Six Decades of Time Parallel Time Integration: Best Current Methods for Parabolic and Hyperbolic Problems

Martin J. Gander
martin.gander@unige.ch

University of Geneva

CEMRACS 2022

Weather Prediction as a Typical Application

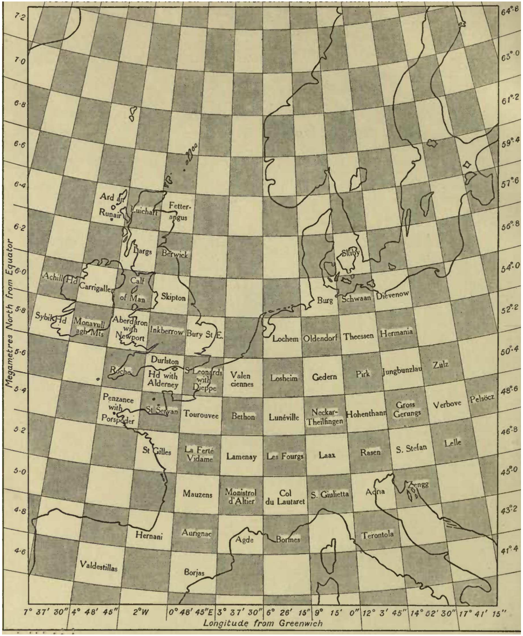
Theophrastus, c. 371 - c. 287 BC

> *"They are less certain when the moon is not full. If the moon looks fiery, it indicates breezy weather for that month, if dusky, wet weather; and, whatever indications the crescent moon gives, are given when it is three days old."*

Richardson (1922): Weather Prediction by Numerical Process (100 years!)

> *"Perhaps some day in the dim future it will be possible to advance the computations faster than the weather advances and at a cost less than the saving to mankind due to the information gained. But that is a dream."*

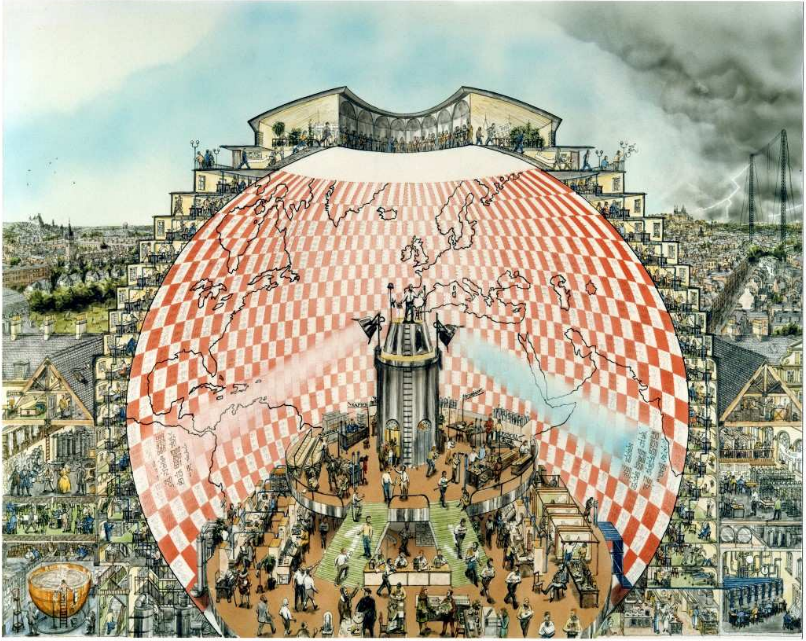
Wheather prediction computation by Richardson



"An arrangement of meteorological stations designed to fit with the chief mechanical properties of the atmosphere"

"Pressure to be observed at the centre of each shaded chequer, velocity at the centre of each white chequer."

Richardson's vision (Stephen Conlin, 1986)



Edward N. Lorenz (1979)

On the prevalence of aperiodicity in simple systems

"As the lone meteorologist at a seminar of mathematicians, I feel that a few words regarding my presence may be in order. Let me begin with some remarks about the mathematics of meteorology.

One of the most familiar problems of interest to meteorologists is weather forecasting. Mathematically this is an initial-value problem. The atmosphere and its surroundings are governed by a set of physical laws which in principle can be expressed as a system of integro-differential equations.

At the turn of the century, the forecast problem was identified by Bjerknes as the problem of solving these equations, using initial conditions obtained from observations of current weather. Detailed numerical procedures for solving these equations were formulated during World War I by Richardson, but the practical solution of even rather crude approximations had to await the advent of computers."

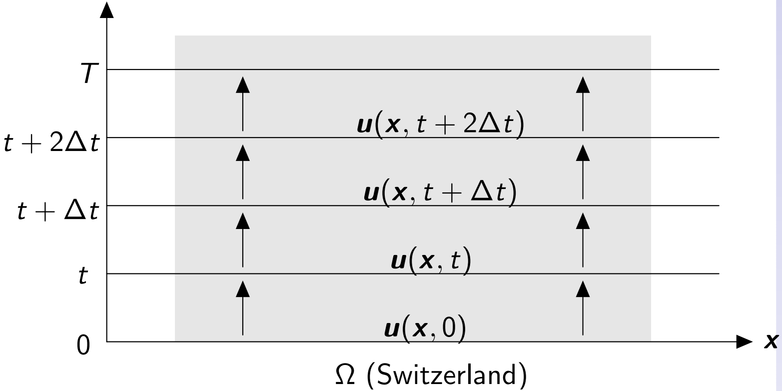
Weather Prediction Equations

$$
\begin{aligned}
\partial_t \boldsymbol{u}(\boldsymbol{x},t) &= \mathcal{L}(\boldsymbol{u}(\boldsymbol{x},t)) && \text{in } \Omega \times (0,T], \\
\boldsymbol{u}(\boldsymbol{x},0) &= \boldsymbol{u}_0(\boldsymbol{x}) && \text{in } \Omega, \\
\mathcal{B}(\boldsymbol{u}(\boldsymbol{x},t)) &= \boldsymbol{g}(\boldsymbol{x},t) && \text{on } \partial\Omega,
\end{aligned}
$$

$\boldsymbol{u}$ contains the wind vector, temperature, pressure etc.
If $\boldsymbol{u}_0$ is known over Switzerland, we can predict the weather, using $\boldsymbol{g}$ from a European weather model.

Computing Weather Prediction Step by Step

The solution $\boldsymbol{u}(\boldsymbol{x}, t + \Delta t)$ depends on the solution $\boldsymbol{u}(\boldsymbol{x}, t)$ for any $\Delta t > 0$:

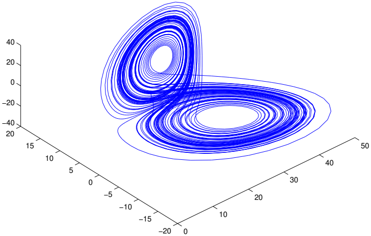


This is an entirely sequential process, impossible to compute the far future before the near future is known!

Solving Evolution Problems in Parallel ?

The Lorenz equations:

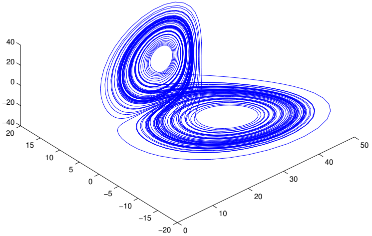
$$
\begin{aligned}
\frac{dx}{dt} &= -\sigma x + \sigma y \\
\frac{dy}{dt} &= -xz + rx - y \\
\frac{dz}{dt} &= xy - bz
\end{aligned}
$$



Solving Evolution Problems in Parallel ?

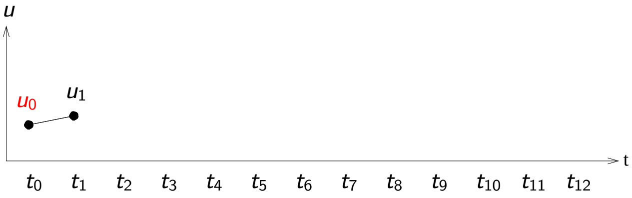
The Lorenz equations:

$$\frac{dx}{dt} = -\sigma x + \sigma y$$
$$\frac{dy}{dt} = -xz + rx - y$$
$$\frac{dz}{dt} = xy - bz$$



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

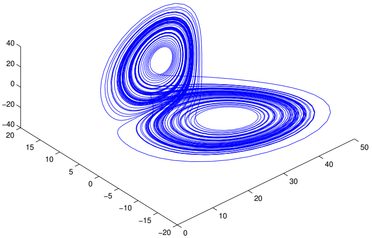
$$u_1 = u_0 + \Delta t f(u_0)$$



Solving Evolution Problems in Parallel ?

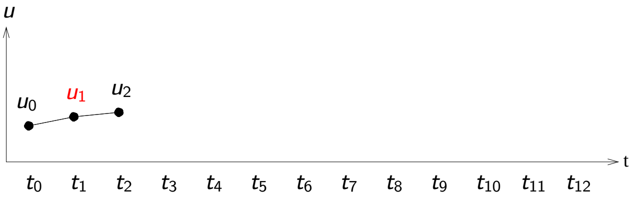
The Lorenz equations:

$$\frac{dx}{dt} = -\sigma x + \sigma y$$
$$\frac{dy}{dt} = -xz + rx - y$$
$$\frac{dz}{dt} = xy - bz$$



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

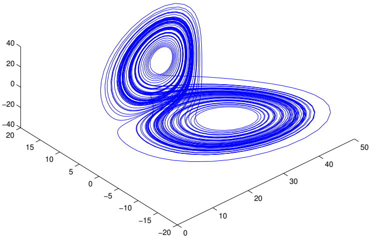
$$u_2 = u_1 + \Delta t f(u_1)$$



Solving Evolution Problems in Parallel ?

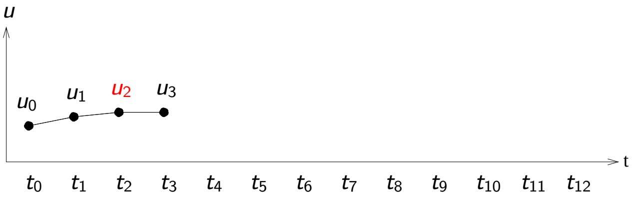
The Lorenz equations:

$$\begin{aligned}\frac{dx}{dt} &= -\sigma x + \sigma y\\ \frac{dy}{dt} &= -xz + rx - y\\ \frac{dz}{dt} &= xy - bz\end{aligned}$$



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

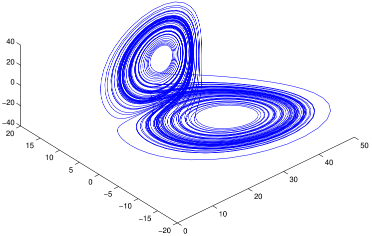
$$u_3 = u_2 + \Delta t f(u_2)$$



Solving Evolution Problems in Parallel ?

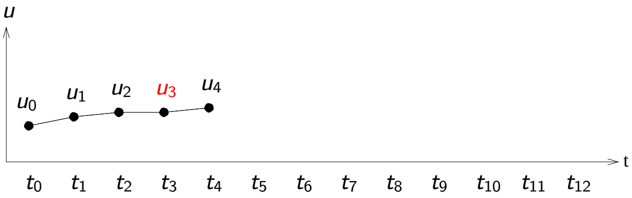
The Lorenz equations:

$$\begin{aligned}\frac{dx}{dt} &= -\sigma x + \sigma y\\ \frac{dy}{dt} &= -xz + rx - y\\ \frac{dz}{dt} &= xy - bz\end{aligned}$$



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

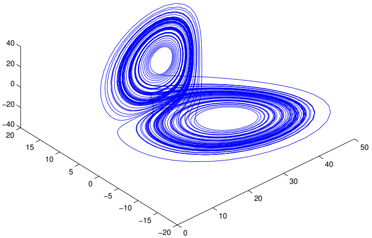
$$u_4 = u_3 + \Delta t f(u_3)$$



Solving Evolution Problems in Parallel ?

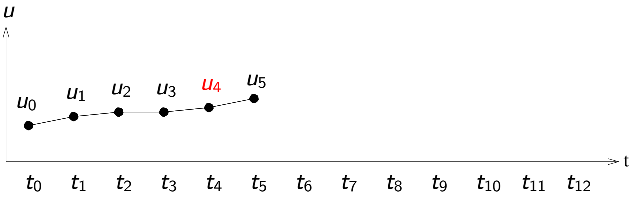
The Lorenz equations:

$$\frac{dx}{dt} = -\sigma x + \sigma y$$
$$\frac{dy}{dt} = -xz + rx - y$$
$$\frac{dz}{dt} = xy - bz$$



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

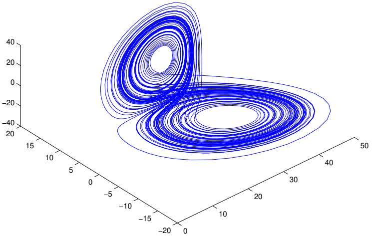
$$u_5 = u_4 + \Delta t f(u_4)$$



Solving Evolution Problems in Parallel ?

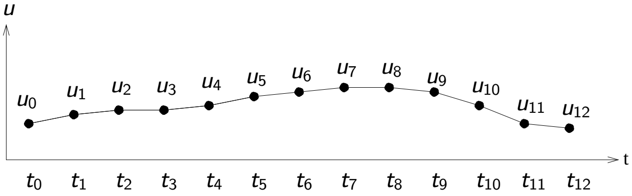
The Lorenz equations:

$$\begin{aligned}\frac{dx}{dt} &= -\sigma x + \sigma y\\ \frac{dy}{dt} &= -xz + rx - y\\ \frac{dz}{dt} &= xy - bz\end{aligned}$$



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

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



Super Computing Systems

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	361,760	19,590.0	25,326.3	2,272
4	Japan Agency for Marine-Earth Science and Technology Japan	**Gyoukou** - ZettaScaler-2.2 HPC system, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 700Mhz ExaScaler	19,860,000	19,135.8	28,192.0	1,350
5	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209

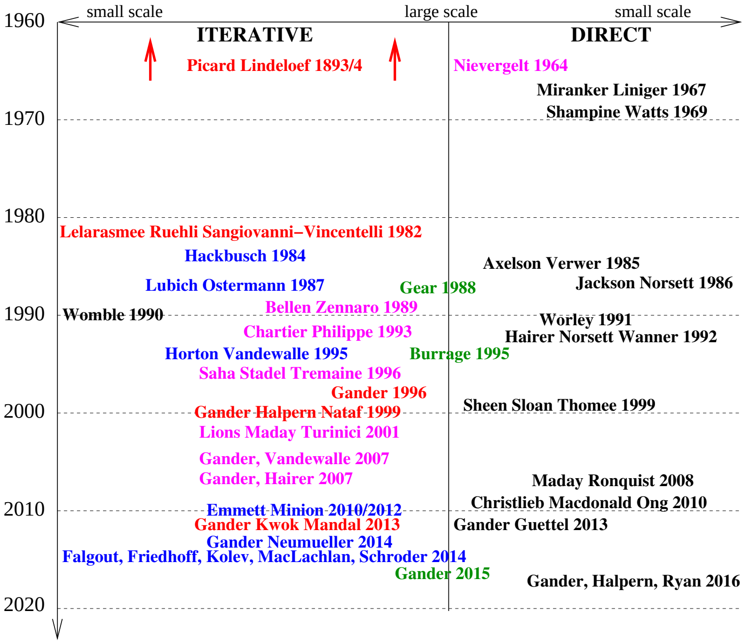
Top 500 list in November 2017

Super Computing Systems

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	1,110,144	151.90	214.35	2,942
4	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096
5	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94.64	125.71	7,438

Top 500 list in June 2022

Time Parallel Methods Over the Course of Time



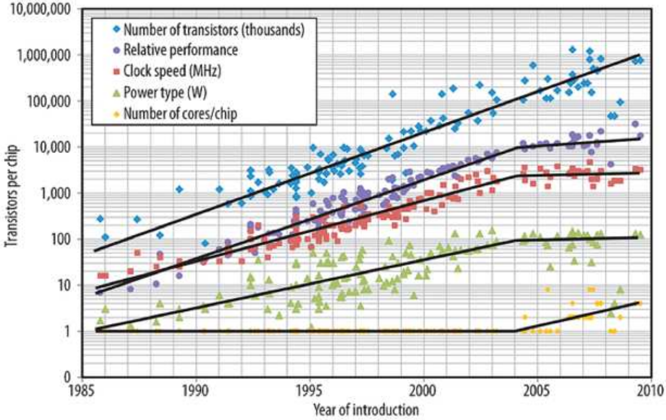
Multiple Shooting Methods

Jörg Nievergelt: Parallel Methods for Integrating Ordinary Differential Equations. Comm. of the ACM, **1964**.

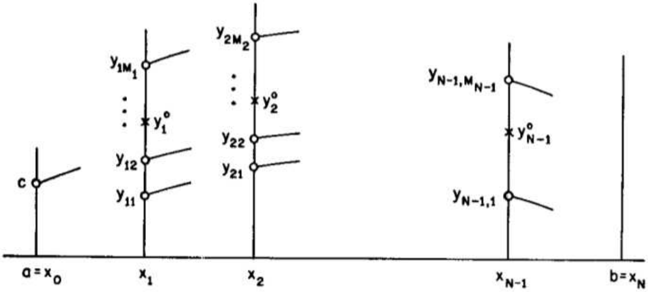
> *"For the last 20 years, one has tried to speed up numerical computation mainly by providing ever faster computers.* Today, as it appears that one is getting closer to the maximal speed of electronic components, *emphasis is put on allowing operations to be performed in parallel. In the near future, much of numerical analysis will have to be recast in a more 'parallel' form."*

Jörg Nievergelt: Parallel Methods for Integrating Ordinary Differential Equations. Comm. of the ACM, **1964**.

> *"For the last 20 years, one has tried to speed up numerical computation mainly by providing ever faster computers.* Today, as it appears that one is getting closer to the maximal speed of electronic components, *emphasis is put on allowing operations to be performed in parallel. In the near future, much of numerical analysis will have to be recast in a more 'parallel' form."*



The idea is to divide the integration interval [a, b] into N equal subintervals $[x_{i-1}, x_i]$, $x_0 = a$, $x_N = b$, $i = 1, 2, \cdots, N$, to make a rough prediction y_i^0 of the solution $y(x_i)$, to select a certain number M_i of values y_{ij}, $j = 1, 2, \cdots, M_i$ in the vicinity of y_i^0, $i = 1, 2, \cdots, N$, and then to integrate *simultaneously* with an accurate integration method $\mathfrak{M}$ all the initial value problems



. The connection between the branches is now brought about by interpolating the end value of the unique branch in $[x_0, x_1]$

Multiple Shooting for IVPs

$$\frac{du}{dt} = f(u), \quad u(0) = u^0, \qquad t \in [0,1]$$

Split $[0,1]$ into $[0,\frac{1}{3}]$, $[\frac{1}{3},\frac{2}{3}]$, $[\frac{2}{3},1]$ and solve

$$\begin{array}{rclrclrcl} \frac{du_0}{dt} &=& f(u_0), & \frac{du_1}{dt} &=& f(u_1), & \frac{du_2}{dt} &=& f(u_2), \\ u_0(0) &=& U_0, & u_1(\frac{1}{3}) &=& U_1, & u_2(\frac{2}{3}) &=& U_2, \end{array}$$

together with the matching conditions

$$U_0 = u^0, \quad U_1 = u_0(\frac{1}{3}, U_0), \quad U_2 = u_1(\frac{2}{3}, U_1)$$

Solving this system with Newton gives the recurrence

$$U_{n+1}^{k+1} = u_n(t_{n+1}, U_n^k) + \frac{\partial u_n}{\partial U_n}(t_{n+1}, U_n^k)(U_n^{k+1} - U_n^k).$$

(see Philippe Chartier and Bernard Philippe 1993)

The Parareal Algorithm

Résolution d'EDP par un schéma en temps "pararéel".
Lions, Maday, Turinici, C. R. Acad. Sci. Paris, 2001

> *"Elle a pour principale motivation les problèmes en temps réel, d'où la terminologie proposée de pararéel."*

Parareal Algorithm for $u' = f(u)$: one needs

1. Coarse solver $G(t_2, t_1, u_1)$
2. Fine solver $F(t_2, t_1, u_1)$

$$
\begin{aligned}
U^0_{n+1} &= G(t_{n+1}, t_n, U^0_n) \\
U^{k+1}_{n+1} &= F(t_{n+1}, t_n, U^k_n) + G(t_{n+1}, t_n, U^{k+1}_n) - G(t_{n+1}, t_n, U^k_n)
\end{aligned}
$$

The Parareal Algorithm

Résolution d'EDP par un schéma en temps "pararéel".
Lions, Maday, Turinici, C. R. Acad. Sci. Paris, 2001

> *"Elle a pour principale motivation les problèmes en temps réel, d'où la terminologie proposée de pararéel."*

Parareal Algorithm for $u' = f(u)$: one needs

1. Coarse solver $G(t_2, t_1, u_1)$
2. Fine solver $F(t_2, t_1, u_1)$

$$\begin{aligned} U_{n+1}^0 &= G(t_{n+1}, t_n, U_n^0) \\ U_{n+1}^{k+1} &= F(t_{n+1}, t_n, U_n^k) + G(t_{n+1}, t_n, U_n^{k+1}) - G(t_{n+1}, t_n, U_n^k) \end{aligned}$$

G, Vandewalle 2007:

1. Parareal is multiple shooting

$$U_{n+1}^{k+1} = u_n(t_{n+1}, U_n^k) + \frac{\partial u_n}{\partial U_n}(t_{n+1}, U_n^k)(U_n^{k+1} - U_n^k)$$

   with the Jacobian approximated by differences on a coarser grid.
2. Parareal is also a time multigrid method with aggressive coarsening.

Precise Convergence Estimate for Parareal

Theorem (G, Hairer 2007)

Let $F(t_{n+1}, t_n, U_n^k)$ denote the exact solution at t_{n+1} and $G(t_{n+1}, t_n, U_n^k)$ be a one step method with local truncation error bounded by $C_1 \Delta T^{p+1}$. If

$$|G(t+\Delta T, t, x) - G(t+\Delta T, t, y)| \leq (1 + C_2 \Delta T)|x - y|,$$

then

$$\max_{1\leq n\leq N} |u(t_n) - U_n^k| \leq \frac{C_1 \Delta T^{k(p+1)}}{k!}(1+C_2\Delta T)^{N-1-k} \prod_{j=1}^{k}(N-j) \max_{1\leq n\leq N} |u(t_n) - U_n^0|$$

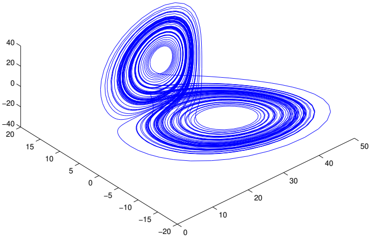
$$\leq \frac{(C_1 T)^k}{k!} e^{C_2(T-(k+1)\Delta T)} \Delta T^{pk} \max_{1\leq n\leq N} |u(t_n) - U_n^0|.$$

G and Hairer: Nonlinear Convergence Analysis for the Parareal Algorithm, Domain Decomposition Methods in Science and Engineering XVII, Springer-Verlag, 2007.

Results for the Lorenz Equations

Suggested by Jean-Pierre Eckmann (2004)

$$\begin{aligned} \dot{x} &= -\sigma x + \sigma y \\ \dot{y} &= -xz + rx - y \\ \dot{z} &= xy - bz \end{aligned}$$

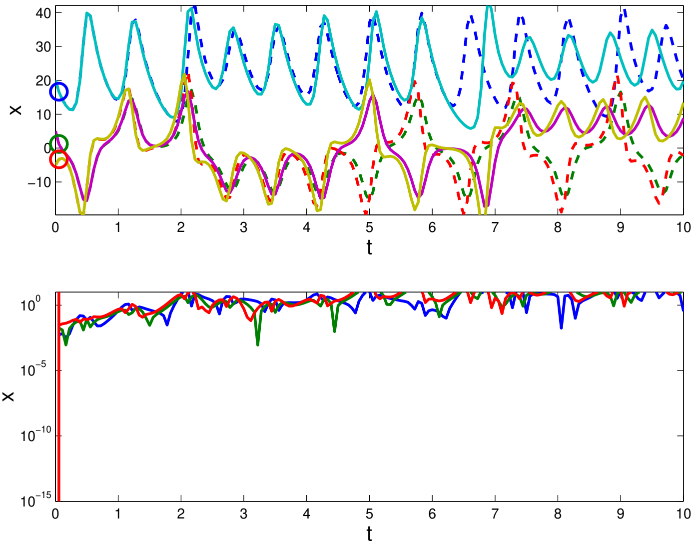


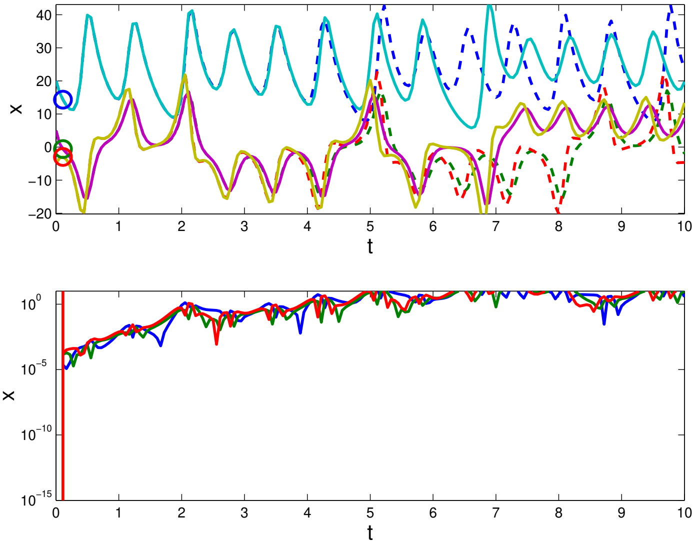
Parameters: $\sigma = 10$, $r = 28$ and $b = \frac{8}{3} \Longrightarrow$ chaotic regime.

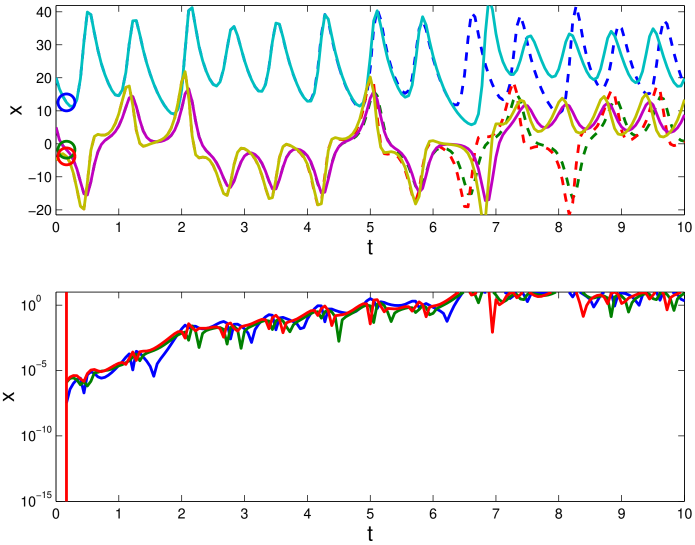
Initial conditions: $(x, y, z)(0) = (20, 5, -5)$

Simulation time: $t \in [0, T = 10]$

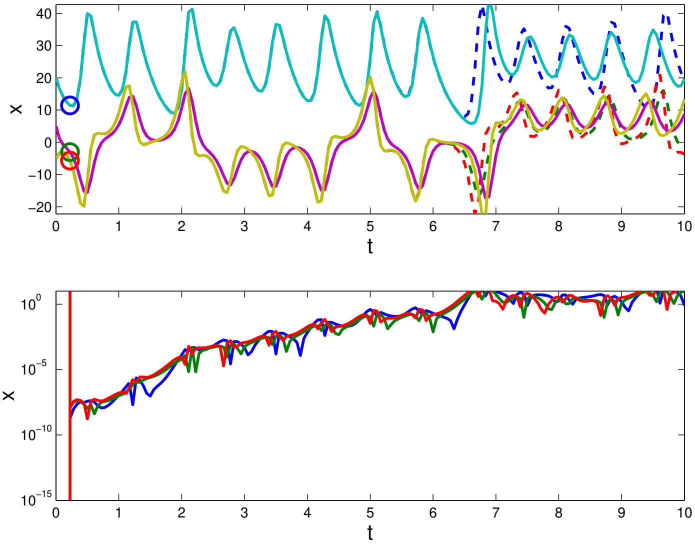
Discretization: Fourth order Runge Kutta, $\Delta T = \frac{T}{180}$,
$\Delta t = \frac{T}{1800}$.

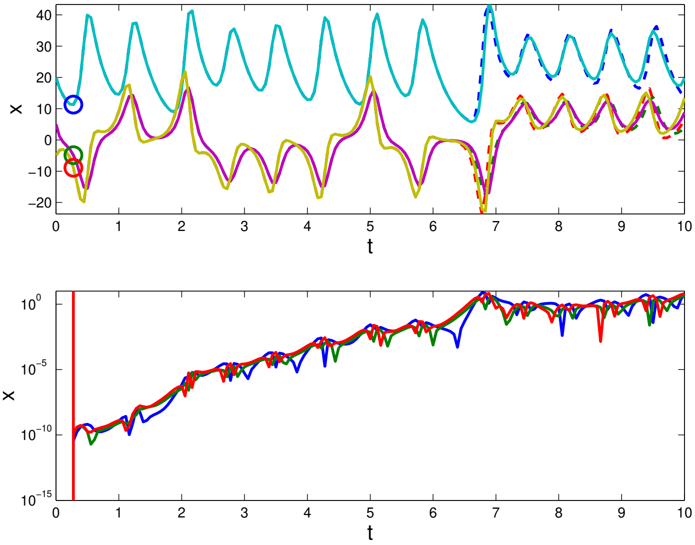

40
30
20
10
0
−10
x
0 1 2 3 4 5 6 7 8 9 10
t
10^{0}
10^{-5}
10^{-10}
10^{-15}
x
0 1 2 3 4 5 6 7 8 9 10
t


40
30
20
10
0
−10
−20
x
t
0
1
2
3
4
5
6
7
8
9
10
10^0
10^{-5}
10^{-10}
10^{-15}

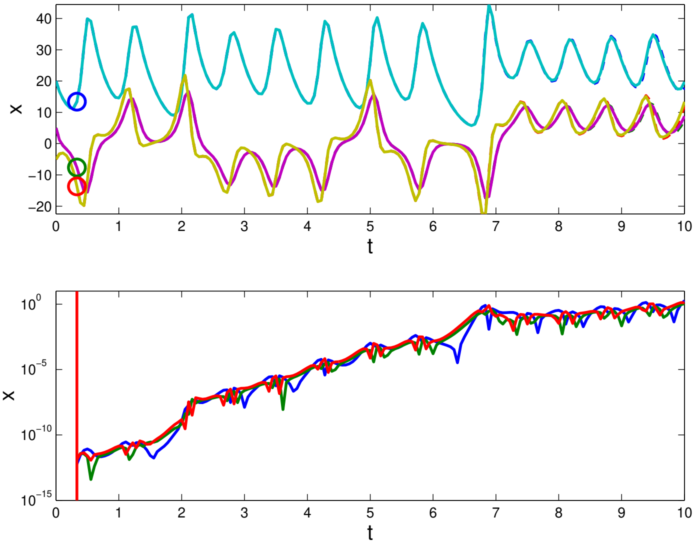


x
t
0
1
2
3
4
5
6
7
8
9
10
40
30
20
10
0
–10
–20
x
t
10^{0}
10^{-5}
10^{-10}
10^{-15}

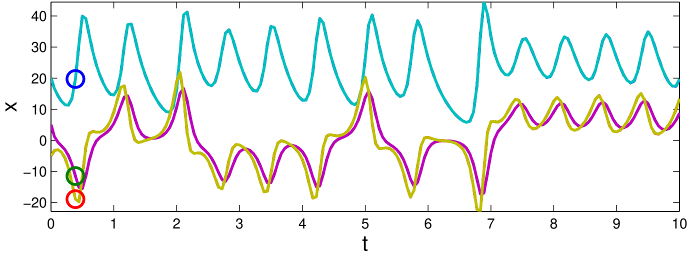

40
30
20
10
0
−10
−20
x
0
1
2
3
4
5
6
7
8
9
10
t
10^0
10^−5
10^−10
10^−15
x
0
1
2
3
4
5
6
7
8
9
10
t



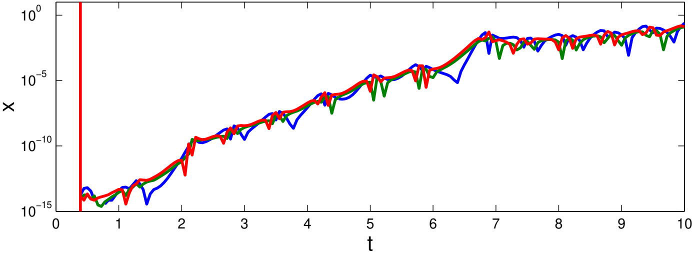
40
30
20
10
0
−10
−20
x
0 1 2 3 4 5 6 7 8 9 10
t
10^0
10^−5
10^−10
10^−15
x
0 1 2 3 4 5 6 7 8 9 10
t



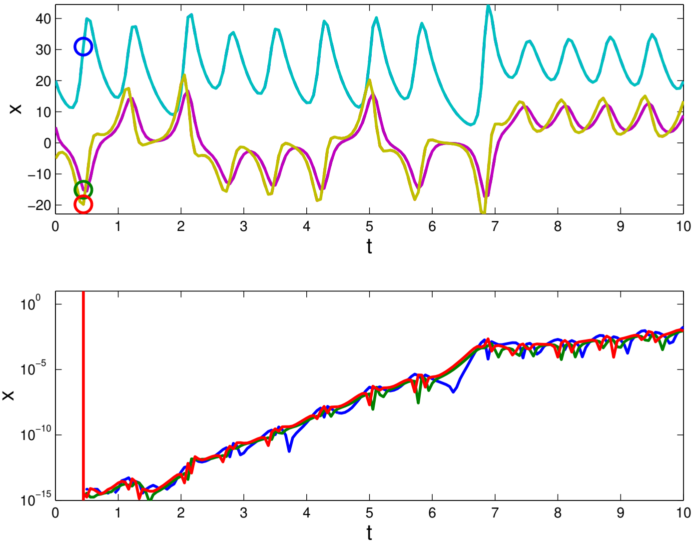
40
30
20
10
0
−10
−20
x
0
1
2
3
4
5
6
7
8
9
10
t
10^0
10^−5
10^−10
10^−15
x
t



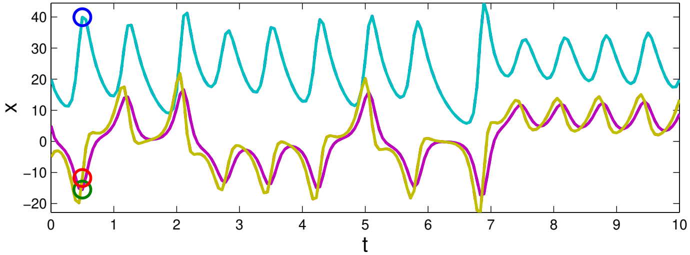
40
30
20
10
0
−10
−20
x
0
1
2
3
4
5
6
7
8
9
10
t

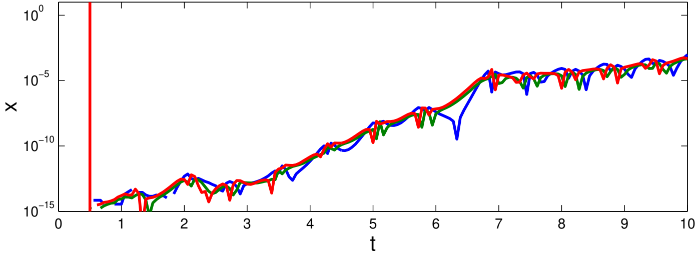


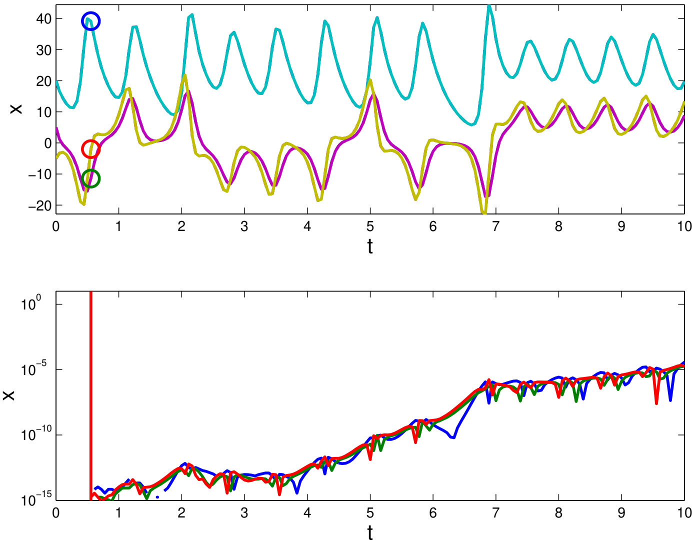
10^{0}
10^{-5}
10^{-10}
10^{-15}
x
0
1
2
3
4
5
6
7
8
9
10
t

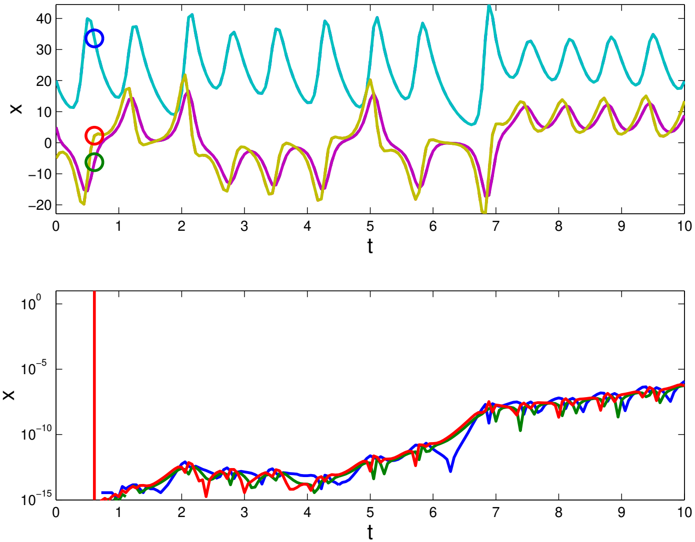


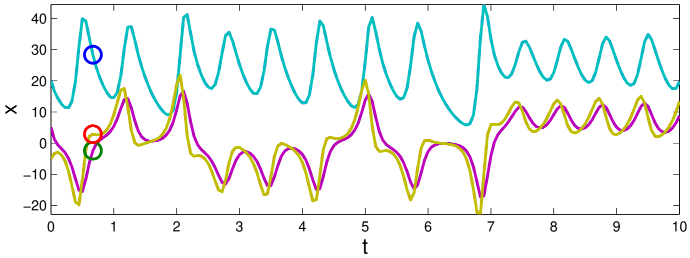
40
30
20
10
0
−10
−20
x
0
1
2
3
4
5
6
7
8
9
10
t
10^0
10^{-5}
10^{-10}
10^{-15}
x
0
1
2
3
4
5
6
7
8
9
10
t


40
30
20
10
0
−10
−20
x
0
1
2
3
4
5
6
7
8
9
10
t

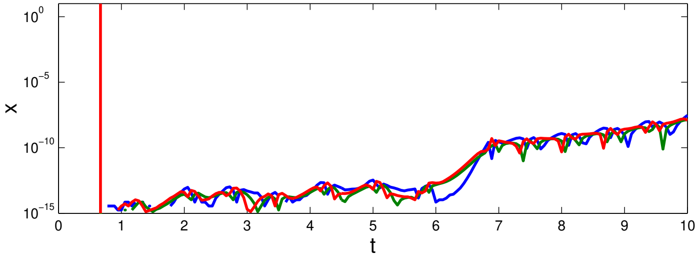

10^{0}
10^{-5}
10^{-10}
10^{-15}
x
0
1
2
3
4
5
6
7
8
9
10
t


40
30
20
10
0
−10
−20
x
0
1
2
3
4
5
6
7
8
9
10
t
10^0
10^{-5}
10^{-10}
10^{-15}
x
0
1
2
3
4
5
6
7
8
9
10
t


40
30
20
10
0
–10
–20
x
0
1
2
3
4
5
6
7
8
9
10
t
10^{0}
10^{-5}
10^{-10}
10^{-15}
x
t



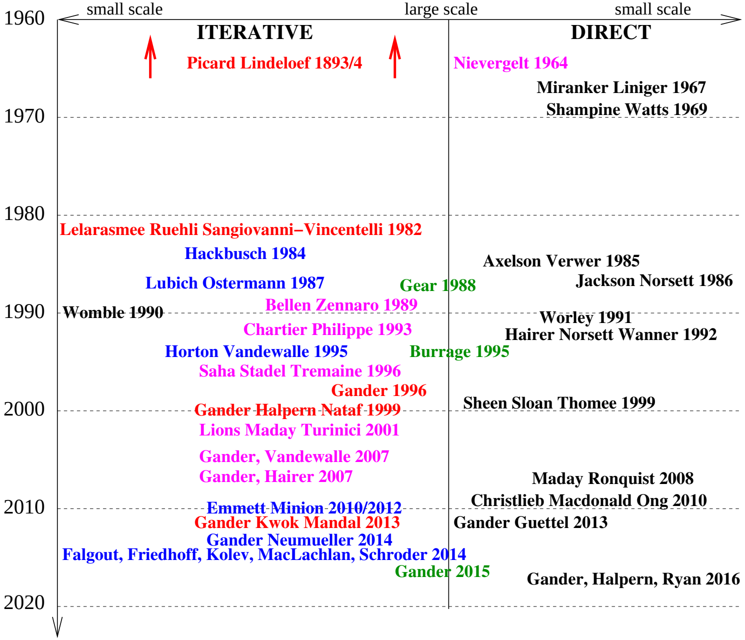
40
30
20
10
0
–10
–20
x
0
1
2
3
4
5
6
7
8
9
10
t



10^{0}
10^{-5}
10^{-10}
10^{-15}
x
0
1
2
3
4
5
6
7
8
9
10
t

Space-Time Multigrid

Time Parallel Methods Over the Course of Time



A New Space-Time Multigrid Method

Discretize the heat equation $\frac{\partial u}{\partial t} = \Delta u + f$ in space and time

$$\begin{pmatrix} A_1 & & & & \\ B_2 & A_2 & & & \\ & B_3 & A_3 & & \\ & & \ddots & \ddots & \\ & & & B_n & A_n \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \\ \vdots \\ \boldsymbol{u}_n \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \boldsymbol{f}_3 \\ \vdots \\ \boldsymbol{f}_n \end{pmatrix}$$

and use block Jacobi smoother for the space-time system with standard restriction and prolongation operators.

Theorem (G, Neumüller (2016))

For the heat equation, and block Jacobi smoother, we have:

- *asymptotically optimal relaxation parameter is* $\omega = \frac{1}{2}$
- *always good smoothing in time (semi-coarsening is always possible)*
- *for* $\frac{\Delta t}{\Delta x^2} \geq C$ *also good smoothing in space*
- *one V-cycle in space suffices to invert* A_j

A New Space-Time Multigrid Method

Discretize the heat equation $\frac{\partial u}{\partial t} = \Delta u + f$ in space and time

$$\begin{pmatrix} A_1 & & & & \\ B_2 & A_2 & & & \\ & B_3 & A_3 & & \\ & & \ddots & \ddots & \\ & & & B_n & A_n \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \\ \vdots \\ \boldsymbol{u}_n \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \boldsymbol{f}_3 \\ \vdots \\ \boldsymbol{f}_n \end{pmatrix}$$

and use block Jacobi smoother for the space-time system with standard restriction and prolongation operators.

Comparison of solution times in seconds using one processor:

dof	forward substitution	space-time multigrid
2 304	3.30	0.06
23 296	3.69	1.02
218 880	9.80	13.19
1 912 576	95.27	136.99
16 015 104	1031.43	1155.12
131 120 896	9970.89	10416.90

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)

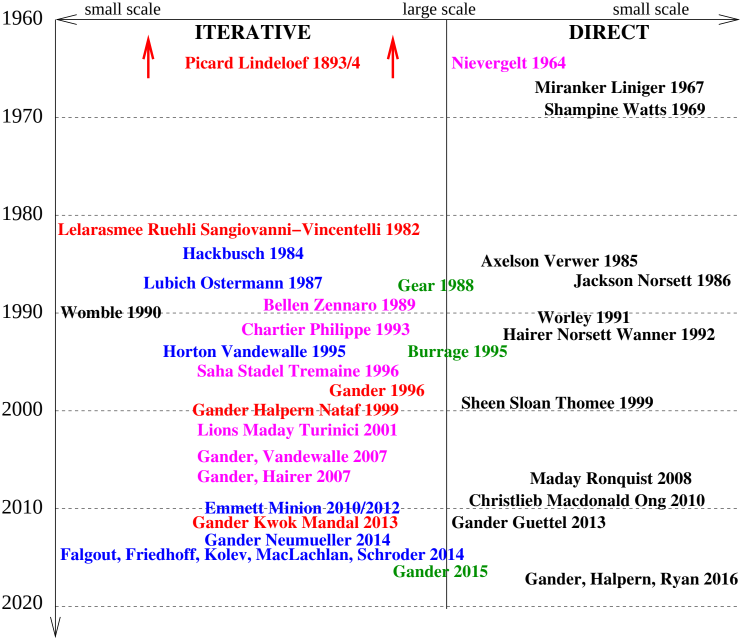
3D Heat Equation Strong Scaling Results

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

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

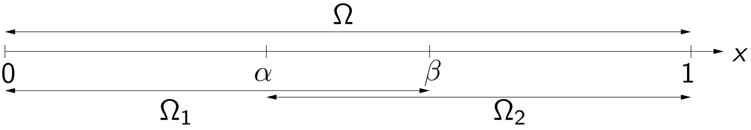
Waveform Relaxation

Time Parallel Methods Over the Course of Time



Schwarz Waveform Relaxation Methods (SWR)

$$\partial_{tt} u = c^2 \Delta u \quad \text{in } \Omega \times (0, T)$$



$$\begin{aligned} \partial_{tt} u_1^n &= c^2 \partial_{xx} u_1^n \text{ in } \Omega_1 \times (0,T) \\ u_1^n(0,t) &= 0 \\ u_1^n(\beta,t) &= u_2^{n-1}(\beta,t) \end{aligned} \qquad \begin{aligned} \partial_{tt} u_2^n &= c^2 \partial_{xx} u_2^n \text{ in } \Omega_2 \times (0,T) \\ u_2^n(1,t) &= 0 \\ u_2^n(\alpha,t) &= u_1^n(\alpha,t) \end{aligned}$$

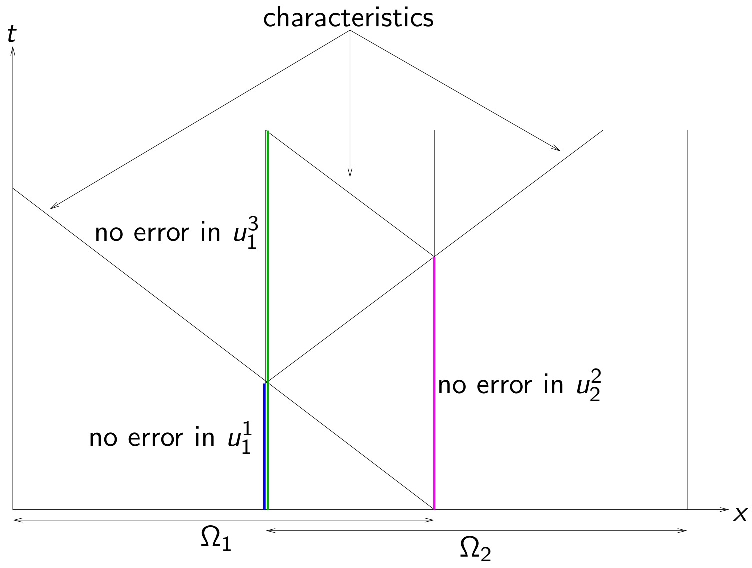
Theorem (Wave equation (G 1997))

Convergence in a finite number of steps, i.e. when

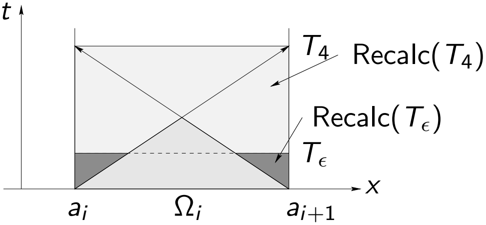
$$n \geq \frac{Tc}{\beta - \alpha}.$$

Also for many subdomains in n-d (G, Halpern 2004)

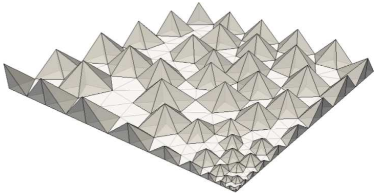
Graphical Convergence Proof



Scalability of SWR without coarse grid

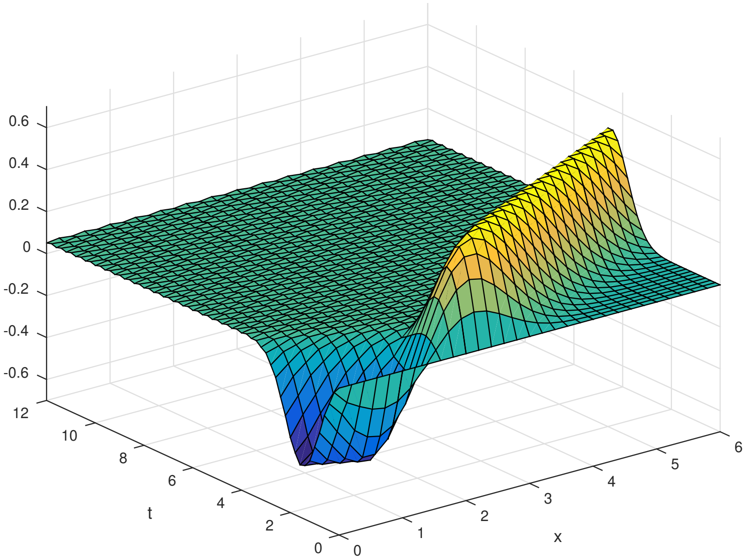


Best current variant based on tent pitching without iteration (Gopalakrishnan, Schöberl and Wintersteiger 2017)

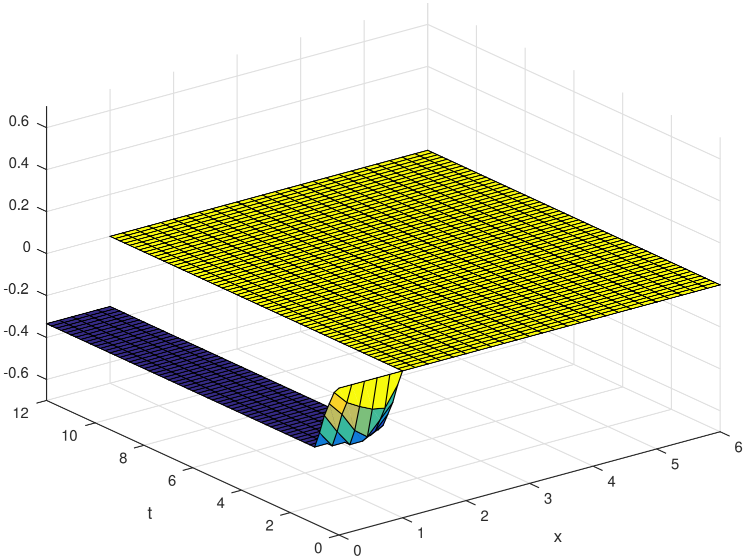


Optimized transmission conditions (G, Halpern, Nataf 1999)
$\Longrightarrow$ Sweeping Preconditioner (Engquist, Ying 2011)

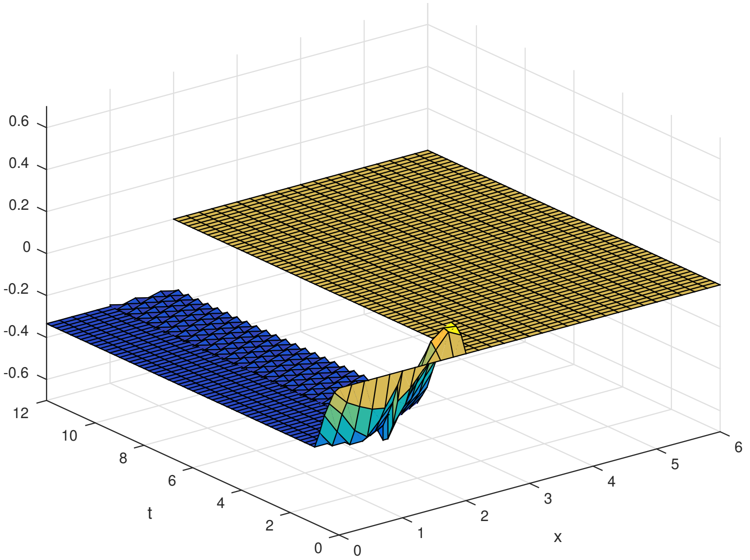
Example: SWR run in a sweeping fashion



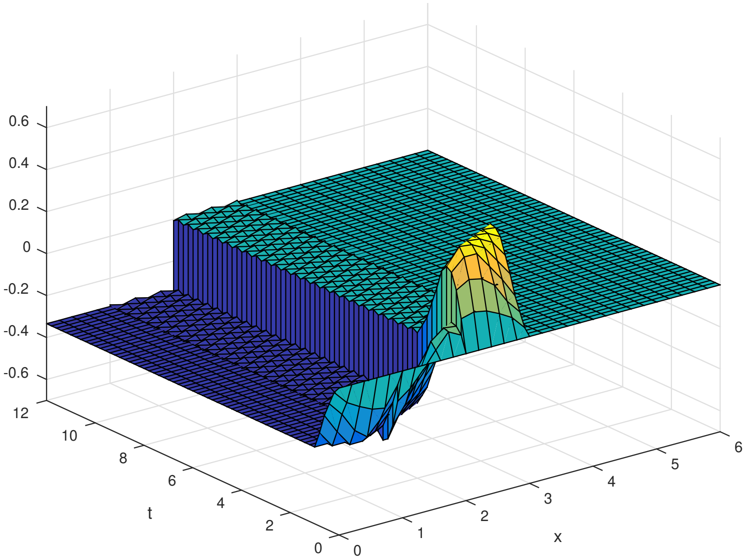
Homogeneous case: Forward Sweep



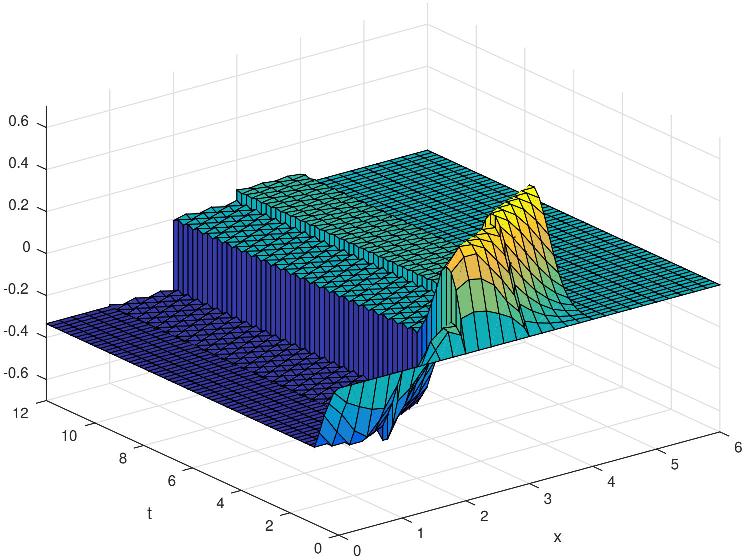
Homogeneous case: Forward Sweep



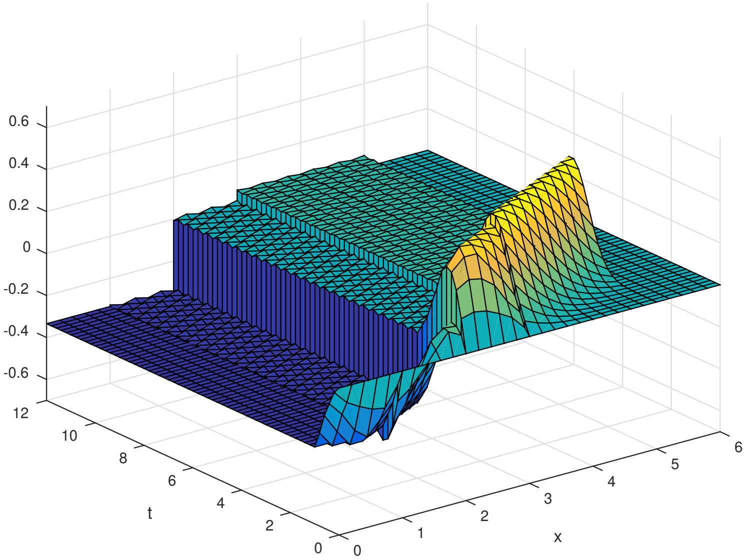
Homogeneous case: Forward Sweep



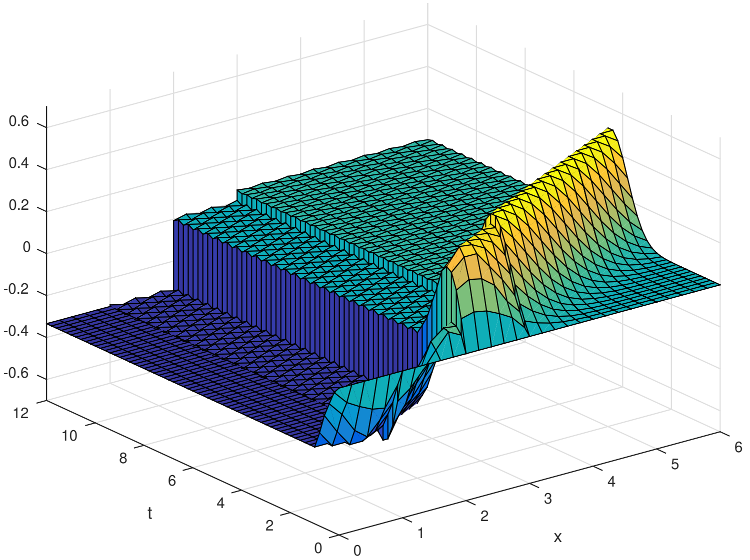
Homogeneous case: Forward Sweep



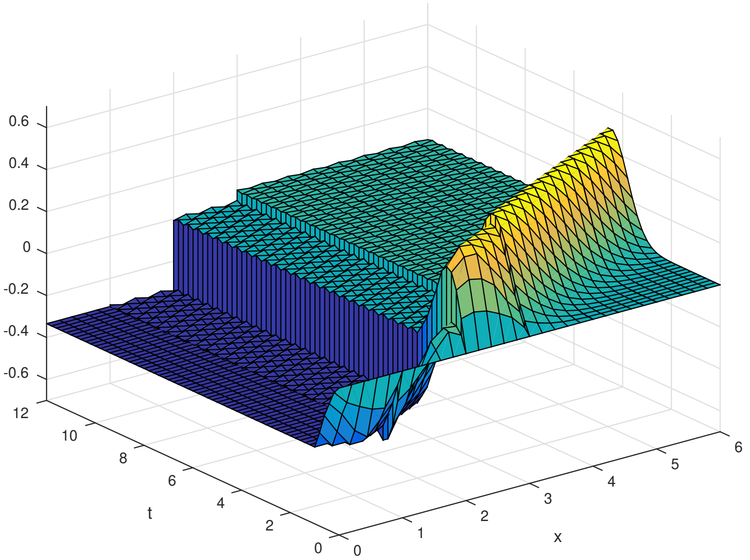
Homogeneous case: Forward Sweep



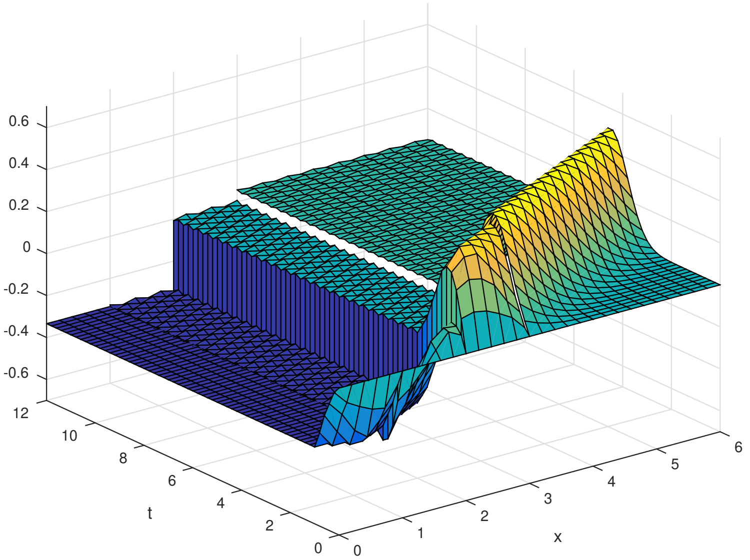
Homogeneous case: Forward Sweep



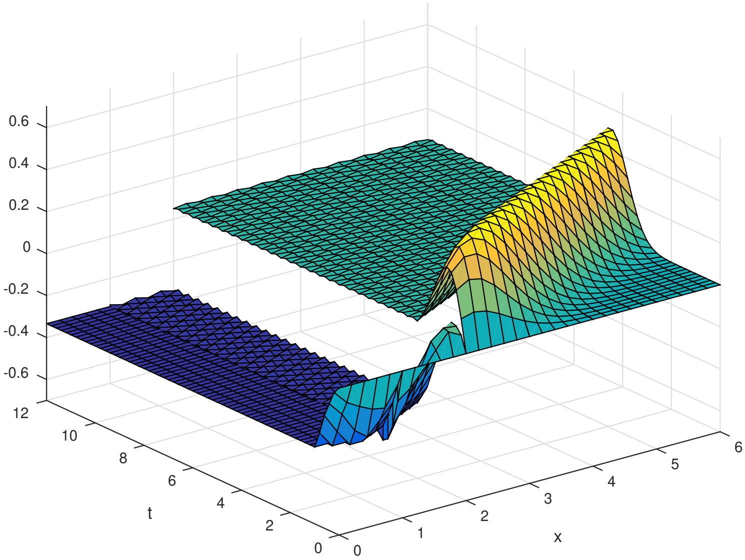
Homogeneous case: Backward Sweep



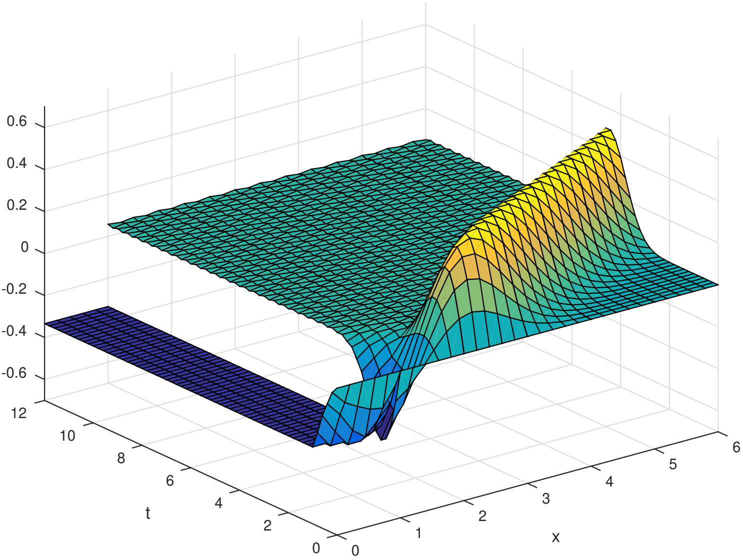
Homogeneous case: Backward Sweep



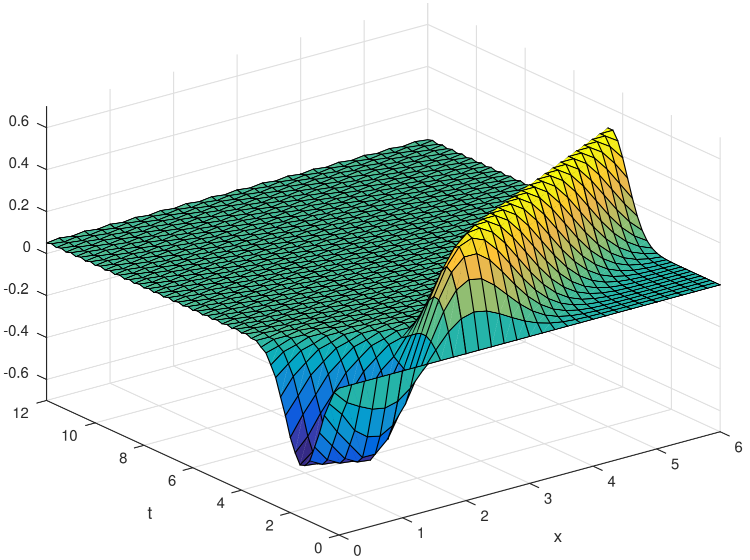
Homogeneous case: Backward Sweep



Homogeneous case: Backward Sweep

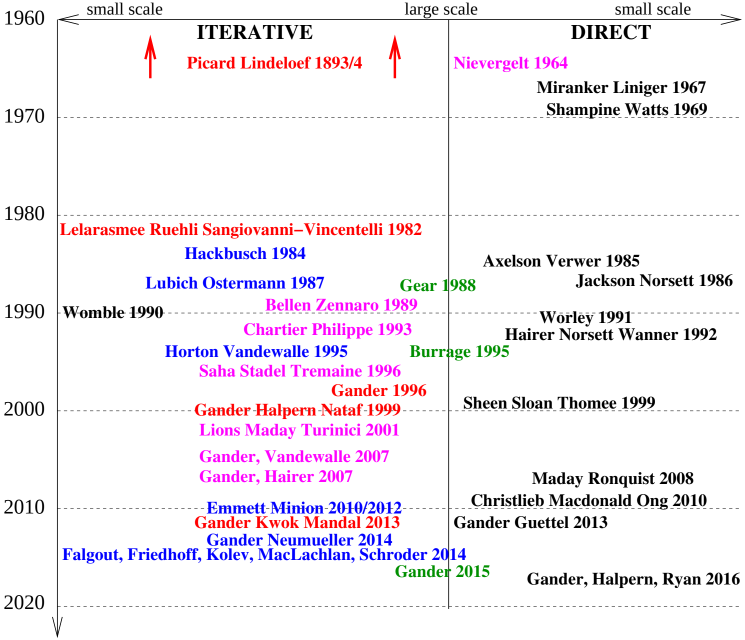


Homogeneous case: Backward Sweep



Direct Time Parallel Methods

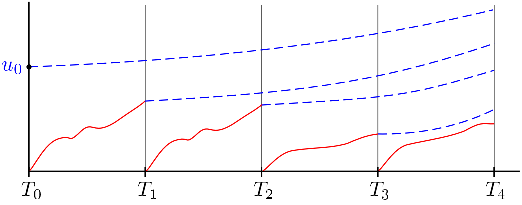
Time Parallel Methods Over the Course of Time



The ParaExp Algorithm

For linear problems $\boldsymbol{u}'(t) = A\boldsymbol{u}(t) + \boldsymbol{g}(t), \quad \boldsymbol{u}(0) = \boldsymbol{u}_0$

ParaExp: use overlapping decomposition



Solve first non-overlapping inhomogeneous problems

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

and then overlapping homogeneous problems

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

The solution is then obtained by summation:

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

Wave Equation Experiment

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

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 %

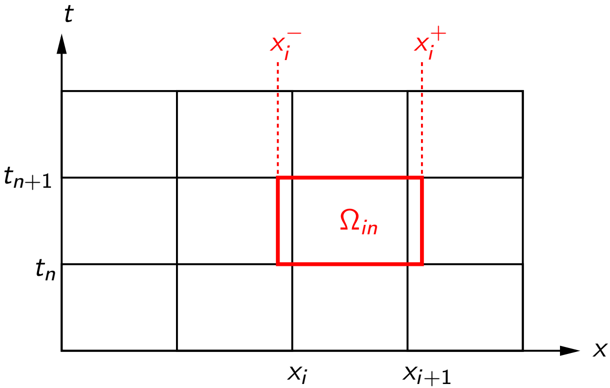
$\Delta x = \frac{1}{101}$, $\Delta t_0 = \min\{5 \cdot 10^{-4}/\alpha, 1.5 \cdot 10^{-3}/f\}$, RK45 and Chebyshev exponential integrator, 8 processors

Combinations

Parareal Schwarz Waveform Relaxation

Model problem: $\partial_t u = \partial_{xx} u$ in $\Omega = (0,1) \times (0,T)$

Decomposition of the space-time domain:



$$\Omega_{in} := (x_i - \frac{L}{2}, x_{i+1} + \frac{L}{2}) \times (t_n, t_{n+1})$$

Parareal Schwarz Waveform Relaxation

Given an initial condition u_0 and boundary conditions g^- and g^+, we define $F_{in}(u_0, g^-, g^+)$ and $G_{in}(u_0, g^-, g^+)$ to be fine and coarse approximations of the solution at $t = t_{n+1}$ of

$$\begin{aligned}
\partial_t u &= \partial_{xx} u, & x \in (x_i^-, x_i^+),\ t \in (t_n, t_{n+1}) \\
u(x, t_n) &= u_0 & x \in (x_i^-, x_i^+) \\
\mathcal{B}_i^- u(x_i^-, t) &= g^- & t \in (t_n, t_{n+1}) \\
\mathcal{B}_i^+ u(x_i^+, t) &= g^+ & t \in (t_n, t_{n+1})
\end{aligned}$$

A Parareal Schwarz Waveform Relaxation Algorithm:
Given initial conditions $u_{0,in}^k(x)$ and boundary conditions $\mathcal{B}_i^- u_{i-1,n}^k(t)$ and $\mathcal{B}_i^+ u_{i+1,n}^k(t)$, we compute

1. All $u_{in}^{k+1} := F_{in}(u_{0,in}^k, \mathcal{B}_i^- u_{i-1,n}^k, \mathcal{B}_i^+ u_{i+1,n}^k)$ in parallel
2. Compute new initial conditions using

$$u_{0,i,n+1}^{k+1} = F(u_{0,in}^k, \mathcal{B}_i^- u_{i-1,n}^k, \mathcal{B}_i^+ u_{i+1,n}^k)$$

$$+G(u_{0,in}^{k+1}, \mathcal{B}_i^- u_{i-1,n}^{k+1}, \mathcal{B}_i^+ u_{i+1,n}^{k+1}) - G(u_{0,in}^k, \mathcal{B}_i^- u_{i-1,n}^k, \mathcal{B}_i^+ u_{i+1,n}^k)$$

Convergence Result

Theorem (G., Song (2017))

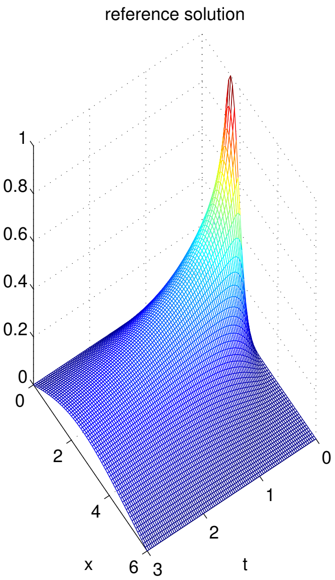
On a bounded time interval $(0, T)$, *the parareal Schwarz waveform relaxation algorithm applied to the heat equation converges superlinearly at a rate governed by the heat kernel,*

$$||e^k|| \leq erfc(\frac{kL}{\sqrt{T}})||e^0||.$$

References:

- Maday, Turinici (2005): overlapping Schwarz/Neumann Neumann for the fine solver
- Guetat (2011): PhD thesis
- G, Jiang, Li (2013): Space-time iteration
- Song, Jiang (2015): Time parallel variant
- G. Jiang, Song (2017): Superlinear convergence estimate

Parareal Schwarz WR Numerical Example



Model problem:

1D Heat equation

$$\partial_t u = \partial_{xx} u$$

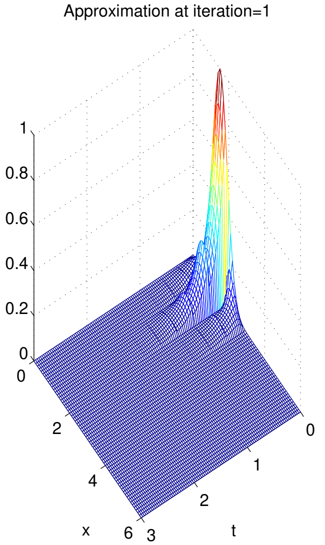
on $\Omega = (0,6) \times (0,T),\ T = 3$

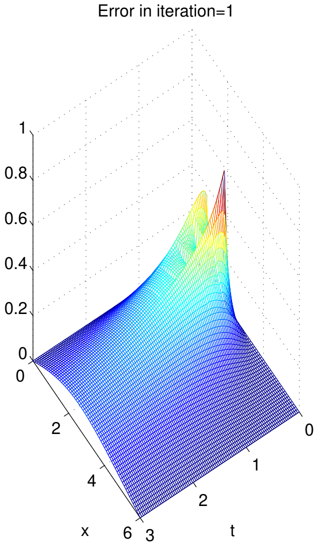
Space time decomposition into 6 spatial subdomains, and 10 time subdomains

Discretization with $\Delta x = \frac{1}{10}$, $\Delta t = \frac{3}{100}$

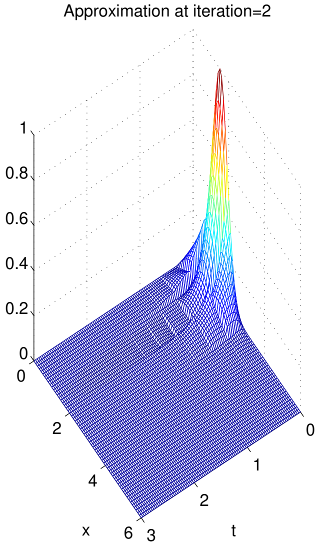
Overlap in space of $2\Delta x$

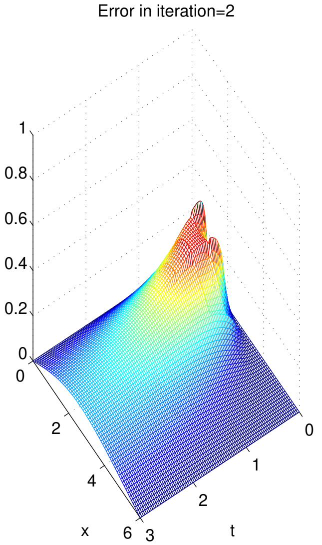
Parareal Schwarz WR: Iteration 1



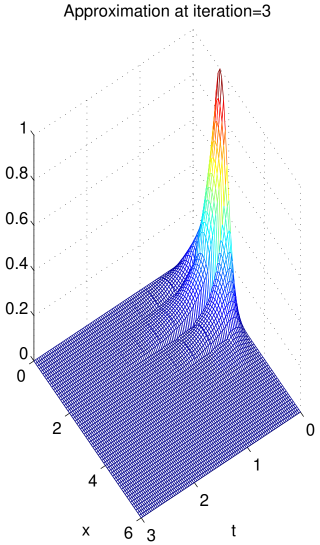


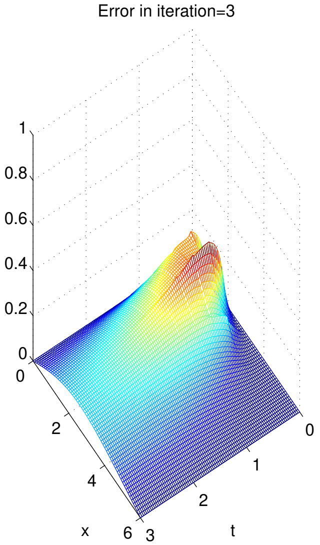
Parareal Schwarz WR: Iteration 2



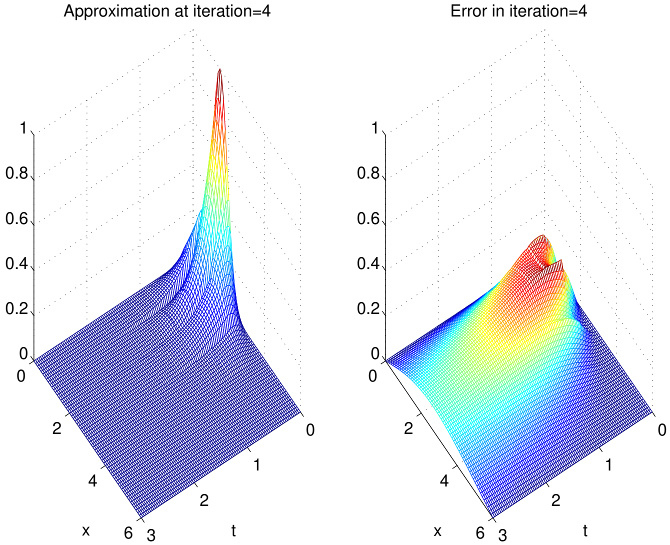


Parareal Schwarz WR: Iteration 3

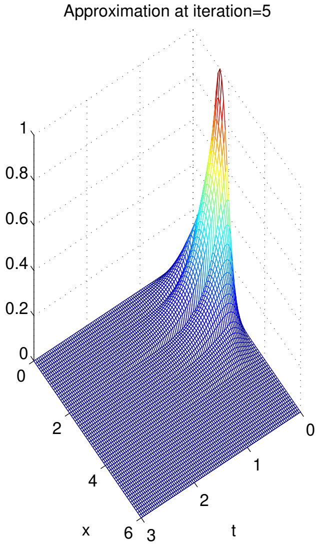


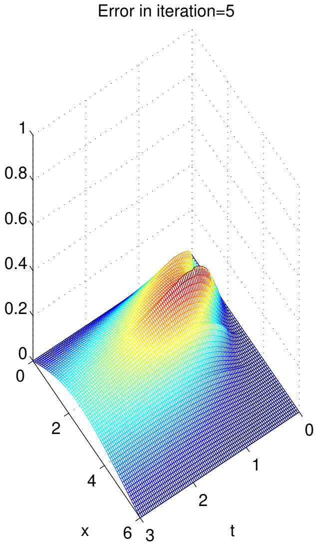


Parareal Schwarz WR: Iteration 4

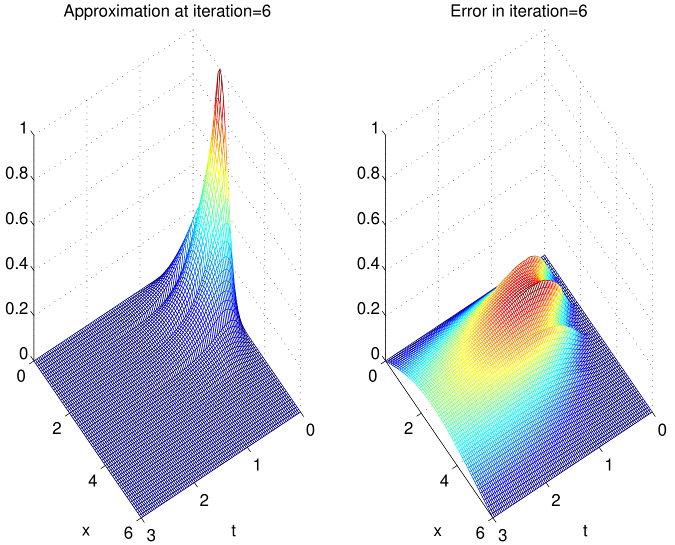


Parareal Schwarz WR: Iteration 5

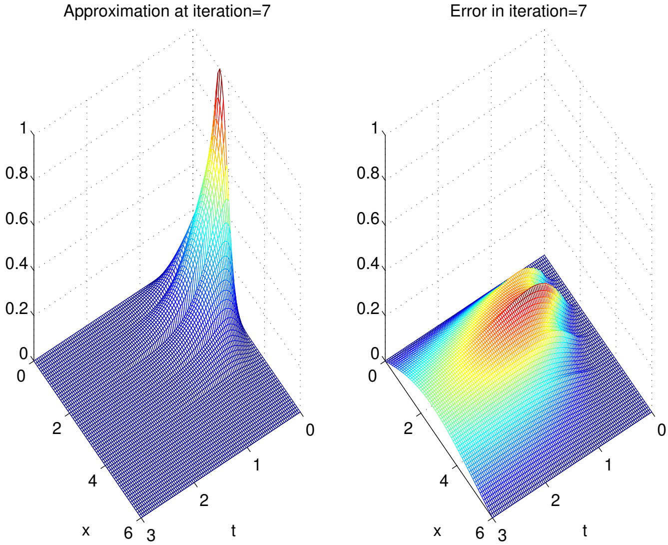




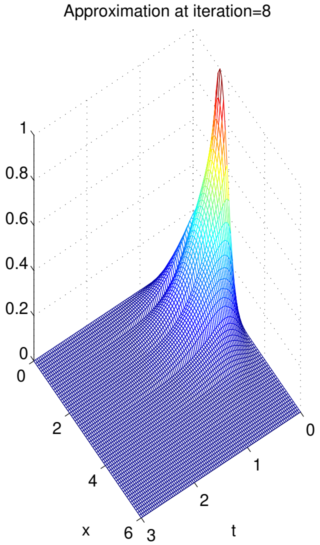
Parareal Schwarz WR: Iteration 6

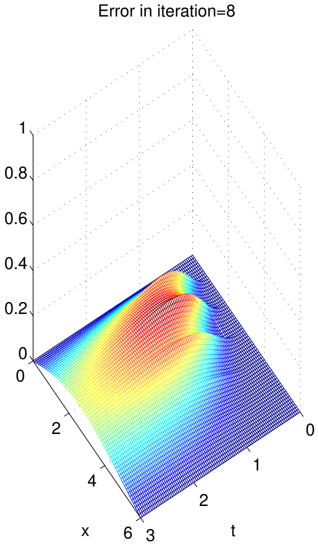


Parareal Schwarz WR: Iteration 7

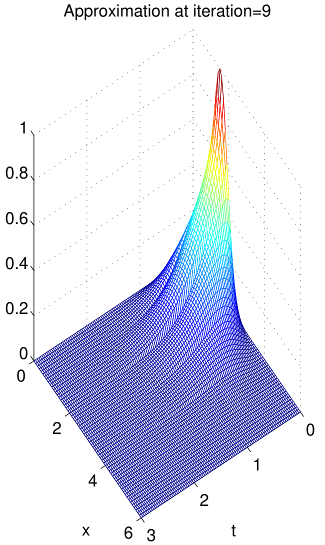


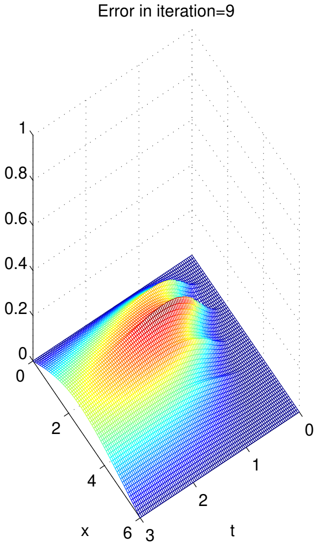
Parareal Schwarz WR: Iteration 8



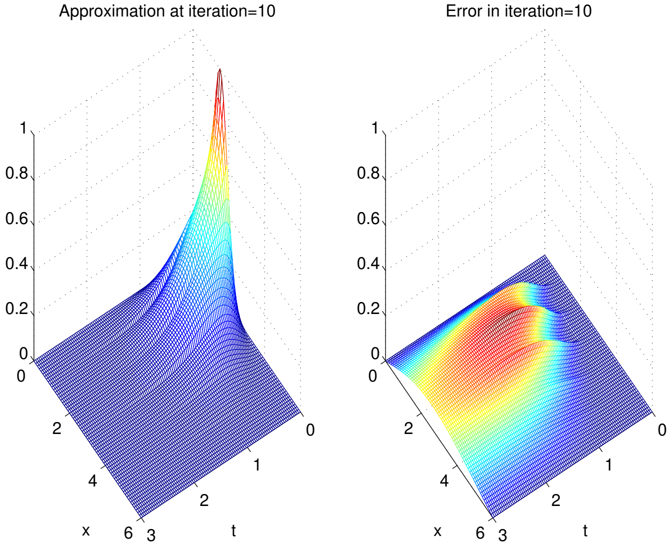


Parareal Schwarz WR: Iteration 9

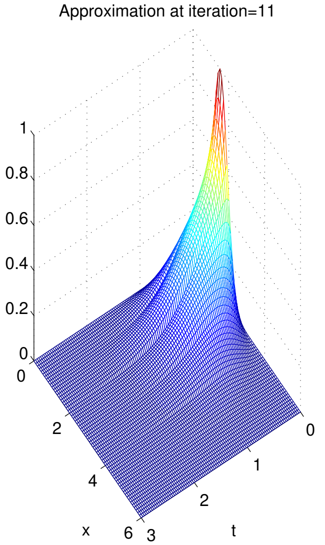


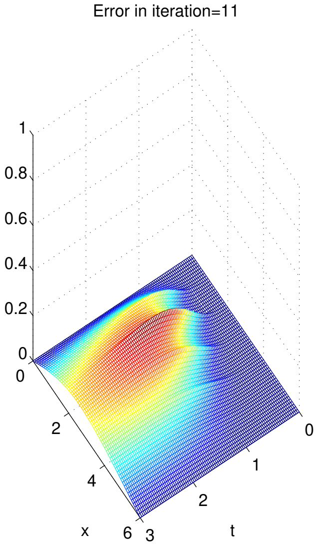


Parareal Schwarz WR: Iteration 10

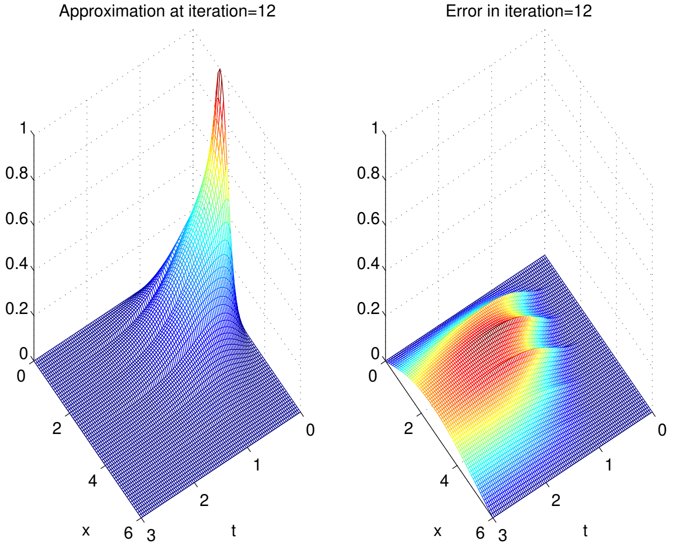


Parareal Schwarz WR: Iteration 11

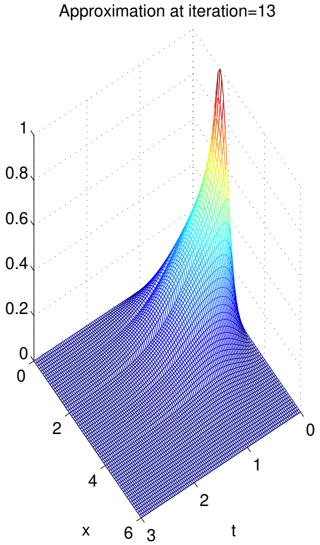


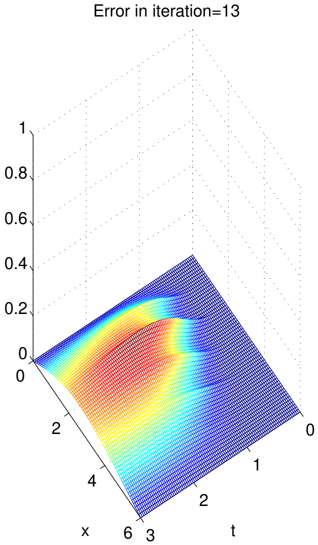


Parareal Schwarz WR: Iteration 12

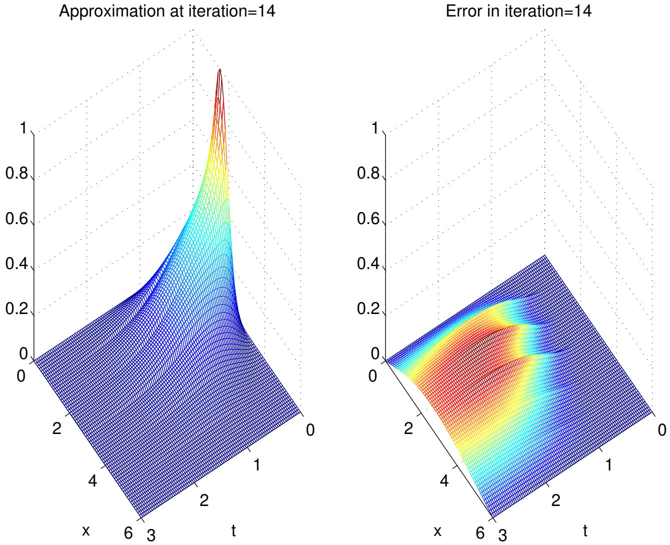


Parareal Schwarz WR: Iteration 13

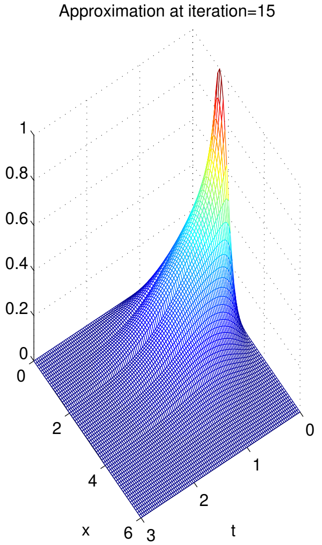


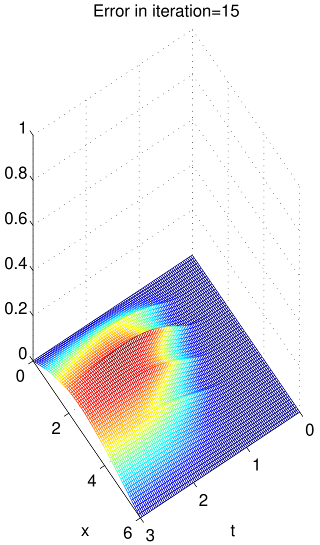


Parareal Schwarz WR: Iteration 14

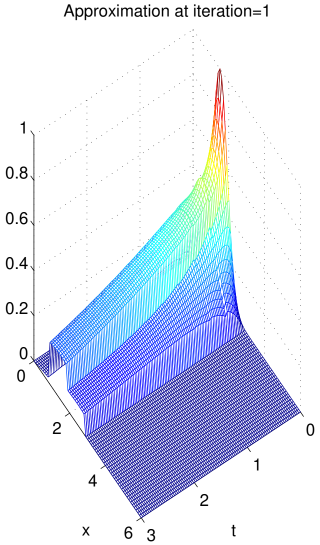


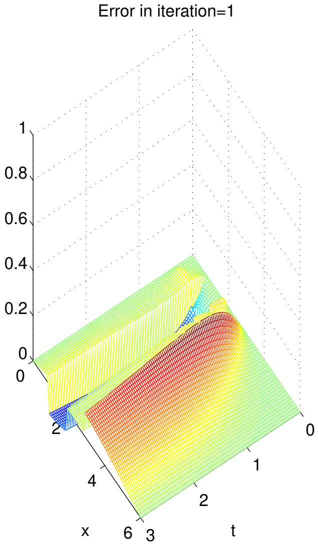
Parareal Schwarz WR: Iteration 15



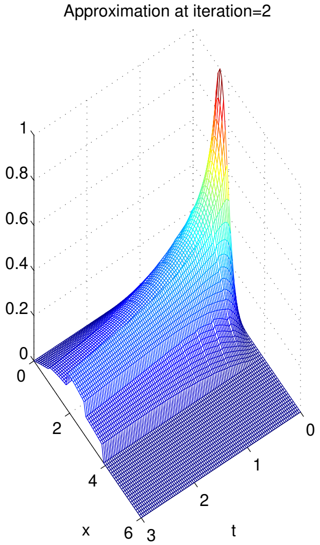


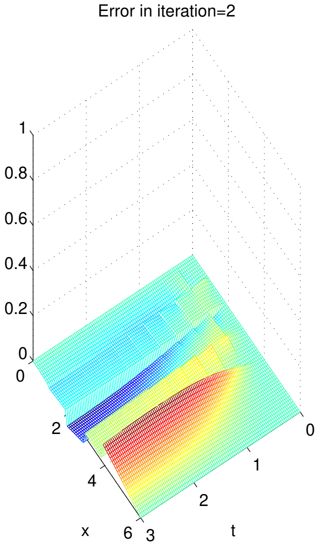
Optimized Parareal Schwarz WR: Iteration 1



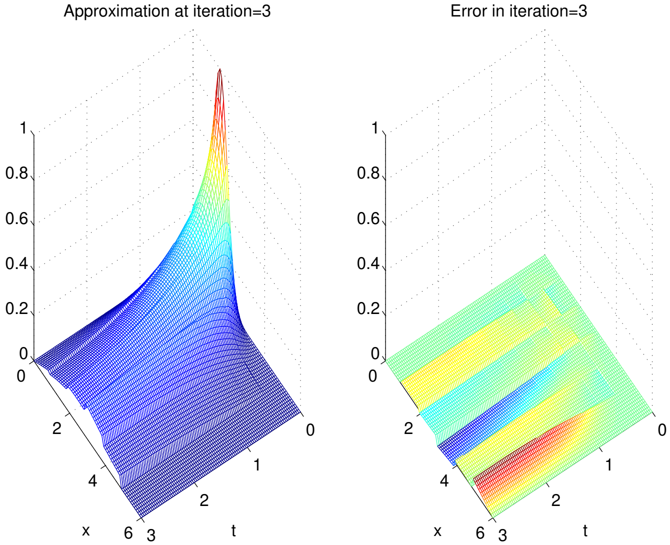


Optimized Parareal Schwarz WR: Iteration 2

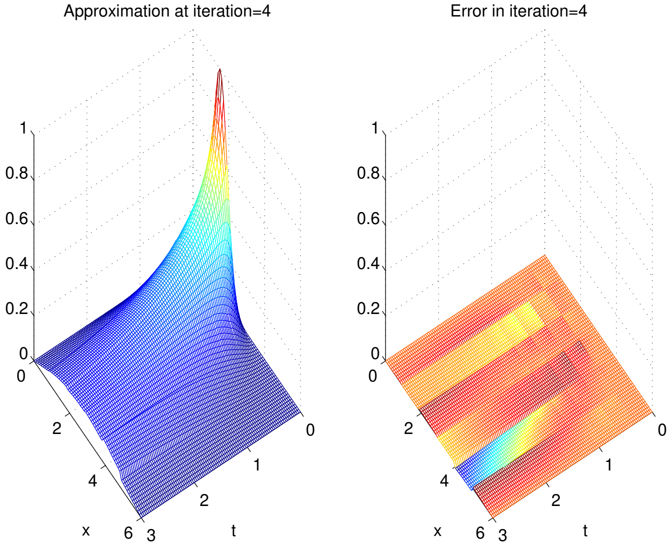




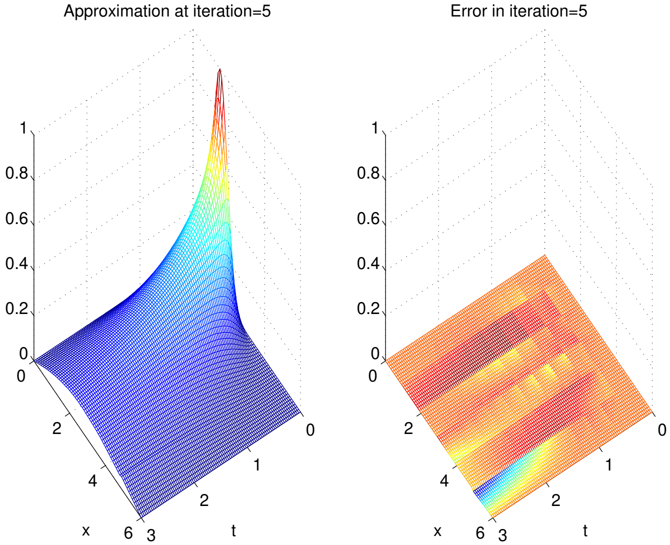
Optimized Parareal Schwarz WR: Iteration 3



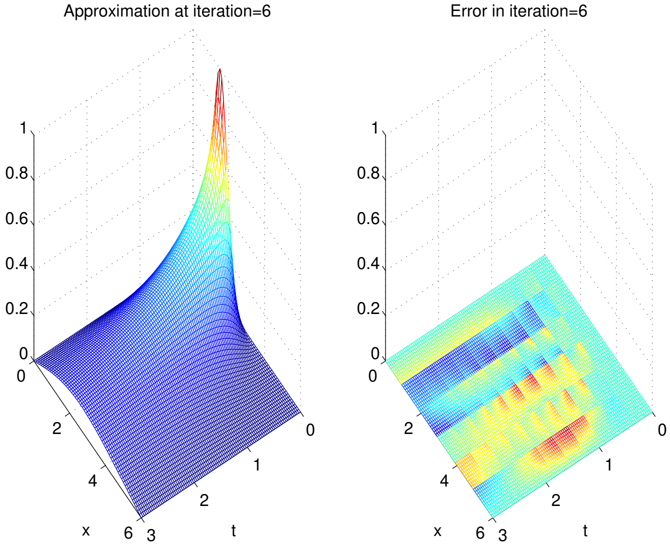
Optimized Parareal Schwarz WR: Iteration 4



Optimized Parareal Schwarz WR: Iteration 5

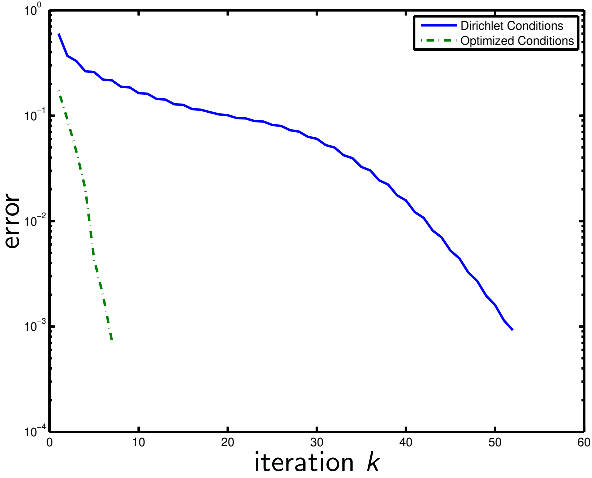


Optimized Parareal Schwarz WR: Iteration 6



Convergence Behavior of PSWR

Convergence comparison between Dirichlet and optimized transmission conditions in space:



Summary

- Time parallelization is currently a very active area of research
- Four classes of methods:
  1. Multiple shooting methods (e.g. parareal)
  2. Space-time multigrid methods
  3. Waveform relaxation methods based on Domain Decomposition (e.g. Schwarz waveform relaxation)
  4. Direct time parallel method (e.g. ParaExp, Diagonalization)
- For diffusive problems all approaches work, but often give less speedup than parallelization in space.
- For wave propagation problems, only certain methods work (e.g. ParaExp, Diagonalization, RIDC)

Review is available at www.unige.ch/∼gander