Time Domain Decomposition Methods

Martin J. Gander
martin.gander@math.unige.ch

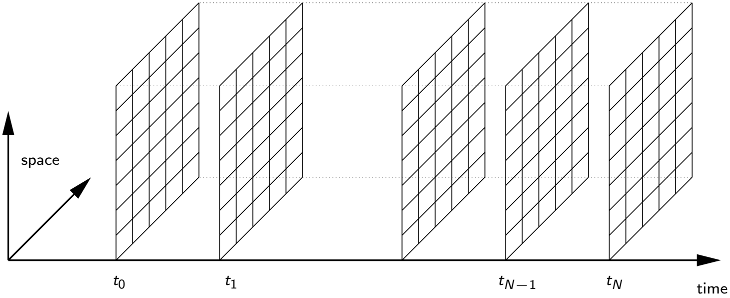
University of Geneva

July 7th, 2006

Evolution Problems

Systems of ordinary differential equations $u' = f(u)$,

or partial differential equations $\dfrac{\partial u}{\partial t} = L(u) + f$.



Is it possible to do useful computations at future time steps, before earlier time steps are known ?

Multiple shooting for boundary value problems

For the model problem

$$u'' = f(u), \quad u(0) = u^0, \quad u(1) = u^1, \qquad x \in [0,1]$$

one splits the spatial interval into subintervals $[0, \frac{1}{3}]$, $[\frac{1}{3}, \frac{2}{3}]$, $[\frac{2}{3}, 1]$, and then solves on each subinterval

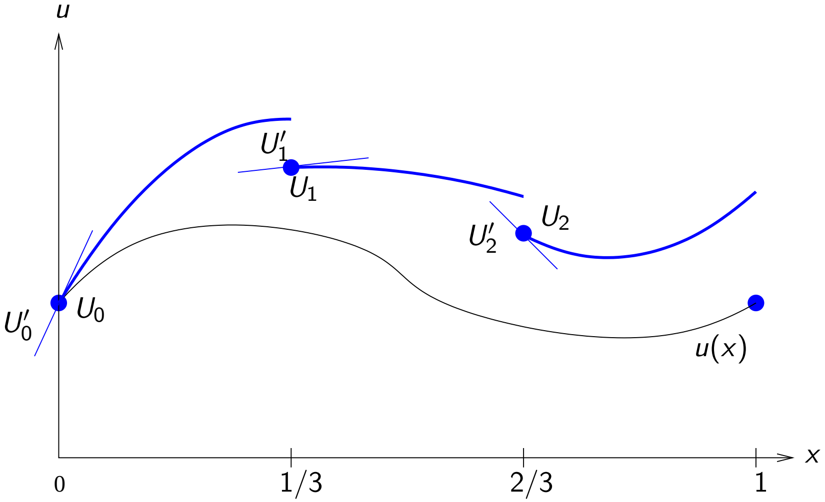
$$\begin{array}{rclcrclcrcl} u_0'' &=& f(u_0), && u_1'' &=& f(u_1), && u_2'' &=& f(u_2),\\ u_0(0) &=& U_0, && u_1(\frac{1}{3}) &=& U_1, && u_2(\frac{2}{3}) &=& U_2,\\ u_0'(0) &=& U_0', && u_1'(\frac{1}{3}) &=& U_1', && u_2'(\frac{2}{3}) &=& U_2', \end{array}$$

together with the matching conditions

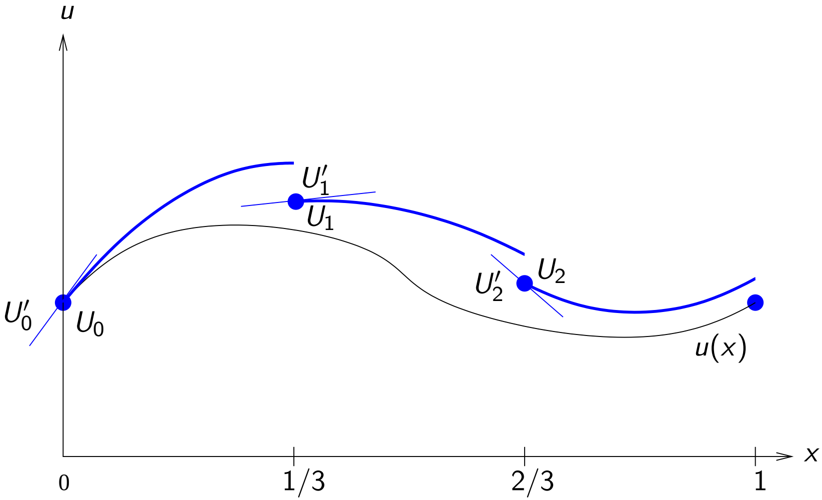
$$\begin{array}{llll} U_0 = u^0, & U_1 = u_0(\frac{1}{3}, U_0, U_0'), & U_2 = u_1(\frac{2}{3}, U_1, U_1'), & \\ & U_1' = u_0'(\frac{1}{3}, U_0, U_0'), & U_2' = u_1'(\frac{2}{3}, U_1, U_1'), & u^1 = u_2(1, U_2, U_2') \end{array}$$

$$\Longleftrightarrow \quad F(\mathbf{U}) = 0, \quad \mathbf{U} = (U_0, U_1, U_2, U_0', U_1', U_2')^T.$$

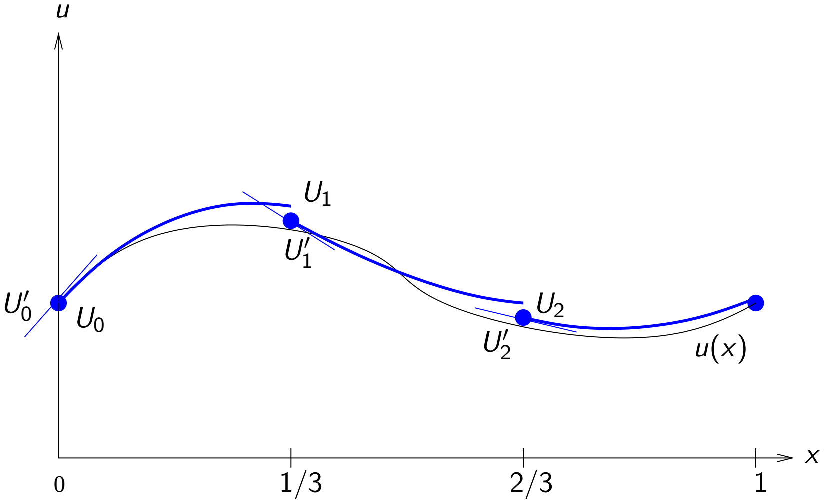
Example: first iteration



Example: second iteration



Example: third iteration



Multiple shooting for initial value problems

For the model problem

$$u' = f(u), \quad u(0) = u^0, \quad t \in [0, 1]$$

one splits the time interval into subintervals $[0, \frac{1}{3}]$, $[\frac{1}{3}, \frac{2}{3}]$, $[\frac{2}{3}, 1]$, and then solves on each subinterval

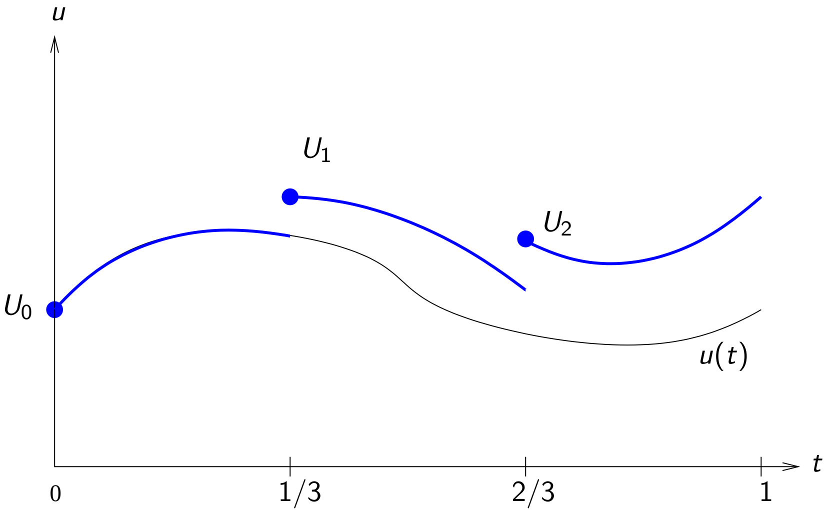
$$\begin{array}{rcl} u_0' &=& f(u_0), \\ u_0(0) &=& U_0, \end{array} \qquad \begin{array}{rcl} u_1' &=& f(u_1), \\ u_1(\frac{1}{3}) &=& U_1, \end{array} \qquad \begin{array}{rcl} u_2' &=& f(u_2), \\ u_2(\frac{2}{3}) &=& U_2, \end{array}$$

together with the matching conditions

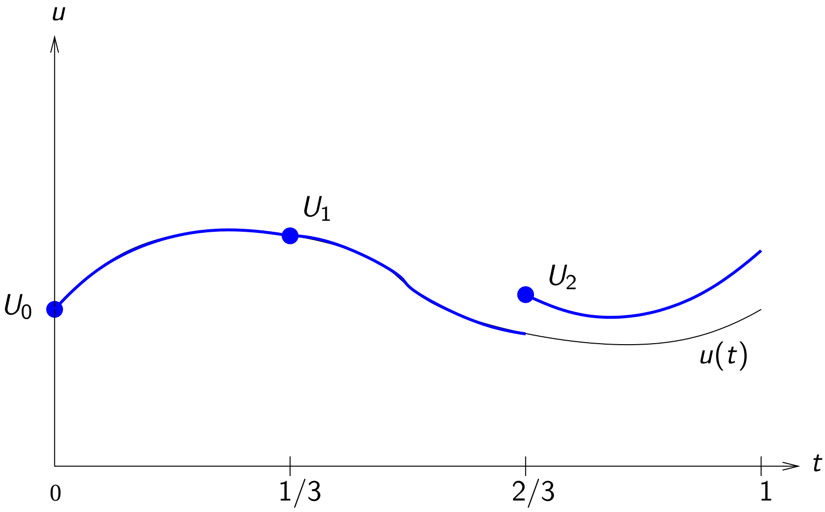
$$U_0 = u^0, \quad U_1 = u_0(\frac{1}{3}, U_0), \quad U_2 = u_1(\frac{2}{3}, U_1)$$

$$\iff \quad F(\mathbf{U}) = 0, \quad \mathbf{U} = (U_0, U_1, U_2)^T.$$

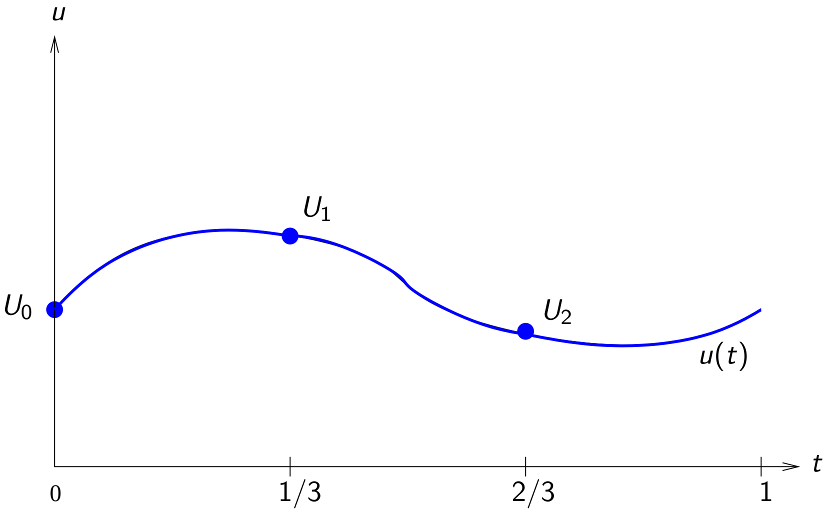
Example: first iteration



Example: second iteration



Example: third iteration



Using Newton's Method

Solving $F(\mathbf{U}) = 0$ with Newton leads to

$$\begin{pmatrix} U_0^{k+1} \\ U_1^{k+1} \\ U_2^{k+1} \end{pmatrix} = \begin{pmatrix} U_0^{k} \\ U_1^{k} \\ U_2^{k} \end{pmatrix} - \begin{bmatrix} 1 & & \\ -\frac{\partial u_0}{\partial U_0}(\frac{1}{3}, U_0^k) & 1 & \\ & -\frac{\partial u_1}{\partial U_1}(\frac{2}{3}, U_1^k) & 1 \end{bmatrix}^{-1} \begin{pmatrix} U_0^k - u^0 \\ U_1^k - u_1(\frac{1}{3}, U_0^k) \\ U_2^k - u_1(\frac{2}{3}, U_1^k) \end{pmatrix}$$

Multiplying through by the matrix, we find the recurrence

$$\begin{aligned} U_0^{k+1} &= u^0, \\ U_1^{k+1} &= u_0(\tfrac{1}{3}, U_0^k) + \tfrac{\partial u_0}{\partial U_0}(\tfrac{1}{3}, U_0^k)(U_0^{k+1} - U_0^k), \\ U_2^{k+1} &= u_1(\tfrac{2}{3}, U_1^k) + \tfrac{\partial u_1}{\partial U_1}(\tfrac{2}{3}, U_1^k)(U_1^{k+1} - U_1^k). \end{aligned}$$

General case with N intervals, $t_n = n\Delta T$, $\Delta T = 1/N$

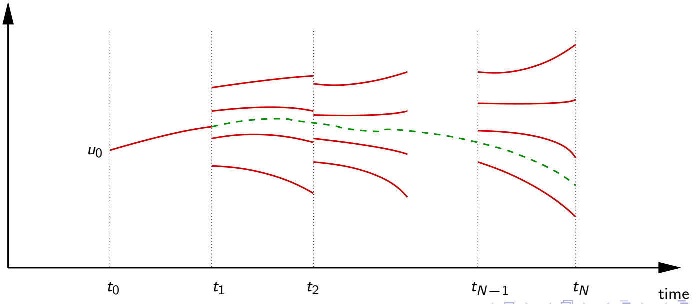
$$U_{n+1}^{k+1} = u_n(t_{n+1}, U_n^k) + \frac{\partial u_n}{\partial U_n}(t_{n+1}, U_n^k)(U_n^{k+1} - U_n^k).$$

History of Time Parallel Algorithms

J. Nievergelt, **Parallel Methods for Integrating Ordinary Differential Equations**. Comm. of the ACM, Vol 7(12), 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."*

$$u' = f(u), \quad u(t_0) = u_0$$

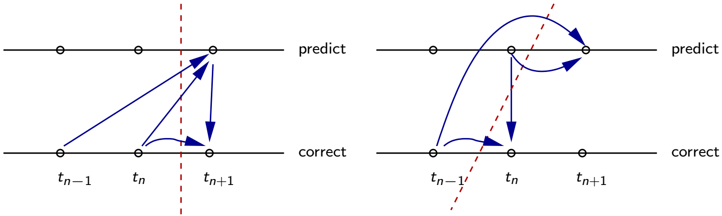


Parallel time stepping

W. Miranker and W. Liniger, **Parallel Methods for the Numerical Integration of Ordinary Differential Equations**. Math. Comp., Vol 21, 1967.

"It appears at first sight that the sequential nature of the numerical methods do not permit a parallel computation on all of the processors to be performed. We say that **the front of computation** *is too narrow to take advantage of more than one processor... Let us consider how we might widen the computation front."*

$$u' = f(u), \quad u(0) = u_0$$



More Recent Space-Time Iterative Methods

- **Waveform Relaxation** Lelarasmee, Ruehli and Sangiovanni-Vincentelli (1982).
- **Parabolic multigrid** Hackbusch (1984); Bastian, Burmeiser and Horton (1990); Oosterlee (1992).
  Multigrid waveform relaxation Lubich and Ostermann (1987); Vandevalle and Piessens (1988).
  Space-time multigrid Horton and Vandevalle (1995)
- **Optimized Schwarz Waveform Relaxation** G, Halpern and Nataf, 1998.
- **Parallel Time Stepping** Womble (1990).
  Deshpande, Malhotra, Douglas and Schultz, Temporal Domain Parallelism: Does it Work (1995) ?

"We show that this approach is not normally useful".

The Parareal Algorithm

J-L. Lions, Y. Maday, G. Turinici, **A "Parareal" in Time Discretization of PDEs, C.R.Acad.Sci. Paris, t.322, 2001.**

The parareal algorithm for the model problem

$$u' = f(u)$$

is defined using two propagation operators:

1. $G(t_2, t_1, u_1)$ is a rough approximation to $u(t_2)$ with initial condition $u(t_1) = u_1$,
2. $F(t_2, t_1, u_1)$ is a more accurate approximation of the solution $u(t_2)$ with initial condition $u(t_1) = u_1$.

Starting with a coarse approximation U_n^0 at the time points $t_1, t_2, \ldots, t_N$, parareal performs for $k = 0, 1, \ldots$ the correction iteration

$$U_{n+1}^{k+1} = G(t_{n+1}, t_n, U_n^{k+1}) + F(t_{n+1}, t_n, U_n^k) - G(t_{n+1}, t_n, U_n^k).$$

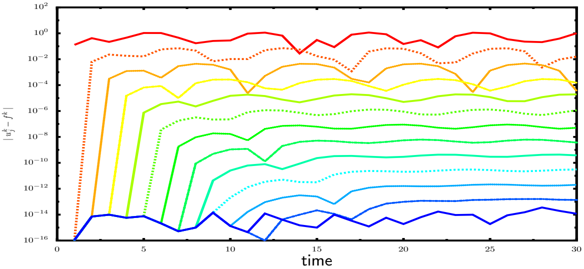
Original Convergence Result for Parareal

Theorem (Lions, Maday, and Turinici, 2001)

If $t_{n+1} - t_n = \Delta T$, G is $O(\Delta T)$ and F is exact, then at iteration k the error for a linear problem is $O(\Delta T^{k+1})$.

Example of the convergence behavior for ΔT fixed:

$u' = -u + \sin t, \quad u(t_0) = 1.0,\ t \in [0, 30]$, trapezoidal rule, $\Delta T = 1.0$ and $\Delta t = 0.01$



Back to Multiple Shooting for IVPs

Theorem (Chartier and Philippe 1993)

If the initial guess $\mathbf{U}^0$ is close enough to the solution, then under appropriate regularity assumptions, the multiple shooting algorithm converges quadratically.

Result (G, Vandevalle 2003)

Approximation of the Jacobian on a coarse time grid leads from

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

to

$$U_{n+1}^{k+1} = F(t_{n+1}, t_n, U_n^k) + G(t_{n+1}, t_n, U_n^{k+1}) - G(t_{n+1}, t_n, U_n^k),$$

which is the parareal algorithm.

Parareal is a Time Multigrid Method

Theorem (G, Vandewalle, 2003)

Let F be method ϕ doing $\bar{m}$ steps and G be method Φ, and let $I_{\Delta t}^{\Delta T}$ be the selection operator at $1, \bar{m}+1, 2\bar{m}+1, \ldots$ and $I_{\Delta T}^{\Delta t}$ be the extension operator with 1 and any values in between.
If in the time multigrid algorithm

- *a block Jacobi smoother is used, $S = EM_{jac}^{-1}$, where $M_{jac} + N_{jac} = M$, and E is the identity, except for zeros at positions $(1,1)$, $(\bar{m}+1, \bar{m}+1)$, $(2\bar{m}+1, 2\bar{m}+1)$ $\ldots$*
- *The initial guess $\mathbf{u}^0$ contains U_n^0 from the parareal initial guess at positions 1, $\bar{m}+1$, $2\bar{m}+1$ $\ldots$*

then it coincides with the parareal algorithm.

A General Convergence Result

For the non-linear IVP $u' = f(u), \quad u(t_0) = u_0$.

Theorem (G, Hairer 2005)

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

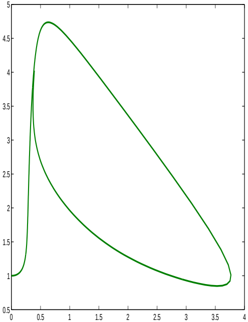
$$|G(t + \Delta T, t, x) - G(t + \Delta T, t, y)| \leq (1 + C_2 \Delta T)|x - y|,$$

then

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

Numerical experiments: Brusselator

$$\begin{aligned} \dot{x} &= A + x^2 y - (B+1)x \\ \dot{y} &= Bx - x^2 y, \end{aligned}$$

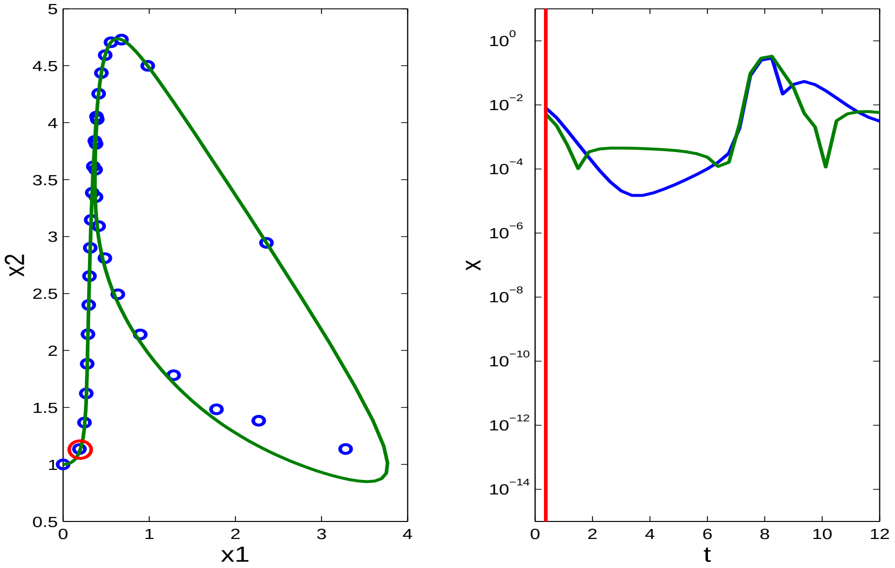


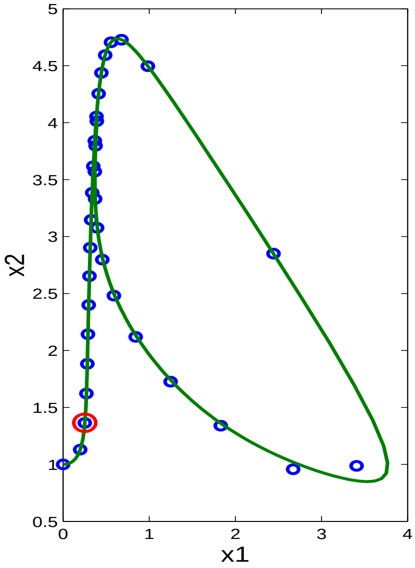
Parameters: $A = 1$ and $B = 3$, $B > A^2 + 1 \Longrightarrow$ limit cycle.

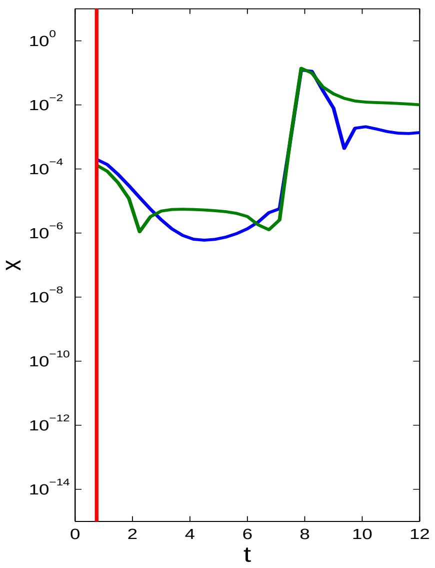
Initial conditions: $x(0) = 0$, $y(0) = 1$.

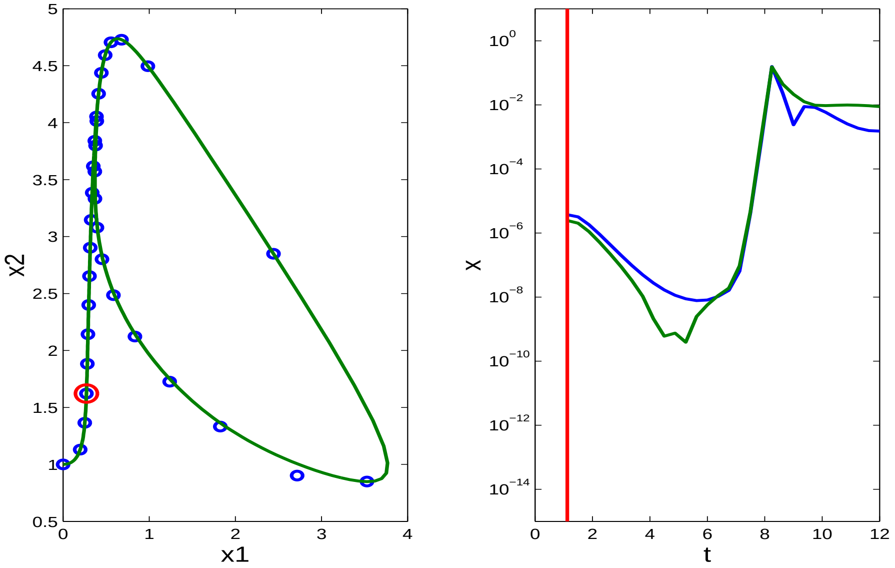
Simulation time: $t \in [0, T = 12]$

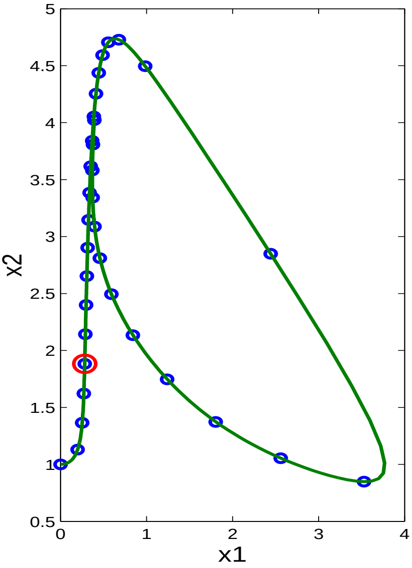
Discretization: Fourth order Runge Kutta, $\Delta T = \frac{T}{32}$, $\Delta t = \frac{T}{320}$.

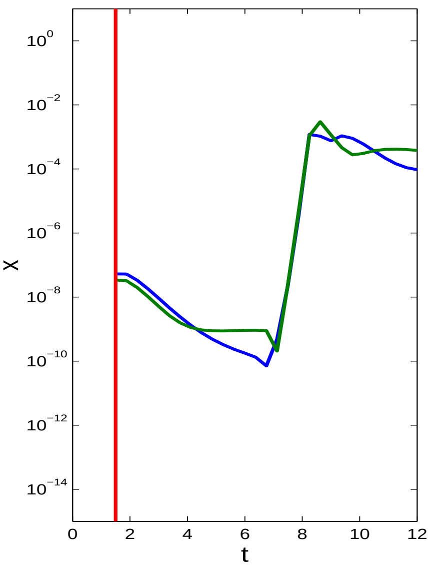

5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
2
3
4
x2
x1
10^0
10^−2
10^−4
10^−6
10^−8
10^−10
10^−12
10^−14
x
0
2
4
6
8
10
12
t

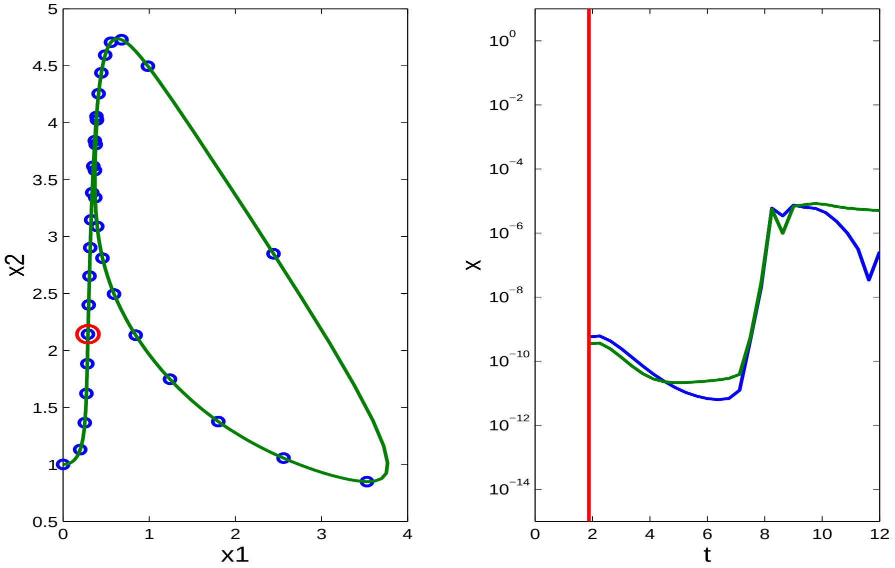

x2
x1
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
2
3
4

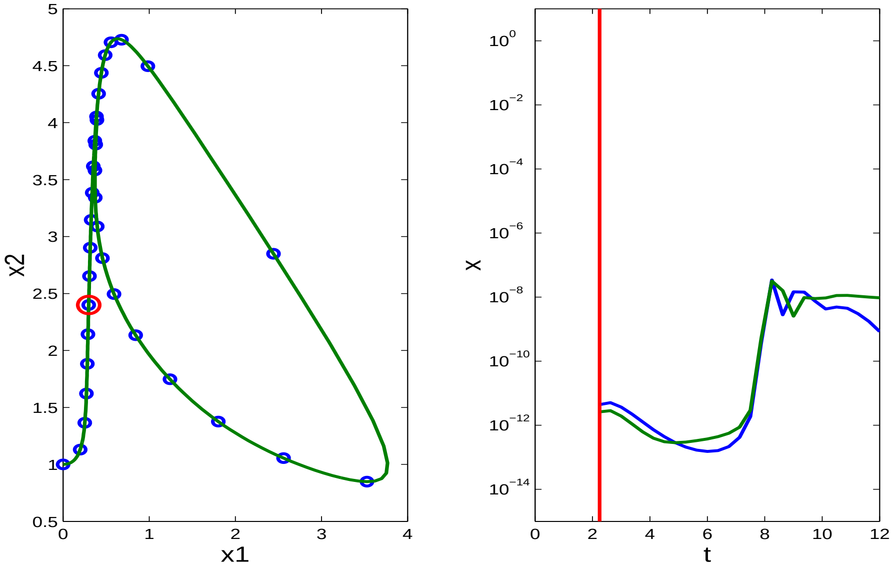

x
t
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
0
2
4
6
8
10
12

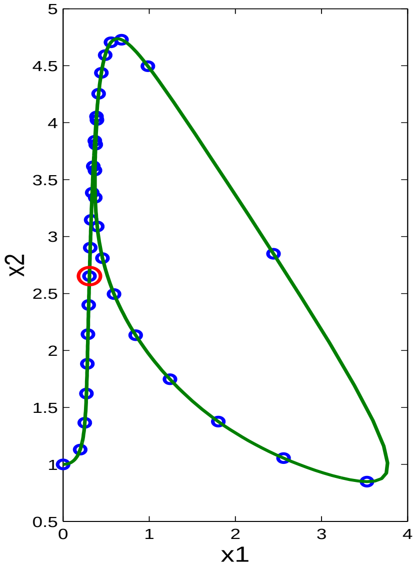

x2
x1
x
t

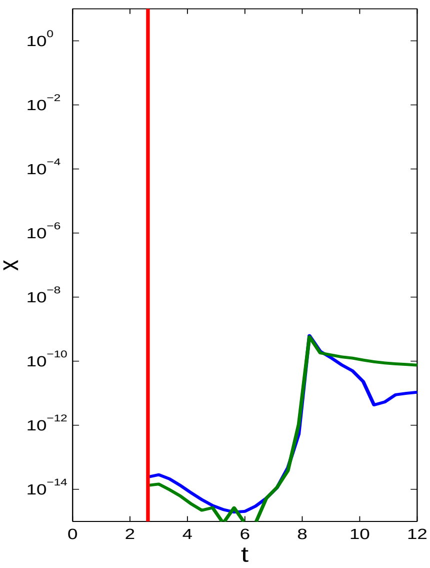

x2
x1
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
2
3
4

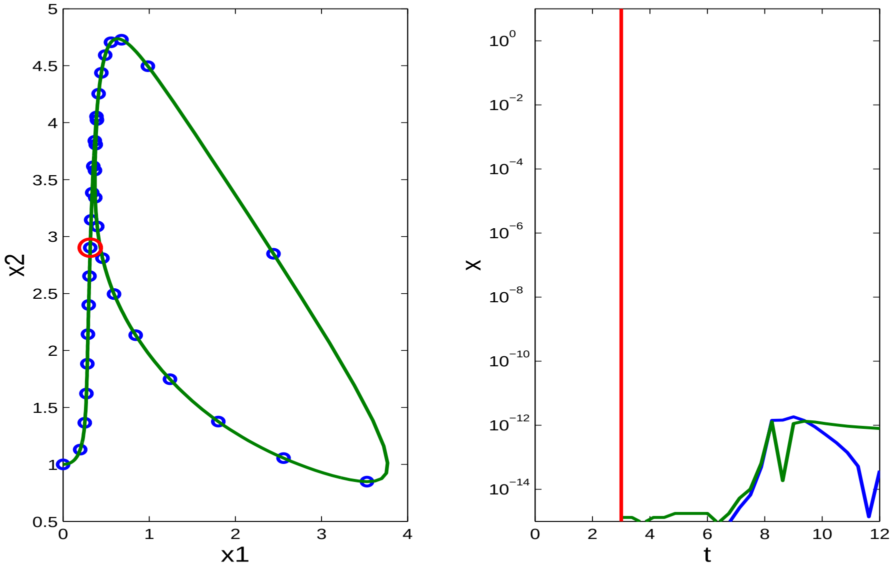

x
t
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
0
2
4
6
8
10
12


x2
x1
x
t
10^0
10^−2
10^−4
10^−6
10^−8
10^−10
10^−12
10^−14


x2
x1
x
t


x2
x1
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
2
3
4


x
t
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
0
2
4
6
8
10
12

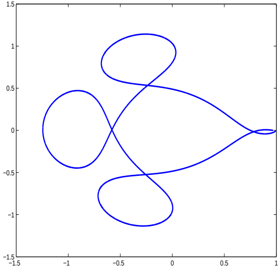

x1
x2
t
x

Numerical experiments: Arenstorf orbit

$$\ddot{x} = x + 2\dot{y} - b\frac{x+a}{D_1} - a\frac{x-b}{D_2}$$

$$\ddot{y} = y - 2\dot{x} - b\frac{y}{D_1} - a\frac{y}{D_2},$$

$$D_1 = ((x+a)^2+y^2)^{(3/2)},\ D_2 = ((x-b)^2+y^2)^{(3/2)}$$



Parameters: $a = 0.012277471,\ b = 1 - a.$

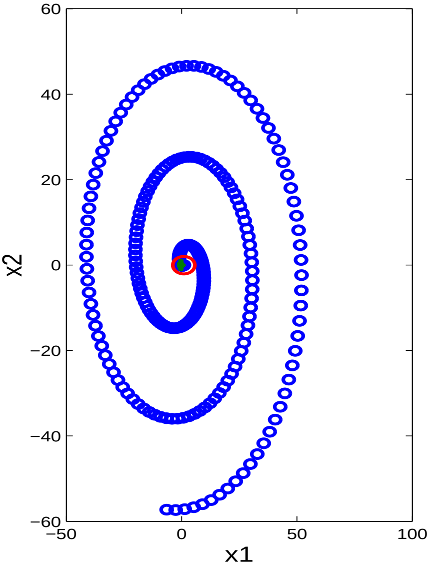
Initial conditions: $x(0) = 0.994,\ \dot{x} = 0,$

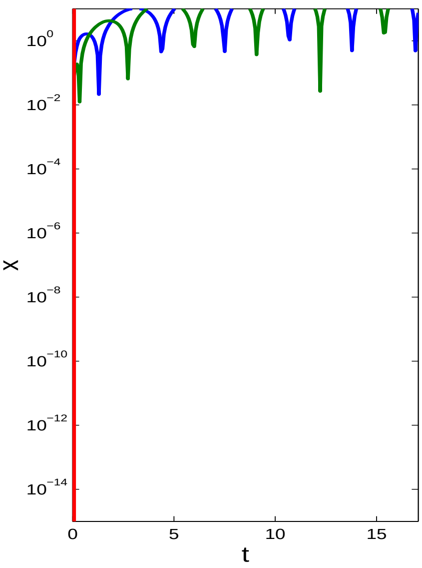
$y(0) = 0,\ \dot{y}(0) = -2.00158510637908$

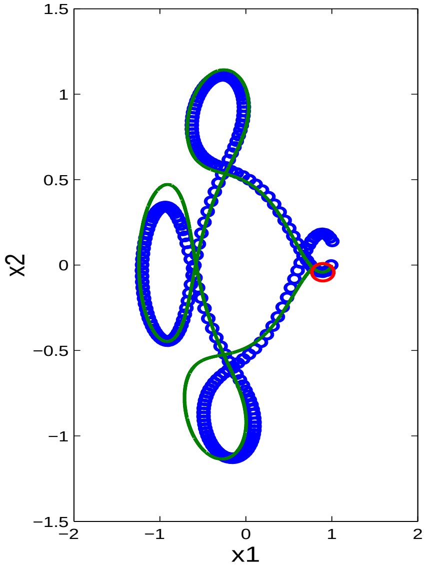
Simulation time: $t \in [0, T = 17.06]$

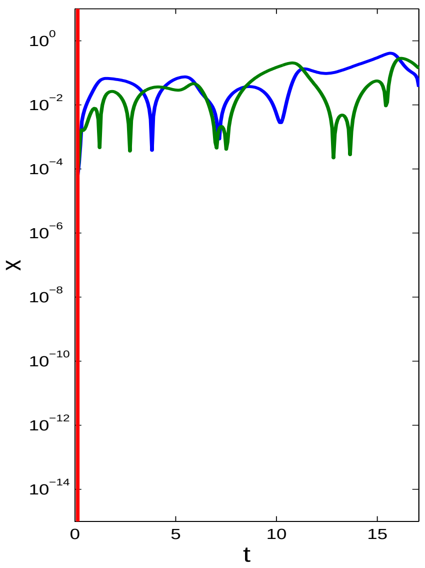
Discretization: Forth order Runge Kutta, $\Delta T = \frac{T}{250},\ \Delta t = \frac{T}{10000}.$

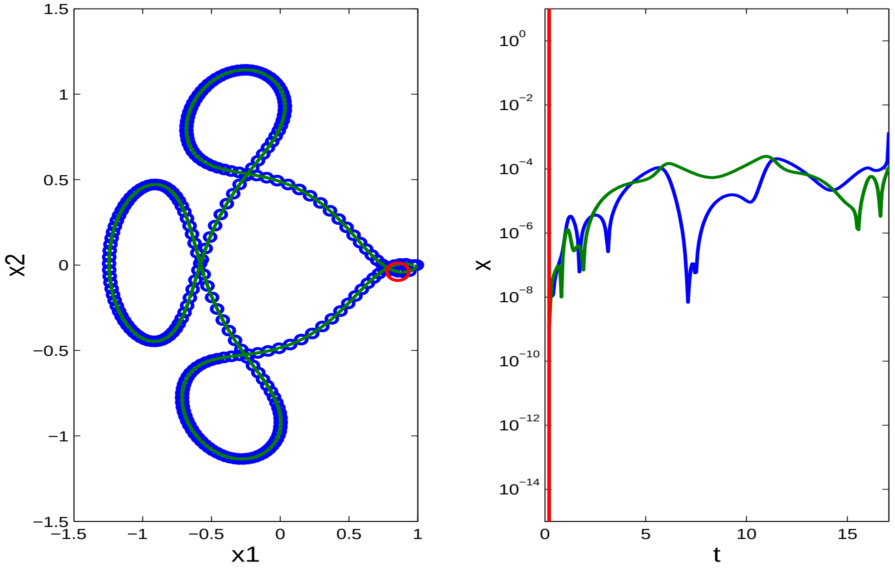
See also Saha, Stadel and Tremaine, a parallel integration method for solar system dynamics, 1997

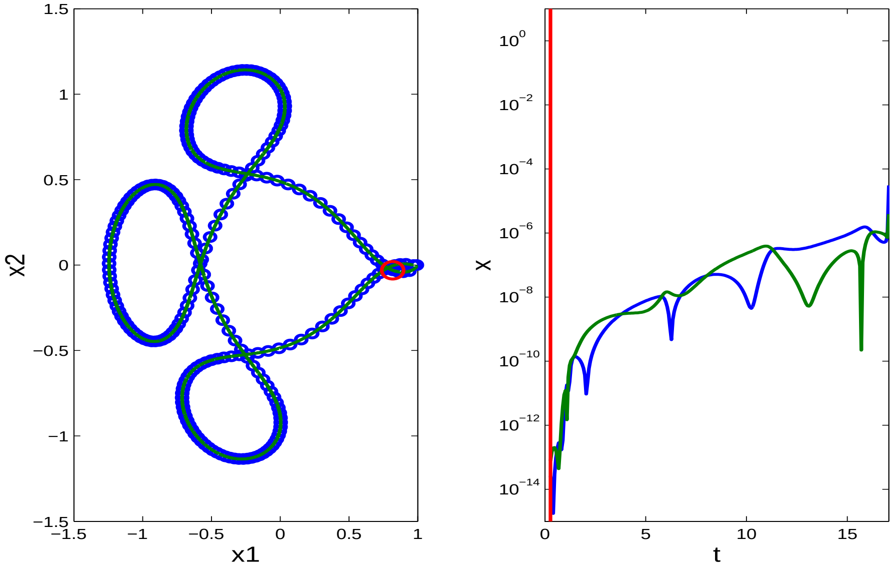

60
40
20
0
−20
−40
−60
−50
0
50
100
x1
x2

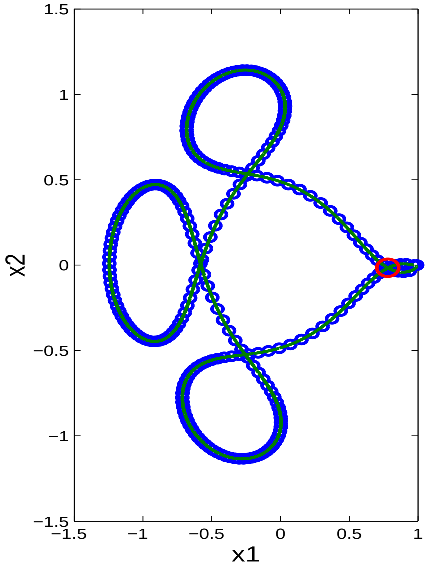

10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
0
5
10
15
t
x

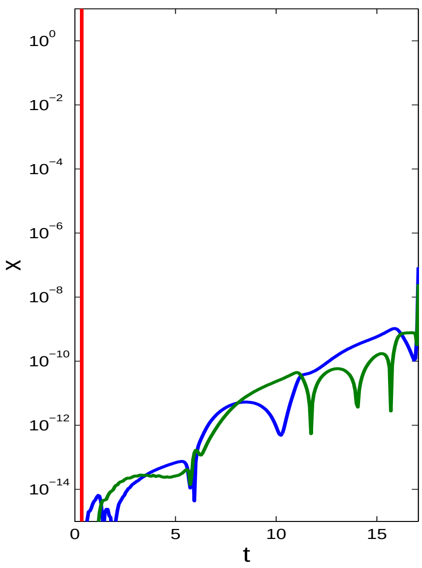

1.5
1
0.5
0
−0.5
−1
−1.5
−2
−1
0
1
2
x1
x2

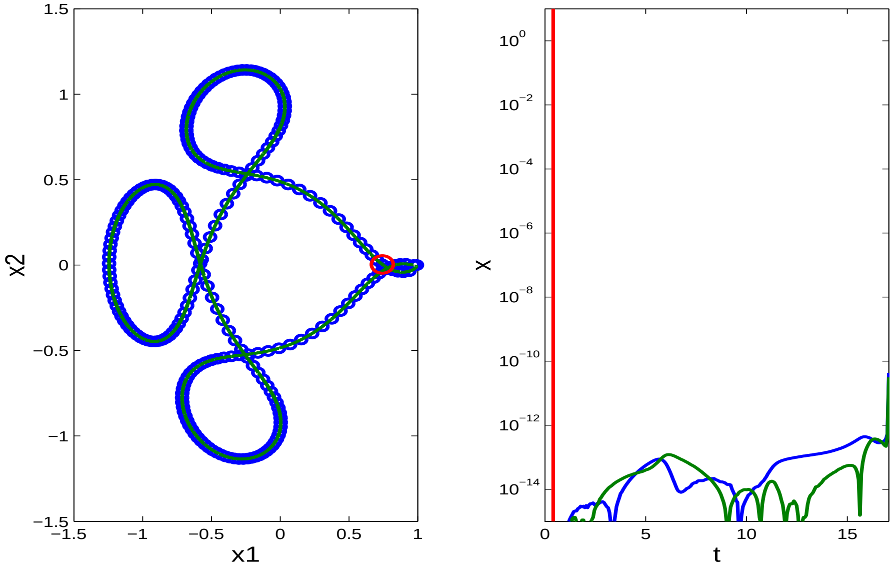

10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
0
5
10
15
t
x


1.5
1
0.5
0
−0.5
−1
−1.5
x2
−1.5
−1
−0.5
0
0.5
1
x1
10^0
10^−2
10^−4
10^−6
10^−8
10^−10
10^−12
10^−14
x
0
5
10
15
t


1.5
1
0.5
0
−0.5
−1
−1.5
x2
−1.5
−1
−0.5
0
0.5
1
x1
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
x
0
5
10
15
t

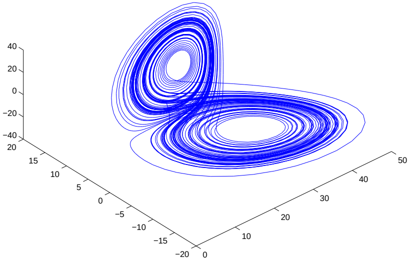

1.5
1
0.5
0
−0.5
−1
−1.5
x2
−1.5
−1
−0.5
0
0.5
1
x1


10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
x
0
5
10
15
t


1.5
1
0.5
0
−0.5
−1
−1.5
x2
−1.5
−1
−0.5
0
0.5
1
x1
10^0
10^−2
10^−4
10^−6
10^−8
10^−10
10^−12
10^−14
x
0
5
10
15
t

Results for the Lorenz Equations

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

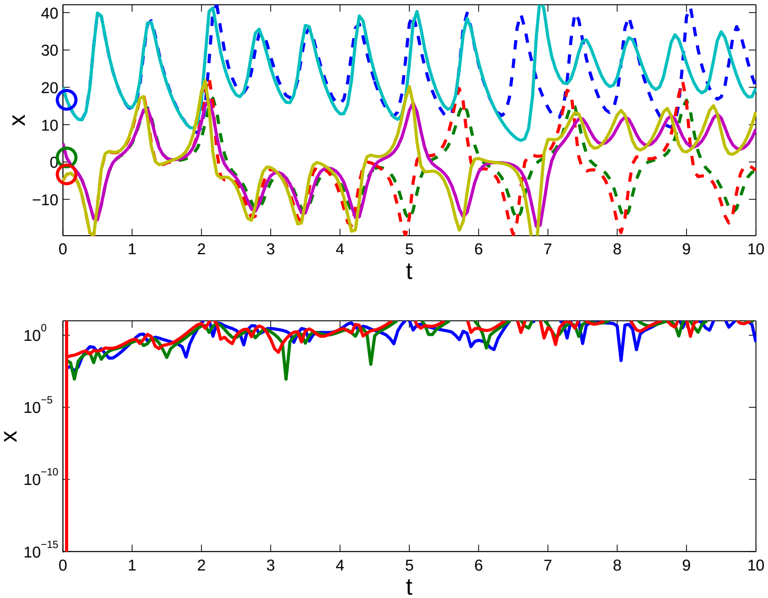


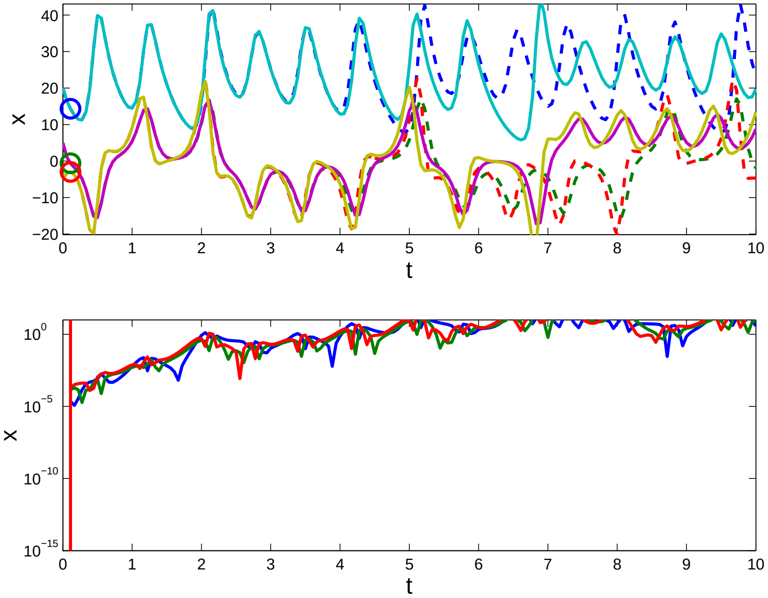
Parameters: $\sigma = 10$, $r = 28$ and $b = \frac{8}{3} \Longrightarrow$ chaotic regime.

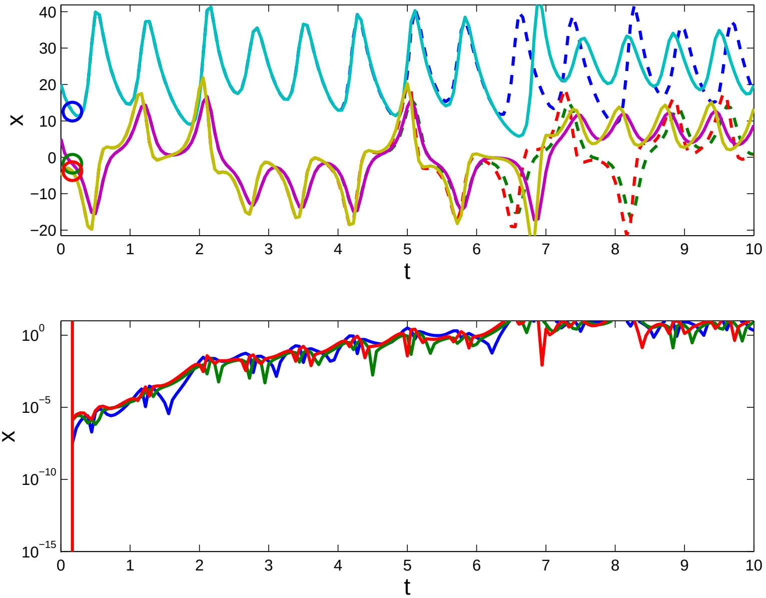
Initial conditions: $(x, y, z)(0) = (20, 5, -5)$

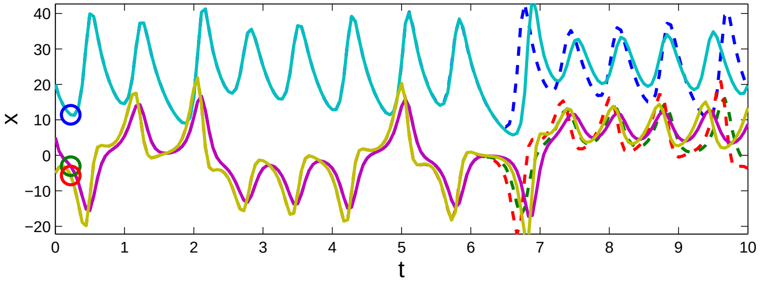
Simulation time: $t \in [0, T = 10]$

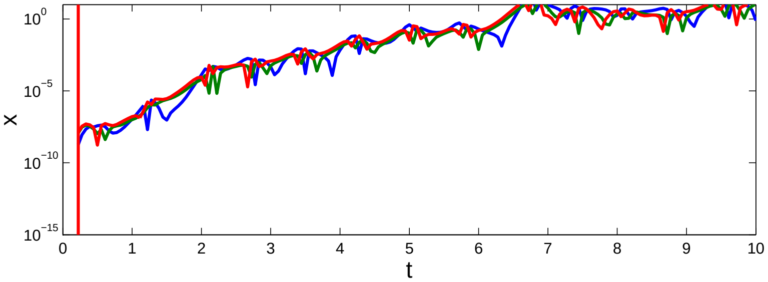
Discretization: Fourth order Runge Kutta, $\Delta T = \frac{T}{180}$, $\Delta t = \frac{T}{1800}$.

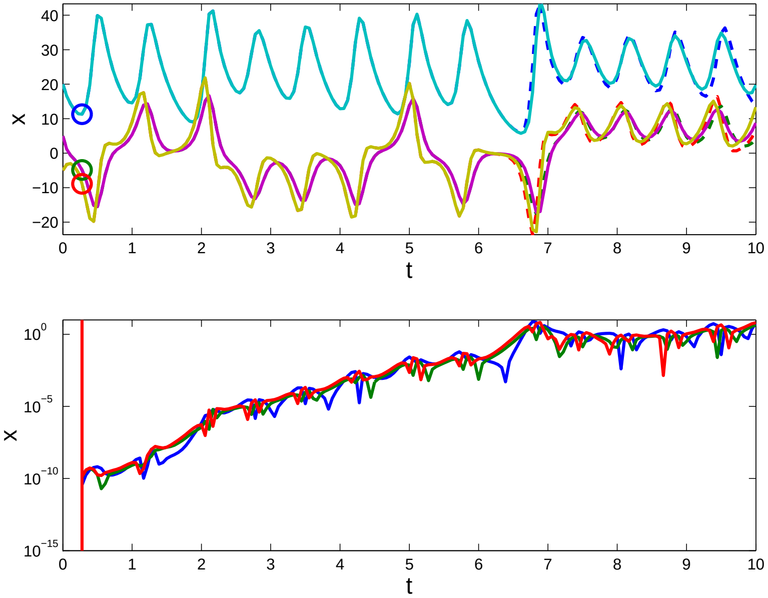

x
t
40
30
20
10
0
−10
0 1 2 3 4 5 6 7 8 9 10
10^{0}
10^{-5}
10^{-10}
10^{-15}

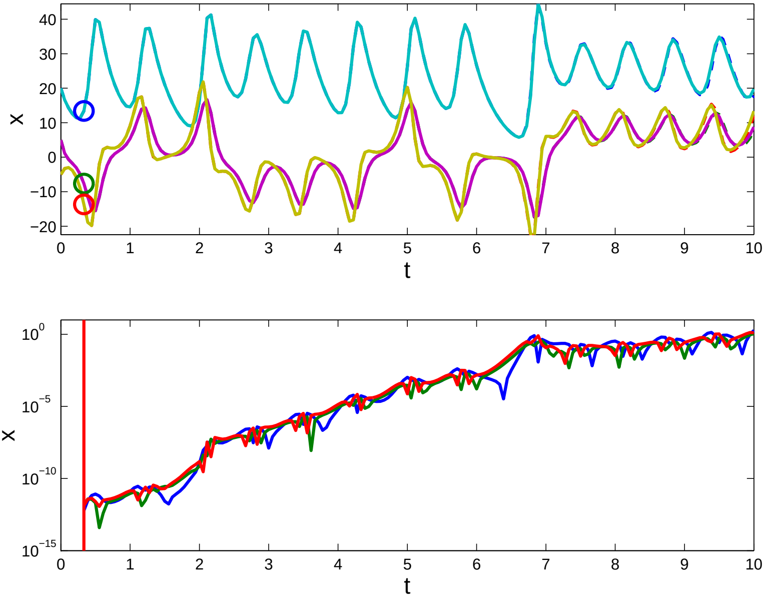

40
30
20
10
0
−10
−20
x
0 1 2 3 4 5 6 7 8 9 10
t
10^{0}
10^{-5}
10^{-10}
10^{-15}
x
0 1 2 3 4 5 6 7 8 9 10
t

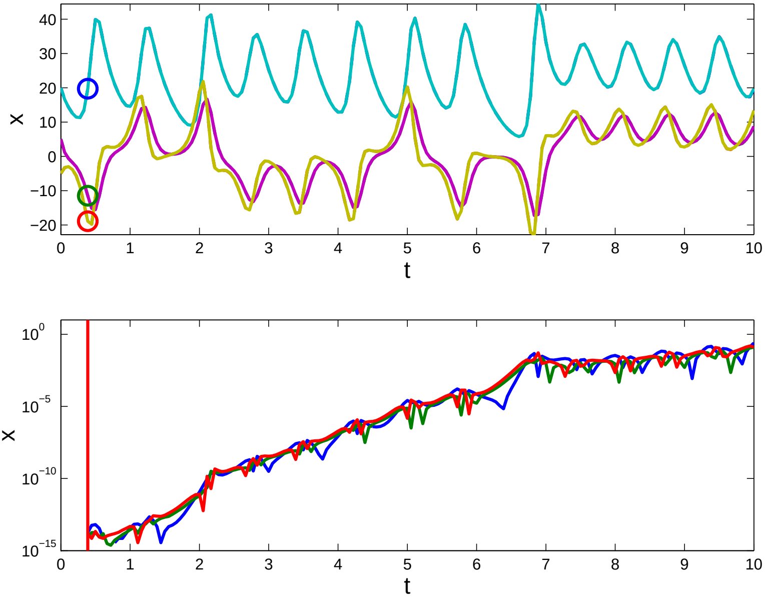

40
30
20
10
0
−10
−20
x
0 1 2 3 4 5 6 7 8 9 10
t
10^{0}
10^{-5}
10^{-10}
10^{-15}
x
0 1 2 3 4 5 6 7 8 9 10
t

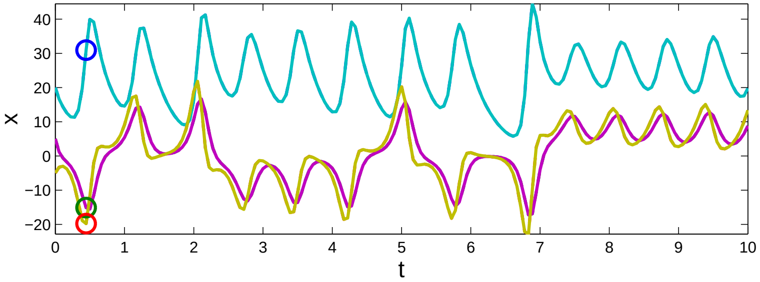

x
40
30
20
10
0
−10
−20
0
1
2
3
4
5
6
7
8
9
10
t

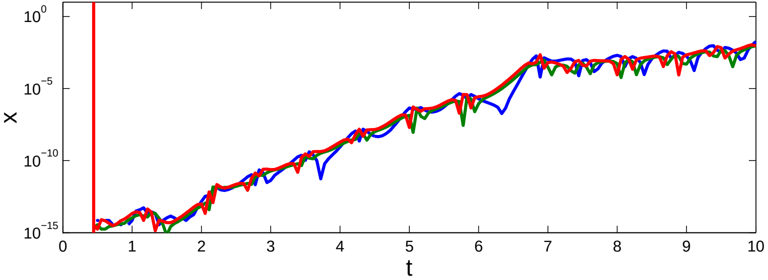

x
10^{0}
10^{-5}
10^{-10}
10^{-15}
0
1
2
3
4
5
6
7
8
9
10
t

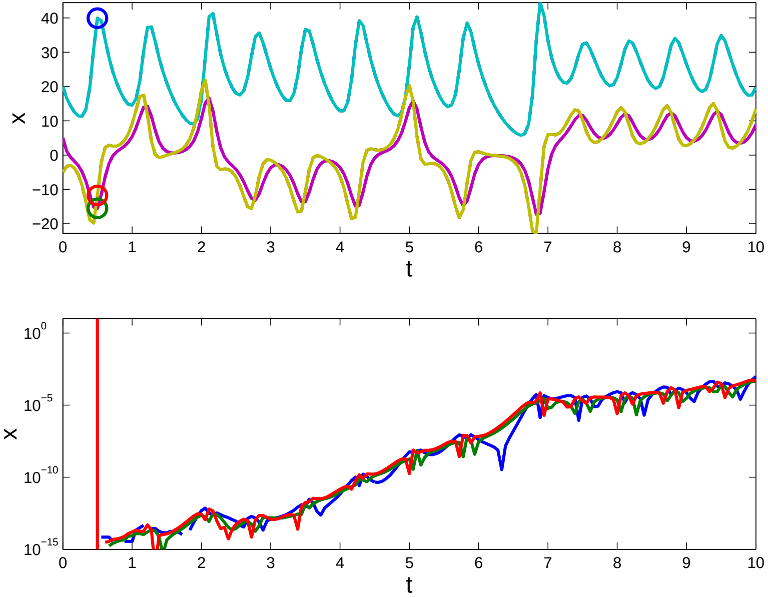

40
30
20
10
0
−10
−20
x
0 1 2 3 4 5 6 7 8 9 10
t
10^0
10^{-5}
10^{-10}
10^{-15}
x
0 1 2 3 4 5 6 7 8 9 10
t

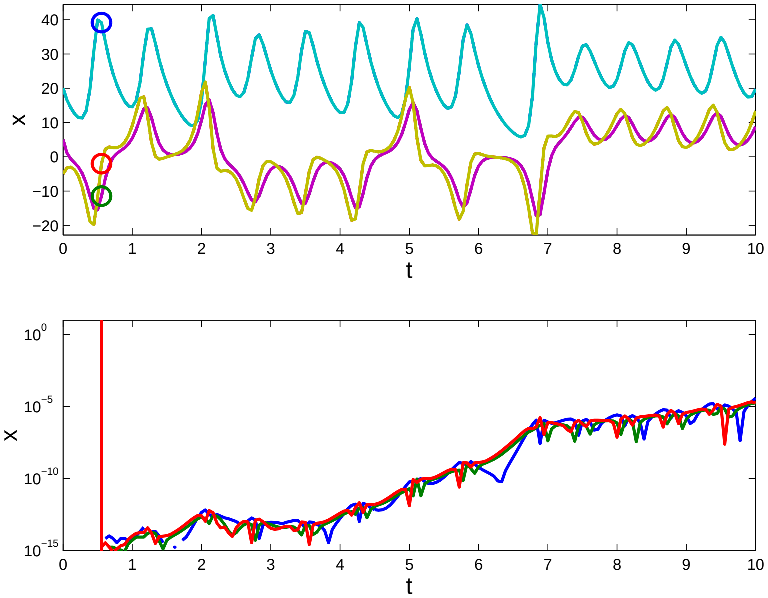

40
30
20
10
0
−10
−20
x
0 1 2 3 4 5 6 7 8 9 10
t
10^{0}
10^{-5}
10^{-10}
10^{-15}
x
0 1 2 3 4 5 6 7 8 9 10
t

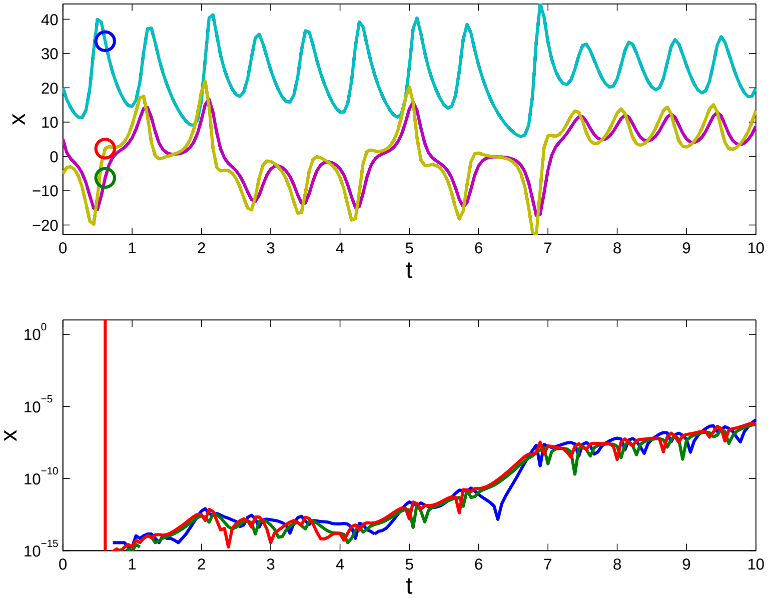

x
t
0
1
2
3
4
5
6
7
8
9
10
40
30
20
10
0
−10
−20
10^0
10^{-5}
10^{-10}
10^{-15}

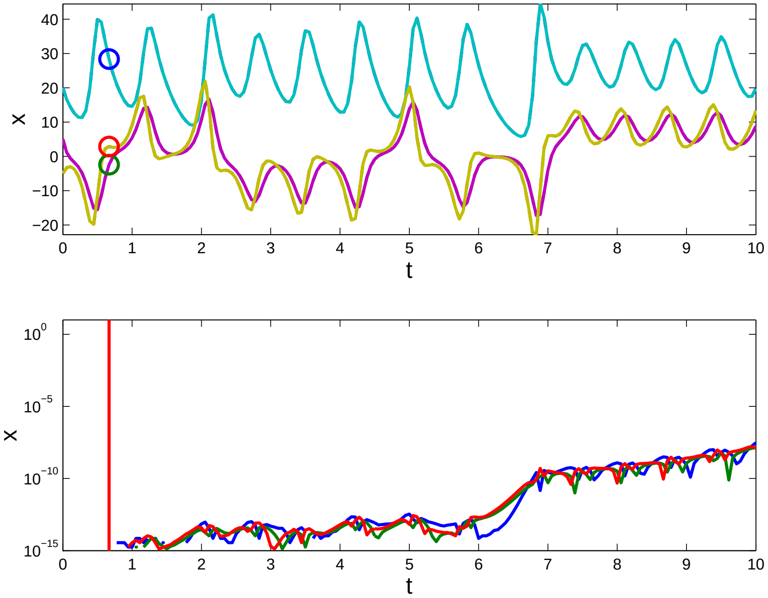

40
30
20
10
0
−10
−20
x
0
1
2
3
4
5
6
7
8
9
10
t

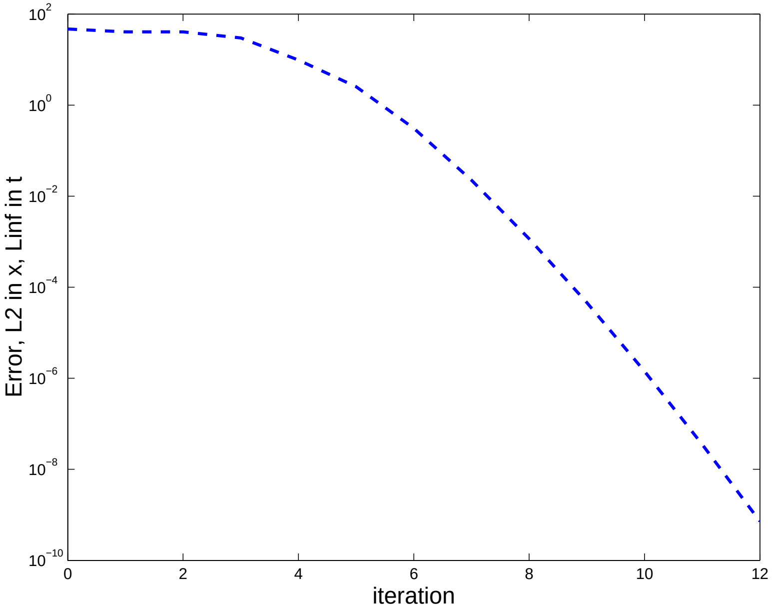

10^0
10^−5
10^−10
10^−15
x
0
1
2
3
4
5
6
7
8
9
10
t


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


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

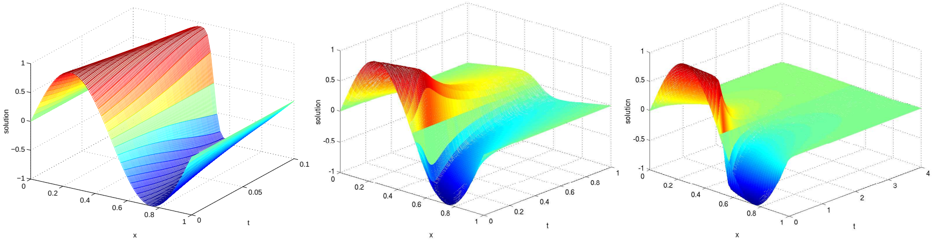

40
30
20
10
0
−10
−20
x
0 1 2 3 4 5 6 7 8 9 10
t
10^{0}
10^{-5}
10^{-10}
10^{-15}
x
0 1 2 3 4 5 6 7 8 9 10
t


10^2
10^0
10^−2
10^−4
10^−6
10^−8
10^−10
0
2
4
6
8
10
12
Error, L2 in x, Linf in t
iteration

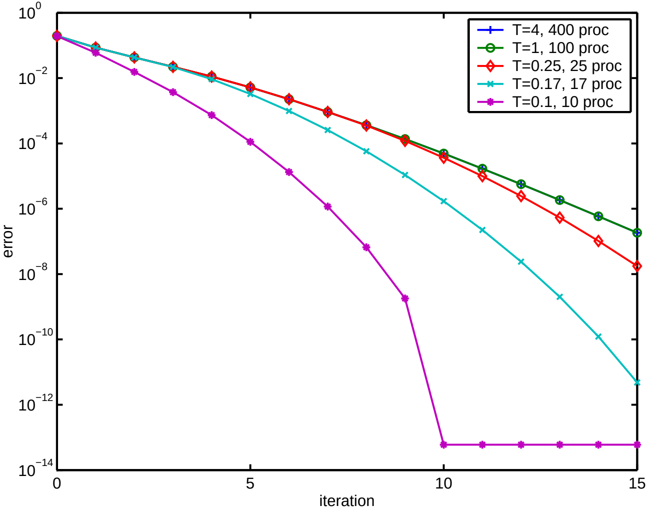
Numerical experiments for PDEs: Burgers equation

$$\begin{aligned} u_t + uu_x &= \nu u_{xx} \quad \text{in } \Omega = [0,1] \\ u(x,0) &= \sin(2\pi x) \end{aligned}$$

Viscosity $\nu = \frac{1}{50}$, homogeneous boundary conditions
Centered finite difference discretization, $\Delta x = \frac{1}{50}$
Backward Euler in time, $\Delta T = \frac{1}{10}$, $\Delta t = \frac{1}{100}$.



Burgers equation: convergence behavior



Convergence for the Heat Equation

Theorem

The parareal algorithm applied to the heat equation $u_t = \Delta u$ discretized with an L-stable method in time converges superlinearly on bounded time intervals,

$$\max_{1\leq n\leq N} ||u(t_n) - U_n^k||_2 \leq \frac{\gamma_s^k}{k!}\prod_{j=1}^{k}(N-j)\max_{1\leq n\leq N} ||u(t_n) - U_n^0||_2,$$

where the constant $\gamma_s < 1$ is universal for each L-stable method. On unbounded time intervals the convergence is linear,

$$\sup_{n>0} ||u(t_n) - U_n^k||_2 \leq \gamma_l^k \sup_{n>0} ||u(t_n) - U_n^0||_2,$$

where $\gamma_l < 1$ is universal for each L-stable method.

Convergence Constants for the Heat Equation

method	order	γ_s	γ_l
BE	1	0.2036321888	0.2984256075
SDIRK 3.1	3	0.1717941220	0.2338191487
SDIRK 3.2	3	0.2073822267	0.1718033767
Radau IIA	5	0.0634592650	0.0677592165

Note that higher order time integration methods lead to faster convergence of the parareal algorithm than lower order methods.

Convergence for pure Advection Problems

Theorem

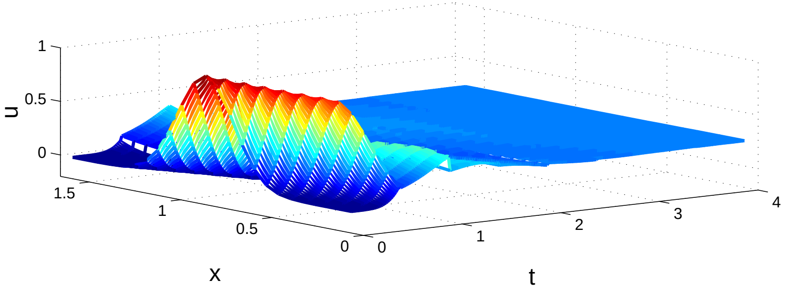
The parareal algorithm applied to the advection equation $u_t = u_x$ *with backward Euler in time converges superlinearly on bounded time intervals,*

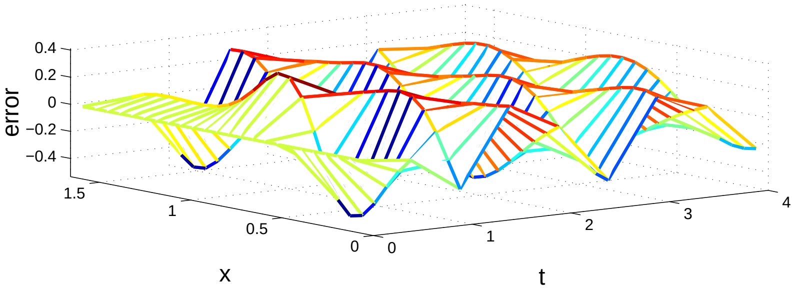
$$\max_{1\leq n\leq N} ||u(t_n) - U_n^k||_2 \leq \frac{\alpha_s^k}{k!}\prod_{j=1}^{k}(N-j)\max_{1\leq n\leq N} ||u(t_n) - U_n^0||_2,$$

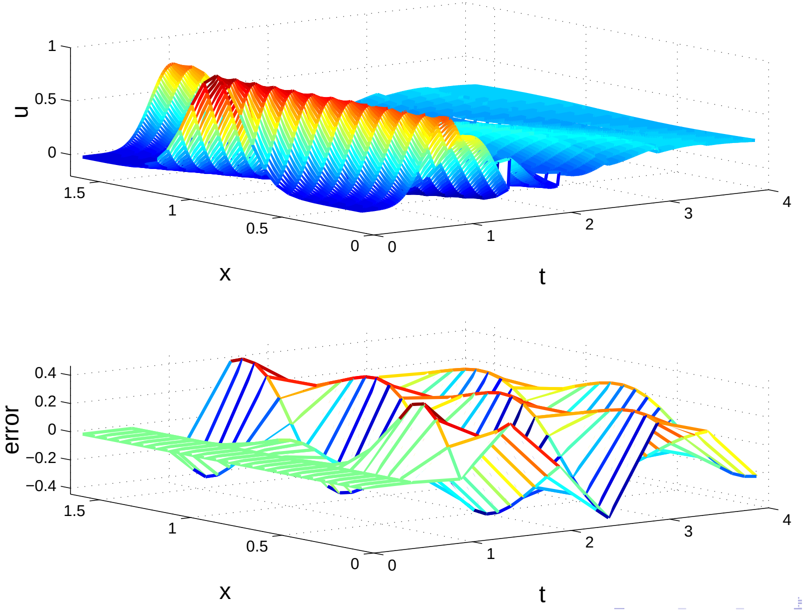
where the constant α_s *is universal,* $\alpha_s = 1.224353426$*.*

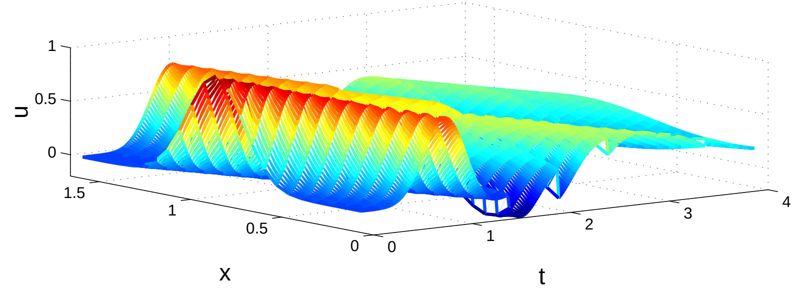
Remarks:

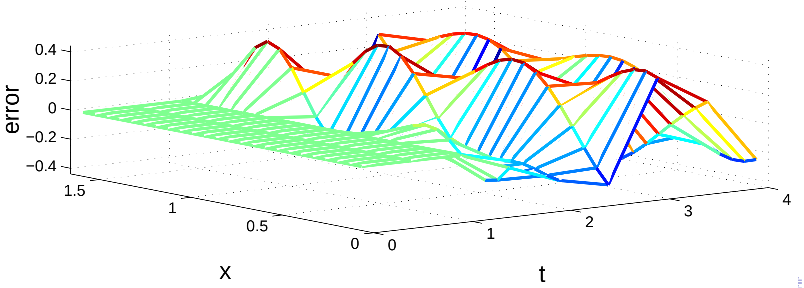
- No convergence result for unbounded time intervals.
- As soon as more than N iterations are needed, the method looses all interest.

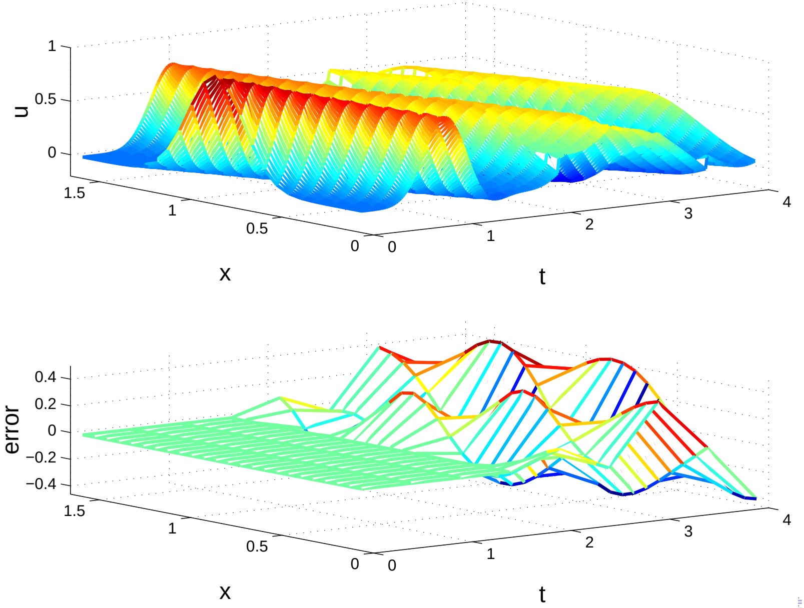

u
1
0.5
0
1.5
1
0.5
0
x
0
1
2
3
4
t

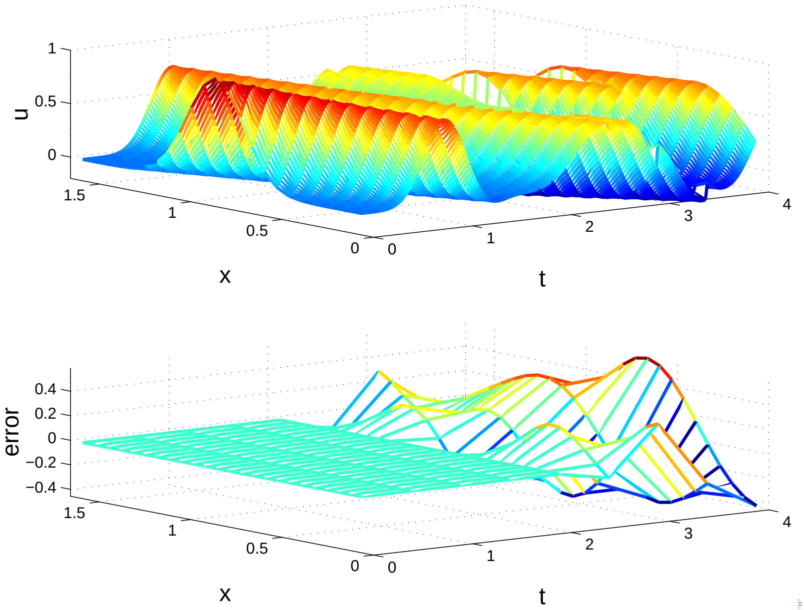

error
0.4
0.2
0
−0.2
−0.4
1.5
1
0.5
0
x
0
1
2
3
4
t

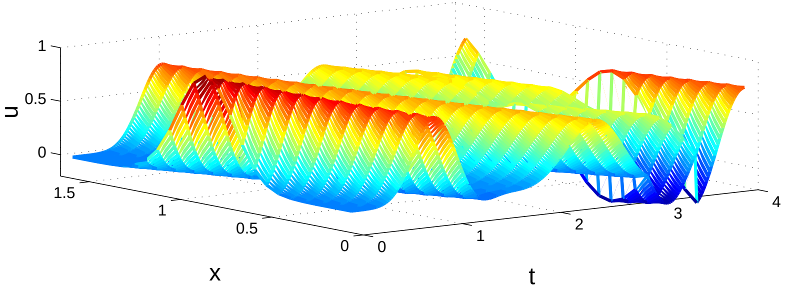

1
0.5
0
u
1.5
1
0.5
0
x
0
1
2
3
4
t
0.4
0.2
0
−0.2
−0.4
error
1.5
1
0.5
0
x
0
1
2
3
4
t

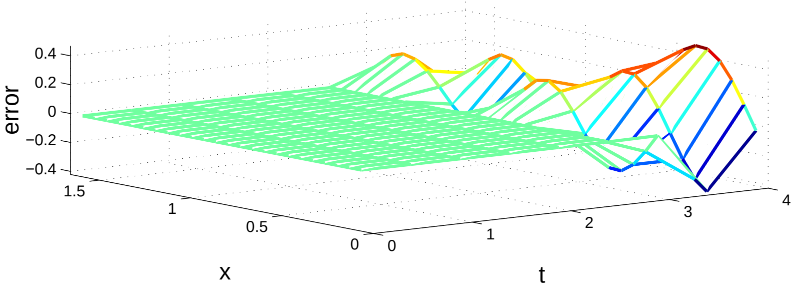

u
1
0.5
0
x
1.5
1
0.5
0
t
0
1
2
3
4

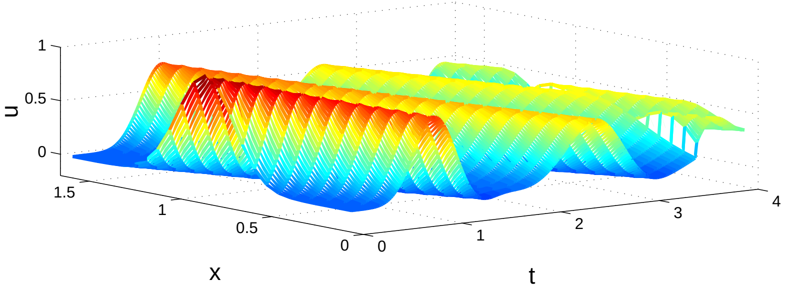

error
0.4
0.2
0
−0.2
−0.4
x
1.5
1
0.5
0
t
0
1
2
3
4

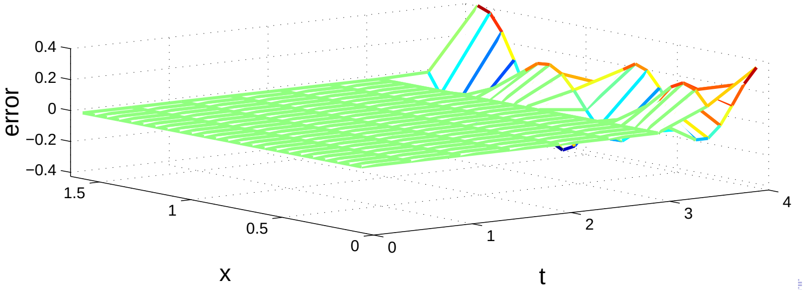

u
1
0.5
0
1.5
1
0.5
0
x
0
1
2
3
4
t
error
0.4
0.2
0
−0.2
−0.4
1.5
1
0.5
0
x
0
1
2
3
4
t

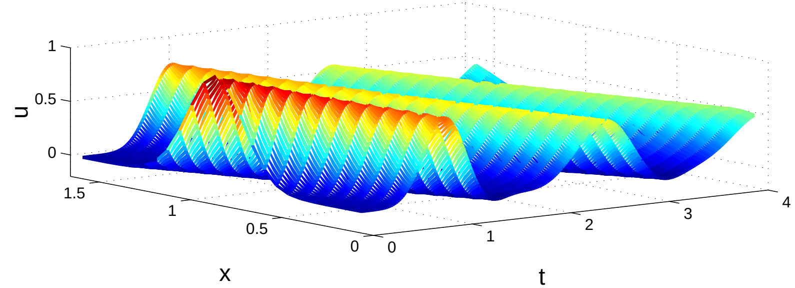

u
1
0.5
0
1.5
1
0.5
0
x
0
1
2
3
4
t
error
0.4
0.2
0
−0.2
−0.4
1.5
1
0.5
0
x
0
1
2
3
4
t

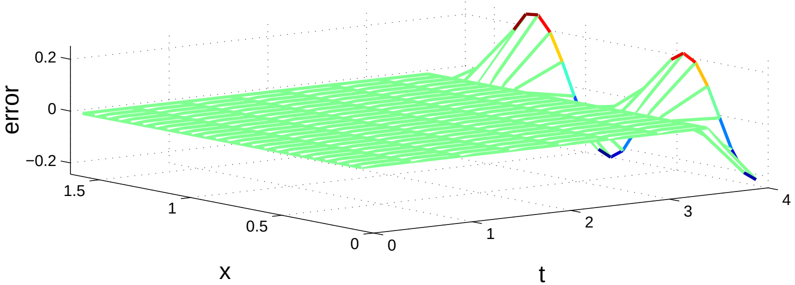

1
0.5
0
u
1.5
1
0.5
0
x
0
1
2
3
4
t

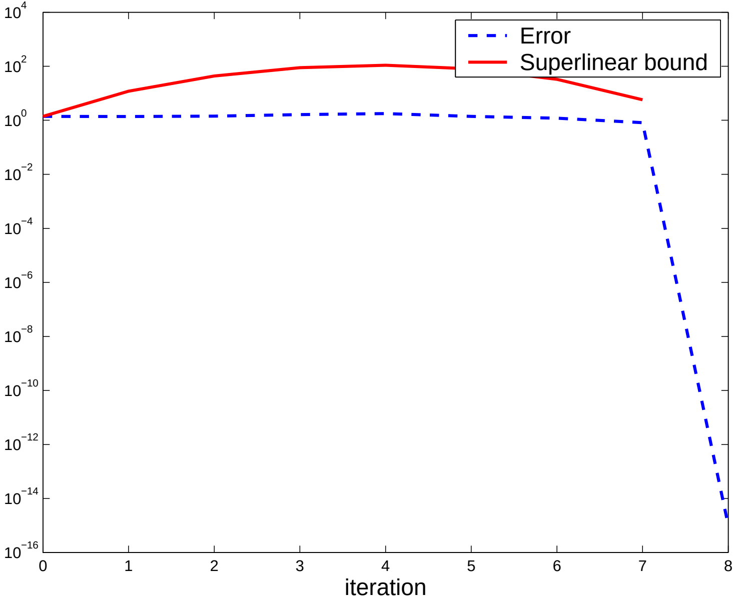

0.4
0.2
0
−0.2
−0.4
error
1.5
1
0.5
0
x
0
1
2
3
4
t

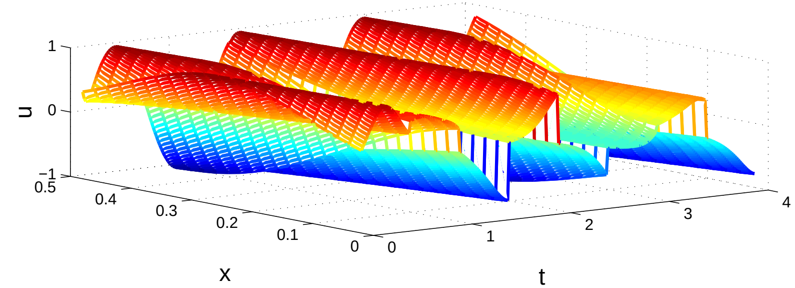

u
1
0.5
0
1.5
1
0.5
0
x
0
1
2
3
4
t

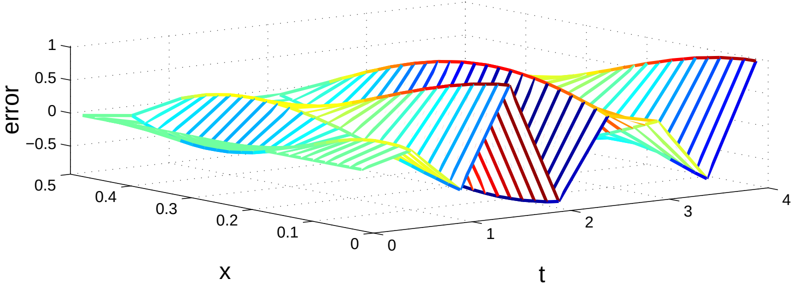

error
0.4
0.2
0
−0.2
−0.4
1.5
1
0.5
0
x
0
1
2
3
4
t

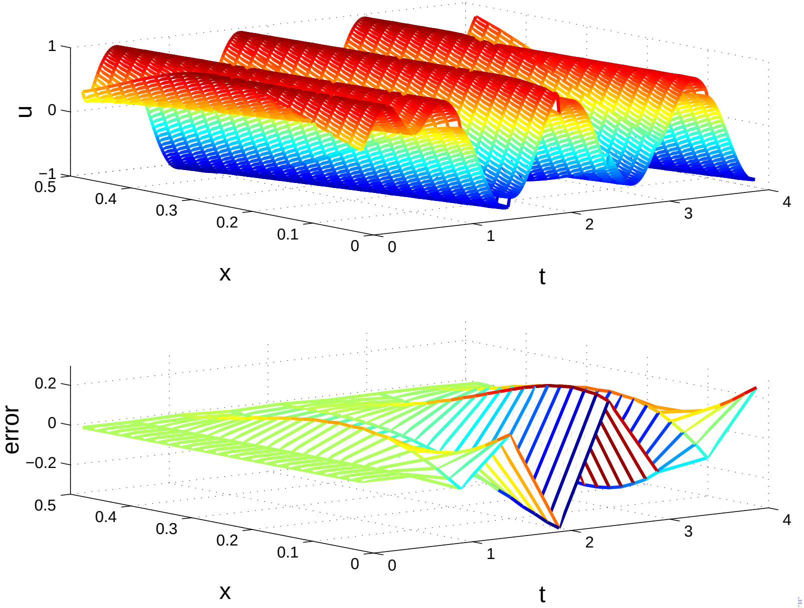

1
0.5
0
u
1.5
1
0.5
0
x
0
1
2
3
4
t

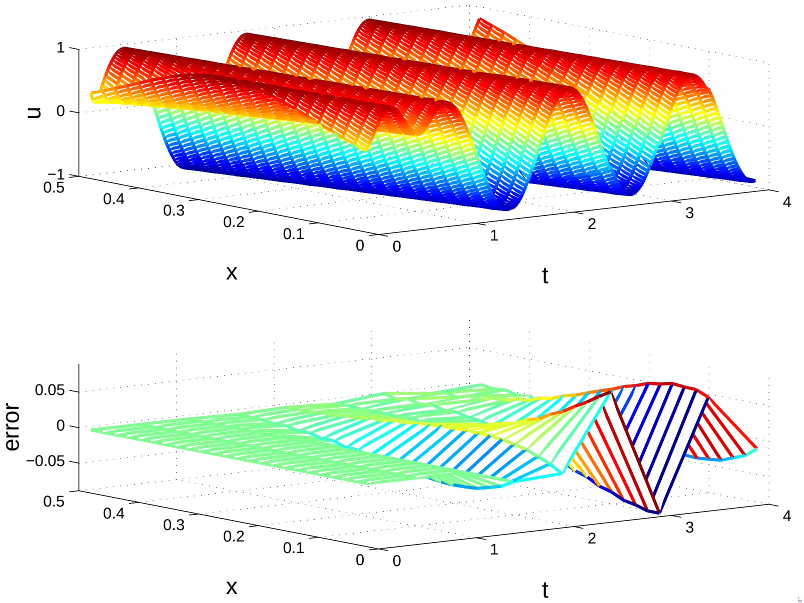

0.2
0
−0.2
error
1.5
1
0.5
0
x
0
1
2
3
4
t

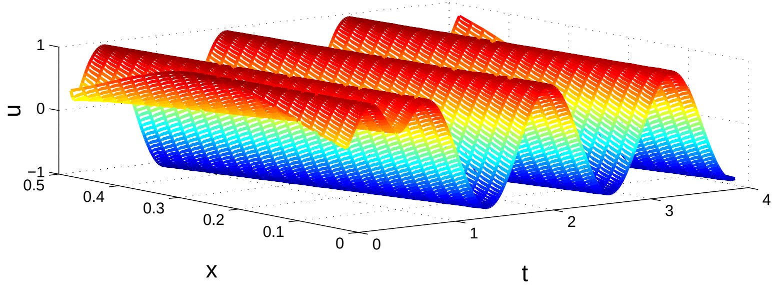

Error
Superlinear bound
iteration
10^{4}
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
10^{-16}
0
1
2
3
4
5
6
7
8

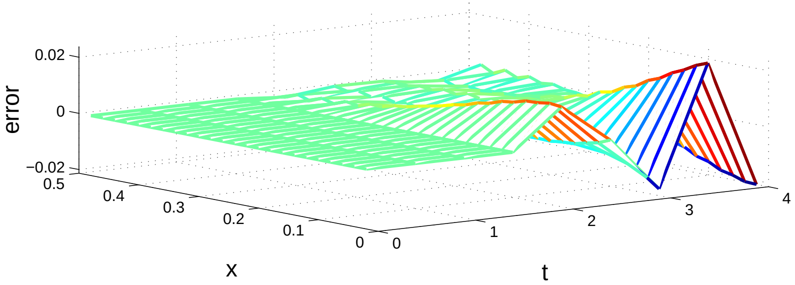

1
0
−1
u
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t

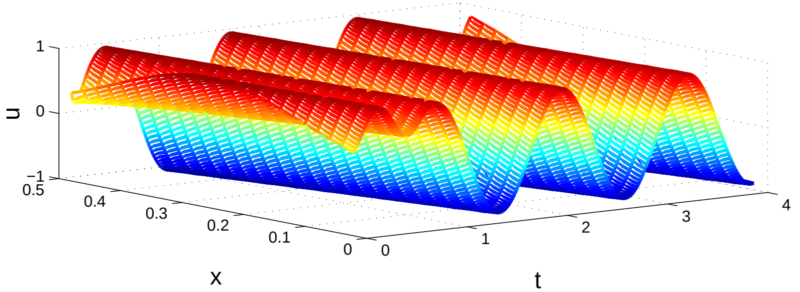

1
0.5
0
−0.5
error
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t

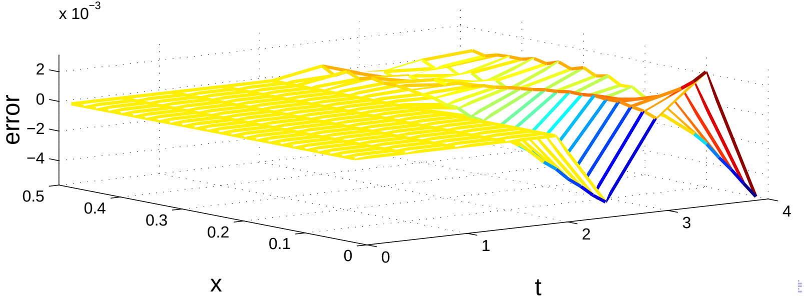

1
0
−1
u
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t
0.2
0
−0.2
error
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t

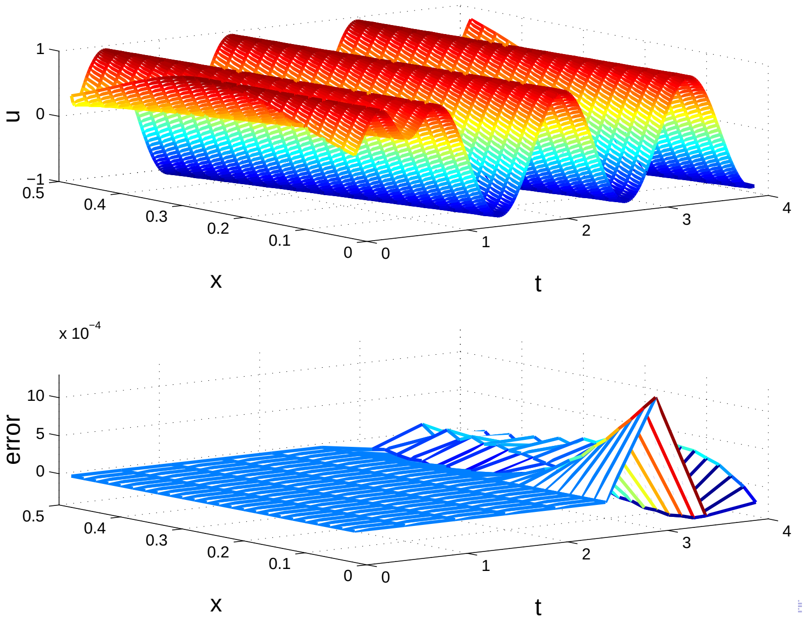

1
0
−1
u
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t
0.05
0
−0.05
error
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t

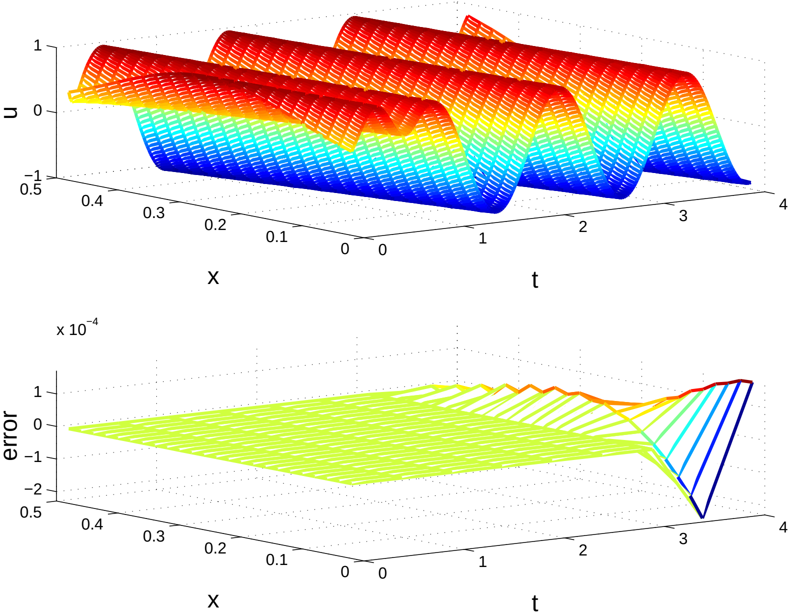

1
u
0
−1
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t

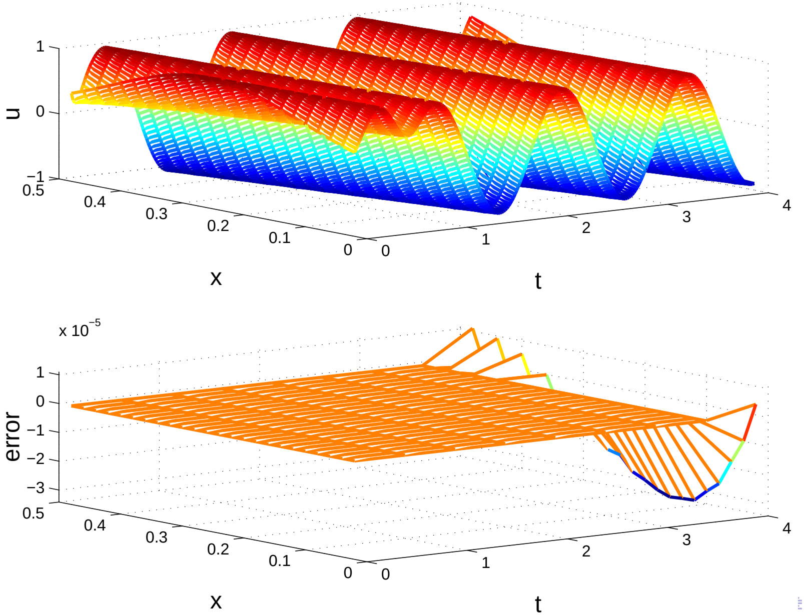

0.02
error
0
−0.02
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t

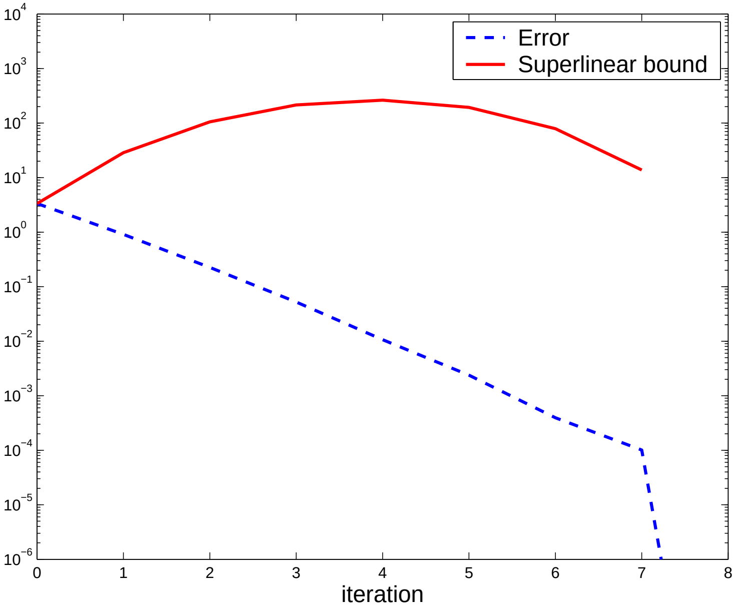

1
0
−1
u
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t


x 10^−3
2
0
−2
−4
error
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t


1
0
−1
u
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t
x 10^−4
10
5
0
error
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t


1
0
−1
u
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
x
t
x 10^−4
1
0
−1
−2
error
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
x
t


1
0
−1
u
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t
x 10^−5
1
0
−1
−2
−3
error
0.5
0.4
0.3
0.2
0.1
0
x
0
1
2
3
4
t


Error
Superlinear bound
iteration
0
1
2
3
4
5
6
7
8
10^4
10^3
10^2
10^1
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
10^-6

Further Applications and Results on Parareal

- **Oscillatory Problems:** Cortial, Farhat, Chandesris (2003, 2006)
- **Control of Quantum Systems:** Maday, Salomon, Turinici (2002, 2006)
- **Reservoir Simulation:** Garrido, Espedal, Fladmark (2003, 2005)
- **Navier-Stokes:** Fischer, Hecht, Maday (2003)
- **Stability Analysis:** Staff and Rønquist (2003)
- **Molecular Dynamics:** Baffico, Bernard, Maday, Turinici, Zerah (2002)
- **Finance:** Bal, Maday (2002)

Google hits for parareal algorithm (6.7.2006): 470

Conclusions

Parallel speedup in time is possible, but the speedup is more modest than in space.

Further results:

- Two multilevel versions of the algorithm.

Future work:

- Study of the hyperbolic case with boundary conditions, and the second order wave equation.
- Analysis of Parareal for DAEs.
- Preservation of symplectic structure in Parareal.