) = 0$$



Transport Equation: Dirichlet and Periodic

$u_t + u_x = f$, $u(0,t) = 0$ and $u(0,t) = u(1,t)$



Parabolic PinT: the Parareal Algorithm

For solving the evolution problem

$$\begin{aligned} \partial_t \boldsymbol{u}(t) &= \boldsymbol{f}(t, \boldsymbol{u}(t)) \quad t \in (0, T], \\ \boldsymbol{u}(0) &= \boldsymbol{u}^0, \end{aligned}$$

Parareal needs two propagation operators:

1. $\boldsymbol{G}(t_2, t_1, \boldsymbol{u}_1)$ is a coarse approximation to the solution $\boldsymbol{u}(t_2)$ with initial condition $\boldsymbol{u}(t_1) = \boldsymbol{u}_1$,
2. $\boldsymbol{F}(t_2, t_1, \boldsymbol{u}_1)$ is a more accurate approximation of the solution $\boldsymbol{u}(t_2)$ with initial condition $\boldsymbol{u}(t_1) = \boldsymbol{u}_1$.

The time interval $(0, T]$ is partitioned into subintervals $(T_{n-1}, T_n]$. Parareal then starts with an initial coarse approximation $\boldsymbol{U}_n^0$ at $T_0, T_1, \ldots, T_N$, and then computes

$$\begin{aligned} \boldsymbol{U}_0^{k+1} &:= \boldsymbol{u}^0, \\ \boldsymbol{U}_{n+1}^{k+1} &:= \boldsymbol{F}(T_{n+1}, T_n, \boldsymbol{U}_n^k) + \boldsymbol{G}(T_{n+1}, T_n, \boldsymbol{U}_n^{k+1}) - \boldsymbol{G}(T_{n+1}, T_n, \boldsymbol{U}_n^k) \end{aligned}$$

Lions, Maday, Turinici (2001): Résolution d'EDP par un schéma en temps "pararéel"

Theorem (Heat Equation (G, Vandewalle 2007))

Let F be exact, G have stability function R_G with $\rho_s := sup_{x<0}|e^x - R_G(x)|$ finite. Then

$$\max_{1\le n\le N} ||u(t_n) - U_n^k||_2 \le \frac{\rho_s^k}{k!}\prod_{\ell=1}^{k}(N-\ell)C_0^N,$$

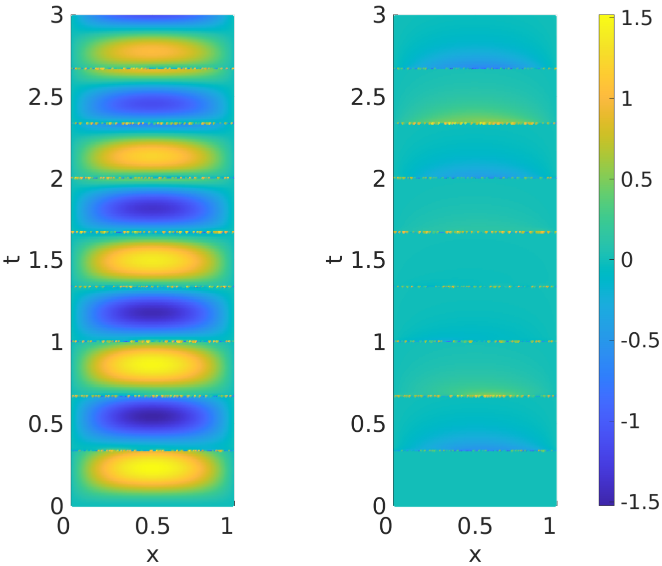
$$C_0^N := \sqrt{\sum_{\omega=1}^{\infty}\max_{1\le n\le N}|\hat{u}(T_n,\omega) - \hat{U}_n^0(\omega)|^2}.$$

If the negative real axis is in the region of absolute stability of G and $\lim_{x\to-\infty}|R_G(x)| < 1$ (A_0-stability), then

$$\sup_{n>0}||u(t_n) - U_n^k||_2 \le \rho_l^k C_0^\infty,$$

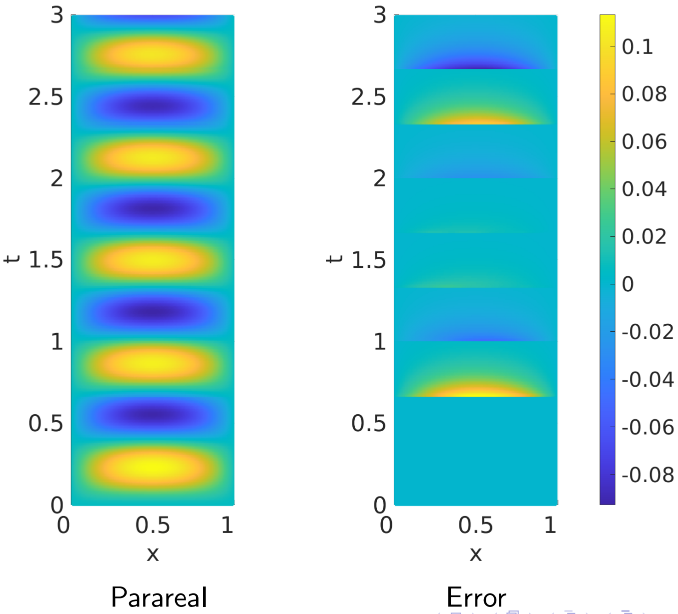
$$\rho_l := \sup_{\omega\in\mathbb{R}}\frac{|e^{-\omega^2\Delta T} - R_G(-\omega^2\Delta T)|}{1-|R_G(-\omega^2\Delta T|}, \quad T_n - T_{n-1} \equiv \Delta T.$$

Heat example: Iteration 1

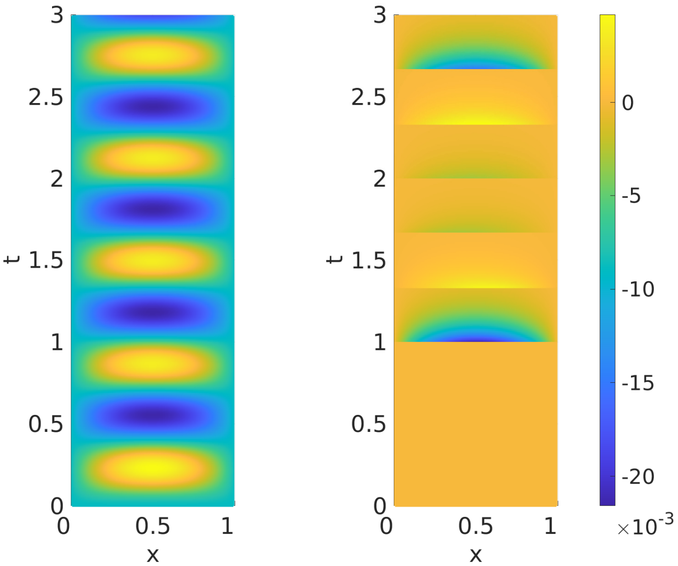


Parareal Error

Heat example: Iteration 2



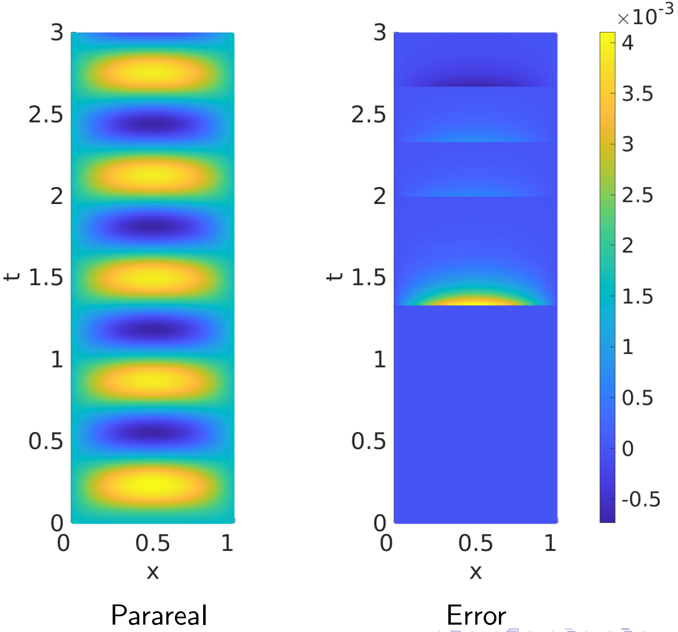
Heat example: Iteration 3



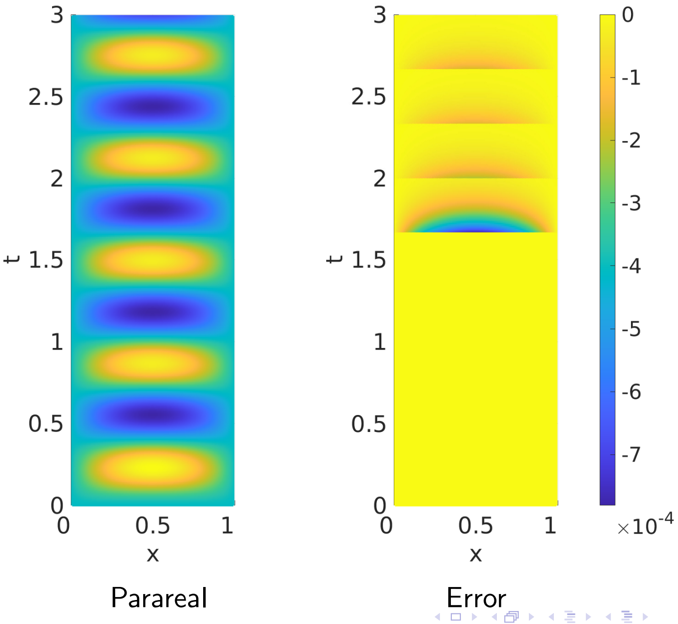
Parareal

Error

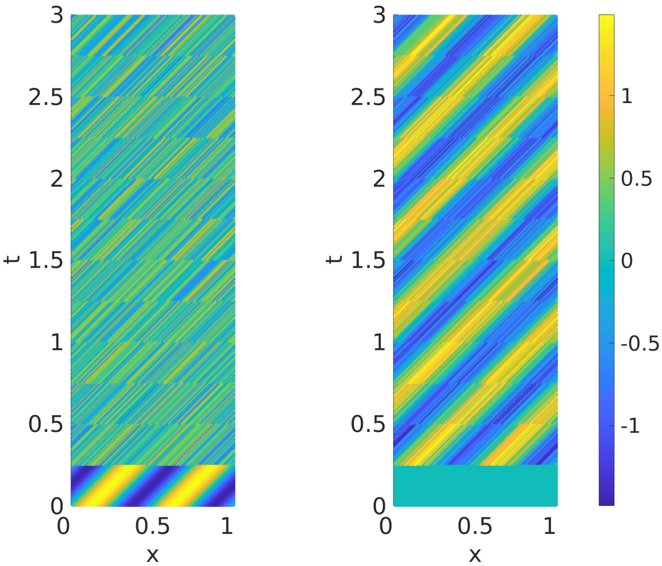
Heat example: Iteration 4



Heat example: Iteration 5



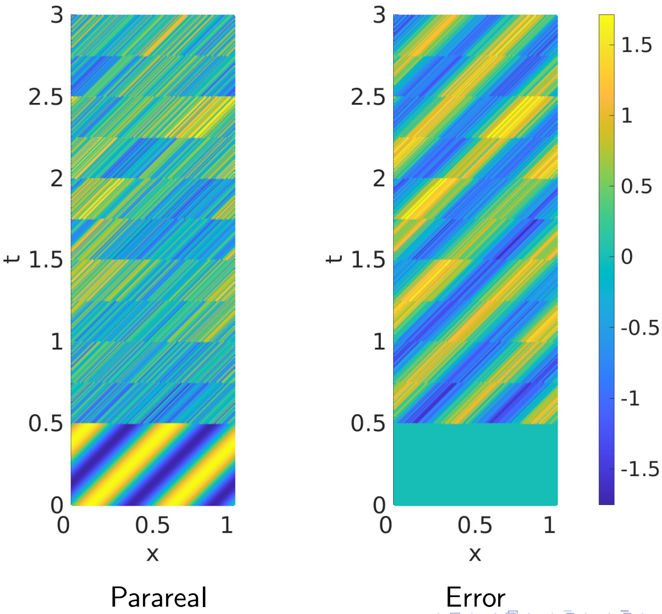
Advection example: Iteration 1



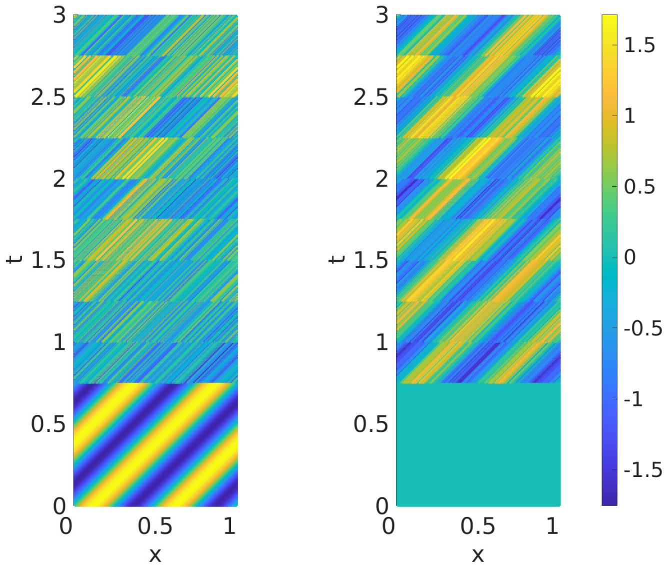
Parareal

Error

Advection example: Iteration 2



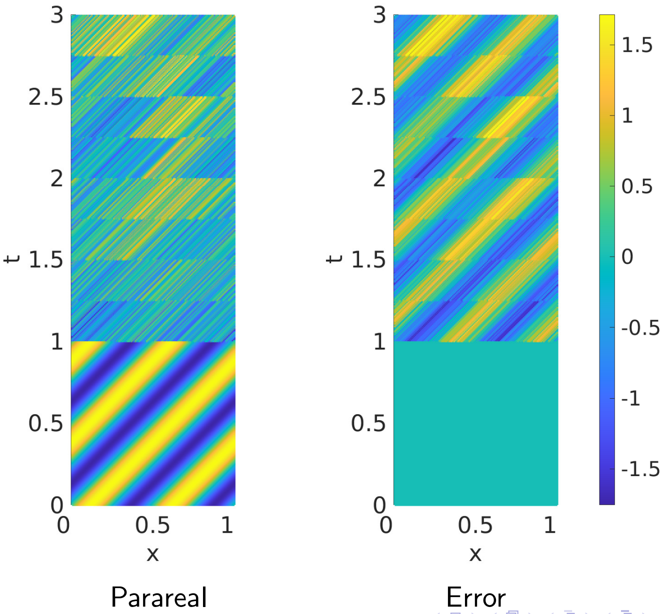
Advection example: Iteration 3



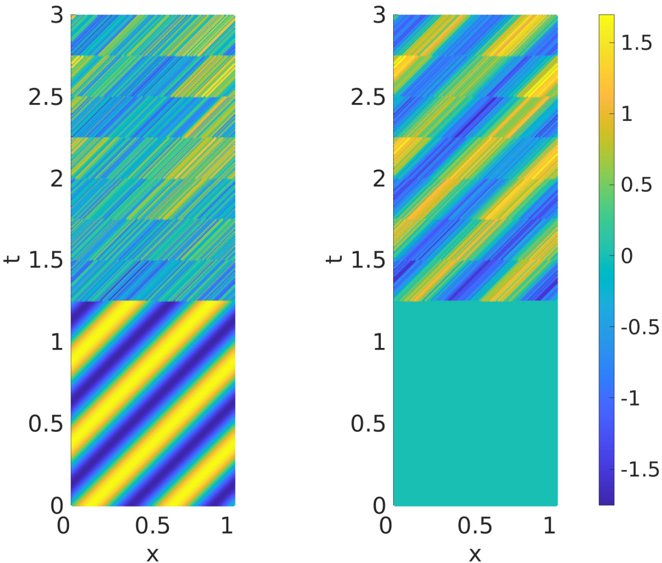
Parareal

Error

Advection example: Iteration 4



Advection example: Iteration 5



Parareal

Error

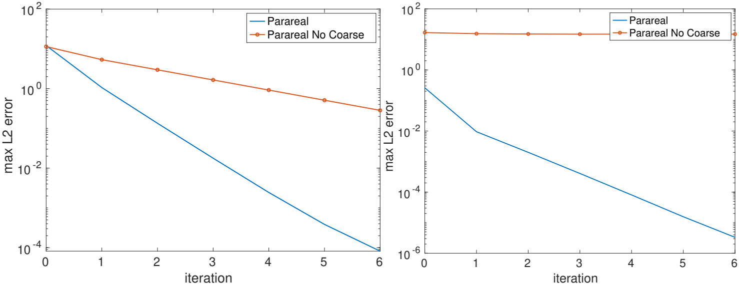
Importance of the Coarse Correction?

What about Parareal without coarse for the heat equation,

$$U_{n+1}^{k+1} := F(T_{n+1}, T_n, U_n^k)$$

instead of the original Parareal algorithm

$$U_{n+1}^{k+1} := F(T_{n+1}, T_n, U_n^k) + G(T_{n+1}, T_n, U_n^{k+1}) - G(T_{n+1}, T_n, U_n^k)$$



$T = 3$, $\Delta T = 1/16$. Dirichlet (left) and Neumann (right)

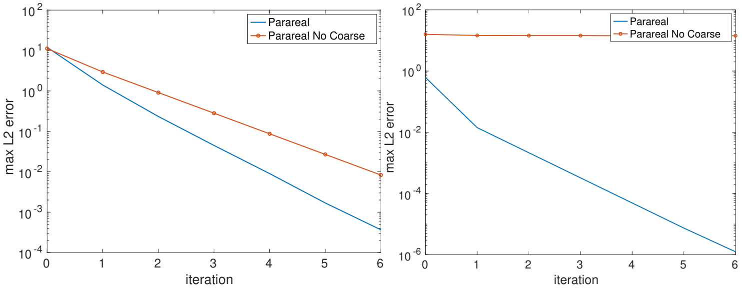
Importance of the Coarse Correction?

What about Parareal without coarse for the heat equation,

$$U_{n+1}^{k+1} := F(T_{n+1}, T_n, U_n^k)$$

instead of the original Parareal algorithm

$$U_{n+1}^{k+1} := F(T_{n+1}, T_n, U_n^k) + G(T_{n+1}, T_n, U_n^{k+1}) - G(T_{n+1}, T_n, U_n^k)$$



$T = 3$, $\Delta T = 1/8$. Dirichlet (left) and Neumann (right)

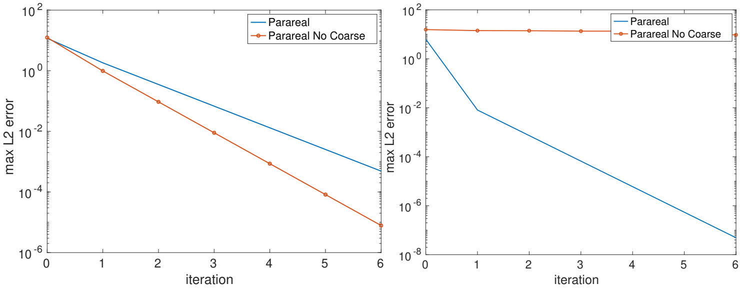
Importance of the Coarse Correction?

What about Parareal without coarse for the heat equation,

$$\boldsymbol{U}_{n+1}^{k+1} := \boldsymbol{F}(T_{n+1}, T_n, \boldsymbol{U}_n^k)$$

instead of the original Parareal algorithm

$$\boldsymbol{U}_{n+1}^{k+1} := \boldsymbol{F}(T_{n+1}, T_n, \boldsymbol{U}_n^k) + \boldsymbol{G}(T_{n+1}, T_n, \boldsymbol{U}_n^{k+1}) - \boldsymbol{G}(T_{n+1}, T_n, \boldsymbol{U}_n^k)$$



$T = 3$, $\Delta T = 1/4$. Dirichlet (left) and Neumann (right)

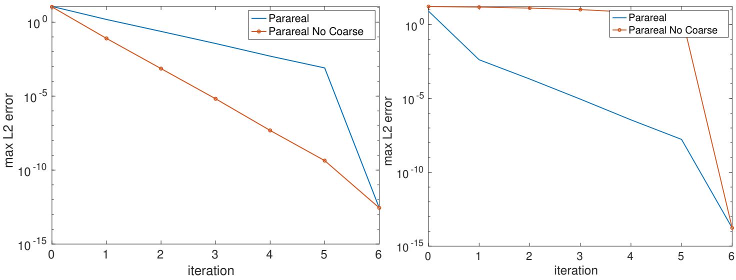
Importance of the Coarse Correction?

What about Parareal without coarse for the heat equation,

$$\boldsymbol{U}_{n+1}^{k+1} := \boldsymbol{F}(T_{n+1}, T_n, \boldsymbol{U}_n^k)$$

instead of the original Parareal algorithm

$$\boldsymbol{U}_{n+1}^{k+1} := \boldsymbol{F}(T_{n+1}, T_n, \boldsymbol{U}_n^k) + \boldsymbol{G}(T_{n+1}, T_n, \boldsymbol{U}_n^{k+1}) - \boldsymbol{G}(T_{n+1}, T_n, \boldsymbol{U}_n^k)$$



$T = 3$, $\Delta T = 1/2$. Dirichlet (left) and Neumann (right)

Convergence Result for a Special Case

Dirichlet BC: Spectral fine propagator F using m_F modes:

$$F(U_n^k,T_n,T_{n+1})=\sum_{m=1}^{m_F}\left(\hat{U}_{m,n}^k e^{-m^2\Delta T}+\int_{T_n}^{T_{n+1}}\hat{f}_m(\tau)e^{-m^2(t-\tau)}d\tau\right)\sin mx$$

Spectral coarse propagator G using m_G modes:

$$G(U_n^k,T_n,T_{n+1})=\sum_{m=1}^{m_G}\left(\hat{U}_{m,n}^k e^{-m^2\Delta T}+\int_{T_n}^{T_{n+1}}\hat{f}_m(\tau)e^{-m^2(t-\tau)}d\tau\right)\sin mx$$

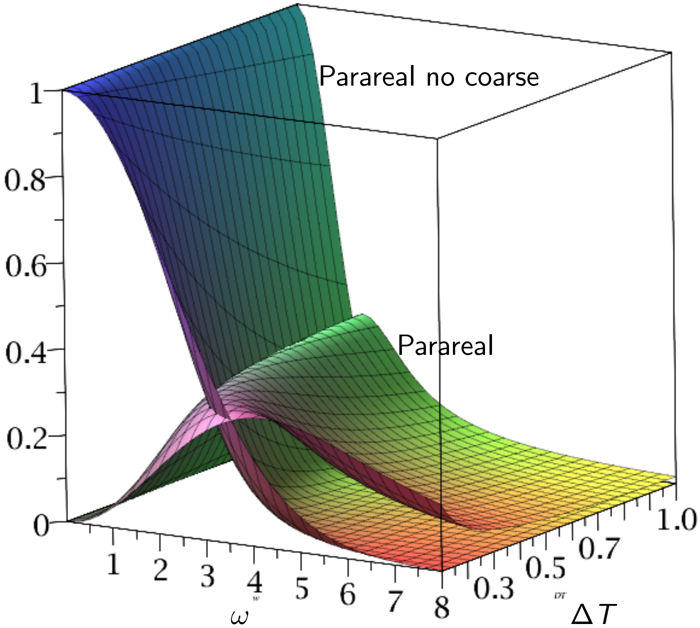
Theorem (G, Ohlberger, Rave (2024))

This parareal algorithm for the heat equation on $(0,\pi)\times(0,T)$ *with Dirichlet boundary conditions satisfies for any initial guess* U_n^0 *the convergence estimate*

$$\sup_n ||U_n^k(\cdot)-u(\cdot,T_n)||_2 \le e^{-(m_g+1)^2 k\Delta T}\sup_n ||U_n^0(\cdot)-u(\cdot,T_n)||_2,$$

and this estimate also holds if the coarse propagator does not contain any modes, $m_G=0$.

Comparison of the Convergence Estimates



Remarks

- **Scalability:** for ΔT constant, convergence does not depend on the number of time subdomains even without G! Equivalent to DD result in space:
  - **Maday, Stamm et al (2013-2014):** Domain decomposition for implicit solvation models
  - **G, Ciaramella (2017, 2018)** Analysis of the parallel Schwarz method for growing chains of fixed-sized subdomains I, II, III
- Similar estimate for Neumann boundary conditions, but need at least the constant mode in G for contraction.
- The same estimate holds in full generality for a more general parabolic equation in arbitrary spatial dimensions, the convergence factor then becomes

$$e^{-\lambda_{m_G+1}\Delta T},$$

where λ_{m_G+1} is the m_G plus first eigenvalue of the corresponding spatial operator.

Much better: Space-Time Multigrid

All at once system for the heat equation in space-time:

$$\underbrace{\begin{bmatrix} (I - L\Delta t) & & & \\ -I & (I - L\Delta t) & & \\ & & \ddots & \ddots \end{bmatrix}}_{A} \underbrace{\begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \end{pmatrix}}_{\boldsymbol{u}} = \underbrace{\begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \vdots \end{pmatrix}}_{\boldsymbol{f}}$$

- Parabolic Multigrid (Hackbusch 1984)
- Multigrid Waveform Relaxation (Lubich and Ostermann 1987)
- Space-Time Multigrid (Horton and Vandewalle (1995)

G, Neumüller (2016): Analysis of a New Space-Time Parallel Multigrid Algorithm for Parabolic Problems

Key idea: use block Jacobi smoother with optimal damping!

3D Heat Equation Weak Scaling Results

cores	time steps	dof	iter	time	fwd. sub.
1	2	59 768	7	28.8	19.0
2	4	119 536	7	29.8	37.9
4	8	239 072	7	29.8	75.9
8	16	478 144	7	29.9	152.2
16	32	956 288	7	29.9	305.4
32	64	1 912 576	7	29.9	613.6
64	128	3 825 152	7	29.9	1 220.7
128	256	7 650 304	7	29.9	2 448.4
256	512	15 300 608	7	30.0	4 882.4
512	1 024	30 601 216	7	29.9	9 744.2
1 024	2 048	61 202 432	7	30.0	19 636.9
2 048	4 096	122 404 864	7	29.9	38 993.1
4 096	8 192	244 809 728	7	30.0	81 219.6
8 192	16 384	489 619 456	7	30.0	162 551.0
16 384	32 768	979 238 912	7	30.0	313 122.0
32 768	65 536	1 958 477 824	7	30.0	625 686.0
65 536	131 072	3 916 955 648	7	30.0	1 250 210.0
131 072	262 144	7 833 911 296	7	30.0	2 500 350.0
262 144	524 288	15 667 822 592	7	30.0	4 988 060.0

Vulcan BlueGene/Q Supercomputer in Livermore (by M. Neumüller)

3D Heat Equation Strong Scaling Results

cores	time steps	dof	iter	time
1	512	15 300 608	7	7 635.2
2	512	15 300 608	7	3 821.7
4	512	15 300 608	7	1 909.9
8	512	15 300 608	7	954.2
16	512	15 300 608	7	477.2
32	512	15 300 608	7	238.9
64	512	15 300 608	7	119.5
128	512	15 300 608	7	59.7
256	512	15 300 608	7	30.0
512	524 288	15 667 822 592	7	15 205.9
1 024	524 288	15 667 822 592	7	7 651.5
2 048	524 288	15 667 822 592	7	3 825.3
4 096	524 288	15 667 822 592	7	1 913.4
8 192	524 288	15 667 822 592	7	956.6
16 384	524 288	15 667 822 592	7	478.1
32 768	524 288	15 667 822 592	7	239.3
65 536	524 288	15 667 822 592	7	119.6
131 072	524 288	15 667 822 592	7	59.8
262 144	524 288	15 667 822 592	7	30.0

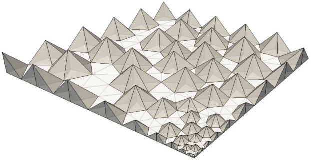
Hard for hyperbolic problems (MGRIT, Falgout et al 2017-2024)

Hyperbolic PinT: Mapped Tent Pitching (MTP)

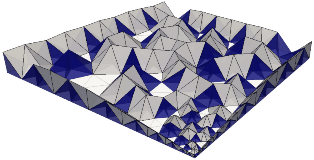
Gopalakrishnan, Schöberl, Wintersteiger (2017):

Mapped Tent Pitching Schemes for Hyperbolic Systems

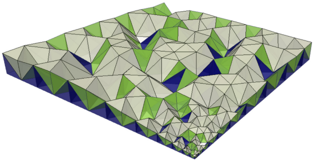
"This paper explores a technique by which standard discretizations, including explicit time stepping, can be used within tent-shaped spacetime domains. The technique transforms the equations within a spacetime tent to a domain where space and time are separable."

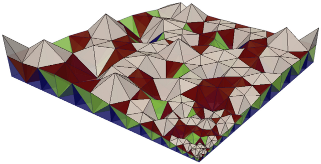


(a) Initial tents forming layer 1.



(b) Layer 2 tents in gray (and layer 1 tents in blue).





Key new idea for MTP

Tent shaped subdomains are mapped to tensor shaped subdomains and then solved by any classical time stepping method, before being mapped back:

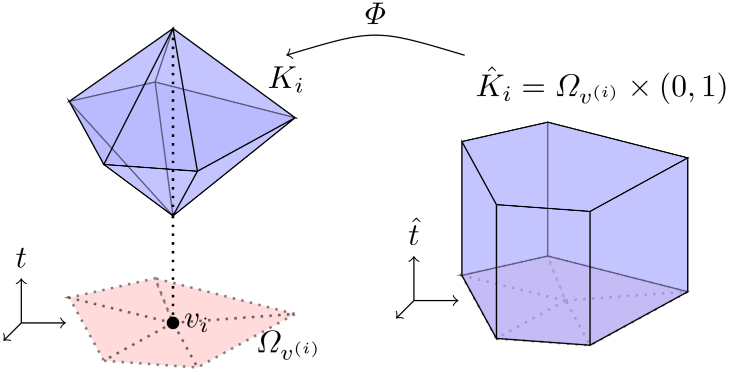
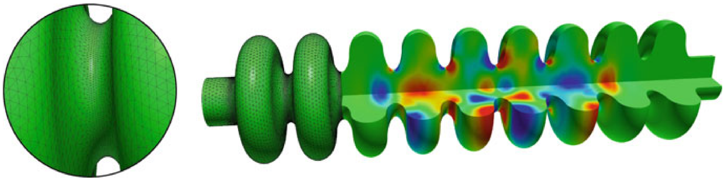


Fig. 2. *Tent mapped from a tensor product domain.*

Application to Maxwell's Equations

Gopalakrishnan, Hochteger, Schöberl, Wintersteiger (2020): An Explicit Mapped Tent Pitching Scheme for Maxwell Equations

"This method is highly parallel, since many tents can be solved independently."



	$p = 2$	$p = 3$
Number of spatial dof	2.938×10^7	5.875×10^7
Number of spacetime dof per slab	1.908×10^9	7.632×10^9
Simulation time per slab	4.6 s	49.2 s
Total simulation time	20 min	3 h 33 min

This data was generated using a shared memory server with 4 E7-8867 CPUs with 16 cores each

Probably the best PinT Maxwell solver currently available!

Red-Black Schwarz Waveform Relaxation

Nievergelt (1964): "…much wider class of numerical procedures in which parallelism is introduced at the expense of redundancy of computation."

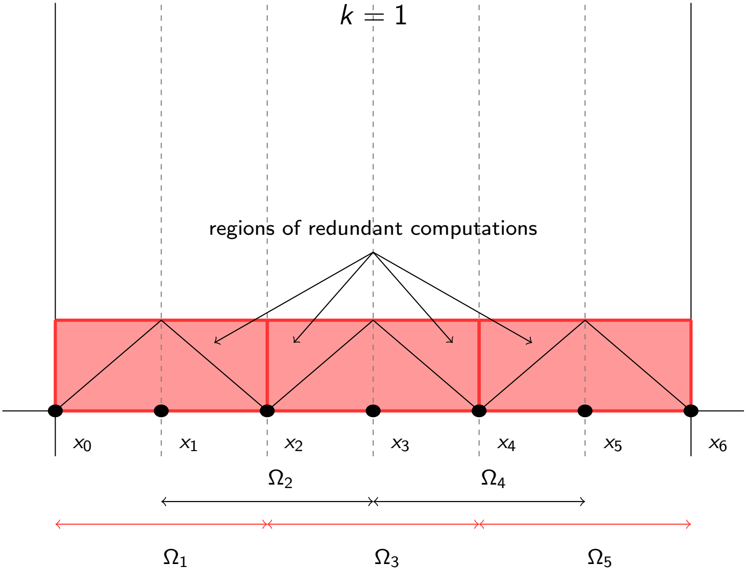
Following this approach, we use Red-Black Schwarz Waveform Relaxation (RBSWR). 1D example:

Wave equation, decomposition of $\Omega = (x_0, x_N)$ into N subdomains $\Omega_j = (x_j, x_{j+2})$, $j = 0, \ldots, N-1$ (decomposition with generous overlap). Let $\mathcal{R} = \{0, 2, 4, \ldots\}$ and $\mathcal{B} = \{1, 3, 5, \ldots\}$:

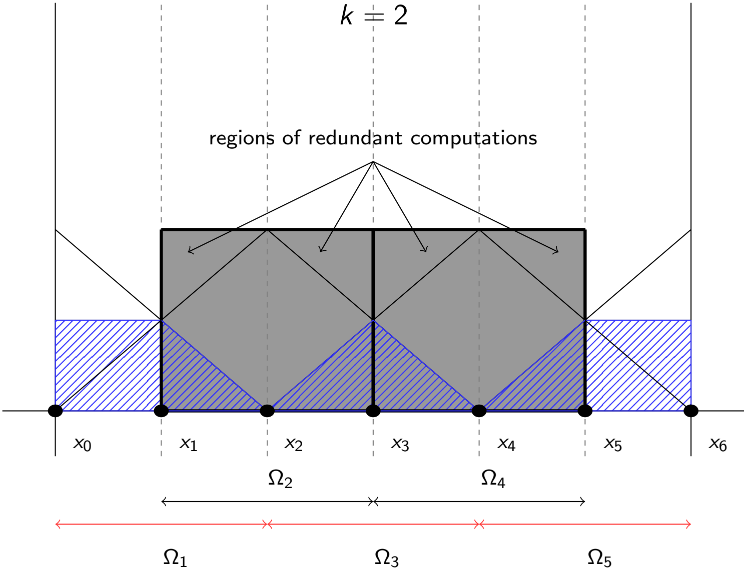
$$
\begin{aligned}
\partial_{tt} u_j^k(x,t) &= c^2 \partial_{xx} u_j^k(x,t) && \text{in } I_j \times (0,T),\\
u_j^k(x_j,t) &= u_{j-1}^{k-1}(x_j,t) && \text{for } t \in [0,T],\\
u_j^k(x_{j+2},t) &= u_{j+1}^{k-1}(x_{j+2},t) && \text{for } t \in [0,T],
\end{aligned}
$$

where k is the iteration index, and $j \in \mathcal{R}$ for k odd and $j \in \mathcal{B}$ for k even.

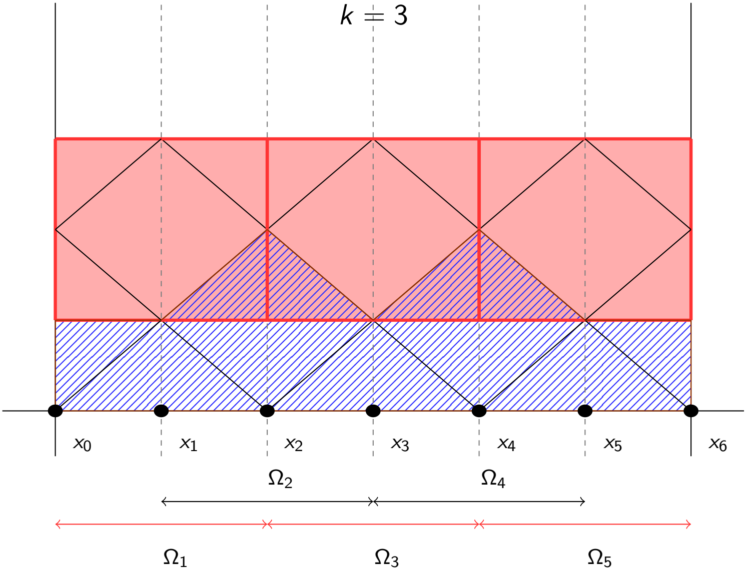
Choosing Tentpole Time Intervals in RBSWR



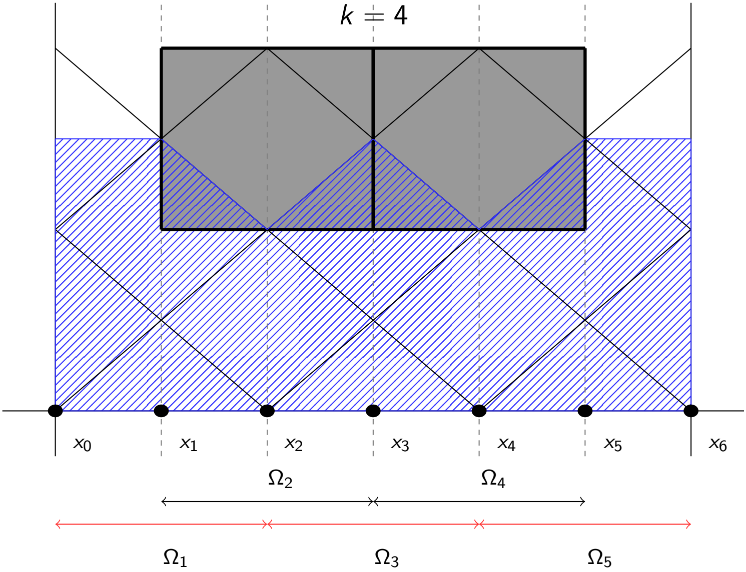
Second Iteration of RBSWR



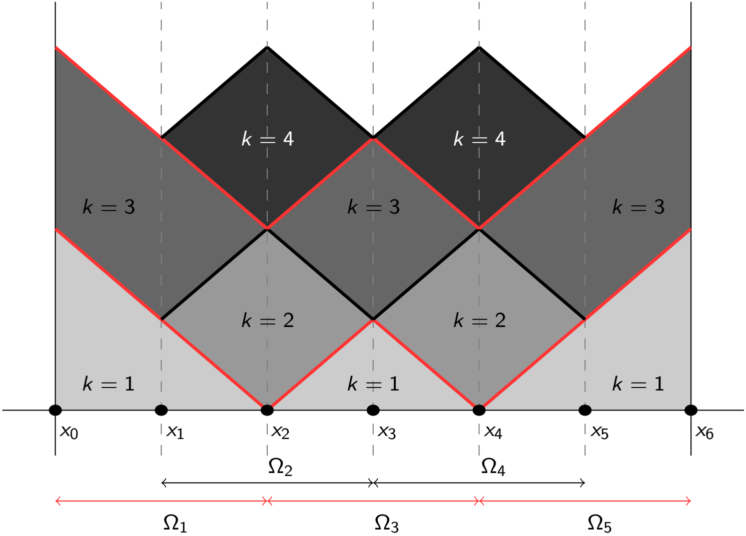
Third Iteration of RBSWR



Fourth Iteration of RBSWR

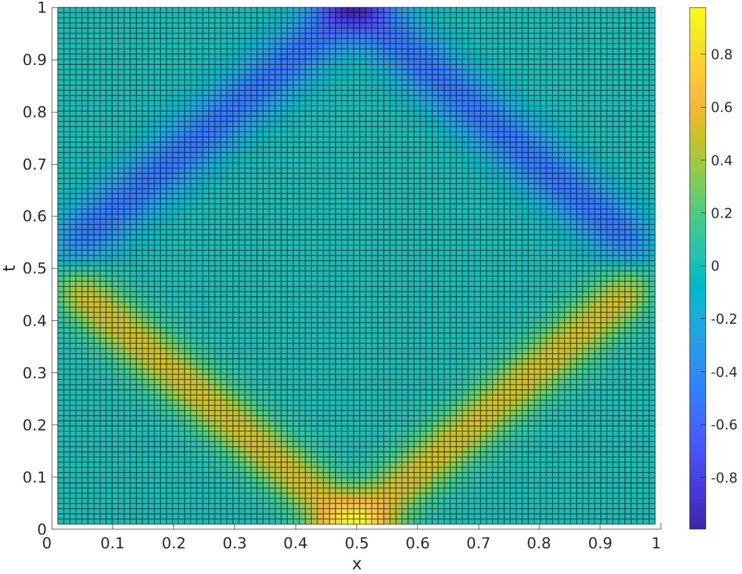


RBSWR As Unmapped Tent Pitching (UTP)



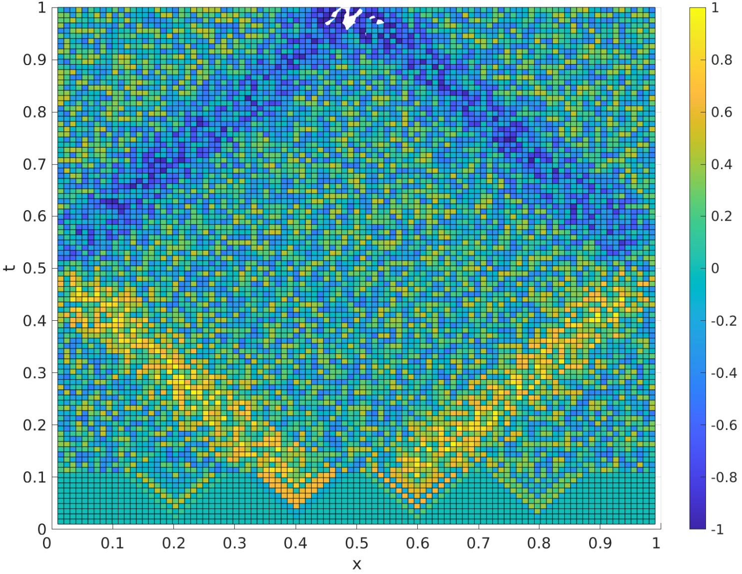
Tent mapping is replaced by redundancy!

Example: wave equation



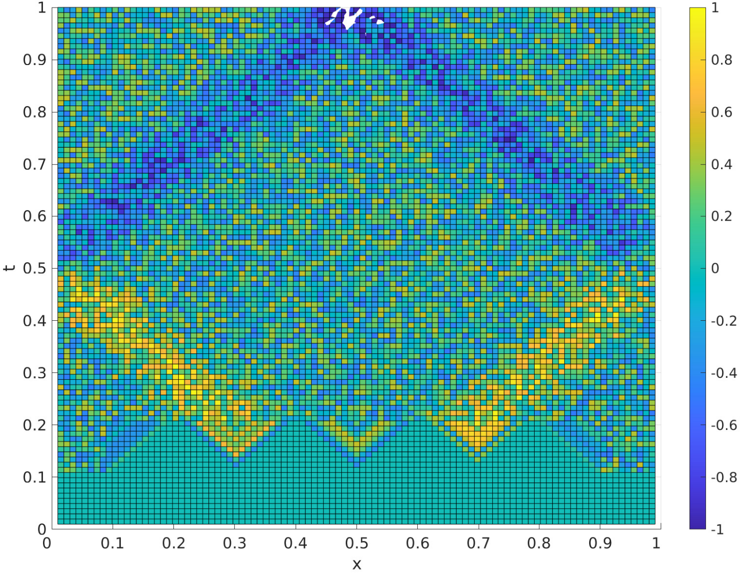
Very easy to code using RAS on the space-time system

Red iteration 1 error



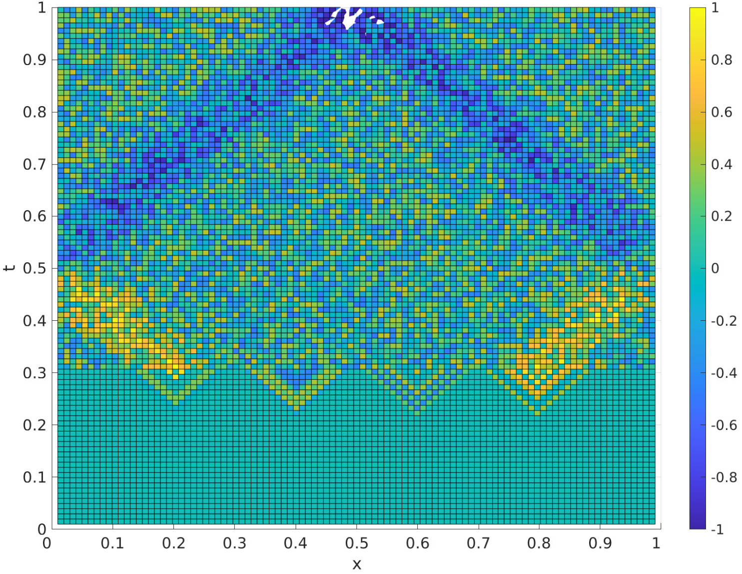
Very easy to code using RAS on the space-time system

Black iteration 1 error



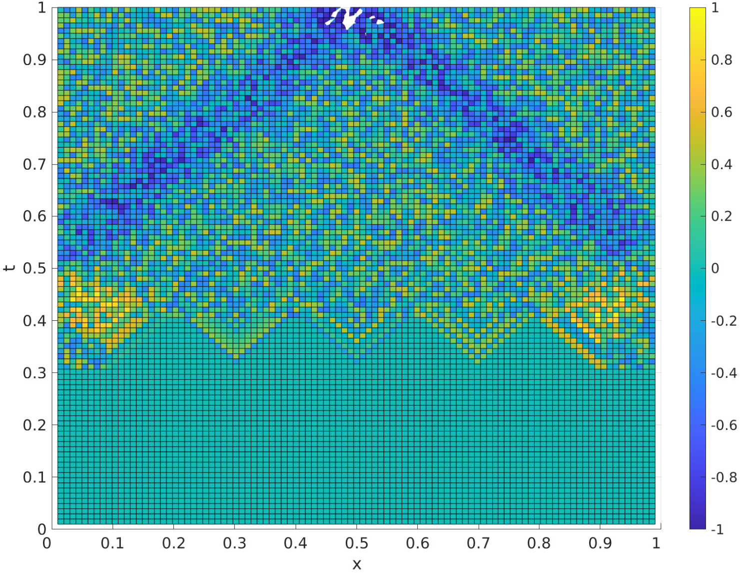
Very easy to code using RAS on the space-time system

Red iteration 2 error



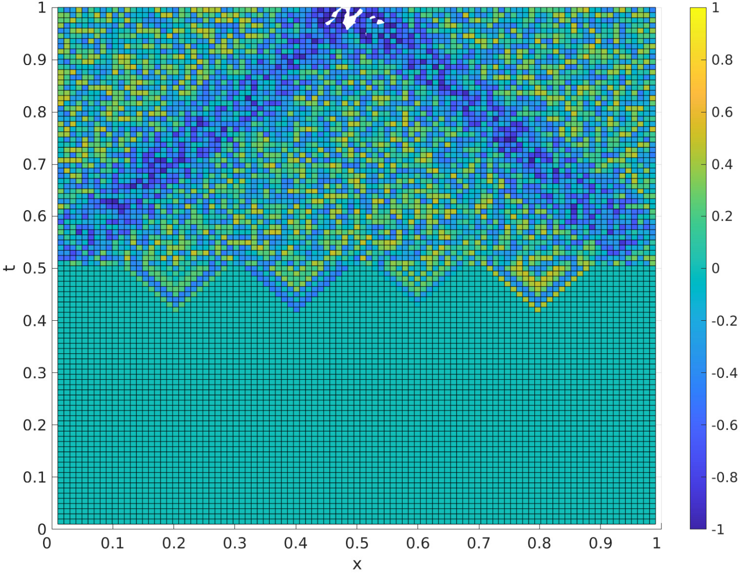
Very easy to code using RAS on the space-time system

Black iteration 2 error



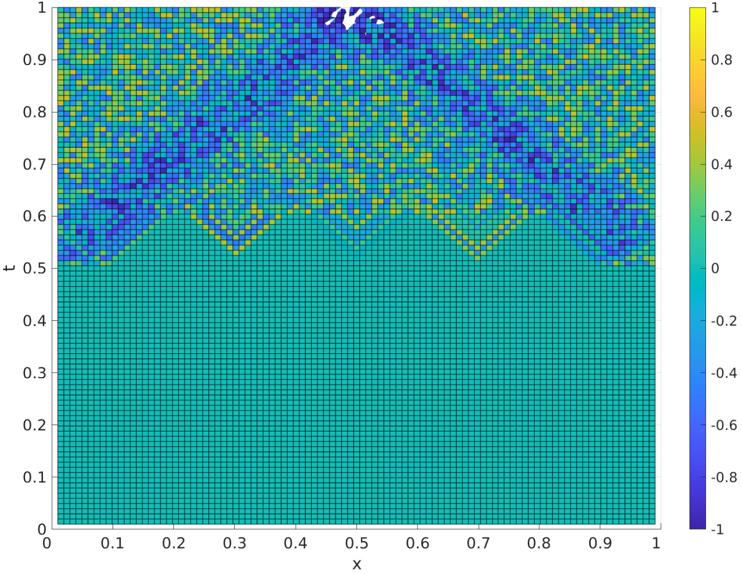
Very easy to code using RAS on the space-time system

Red iteration 3 error



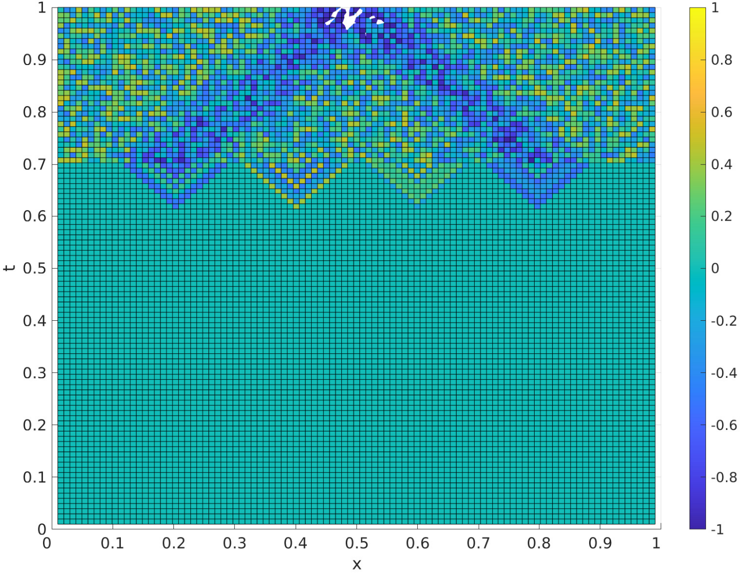
Very easy to code using RAS on the space-time system

Black iteration 3 error



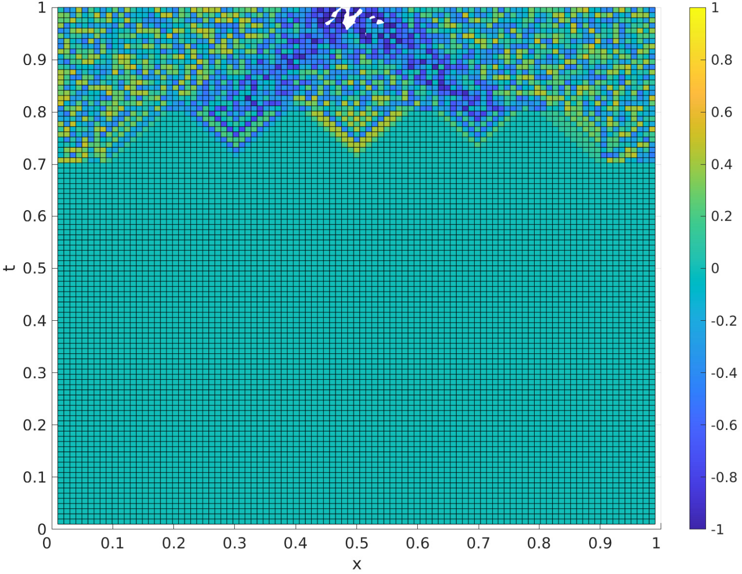
Very easy to code using RAS on the space-time system

Red iteration 4 error



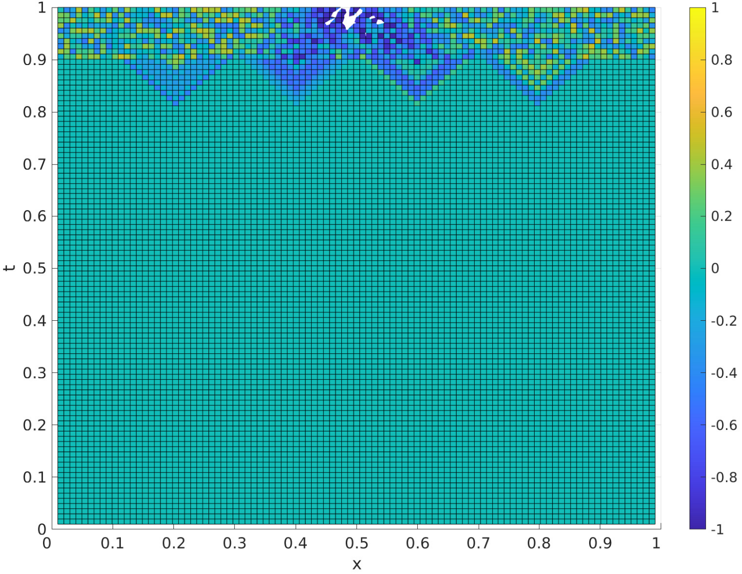
Very easy to code using RAS on the space-time system

Black iteration 4 error



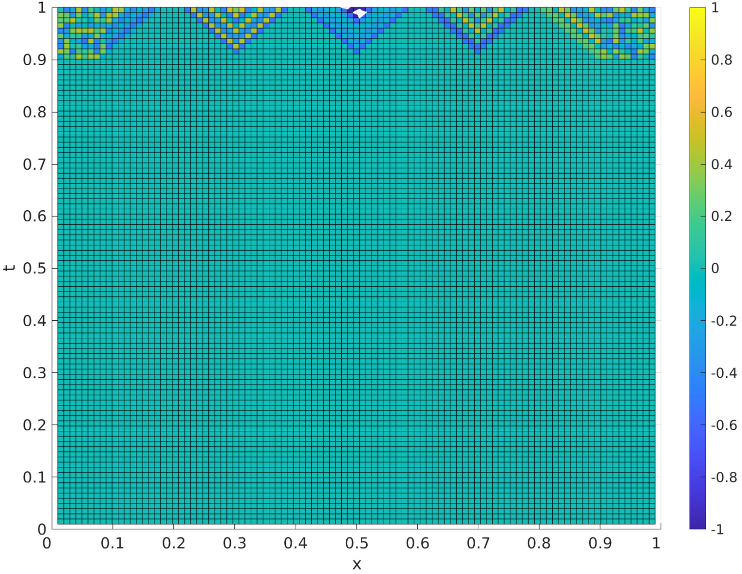
Very easy to code using RAS on the space-time system

Red iteration 5 error



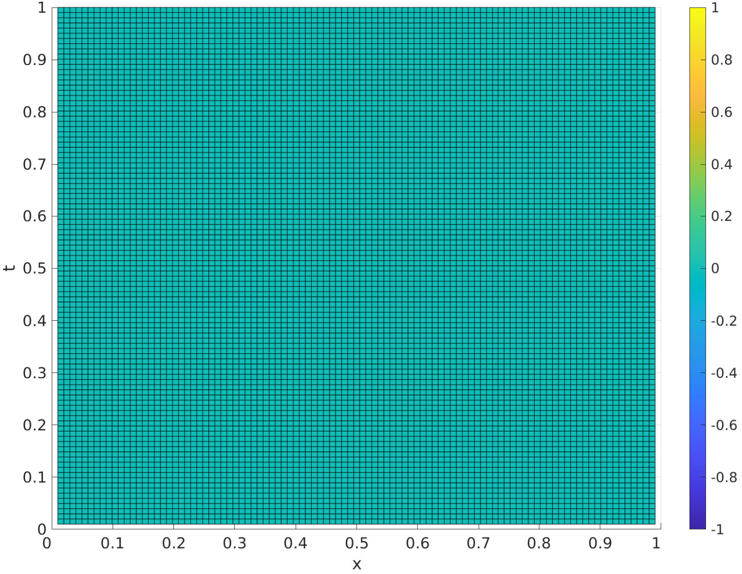
Very easy to code using RAS on the space-time system

Black iteration 5 error



Very easy to code using RAS on the space-time system

Red iteration 6 error



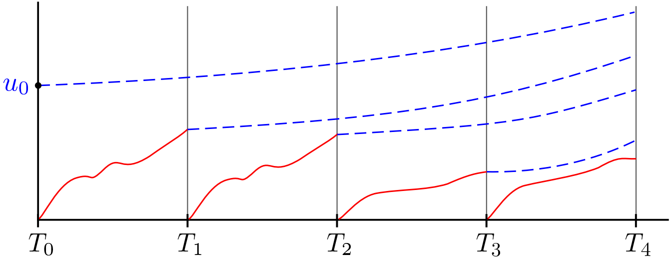
Very easy to code using RAS on the space-time system

ParaExp

Consider the linear system of evolution equations

$$\boldsymbol{u}'(t) = A\boldsymbol{u}(t) + \boldsymbol{g}(t), \quad t \in [0, T], \quad \boldsymbol{u}(0) = \boldsymbol{u}_0.$$

ParaExp is based on a completely overlapping decomposition of the time interval $[0, T]$ into subintervals, e.g. $[0, T_4 := T]$, $[T_1, T_4]$, $[T_2, T_4]$, and $[T_3, T_4]$.



G., Güttel (2013): ParaExp: a Parallel Integrator for Linear Initial-Value Problems

Direct parallel solve in two steps

1. solve the non-overlapping inhomogeneous red problems

$$\boldsymbol{v}_j'(t) = A\boldsymbol{v}_j(t) + \boldsymbol{g}(t), \quad \boldsymbol{v}_j(T_{j-1}) = 0, \quad t \in [T_{j-1}, T_j]$$

2. solve the overlapping homogeneous blue problems

$$\boldsymbol{w}_j'(t) = A\boldsymbol{w}_j(t), \quad \boldsymbol{w}_j(T_{j-1}) = \boldsymbol{v}_{j-1}(T_{j-1}),\ t \in [T_{j-1}, T]$$

By linearity, the solution is then obtained by summation,

$$\boldsymbol{u}(t) = \boldsymbol{v}_k(t) + \sum_{j=1}^{k} \boldsymbol{w}_j(t) \quad \text{with } k \text{ s.t.} \quad t \in [T_{k-1}, T_k].$$

Blue problems even over long time are very cheap:

1. Approximate $\boldsymbol{a}_n(t) \approx exp(tA)\boldsymbol{v}$ from a Krylov space built with $S := (I - A/\sigma)^{-1}A$
2. $\exp(tA)\boldsymbol{v} \approx \sum_{j=0}^{n-1} \beta_j(t) p_j(A)\boldsymbol{v}$, where p_j are polynomials or rational functions.

ParaExp for the Wave Equation

$$\begin{aligned}\partial_{tt}u(t,x) &= \alpha^2\partial_{xx}u(t,x) + \text{hat}(x)\sin(2\pi f t), \quad x,t\in(0,1)\\ u(t,0) &= u(t,1) = u(0,x) = u'(0,x) = 0\end{aligned}$$

α^2	f	serial		parallel			efficiency
		τ_0	error	$\max(\tau_1)$	$\max(\tau_2)$	error	
0.1	1	2.54e−01	3.64e−04	4.04e−02	1.48e−02	2.64e−04	58 %
0.1	5	1.20e+00	1.31e−04	1.99e−01	1.39e−02	1.47e−04	71 %
0.1	25	6.03e+00	4.70e−05	9.83e−01	1.38e−02	7.61e−05	76 %
1	1	7.30e−01	1.56e−04	1.19e−01	2.70e−02	1.02e−04	63 %
1	5	1.21e+00	4.09e−04	1.97e−01	2.70e−02	3.33e−04	68 %
1	25	6.08e+00	1.76e−04	9.85e−01	2.68e−02	1.15e−04	75 %
10	1	2.34e+00	6.12e−05	3.75e−01	6.31e−02	2.57e−05	67 %
10	5	2.31e+00	4.27e−04	3.73e−01	6.29e−02	2.40e−04	66 %
10	25	6.09e+00	4.98e−04	9.82e−01	6.22e−02	3.01e−04	73 %

Finite differences and RK45 for the red problems, Chebyshev exponential integrator for the blue ones, 8 processors

ParaDiag

Model Problem: discretize a linear evolution problem $u_t = Lu + f$ using Backward Euler,

$$\begin{pmatrix} \frac{1}{\Delta t} - L & & & \\ -\frac{1}{\Delta t} & \frac{1}{\Delta t} - L & & \\ & \ddots & \ddots & \\ & & -\frac{1}{\Delta t} & \frac{1}{\Delta t} - L \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} f_1 + \frac{1}{\Delta t} u_0 \\ f_2 \\ \vdots \\ f_N \end{pmatrix}$$

Using the Kronecker symbol, this linear system can be written in compact form as

$$(B \otimes I_x - I_t \otimes L)\boldsymbol{u} = \boldsymbol{f},$$

I_x and I_t identity matrices and B is the time stepping matrix

Maday, Rønquist (2008): Parallelization in time through tensor-product space–time solvers

Brugnano, Trigiante (1996-): Boundary value methods for ODEs

ParaDiag-I for diagonalizable B

$$B := \begin{pmatrix} \frac{1}{\Delta t_1} & & & \\ -\frac{1}{\Delta t_2} & \frac{1}{\Delta t_2} & & \\ & \ddots & \ddots & \\ & & -\frac{1}{\Delta t_N} & \frac{1}{\Delta t_N} \end{pmatrix}.$$

With $B = SDS^{-1}$, one can rewrite the system in factored form, namely

$$(S \otimes I_x)(\text{diag}(D - L))(S^{-1} \otimes I_x)\boldsymbol{u} = \boldsymbol{f},$$

and we can hence solve it in 3 steps:

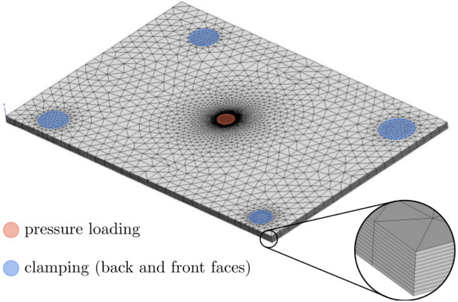
$$\begin{array}{lrcl} (a) & (S \otimes I_x)\boldsymbol{g} & = & \boldsymbol{f}, \\ (b) & (\frac{1}{\Delta t_n} - L)\boldsymbol{w}^n & = & \boldsymbol{g}^n, \quad 1 \leq n \leq N, \\ (c) & (S^{-1} \otimes I_x)\boldsymbol{u} & = & \boldsymbol{w}. \end{array}$$

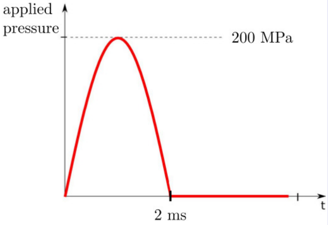
The expensive step (b) solving with the system matrix L can be done entirely in parallel for all time levels t_n.

Industrial Elasticity Example

Response of a carbon/epoxy laminated composite panel to an impact-like loading, modeled by the elasticity equations

$$\rho \ddot{\boldsymbol{u}} = \mathrm{div}(\sigma) + \boldsymbol{f}, \quad \epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right).$$

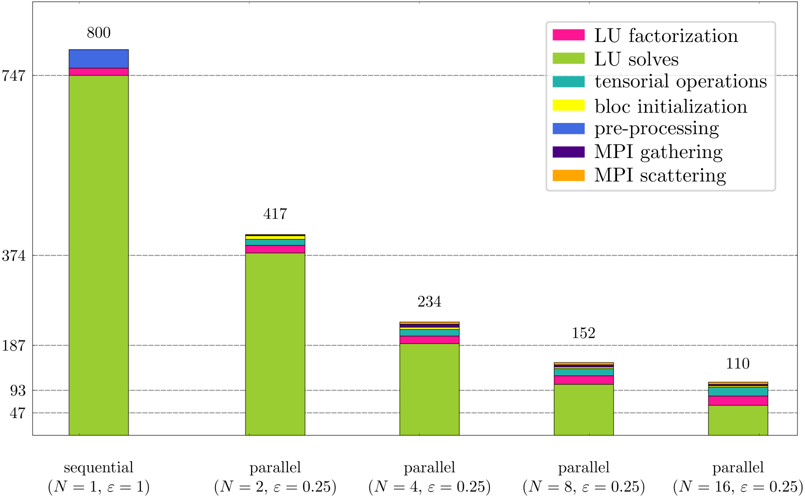




152607 dofs, 2000 time steps over the 10ms simulation range

G, Halpern, Rannou, Ryan (2019): A Direct Time Parallel Solver by Diagonalization for the Wave Equation

Strong Scaling Results



Computing times (in seconds) for the industrial elasticity problem (all computations performed by J. Rannou 2017)

ParaDiag-II: non-diagonalizable B

Can one make B with equal time steps diagonalizabe ?

$$\tilde{B} = \begin{bmatrix} \frac{1}{\Delta t} & & & -\frac{\alpha}{\Delta t} \\ -\frac{1}{\Delta t} & \frac{1}{\Delta t} & & \\ & \ddots & \ddots & \\ & & -\frac{1}{\Delta t} & \frac{1}{\Delta t} \end{bmatrix}$$

But then we solve the wrong problem, need to iterate:

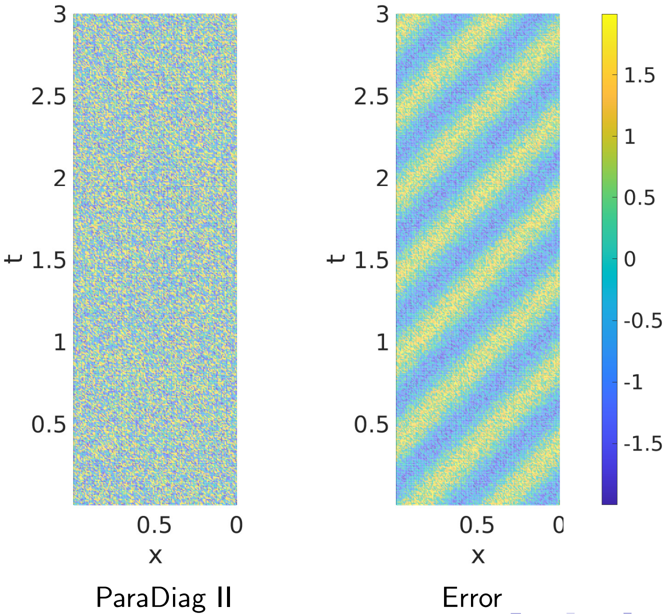
$$A = (B \otimes I_x - I_t \otimes L), \quad \tilde{A} = (\tilde{B} \otimes I_x - I_t \otimes L),$$

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

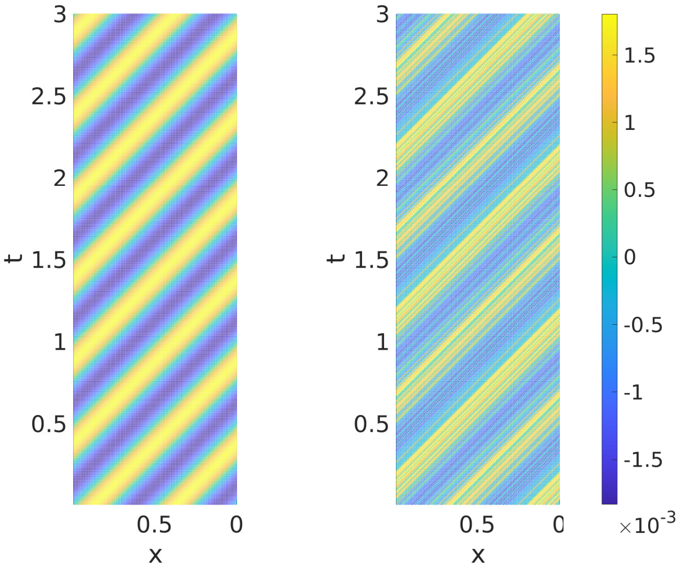
Independently discovered:

- $\alpha = 1$: **McDonald, Pestana, Wathen (2018):** Preconditioning and iterative solution of all-at-once systems for evolution partial differential equations.
- Optimized α: **Shulin Wu (2018):** Toward parallel coarse grid correction for the parareal algorithm (2018)

ParaDiag II on advection: Initial Guess



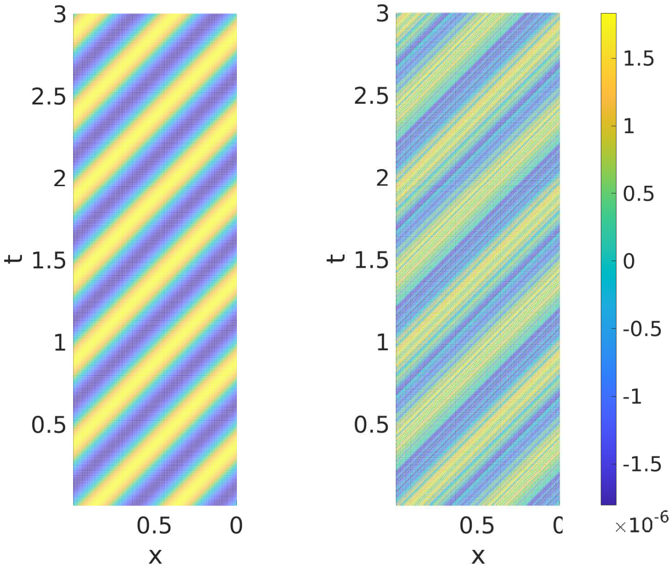
ParaDiag II on advection: Iteration 1



ParaDiag II

Error

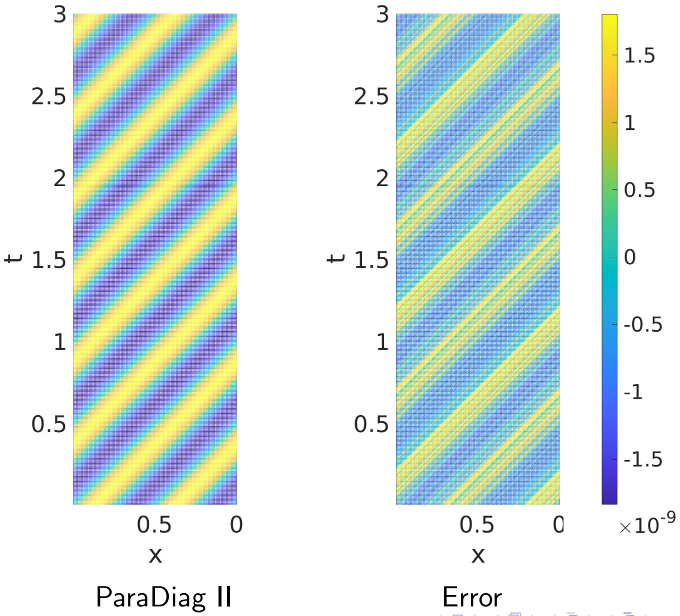
ParaDiag II on advection: Iteration 2



ParaDiag II

Error

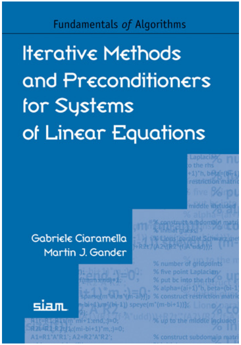
ParaDiag II on advection: Iteration 3

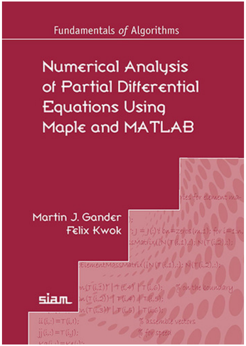


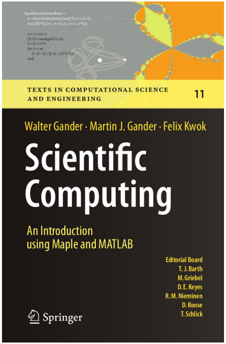
Conclusions

- Parabolic problems are natural for PinT, hyperbolic problems are not!
- There are many good PinT methods for parabolic problems: **Parareal** and variants (PITA, PFASST, MGRIT), **Space-Time Multigrid** (STMG), **Domain Decomposition Waveform relaxation** (Schwarz WR, Dirichlet-Neumann WR, Neumann-Neumann WR)
- **Domain Decomposition Waveform relaxation** methods are also suitable for hyperbolic problems $\Longrightarrow$ Mapped and Unmapped Tent Pitching.
- **ParaExp** and **ParaDiag** methods are also suitable for hyperbolic problems (see also G, Palitta 2024).

New PinT book (2024): Time Parallel Time Integration


Fundamentals of Algorithms
Iterative Methods and Preconditioners for Systems of Linear Equations
Gabriele Ciaramella
Martin J. Gander
siam


Fundamentals of Algorithms
Numerical Analysis of Partial Differential Equations Using Maple and MATLAB
Martin J. Gander
Felix Kwok
siam


TEXTS IN COMPUTATIONAL SCIENCE AND ENGINEERING
11
Walter Gander · Martin J. Gander · Felix Kwok
Scientific Computing
An Introduction using Maple and MATLAB
Editorial Board
T. J. Barth
M. Griebel
D. E. Keyes
R. M. Nieminen
D. Roose
T. Schlick
Springer